**Перечень теоретических вопросов к экзамену**

1. **Задачи сопровождения информационной системы**

Сопровождение ПО – процесс улучшения, оптимизации и устранения дефектов ПО после передачи в эксплуатацию.

Сопровождение – одна из фаз ЖЦ ПО следующая за фазой передачи ПО в эксплуатацию. В ходе сопровождения в программу вносятся изменения для того, чтобы исправить обнаруженные в процессе использования дефекты и недоработки, а также для добавления новой функциональности с целью повысить удобство использования и применимость ПО.

Сопровождение включает в себя следующий комплекс мероприятий:

1. тех поддержка

2. восстановление конфигурации и её изменение

3. проведение тех консультаций для персонала

4. оптимизация работы

5. ведение тех документации

6. профилактические мероприятия включая определение соответствия, используемого по целям и задачам заказчика и при необходимости установка новых версий.

Сопровождение ИС состоит из больших и разноплановых задач:

1. эксплуатация ИС – решение этой задачи начинается с установки прикладного ПО в определённом программно-аппаратном окружении и настройкой ПО.

В соответствии с документацией разработчика таким образом, чтобы обеспечить максимальную надёжность и производительность работы приложения.

2. внесение изменений в ИС. Изменения могут включать донастройки тиражируемого ПО или доработки заказного ПО. И то и другое требует привлечение консультантов, а также программистов, обладающих соответствующими компетенциями.

1. **Ролевые функции и организация процесса сопровождения**

* **Организация процесса сопровождения** подразумевает выполнение следующих действий:
  + определение цели и состава процессов сопровождения;
  + определение причин и видов изменения программного средства в процессе его сопровождения;
  + организация процессов и передача на сопровождение разработанного программного средства;
  + заключение договора между заказчиком и исполнителем на сопровождение программного средства;
  + разработка концепции методов и процессов сопровождения ПП;
  + разработка спецификации требований на модификации при сопровождении программного средства;
  + утверждение заказчиком концепции, договора и технического задания на сопровождение ПП;
  + организация контроля реализации концепции и договора на сопровождение программного средства.

Технология сопровождения должна обеспечить координированное развития множества версий прогр ср-ва и их компонентов каждая из которых даёт высокое кач-во и специфические функции.

Сопровождаемость – возможность регламентированной модификации, отражающей простоту внесения изменений в прогр. продукт после его ввода в эксплуатацию. Сопровождаемость должна быть определена до начала первичной разработки прогр. ср-ва соотв. Соглашениям между заказчиком и разработчиком. Разработчик готовит план обеспечения сопровождения, в котором должны быть отражены конкретные методы, ресурсы и последовательность работ

Требования к процессам сопровождения определяются группой основных факторов, влияющих на реализацию модификации прогр ср-в, образующих концептуальную цепочку: треб на изменение -> изменяемые функции -> размер или масштаб изменений -> стратегия модификации -> ресурсы необходимые для их реализации. Эта логическая схема обычно используется при последовательном анализе процессов сопровождения сложных прогр ср-в при этом основным критерием оценки сопровождения является совершенствование функциональной пригодности и улучшение характеристик кач-ва прогр. продукта. Осн. процесс эксплуатации в ЖЦ может инициировать процесс сопровождения прогр ср-ва путём представления предложений путём модификации отчёта о дефектах

Процесс сопровождения прогр ср-ва в соотв со стандартом ИСО 12207 и детализацией этого раздела в стандарте ИСО 14764 используют основной стандартизированный процесс разработки комплекса программ и вспомогательные процессы документирования, управления конфигурацией, обеспечение кач-ва, верификации, аттестации, совместного анализа, аудита и устранения дефектов.

1. **Сценарий сопровождения информационной системы**

Сценарий – это некоторая последовательность действий, иллюстрирующая поведение системы. В разработке ПО и сист проектир в сценарии использования представляют собой описание поведения системы, которые отвечают на внешние запросы. Сценарии использования описывают то, что и кто может сделать с рассматриваемой системой. Методика сценариев использования применяется для выявления требований к поведению системы.

Сценарии использования сосредоточены на том, как достигнуть конкретной цели или задачи, необходимой пользователю, как внешнему агенту от данной системы. Для большинства разрабат программ это означает необходимость разработки множества сценариев для определения общего набора решаемых задач.

Степень детализации сценариев зависит от сложности и тех стадии разрабат проекта. Сценарии использования описывают, что и как конкретно должна делать система в ответ на запросы пользователей, не касаясь при этом дизайна интерфейсов.

1. **Услуги по сопровождению информационных систем**

Услуги по сопровождению системы включают:

1. приём запросов

Получ запросов от

1) уполномоченных представителей заказчика

2) специалистов первой линии тех поддержки

Классификация запросов:

Вопрос – тип запроса требующий пояснения по базовому функционалу системы

Ошибка – тип запроса требующий реакции по ситуации. Которая не является частью нормального функционирования системы

Предложение по улучшению – тип запроса для пожелания заказчиков по улучшению системы

2. немедленное реагирование

1) фиксация и исправление ошибок и исправление ошибок и инцедентов в сроки соотв критичности запроса

2) применение для недопущения воспроизведения инцедента в будущем

3) предоставление консультаций по вопросам администрирования

4) проведение диагностических работ при возникновении нештатных ситуаций

3. периодичные работы

1) технолог аудит не менее 1 раза в квартал

2) документирование и информ обеспеч(обнов руководств, регламентов, инструкций, опис изменений настроек по результатам работ)

3) участие и/или оказание консультационных услуг в развёртывании в доп функциональности

4) подготовка периодической отчетности о трудозатратах на поддержку и контролю кач-ва оказанных услуг.

Состав услуг по развитию системы:

1) предоставление консультационных услуг по вопросам проектирования, модификации, настройки и разработки доп функциональности прикладного ПО

2) проведение анализа треб: сбор треб. Подготовка вариантов реализации, оценка сроков выполнения работ, расчет предварительной стоимости выполнения работ

3) формализация треб составления архитектурной и аналитической документации по проектам

4) разработка, доработка прогр кода

5) проведение кода ревью разработ компонентов

6) тестирование

7) перенос результатов разработки на среды заказчика

8) написание сопроводительной документации

9) гарантийная поддержка результатов услуг

1. **Договор на сопровождение информационной системы**

Варианты сопровождения:

**по заявкам**

Обслуживание ИС производится структурным подразделением заказчика в случае возникновения аварийных ситуаций (отказ оборудования, отказ программного обеспечения) и заключается в разовом обращении в нашу компанию. При этом, заказчику необходимо самостоятельно выявить неисправный элемент и принять решение об организации его восстановления, что может привести к длительным срокам восстановления. Сюда же накладывается время реакции на обращение нашими специалистами, так как вызов неплановый.

**• локальное обслуживание по договору**

Установление договорных отношений с нашей компанией на обслуживание отдельных компонент системы, например, технических и/или программных. Наиболее распространенной услугой является обслуживание ККМ в различных временных режимах со сроками реакции, например, до 4-х, 8-и, 24-х часов с момента регистрации вызова.

Качество услуг обеспечивается созданием в компании так называемого «страхового фонда»: мобильной инженерной группы и материальных ресурсов (подменного оборудования, запчастей, расходных материалов), ориентированных на устранение экстренных ситуаций. Таким образом, заказчик, не производя полного объема затрат на создание такой инфраструктуры, фактически в состоянии воспользоваться ею в нужный момент.

**• регламентированое сопровождение**

Данный уровень сопровождения характеризуется передачей нашей компании ряда активных функций по обеспечению жизнедеятельности ИС. Здесь объем услуг определяется основной целью — обеспечение бесперебойного функционирования ИС — и включает комплекс профилактических мероприятий, направленных на предупреждение возникновения сбоев: мониторинг состояния ресурсов и параметров компонент ИС; планирование штатных административных процедур; администрирование (в том числе удаленное) серверов, БД, пользователей, сетей и сетевых ресурсов; организация систем резервного копирования и систем безопасности.

• **аудит/консультирование**

Обеспечение жизнедеятельности ИС не сводится исключительно к сопровождению ее текущего состояния. Как любой живой бизнес, ИС должна отражать его состояние в изменении и развитии. Процесс развития должен соответствовать развитию направлений основного бизнеса, а меры, его обеспечивающие, должны быть реализованы в рамках единой стратегии с обязательным условием сохранения преемственности реализованных решений.

Активные меры по планированию развития компонент ИС заключаются в обеспечении достаточности ресурсов и прогнозирования их изменения на основании аудита информационных решений. В рамках аудита выполняются планирование мероприятий по модернизации аппаратных и программных компонент ИС; разработка поэтапных планов и методов развития и модернизации компьютерных систем; разработка временных графиков и оценка стоимости приобретения оборудования и т. д.

Проектам развития предшествует стадия технической и коммерческой экспертизы, которая уже традиционно выполняется с привлечением консультантов специализированных и консалтинговых компаний. Наша компания может быть привлечена для выполнения работ по всем проводимым активным мерам в качестве системного интегратора.

• **полное сопровождение**

Передача функций по обеспечению жизненного цикла ИС как единого технологического комплекса нашей компании на принципах out-sourcing, что включает в себя передачу исполнителю функций обеспечения ресурсами; создания компонент ИС; внедрения прикладных задач; проведения обучение персонала, выполнения информационных услуг и т. д.

Договор на сопровождение:

1 предмет договора

2 права и обязанности сторон

3 учет раб времени и оказанных услуг

4 стоимость услуг, порядок и сроки расчетов

5 срок действия договора

6 ответственность сторон

7 конфиденциальность

8 интеллект собственность

9 обстоятельства непреодолимой силы

10 разрешение споров

11 юрид адреса и банковские реквизиты сторон

1. **Анализ исходных программ и компонентов программного средства**

оценивание корректности программных средств можно представить двумя видами работ:

* верификацией — последовательным ***прослеживанием*** *сверху вниз* реализации требований к системе и ПС программными компонентами нижних уровней;
* определением полноты ***покрытия тестами*** их структуры и проверками выполнения исходных требований к ПС и его компонентам.
* Покрытие тестами может оцениваться по степени охвата тестированием ***набора требований*** к программе или по ***покрытию тестами структуры*** программы. Прослеживание и покрытие тестами набора требований к программам трудно формализовать и оценить их влияние на достигаемую корректность ПС. Имеющийся опыт показывает, что такой анализ вполне доступен для неформального анализа, но относительно слабее влияет на корректность, чем недостаточное тестирование структуры программ. Поэтому ниже внимание сосредоточено на оценивании тестирования и корректности структурного покрытия программ.
* Анализ и оценивание покрытия тестами структуры программ позволяет выявить дефекты и ошибки, угрожающие наиболее тяжелыми последствиями при функционировании ПС. Ограниченные ресурсы трудоемкости и времени на тестирование сложных комплексов программ для обеспечения их максимальной корректности приводят к необходимости рационального использования доступных ресурсов, прежде всего, для устранения самых опасных ошибок. Тестирование фрагментов структуры программы не гарантирует полное отсутствие ошибок в них, однако существенно снижает их вероятность. Пропущенные при тестировании фрагменты структуры заведомо могут содержать невыявленные ошибки, которые негативно отражаются на корректности программ. Таким образом, тестирование структурного покрытия программ является ***необходимым*** условием обеспечения их относительной корректности, однако оно ***недостаточно*** для полного обеспечения корректности их функционирования. В то же время тестирование и покрытие тестами структуры программ наиболее доступно формализации для оценивания достигаемой относительной корректности и вероятности отсутствия ошибок в программе и позволяет устранять наиболее опасные ошибки, угрожающие отсутствием или полным искажением требуемых результатов на выходе ПС.

1. **Программная инженерия и оценка качества**

Для создания ядра знаний ПО был создан международный комитет при американском объединении компьютерных специалистов ACM (Association for Computing Machinery) и институте инженеров по электронике и электротехнике IEEE Computer Society. В комитет вошли специалисты мирового уровня в области информатики и разработки ПО, которые внесли свой опыт и знания, а также систематизировали накопленные разнородные знания и определили (1999г., 2001г., 2004г.) ядро профессиональных знаний SWEBOK (Software Engineering Body Knowledge) программной инженерии [20], как основы проектирования ПО. Ядро включает сумму знаний, распределенную по 10 специализированным областям, которые отражают отдельные процессы проектирования ЖЦ ПО и методы их поддержки.

Программная инженерия (Software Engineering) является отраслью Computer science, изучает вопросы построения компьютерных программ, отражает закономерности ее развития, обобщает опыт программирования в виде комплекса общих знаний и правил регламентации инженерной деятельности разработчиков ПО. В этом определении важно рассмотреть два основных аспекта.

Требования – это свойства, которыми должно обладать ПО для адекватного задания функций, а также условия и ограничения на ПО, данные, среду выполнения и технику. Требования отражают потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПО. Заказчик и разработчик совместно проводят сбор требований, их анализ, пересмотр, определение необходимых ограничений и документирование. Различают требования к продукту и к процессу, а также функциональные и нефункциональные требования, системные требования. Программные требования определяют требования к процессу, ОС, режиму выполнения ПО, выбору платформы и т.п. 18 Функциональные требования задают назначение системы, а нефункциональные – условия выполнения ПО. Системные требования описывают требования к программной системе, состоящей из взаимосвязанных программных и аппаратных подсистем и разных приложений. Требования могут оцениваться количественно (например, количество запросов в сек., средний показатель ошибок не должен превышать 1.5% от объема вводимой информации и т.п.). Значительная часть требований относится к атрибутам качества: безотказность, надежность и др. Область знаний «Требования к ПО (Software Requirements)» состоит из следующих разделов: – инженерия требований (Requirement Engineering), – выявление требований (Requirement Elicitation), – анализ требований (Requirement Analysis), – спецификация требований (Requirement Specification). – проверка требований (Requirement validation), – управление требованиями (Requirement Menegement).

Проектирование ПО – процесс определения архитектуры, компонентов, интерфейсов, других характеристик системы и конечного результата. Область знаний «Проектирование ПО (Software Design)» состоит из следующих разделов: – базовые концепции проектирования ПО (Software Design Basic Concepts), – ключевые вопросы проектирования ПО (Key Issue in Software Design), – структура и архитектура ПО (Software Structure and Architecture), 20 – анализ и оценка качества проектирования ПО (Software Design Quality Analysis and Evaluation), – нотации проектирования ПО (Software Design Notations), – стратегия и методы проектирования ПО (Software Design Strategies and Methods)

Архитектура проекта – высокоуровневое представление структуры, задаваемое с помощью паттернов, компонентов и их идентификация. Описание архитектуры содержит описание логики отдельных компонентов системы, достаточное для проведения работ по кодированию, и связей между ними. Существуют и другие виды структур, основанные на проектировании образцов, шаблонов, семействе программ и их каркасов. Паттерн – это конструктивный элемент ПО, который задает взаимодействие объектов (компонентов) проектируемой системы, определение ролей и ответственности исполнителей. Основным языком задания этого элемента является UML. Паттерн может быть: структурным, в котором определяются типовые композиции структур из объектов и классов диаграммами классов, объектов, связей и др.; поведенческим, определяющим схемы взаимодействия классов объектов и их поведение диаграммами активностей, взаимодействия, потоков управления и др.; креативным, отображающим типовые схемы распределения ролей экземпляров объектов диаграммами взаимодействия, кооперации и др.

Нотации проектирования позволяют представить артефакты ПО и его структуру, а также поведение системы. Существует два типа нотаций: структурные, поведенческие и множество различных их представлений. Структурные нотации являются графическими, они используются для представления структурных аспектов проектирования, компонентов и их взаимосвязей, элементов архитектуры и их интерфейсов. К ним относятся формальные языки спецификаций и 21 проектирования: ADL (Architecture Description Language), UML (Unified Modeling Language), ERD (Entity–Relation Diagrams), IDL (Interface Description Language), классы и объекты, компоненты и классы (CRC Cards), Use Case Driven и др. Нотации включают языки описания архитектуры и интерфейса, диаграммы классов и объектов, диаграммы сущность-связь, компонентов, развертывания, а также структурные диаграммы и схемы. Поведенческие нотации отражают динамический аспект поведения систем и их компонентов. Таким нотациям соответствуют диаграммы: Data Flow, Decision Tables, Activity, Colloboration, Pre-Post Conditions, Sequence, таблицы принятия решений, формальные языки спецификации, языки проектирования PDL и др.

Конструирование ПО – создание работающего ПО с привлечением методов верификации, кодирования и тестирования компонентов. К инструментам конструирования ПО отнесены языки программирования и конструирования, а также программные методы и инструментальные системы (компиляторы, СУБД, генераторы отчетов, системы управления версиями, конфигурацией, тестированием и др.). К формальным средствам описания процесса конструирования ПО, взаимосвязей между человеком и компьютером и с учетом среды окружения отнесены языки конструирования. Область знаний «Конструирование ПО (Software Construction)» включает следующие разделы:

– снижение сложности (Reduction in Complexity), – предупреждение отклонений от стиля (Anticipation of Diversity), – структуризация для проверок (Structuring for Validation), – использование внешних стандартов (Use of External Standards)

Тестирование ПО – это процесс проверки работы программы в динамике, основанный на выполнении конечного набора тестовых данных и сравнения полученных результатов с запланированными вначале. Область знаний «Тестирование ПО (Software Testing)» включает следующие разделы: – основные концепции и определение тестирования (Testing Basic Concepts and definitions), – уровни тестирования (Test Levels), – техники тестирования (Test Techniques), – метрики тестирования (Test Related Measures), – управление процессом тестирования (Managing the Test Process).

Сопровождение ПО – совокупность действий по обеспечению работы ПО, а также по внесению изменений в случае обнаружения ошибок в процессе эксплуатации, по адаптации ПО к новой среде функционирования, а также по повышению производительности или других характеристик ПО. В связи с решением проблем 2000года сопровождение стало рассматриваться как более важный процесс, который должен строго обеспечиваться и обновляться участниками разработчиков. Новая версия системы должна решать те же задачи, иметь план переноса информации БД и учет стоимости сопровождения. Сопровождение (согласно стандартов ISO/IEC 12207 и ISO/IEC 14764) считается модификацией программного продукта в процессе эксплуатации при условии сохранения целостности продукта. Область знаний «Сопровождение ПО (Software maintenance)» состоит из следующих описаний разделов: – основные концепции (Basic Concepts), – процесс сопровождения (Process Maintenance), – ключевые вопросы сопровождения ПО (key Issue in Software Maintenance) , – техники сопровождения (Techniques for Maintenance).

Управление конфигурацией – дисциплина идентификации компонентов системы, определения функциональных и физических характеристик аппаратного и программного обеспечения для проведения контроля внесения изменений и трассирования конфигурации на протяжении ЖЦ. Это управление соответствует одному из вспомогательных процессов ЖЦ (ISO/IEC 12207), выполняется техническим и административным руководством проекта и заключается в контроле указанных характеристик конфигурации системы и их изменении; составления отчета о внесенных изменениях в конфигурацию и статус их реализации; проверки соответствия внесенных изменений заданным требованиям. Конфигурация системы – состав функций, программных и физических характеристик программ или их комбинаций, аппаратного обеспечения, обозначенные в технической документации системы и реализованные в продукте. Конфигурация ПО включает набор функциональных и физических характеристик ПО, заданных в технической документации и достигнутых в готовом продукте. Т.е это сочетание разных элементов продукта вместе с заданными процедурами сборки и отвечающие определенному назначению. Элемент конфигурации – график разработки, проектная документация, исходный и исполняемый код, библиотека компонентов, инструкции по установке системы и др

Управление инженерией ПО (менеджмент) – руководство работами команды разработчиков ПО в процессе выполнения плана проекта, определение критериев и оценка процессов и продуктов проекта с использованием общих методов управления, планирования и контроля работ. Как любое управление, менеджмент ПО предполагает планирование, координацию, измерение, контроль и отчет по процессу управления проектом; представляет собой системную, дисциплинированную и измеряемую разработку ПО. Координацию людских, финансовых и технических ресурсов при реализации задач проекта выполняет менеджер проекта, аналогично тому, как это делается в технических проектах. В его обязанности входит соблюдение запланированных бюджетных и временных характеристик и ограничений, стандартов и сформулированных требований. Общие вопросы управления проектом содержится в ядре знаний РMBOK [19] в разделе Management Process Activities, а также в стандарте ISO/IEC 12207 – Software life cycle processes [14], где управление проектом рассматривается как дополнительный и организайионный процесс ЖЦ,

1. **Процессы управления качеством программного обеспечения**

Управление качеством программного обеспечения (SQM, Software Quality Management) применяется ко всем аспектам процессов, продуктов и ресурсов. SQM определяет процессы, владельцев процессов, а также требования к процессам, измерения процессов и их результатов, плюс – каналы обратной связи. Процессы управления качеством содержат много действий. Некоторые из них позволяют напрямую находить дефекты, в то время, как другие помогают определить где именно может быть важно провести более детальные исследования, после чего, опять-таки, проводятся работы по непосредственному обнаружению ошибок. Многие действия также могут вестись с целью достижения и тех и других целей.

1. **Качество программного обеспечения**

Качество ПО – набор характеристик продукта или сервиса, которые характеризуют его способность удовлетворить установленным или предполагаемым потребностям заказчика. Понятие качества имеет разные интерпретации в зависимости от конкретной системы и требований к программному продукту. Кроме того, в разных источниках таксономия и модели качества отличаются. Каждая модель имеет различное число уровней и общее число характеристик качества.

Планирование качества программного обеспечения включает:

(1) Определение требуемого продукта в терминах характеристик качества (см., например, область знаний “Управление программной инженерией”).

(2) Планирование процессов для получения требуемого продукта (см., например, области знаний “Проектирование” и “Конструирование”).

*Процессы управления качеством должны адресоваться вопросам, насколько хорошо продукт будет удовлетворять потребностям заказчика и требованиям заинтересованных лиц, обладать ценностью для заказчика и заинтересованных лиц и качеством, необходимым для соответствия сформулированным требованиям к программному обеспечению.*

SQM может использоваться для оценки и конечных и промежуточных продуктов.

Некоторые из специализированных процессов SQM определены в стандарте 12207:

• Процесс обеспечения качества (quality assurance process)

• Процесс верификации (verification process)

• Процесс аттестации (validation process)

• Процесс совместного анализа (joint review process)

• Процесс аудита (audit process)

Все эти процессы поддерживают стремление к достижению качества и, кроме того, помогают в поиске возможных ошибок. *Верификация* – попытка обеспечить *правильную разработку продукта* (продукт построен правильным образом; обычно, для промежуточных, иногда, для конечного продукта, *прим. автора*), в том значении, что получаемый в рамках соответствующей деятельности продукт соответствует спецификациям, заданным в процессе предыдущей деятельности. *Аттестация* – попытка обеспечить *создание правильного продукта* (построен правильный продукт; обычно, в контексте конечного продукта, *прим. автора*), с точки зрения достижения поставленной цели. Оба процесса – верификация и аттестация – начинаются на ранних стадиях разработки и сопровождения. Они обеспечивают исследованию (экспертизу) ключевых возможностей продукта как в контексте непосредственно предшествующих результатов (промежуточных продуктов), так и с точки зрения удовлетворения соответствующих спецификаций.

1. **Реинжиниринг программного обеспечения**

Определение реинжиниринга бизнеса, данное М. Хаммером и Дж. Чампи2 содержит четыре ключевых слова: «фундаментальный», «радикальный», «резкий (скачкообразный)» и «процесс» (наиболее важное слово). Фундаментальный. На начальной стадии реинжиниринга необходимо ответить на такие основные вопросы: 1) почему компания делает то, что она делает? 2) почему компания делает это таким способом? 3) какой хочет стать компания? Отвечая на эти вопросы, специалисты должны переосмыслить текущие правила и положения (зачастую не сформулированные в письменной форме) ведения бизнеса и часто оказывающиеся устаревшими, ошибочными или неуместными. Радикальный. Радикальное перепроектирование - это изменение всей существующей системы, а не только поверхностные преобразования, т.е. входе радикального перепроектирования предлагаются совершенно новые способы выполнения работы. Резкий (скачкообразный). Реинжиниринг не применяется в тех случаях, когда необходимо улучшение либо увеличение показателей деятельности компании на 10-100%, а используются более традиционные методы (от произнесения зажигательных речей перед сотрудниками до проведения программ повышения качества), применение которых не сопряжено со значительным риском. Реинжиниринг целесообразен только в тех случаях, когда требуется достичь резкого (скачкообразного) улучшения показателей деятельности компании (500-1000% и более) путем замены старых методов управлении новыми.

Проект реинжиниринга бизнеса обычно включает четыре этапа:

1. Разработка образа-видения *(vision)* будущей компании. На этом этапе компания строит картину того, как следует развивать бизнес, чтобы достичь стратегических целей;  
2. Анализ существующего бизнеса — проводится исследование компании и составляются схемы ее работы в настоящий момент;  
3. Разработка нового бизнеса — создаются новые и (или) изменяются прежние процессы и поддерживающая их информационная система, тестируются новые процессы;  
4. Внедрение проекта нового бизнеса.

**Принципы перепроектирования бизнес-процессов**

Бизнес-процессы весьма разнообразны, но существуют определеннее требования, которым все они должны отвечать. Можно выделить следующие принципы организации бизнес-процесов, сформированных в ходе проведения реинжиниринга.

*1. Интегрирование бизнес-процессов.* Наиболее характерное свойство перепроектированных процессов — отсутствие сборочных конвейеров как способа координации работы персонала с относительно простыми трудовыми функциями. При выполнении сложных трудовых функций требуется иная организация работ. На практике, конечно, не всегда удается свести все этапы процесса к работе, выполняемой одним человеком. В этом случае создается команда, которая несет ответственность за данный процесс. Возможны сбои и ошибки, но потери будут значительно меньше, чем при традиционной организации работ.

*2. Горизонтальное сжатие бизнес-процессов.* Сравнительные оценки, выполненные компаниями, которые провели реинжиниринг, показывают, что переход от традиционной организации работ к выполнению процесса одним человеком позволяет снизить численность персонала и ускорить выполнение процесса примерно в 10 раз. Уменьшается количество ошибок и отпадает необходимость держать специалистов для устранения этих ошибок. За счет уменьшения численности работающих и четкого распределения ответственности между ними улучшается управляемость.

*3. Децентрализация ответственности (вертикальное сжатие бизнес-процессов).* Исполнители принимают самостоятельные решения в случаях, в которых раньше они традиционно должны были обращаться к руководству.

*4. Логика реализации бизнес-процессов.* Линейное выполнение работ заменяется логическим порядком (т.е. часто работы осуществляются параллельно). Это экономит время, которое тратилось на взаимоувязку работ на разных участках.

*5. Диверсификация бизнес-процессов.* Существуют различные варианты процессов выполнения. Традиционный процесс, ориентированный на производство массовой продукции, должен выполняться одинаково для всех входов, приводя к согласованным выходам. Традиционные процессы обычно оказываются очень сложными, так как они весьма детализированные и во многом рассчитаны на исключения и частные случаи.

*6. Разработка различных версий бизнес-процессов* в условиях постоянно меняющегося рынка необходима, чтобы процессы имели различные варианты в зависимости от ситуаций, входов и состояния рынка. Новые процессы, имеющие различные версии, начинаются с проверочного шага, на котором определяется, какая версия процесса наиболее подходит для текущей ситуации. Поэтому новые процессы в отличие от традиционных проще и понятнее, так как каждый вариант ориентирован только на одну, соответствующую ему ситуацию.

*7. Рационализация горизонтальных связей.* Создание линейных функциональных подразделений. Работа выполняется в том месте, где это наиболее целесообразно. Раньше в компаниях работа была организована по «тематическому» принципу в соответствующих подразделениях: расчетный отдел, транспортный отдел, отдел снабжения и т.д., поэтому если расчетному отделу требовались карандаши, то он обращался в отдел снабжения с заявкой. Этот отдел находил производителя, договаривался о цене, размещал заказ, осматривал товар, оплачивал его и передавал в расчетный отдел. Этот процесс длителен и неэкономичен. Анализ, проведенный в одной из компаний, показал, что затраты на приобретение батарейки за 3 долл. составили 100 долл. При реинжиниринге чаще всего создаются горизонтальные управленческие связи между подразделениями. Это позволяет устранить излишнюю интеграцию.

*8. Рационализация управленческого воздействия.* Речь идет об уменьшении числа проверок и снижении степени управленческого воздействия, которые не приводят непосредственно к получению материальных ценностей. Поэтому задача реинжиниринга — осуществлять их только в той мере, в которой это экономически целесообразно.

*9. Культура решения задачи.* Предполагается минимизация согласований, так как они тоже не имеют материальной ценности. Задача реинжиниринга — минимизировать согласования в ходе исполнения процесса путем сокращения внешних контактов.

*10. Рационализация связей «компания — заказчик».* Совершенствование оргструктуры фирмы должно создать условия, при которых уполномоченный менеджер обеспечивает единый канал связей.

*11. Уполномоченный менеджер.* Этот принцип применяется в тех случаях, когда шаги процесса либо сложны, либо распределены таким образом, что их интеграция силами небольшой команды невозможна. Уполномоченный менеджер является буфером между сложным процессом и заказчиком. Менеджер во взаимоотношениях с заказчиком выступает ответственным за весь процесс. Чтобы сыграть эту роль, менеджер должен быть способен отвечать на вопросы заказчика и решать его проблемы. Содержание задачи обусловливает необходимость обеспечения доступа менеджера ко всем информационным системам, используемым в этом процессе, а также к его исполнителям.

*12. Сохранение положительных моментов централизации управления.* На практике это достигается путем совершенствования информационного обеспечения дивизиональной организации управления. Современные ИТ дают возможность подразделениям компании действовать автономно, сохраняя возможность пользования централизованными данными. Таким образом, компания может устранить бюрократические региональные структуры, необходимые для обслуживания территориально разобщенной клиентуры, и одновременно повысить качество обслуживания.

На процесс реинжиниринга оказывают существенное влияние следующие факторы.

*1. Мотивация.* Мотив осуществления проекта реинжиниринга должен быть ясно определен и зафиксирован. При этом высшее руководство должно быть абсолютно убеждено, что этот проект действительно даст значительный результат, и понимать, что полученный результат вызовет изменение структуры компании. Чтобы обеспечить успех, руководство должно верить в необходимость реинжиниринга, проводимого в масштабах всей компании, и предоставить в распоряжение команды по реинжинирингу лучшие силы.

*2. Руководство.* Проект должен выполняться под управлением руководителей компании; руководитель, возглавляющий проект реинжиниринга, должен иметь большой авторитет и нести за него ответственность. Для успеха проекта очень важно твердое и умелое управление. Руководитель проекта должен понимать, что возникнут трудности, неизбежные при построении новой компании: он должен сопротивляться «давлению» старых порядков и убедить своих сотрудников в том, что проект не только выполним, но и необходим для выживания компании. Он обязан прилагать все усилия для продвижения проекта и своевременного его завершения.

*3. Сотрудники.* В команде, выполняющей проект реинжиниринга и контролирующей его проведение, необходимо участие сотрудников, выделенных соответствующими полномочиями и способных создать атмосферу сотрудничества. Сотрудники должны понимать, почему проект приведен в действие (другими словами, они должны оценивать проблемы, которые мешают бизнесу), принимать свои новые обязанное быть способными выполнять их, посвящать реинжинирингу необходим время и обоснованно двигаться к успеху. По сути, все работники должны освоить и устойчиво реализовывать новый набор образцов поведения. Опыт показывает, что относительно просто объяснить новый способ работы персоналу нижнего уровня, но людям, занимающим должности менеджеров, намного труднее понять то, что предлагает новая компания. Группа, на которую следует обратить особое внимание специалистов,— менеджеры среднего уровня. Американский исследователь Б. Виллох определяет три категории менеджеров такого уровня:

1. «тигры» — молодые карьеристы, которые хотя и участвуют в проекте по реинжинирингу с энтузиазмом, имеют тенденцию концентрироваться на собственных задачах в ущерб общим целям проекта;  
2. «ослы» — старейшие сотрудники, достигшие пика карьеры, которые хотят спокойствия и стабильности в компании; они могут серьезно навредить проекту;  
3. «акулы» — сотрудники, которые разработали процедуры и инструкции для управления операциями компании; они часто имеют реальную силу в компании и могут создать огромные проблемы, саботируя реальные перемены в жизни компании.

*4. Коммуникации.* Новые задачи компании должны быть четко сформулированы и понятны каждому сотруднику. Успешность реинжиниринга зависит от того, насколько руководство и рядовые сотрудники понимают, как достичь стратегических целей компании.

*5. Бюджет.* Проект должен иметь свой бюджет, особенно если планируется интенсивное использование ИТ. Часто ошибочно считают, что реинжиниринг возможен на условиях самофинансирования. Поэтому реинжиниринг нужно рассматривать как венчурный по характеру проект.

*6. Технологическая поддержка.* Для проведения работ по реинжинирингу необходима поддержка — соответствующие методики и инструментальные средства. Реинжиниринг обычно включает в себя построение информационной системы для поддержки нового бизнеса.

*7. Консультации.* Эксперты (консультанты) могут оказать существенно помощь исполнителям, впервые осуществляющим реинжиниринг. Важно, чтобы консультанты исполняли поддерживающую, а не управляющую роль, и не входили в штат компании. Поэтому руководитель проекта реинжиниринга должен быть грамотным заказчиком услуг консультантов. К факторам, способствующим успеху реинжиниринга, можно отнести и такие, как риск, четко определенные роли и обязанности и осязание результаты.

1. **Цели и регламенты резервного копирования данных**

Администратор БД должен: ▫ минимизировать число операций по восстановлению данных, ▫ предупреждать проблемы до их возникновения, ▫ максимально ускорить восстановление при возникновении неполадок, ▫ в минимальный срок выполнить проверку успешности восстановления данных.

Обеспечение отказоустойчивости информационного хранилища • Для обеспечения отказоустойчивости рекомендуется: ▫ использование RAID-массив, обеспечивающий отказоустойчивость дисковой подсистемы; ▫ использование аппаратуры, обеспечивающей защиту от скачков сетевого напряжения; ▫ регулярное резервное копирование данных.

Предупреждение неисправностей в работе • Одна из задач администратора БД – предупреждение неисправностей и разработка схемы их устранения. • К числе неисправностей, возникающих в системе БД: ▫ потеря информации на диске с файлами данных; ▫ потеря информации на диске с журналом транзакций; ▫ потеря информации на системном диске; ▫ отказ сервера; ▫ стихийные бедствия; ▫ кража данных, кража информации; ▫ кража носителя резервных данных; ▫ неисправный носитель резервных данных; ▫ неисправное устройство восстановления данных; ▫ непреднамеренная ошибка пользователя; ▫ злонамеренные действия сотрудников.

Определение политики восстановления данных • При планировании процедур резервного копирования и восстановления данных необходимо определить следующие параметры: ▫ Какова стоимость простоя БД? ▫ Необходимость использования RAID-массивов? ▫ Каково время на восстановление данных с резервной копии БД? ▫ Поможет ли частое резервное копирование значительно сократить время на восстановление данных? ▫ Имеются ли в организации дополнительные устройства восстановления данных с резервных копий?

Типы резервного копирования БД

Полное резервное копирование БД • При полном резервном копировании БД копируются все экстенты БД, в том числе изменения внесенные во время ее последнего резервного копирования. • Копируются все пользовательские данные и объекты БД, системные таблицы, индексы и пользовательские БД. ▫ Полное резервное копирование применимо для любой модели восстановления данных • Первое полное резервное копирование рекомендуется выполнять после первоначального заполнения БД. • Впоследствии полное копирование выполняется при вводе большого объема данных. • Базовая команда выполнения резервного копирования: ▫ BACKUP DATABASE database\_name TO DISK = ‘disk\directory\file’ WITH INIT

Дифференциальное резервное копирование БД

• При дифференциальном копировании БД копируются все экстенты, измененные с последнего полного резервного копирования. Этот процесс включает в себя копирование БД отражает только самые последние изменения данных. • Основная цель дифференциального резервного копирования – уменьшение числа резервных копий журнала транзакций необходимых для восстановления данных. • Дифференциальное копирование требует меньше времени и места на носителе. • Для увеличения скорости дифференциального копирования SQL Server прослеживает все изменения в экстентах, которые произошли с момента последнего резервного копирования БД. Такие изменения отражаются на странице карты изменений (DCM). • Базовая команда выполнения копирования: ▫ BACKUP DATABASE database\_name TO DISK = ‘disk\directory\file’ WITH DIFFERENTIAL

Резервное копирование журнала транзакций

• Резервное копирование заключается в последовательной записи всех зарегистрированных транзакций с момента последнего резервного копирования журнала. • Использование резервных копий журнала позволяет восстановить данные до состояния на определенный момент времени. ▫ Резервные копии журнала транзакций могут быть использованы только в полной модели восстановления и в модели с неполным протоколированием. • Скорость выполнения резервного копирования журнала зависит от скорости транзакций, используемой модели восстановления и от объема операций. • Резервное копирование журнала транзакций относится к активной части журнала. • После завершения резервного копирования журнала транзакций SQL Server удаляет все виртуальные файлы журнала, которые не содержат активную часть журнала. • Базовая команда выполнения резервного копирования журнала транзакций: ▫ BACKUP LOG database\_name TO DISK = ‘disk\directory\file’ WITH INIT

Типы восстановления данных

|  |  |
| --- | --- |
| Тип восстановления | Описание |
| Полное восстановление БД | Восстановление с помощью полной резервной копии БД, последней дифференциальной копии и всех резервных копий журнала транзакций |
| Восстановление по состоянию на определенную дату | Восстановление всей БД, по состоянию на определенную дату с использованием полной информации о транзакциях в резервной копии журнала транзакций и копий БД |
| Восстановление файла или группы файлов | Полное восстановление файла с использованием резервных копий файла, последней дифференциальной копии файла и всех копий журнала транзакций |
| Восстановление до заданной транзакции | Восстановление БД до определенного состояния с использованием полной информации о транзакциях, а также резервных копий БД |

Типы восст данных

Автоматический процесс восстановления данных

• Задача автоматического процесса восстановления данных – гарантированность логической целостности данных в каждой БД после запуска сервера. • В автоматическом процессе используются информация из активной части журнала транзакций каждой БД. • Анализируются транзакции, зарегистрированные с момента последней проверки целостности БД. ▫ SQL Server определяет все подтвержденные транзакции и выполняет их повторно, затем выделяет все неподтвержденные транзакции и отменяет их. • В процессе проверки SQL Server проверяет системные базы данных в следующей последовательности: master, model, msdb, а также БД публикуемые для репликации, затем все пользовательские БД. • В конце процесса удаляется и вновь создается БД tempbd.

Восстановление данных вручную

• При восстановлении БД вручную используются одна или более резервные копии БД и выполняется процедура полного или частичного восстановления данных. • При восстановлении базы данных до состояния, в котором она находилась к моменту завершения последнего резервного копирования журнала транзакций, то начинают с последней полной резервной копии БД. ▫ Полная копия может быть восстановлена на любом сервере БД.

Восстановление данных по состоянию на определенный момент времени

Если требуется восстановление данных на определенный момент времени, то возможны следующие варианты: ◦ восстановление БД до состояния на определенный момент времени; ◦ восстановление БД до определенного места в журнале транзакций. Для восстановления данных по состоянию на определенный момент времени следует использовать полную резервную копию БД. Затем необходимо восстанавливать резервные копии журнала транзакций к выбранному моменту времени. При восстановлении последней резервной копии устанавливается срок восстановления внутри данной резервной копии и выбирается режим восстановления до указанного срока. Для восстановления данных до определенного состояния, необходимо размещать в журнале транзакций специальные метки. При добавлении такой метки в таблицу БД msdb вносятся соответствующая запись. В процессе восстановления можно восстанавливать данные до определенной отметки.

Планирование процедур резервного копирования баз данных

• При планировании процедур резервного копирования рекомендуется выполнять архивирование всех БД, которые требуют полного восстановления после любых сбоев. • Обязательно необходимо полное резервное копирование БД master. • Другая системная БД – msdb содержит информацию о работе агента SQL Server, репозитарий службы Meta Data Services, а также архив всех выполненных операций резервного копирования БД.

Разработка и реализация плана восстановления данных

• План резервного копирования и восстановления данных требует: ▫ Описания каждого установленного экземпляра SQL Server, указав: версию ОС, установленных пакетов обновления, версию SQL Server, имена файлов журнала транзакций и данных, имена серверов, сетевые библиотеки, имя учетной записи, используемой службами; ▫ Описание БД, частоту архивирования, указание причины выбора способа архивирования; ▫ Степень автоматизации восстановления данных (расписание восстановления данных из резервных копий); ▫ Определение лица, ответственного за резервное копирование; ▫ Определение каким образом выполняется проверка качества выполнения операций восстановления данных; ▫ Выбор места для хранения носителя резервной копии данных; ▫ Определение сроков хранения носителей резервных копий данных; ▫ Документирование процессов создания копий и аппаратного обеспечения сервера.

Носители и устройства резервного копирования

При описании процессов резервного копирования и восстановления данных используются следующие термины: ◦ резервная копия - полная или частичная копия БД, журнала транзакций, файла или группы файлов; ◦ устройство резервного копирования – физический файл или ленточный накопитель, используемый для записи резервной копии ◦ файл резервной копии – файл с набором резервных копий; ◦ носитель резервной копии – физический носитель, на которой сохраняют набор резервных копий с использованием файла резервной копии; ◦ набор резервных копий – резервная копия, созданная за одну операцию резервного копирования; ◦ семейство носителей – все носители в наборе носителей, записанные одним устройством для одного набора резервных копий; ◦ заголовок носителя – предоставляет информацию о содержимом носителя; ◦ набор носителей – все носители, задействованные в операциях резервного копирования.

Выбор носителя резервной копии • Носителем резервной копии может быть:. ▫ Ленточные носители (стримеры) – традиционное средство выполнения резервного копирования. Основной недостаток – медленное выполнение операций резервного копирования. ▫ Диск – удобное средство резервного копирования, однако стоимость хранения данных больше, чем при использовании стримера.

1. **Сохранение и откат рабочих версий системы. Сохранение и восстановление баз данных**

Программа реализует следующие функции:

*Регулярное резервное копирование данных* – позволяет создать архивную копию всей базы данных (Backup). При этом могут создаваться многофайловые архивы, что определяется настройкой программы.

*Профилактика базы данных (Backup/Restore)* – включает в себя выполнение сразу двух операций: резервного копирования данных (Backup) и восстановления данных (Restore). Профилактика выполняется как для получения резервной копии базы данных, так и для очистки базы данных от временных записей, что позволяет уменьшить размер БД. На этапе восстановления могут создаваться многофайловые БД, что определяется настройкой программы. При восстановлении все БД отключаются немедленно, а также происходит перезапуск сервера БД. Восстановление по этой причине возможно только на сервере БД. Следует отметить, что функция профилактики БД имеет приоритет выполнения перед функцией регулярного резервного копирования данных. В случае, когда в расписании для этих функций указаны одинаковые параметры запуска (например, совпадает день и время запуска), то будет выполнена только профилактика БД. Это обусловлено тем, что профилактика уже включает в себя операцию резервного копирования данных (BackUp).

Перед выполнением восстановления БД (Restore) создается копия текущей БД, с расширением docpoint2.\_\_b . Копия БД необходима для быстрого отката, при возникновении нештатных ситуаций. Для восстановления работоспособности системы достаточно переименовать файл(ы) копии БД. Например, копия БД docpoint2.\_\_b переименовывается в docpoint2.gdb. При каждом запуске «Планировщика резервного копирование» проверяется, есть ли файл БД docpoint2.gdb. В случае если файл не найден (такое может произойти например в случае если во время выполнения профилактики БД произошло отключение сервера), то выдается текстовое сообщение о том что файл БД отсутствует и администратору системы предлагается восстановить базу из копии (автоматически переименовать docpoint2.\_\_b в docpoint2.gdb )

*Починка БД* – при выполнении починки БД происходит также и проверка целостности БД. При починке БД происходит отключение всех подключенных на момент проверки пользователей и включение БД при заве ршении операции. При этом починка БД происходит в два этапа, сначала БД проверяется на наличие ошибок, после чего выдается запрос о том, были ли обнаружены ошибки в БД, и нужно ли выполнять исправление БД (дело в том, что в некоторых случаях рекомендуется сделать резервную копию файла, поскольку процесс восстановления будет вносить в БД изменения). В случае же если ошибки не найдены, то проверку нужно прекращать. Сама проверка должна производиться только в случае повреждения БД. Следует четко различать ошибки повреждения БД и, например, ошибки подключения к БД по причине отсутствия соединения. Обычно ошибки в БД могут возникать по причине нестандартного отключения электропитания.

*Восстановление БД –* восстановление БД из резервной копии происходит автоматически при выполнении функции «Профилактика БД», однако, эта операция может быть сделана по запросу пользователя. Для этого необходимо в меню «Операции» выбрать пункт меню «Восстановить БД». Резервная копия БД находится в каталоге определенном в настройках программы.

*Удаление документов по указанную дату –* используется для удаления «устаревших» документов и уменьшения размера БД. Пользователь может выбрать дату и время, на которую требуется удалить документы. Удаление документов происходит от самой ранней даты изменения документа найденной в БД, до даты выбранной пользователем. По умолчанию дата устанавливается из расчета, текущая дата минус один год. Периодически необходимо выполнять ревизию документов, для того чтобы те документы которые необходимо отставить, имели относительно свежую дату их изменения. Для этого, например, можно иметь некий переход в технологической схеме, прохождение через который менял бы дату изменения документа. Информация о количестве удаленных документов сохраняется в лог файле BackUtil.log.

*Очистка BODY\_LOG*– удаление в каталоге BODY\_LOG временных файлов не связанных ни с одним документом БД.

При откате (RollBack) транзакции, например, когда пользователь отказывается от сохранения изменений в документе, работая в ПЗ «Модуль обработки документов», в каталоге каталога BODY\_LOG формируются временные файлы. Удаление таких файлов, не связанных ни с одним из документов, выполняется автоматически при выполнении функций регулярного резервного копирования и профилактике БД. Однако очистка каталога может быть выполнена по запросу пользователя. Для этого следует выбрать в меню «Операции» пункт «Очистить BODY\_LOG». Эта операция, а также период, за который будет произведена очистка, возможны при соответствующей настройке программы.

*Физическое копирование БД -* при выполнении этой операции происходит только копирование БД, без проведения процедуры backup/restore. Копирование производится в каталог для резервных копий БД.

*Возврат состояния до последней операции*– при выполнении этой операции происходит переименовывание копии базы оставшейся после последнего выполнения операции «Профилактика БД». Важно запомнить, что при выполнении этой операции текущий вариант базы будет перезаписан более старой ее копией, без возможности возврата к нему.

Если на сервере установлено несколько БД АИСТ-М, то перед выполнением операции будет выдано окно со списком всех баз зарегистрированных в системе. И только после выбора конкретной базы будет выполнена выбранная операция

Как происходит восстановление уже поврежденной БД в случае внезапного отключения электропитания? Для понимания этого сравним БД с файловой системой FAT. Для того чтобы восстановить конкретный файл в файловой системе, выполняются 2 действия:

1. Проверка файловой системы на целостность (логическую), например утилитой scandisk.

2. Проверка остался ли этот файл без изменений (например, специальной программой, работающей с файлом).

Аналогичные действия необходимо произвести над неисправной БД.

1. Проверить целостность БД. База данных, как и файловая система, построена на страницах, которые выделяются в одном пространстве по мере необходимости.

2. Выполнить одну за другой функции «Выполнить резервное копирование» (Backup) и «Выполнить восстановление БД» (Restore).

Эти операции позволят выяснить, удачно ли прошло логическое восстановление БД. Если в ходе этих операций не возникло никаких ошибок, то база данных логически цела, и может использоваться для работы. Исключением может быть «потеря» нескольких документов, по причине того, что они в момент выключения находились в памяти операционной системы и не были сохранены (или были сохранены частично) на диске. Если базу восстановить не удается, то при наличии резервной копии (если она создается регулярно) есть возможность просто восстановить данные на момент последнего резервного копирования, при этом вся работа, происходившая после создания резервной копии будет утеряна, но будут восстановлены в полном объеме данных за весь период до резервного копирования.

Для восстановления БД в случае ее поломки (повреждения), не связанной с отключением электропитания, существует другой алгоритм.

Важно запомнить, что ввиду особой сложности этой процедуры, ее может выполнять только вручную администратор системы. При этом восстановление выполняется только для одной конкретной БД. Восстановление требует остановки работы подразделения, т.к. во время операции производится остановка сервиса СУБД.

1. Перед выполнением операции будет выполнен запуск скрипта, если он прописан (по умолчанию отсутствует).

2. Если в системе зарегистрировано более одной БД, то пользователю выдается список зарегистрированных БД, из которого он должен выбрать, какую из БД он будет восстанавливать.

3. Выбор пользователем файла - архивной копии. По умолчанию путь к хранению архивной копии будет указан тот, куда производится автоматическое архивирование БД.

4. Запрос на сохранение копии текущей БД, перед восстановлением путем копирования файла. Копирование может и не производится. При выборе варианта с сохранением будет запрошен путь, куда сохранять копию БД.

5. Проверяется состояние службы «AutoServiceDocPoint», если она запущена, то выполняется ее остановка. Если сервис был запущен, то состояние сохраняется, для последующего запуска службы, после выполнения операции.

6. Проверяется состояние службы «ControlServerService», если она запущена, то выполняется ее остановка. Если сервис был запущен, то состояние сохраняется, для последующего запуска службы, после выполнения операции.

7. Останавливается сервис управления СУБД (FireBird).

8. Выполняется изменение расширения для файлов БД с gdb на \_\_b. С целью иметь копию БД в ее первоначальном виде (в случае неудачного восстановления БД), а также с целью недоступности БД по «привычному» пути. Т.е. при попытке после старта сервиса СУБД выполнить подключение к БД, пользователь получит сообщение о том, что БД не найдена. Если в процессе переименования возникли ошибки, то сервис управления СУБД (FireBird) будет запущен.

9. Выполняется старт службы СУБД (FireBird) и выполняется пауза, указанная в параметре «Задержка (в миллисекундах) между остановкой/запуском сервисов». По умолчанию, после установки системы пауза равна 2000 (2 секунды). Пауза предназначена для времени инициализации запущенного сервиса.

10. Выполняется восстановление БД. Восстановление производится не в файл с расширением gdb, а в файл с расширением \_db. С той целью, чтобы пользователи не могли подключаться к БД во время восстановления. Размер страницы БД берется из параметра «Размер страницы БД при восстановлении» настроек, по умолчанию этот параметр равен 4096.

10.1. При необходимости (в зависимости от выбора пользователя), производится копирование БД. Файл копируется из \_\_b по указанному пути в gdb.

10.2. Перед восстановлением выполняется контроль свободного места на диске. Проверка производится следующим образом если свободного места на диске плюс место, занимаемое ранее переименованным файлом БД (с расширением \_\_b) больше, чем архивная копия этой БД +10%, то места достаточно для выполнения операции. Информация о расчете свободного места пи любом исходе заносится в протокол. В случае нехватки места на диске операция завершается аварийно (запись об этом событии попадает в протокол) и происходит обратное п.5 переименование фала БД с расширением gdb.

10.3. Выполняется восстановление БД в файл с расширением \_db через Firebird API, вся информация о ходе выполнения восстановления попадает в файл протокола. Для совместимости поддерживается восстановление многофайловых архивов. Архивы БД, берутся по месту, указанному в настройках хранения архивных копий, для дополнительно зарегистрированных БД из подкаталогов DB1, DB2 и т.д.

10.4. Восстановление, в отличие от предыдущих версий данного программного продукта, производится в один файл (многофайловость не поддерживается), а поэтому производится дополнительный контроль присутствия вторичных файлов БД (с расширениями gd1m gd2 и т.д. и \_\_1, \_\_2 и т.д.), если такие файлы найдены, то они являются мусором и удаляются, о чем информация заносится в протокол.

10.5. Если восстановление завершено успешно, то файл, куда производилось восстановление БД (\_db) изменяет свое расширение на gdb. Иначе происходит переименование обратное п.5. из файла с расширением \_\_b в файл с расширением gdb (откат к предыдущей версии).

11. Если до операции сервис AutoServiceDocPoint не был запущен, то производится его запуск.

12. Если до операции сервис «ControlServerService» не был запущен, то производится его запуск.

13. После выполнения операции будет выполнен запуск соответствующего скрипта «ПОСЛЕ восстановления БД», если он прописан (по умолчанию отсутствует).

Важным элементом сохранения информации являются профилактические действия, которые следует производить для предотвращения поломок баз данных, – это постоянное создание резервных копий. Это самый надежный способ от повреждений базы данных. Только резервное копирование дает полную гарантию сохранности данных. Но в результате резервного копирования может получиться и невосстановимая копия, поэтому восстановление базы из копии никогда не должно выполняться путем записи поверх оригинала и резервное копирование должно следовать определенным правилам. Во-первых, резервное копирование должно быть как можно более частым, во-вторых, оно должно быть последовательным, и, в-третьих, резервные копии нужно проверять на возможность восстановления.

Частое резервное копирование означает, что нужно достаточно часто делать резервную копию, например один раз в сутки. Чем меньше промежуток данных между резервным копированием базы данных, тем меньше данных потеряется в результате сбоя.

1. **Защита информации**

*Защита информации* – деятельность, направленная на сохранение государственной, служебной, коммерческой или личной тайн, а также на сохранение носителей информации любого содержания.

*Система защиты информации* – комплекс организационных и технических мероприятий по защите информации, проводимых на объекте управления с применением средств и способов в соответствии с концепцией, целью и замыслом защиты.

Средства защиты информации

1. ***Технические средства*** – реализуются в виде электрических, электромеханических, электронных устройств. Всю совокупность технических средств принято делить на:

* ***аппаратные*** – устройства, встраиваемые непосредственно в аппаратуру, или устройства, которые сопрягаются с аппаратурой систем обработки данных по стандартному интерфейсу (схемы контроля информации по четности, схемы защиты полей памяти по ключу, специальные регистры);
* ***физические*** – реализуются в виде автономных устройств и систем (электронно-механическое оборудование охранной сигнализации и наблюдения и т.п.);

1. ***Программные средства*** – программы, специально предназначенные для выполнения функций, связанных с защитой информации:
2. ***Организационные средства*** – организационно-правовые мероприятия, осуществляемые в процессе создания и эксплуатации систем обработки данных для обеспечения защиты информации.
3. ***Законодательные средства*** – законодательные акты страны, которыми регламентируются правила использования и обработки информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил;
4. ***Морально-этические средства***.

Методы защиты информации

1. ***Управление доступом***, включающее следующие функции защиты:

* *идентификацию пользователя* (присвоение персонального имени, кода, пароля и опознание пользователя по предъявленному идентификатору);
* *проверку полномочий*, соответствие дня недели, времени суток, запрашиваемых ресурсов и процедур установленному регламенту;
* *разрешение и создание условий работы в пределах установленного регламента*;
* *регистрацию обращений к защищаемым ресурсам*;
* *реагирование* (задержка работ, отказ, отключение, сигнализация) при попытках несанкционированных действий.
* *криптографическое шифрование – готовое к передаче сообщение(текст, речь, графика)* зашифровывается, т.е. преобразуется в шифрограмму. Когда санкционированный пользователь получает это сообщение, он дешифрует его посредством обратного преобразования криптограммы.

**Криптография** – наука об обеспечении секретности и/или подлинности (аутентичности) передаваемых сообщений.

Процесс шифрования/дешифрования единственным образом определяется так называемым *шифрующим ключом* (это либо набор команд, либо узел аппаратуры, либо программа, либо все это вместе).

При передаче информации по линиям связи большой протяженности криптографическое закрытие является одним из самых надежных способов ее защиты.

Шифрование может быть симметричным и асимметричным.

*Симметричное* основывается на использовании одного и того же секретного ключа для шифрования и дешифрования.

*Асимметричное* характеризуется тем, что для шифрования используется один ключ, являющийся общедоступным, а для дешифрования – другой, являющийся секретным.

1. ***Механизм цифровой****(****электронной****)****подписи***, основывающийся на алгоритмах асимметричного шифрования и включающий две процедуры: формирование подписи отправителя и ее распознавание (верификацию) получателем.

Причем для шифрования используется секретный ключ отправителя, а вторая процедура основывается на использовании общедоступного ключа, знания которого достаточно для опознавания отправителя.

1. ***Механизмы контроля доступа*** осуществляют проверку полномочий объектов АИТ (программ и пользователей) на доступ к ресурсам сети.
2. ***Механизмы обеспечения целостности данных*** (например, отправитель дополняет передаваемый блок данных криптографической суммой, а получатель сравнивает ее с криптографическим значением, соответствующим принятому блоку. Несовпадение свидетельствует об искажении информации в блоке).
3. ***Механизмы управления маршрутизацией***обеспечивает выбор маршрутов движения информации по коммуникационной сети таким образом, чтобы исключить передачу секретных сведений по небезопасным физически ненадежным каналам идр***.***
4. **Организация процесса обновления в информационной системе**

Обновление – это дополнение к программному обеспечению, которое предотвращает или устраняет неполадки в нем. Помимо этого, оно также повышает безопасность, а также улучшает производительность компьютера. Обновления программного обеспечения расширяют функциональность системы и устраняют несовместимость с программным/аппаратным обеспечением.

Своевременное выполнение обновлений поможет избежать следующих негативных последствий:

развитие информационной системы под требования бизнеса только за счет собственных разработок;

увеличение количества собственных разработок приводит только к установке нот и обновлений, устраняющих ошибки и необходимых для выполнения требований изменения законодательства. Со временем установка таких обновления становится все более трудоемкой;

увеличение стоимости поддержки информационных систем;

завершение поддержки производителей устаревших баз данных и операционных систем;

снижение общего уровня безопасности системы.

Типы обновлений

* Частичное обновление
* Полное обновление
* Промежуточное
* Квартальное
* Частичное обновление

Каждое частичное обновление, выданное разработчиком, должно иметь ссылку на задачу, в которой описана проблема. Обновление ставится сначала на тестовую базу.

Проверяется работоспособность функционала, по задаче, созданной на ресурсе. После установки обновления на тестовую базу, в случае соответствия функционала, описанному в задаче, обновление устанавливается на рабочую базу и ожидает проверки конечного пользователя.

Частичное обновление выкладывается в случае возникновения критической ошибки в работе функционала, а так же при потребности передачи функционала до установки полного обновления.

Полное обновление

В зависимости от потребности может собираться как полное квартальное обновление, так и полное промежуточное. Промежуточное Обновление выкладывается в отдельную задачу для установки на тестовую базу (содержит описание и список задач по фильтру). В первый рабочий день обновление устанавливается на тестовую базу данных. Срок тестирования 2 дня. В случае обнаружения критических ошибок срок установке сдвигается до их исправления.

Квартальное

Обновление выкладывается в отдельную задачу для установки на тестовую базу. В течение 3х дней обновление должно быть установлено на тестовую базу. Тестовую базу перед установкой обновления следует актуализировать до состояния рабочей базы данных на текущий момент. Со дня установки релиза начинается тестирование согласно Приложению 1 к регламенту Тестирования и установки обновлений (срок тестирования 15 рабочих дней)

Простое обновление

Процесс обновления систем заключается в последовательном выполнении следующих этапов:

Обследование. На этом этапе определяется уровень обновлений системы, подключение дополнительных функциональных возможностей. Проводится сбор информации об объеме внедрения бизнес-процессов, операций по ним и определяется объем тестирования. Также осуществляется анализ объема, модифицированного ПО и собственных разработок.

Подготовка плана перехода. Производится подготовка тестовой системы (копия продуктивной), ее обновление, анализ и корректировка затронутого модифицированного ПО; тестирование работы системы, регистрация и решение проблем; создание перечня мероприятий для перехода. Затем проводится повторное разворачивание тестовой системы, ее обновление, тестирование и применение плана мероприятий.

Производится планирование сроков этапов перехода, оценка рисков и возможность дополнительных мероприятий по их снижению. В итоге определяется период неработоспособности продуктивной системы, разрабатывается и утверждается документ «План перехода» Выполнение плана перехода. Заключается в последовательном выполнении мероприятий, описанных в документе "План перехода".

Поддержка пользователей. После переноса обновлений в продуктивную систему заказчика осуществляется оперативная поддержка пользователей и решение оставшихся проблем. Миграция Процесс миграции систем включает в себя следующие этапы:

Обследование. На этом этапе определяется перечень мероприятий.

Подготовка плана миграции. Производится применение перечня мероприятий, подготовка стенда, проверяется работоспособность системы.

Затем проводится тестовая миграция, в ходе которой уточняется, обновляется перечень мероприятий, а также определяется их длительность. На основе результатов этого процесса производится планирование сроков этапов миграции, оценка рисков и возможность дополнительных мероприятий по их снижению. В итоге определяется период неработоспособности продуктивной системы, разрабатывается и утверждается документ «План перехода» Выполнение плана миграции. Заключается в последовательном выполнении мероприятий, описанных в документе.

Поддержка пользователей. После миграции осуществляется оперативная поддержка пользователей и решение оставшихся проблем.

1. **Регламенты обновления**

**Регламенты обновления** — это документы или набор правил, определяющих, как и когда следует проводить обновления программного обеспечения, систем или процессов. Их цель — обеспечить стабильность и безопасность работы системы, минимизируя риски и потери данных. Основные моменты, которые обычно описываются в регламентах обновления:

1. **Типы обновлений**:
   * **Плановые** — заранее запланированные обновления, проводимые регулярно.
   * **Необходимые** — обновления, связанные с исправлением критических ошибок
   * **Экстренные** — обновления, требующие немедленного применения из-за угроз безопасности.
2. **Процесс обновления**:
   * **Проверка совместимости** — анализ совместимости обновлений с текущими версиями программного обеспечения.
   * **Тестирование** — выполнение тестов на новой версии, чтобы исключить ошибки.
   * **Резервное копирование** — перед обновлением создается резервная копия данных и настроек, чтобы восстановить систему в случае сбоя.
   * **Инсталляция и настройка** — обновление непосредственно системы или программы.
3. **Ответственные лица** — в регламенте указывается, кто отвечает за подготовку, выполнение и проверку обновлений (например, администраторы системы, службы технической поддержки).
4. **Периодичность обновлений** — могут быть указаны временные рамки для плановых обновлений (ежемесячные, ежеквартальные и т. д.).
5. **Оповещения и уведомления** — регламент может предусматривать уведомление пользователей о предстоящих обновлениях, чтобы они были готовы к временным сбоям.
6. **Риски и безопасность** — описываются меры по снижению рисков, таких как падеж
7. **Обратная совместимость** — если обновление затрагивает важные функции, может быть предусмотрено тестирование на совместимость с устаревшими версиями.

Регламенты обновления помогают поддерживать актуальность системы, минимизировать технические сбои и гарантировать, что обновления не нарушат работу пользователей.

1. **Требования к архитектуре информационной системы для обеспечения безопасности ее функционирования**

В "Оранжевой книге" надежная информационная система определяется как "система, использующая достаточные аппаратные и программные средства, чтобы обеспечить одновременную достоверную обработку информации разной степени секретности различными пользователями или группами пользователей без нарушения прав доступа, целостности и конфиденциальности данных и информации, и поддерживающая свою работоспособность в условиях воздействия на нее совокупности внешних и внутренних угроз".

Это качественное определение содержит необходимое достаточное условие безопасности. При этом не обусловливается, какие механизмы и каким образом реализуют безопасность - практическая реализация зависит от многих факторов: вида и размера бизнеса, предметной области деятельности компании, типа информационной системы, степени ее распределенности и сложности, топологии сетей, используемого программного обеспечения и т. д.

Идеология открытых систем (см. гл. 19) существенно отразилась на методологических аспектах и направлении развития сложных ИС. Она базируется на строгом соблюдении совокупности профилей, протоколов и стандартов де-факто и де-юре. Программные и аппаратные компоненты по этой идеологии должны отвечать важнейшим требованиям: переносимости и возможности согласованной, совместной работы с другими удаленными компонентами. Это позволяет обеспечить совместимость компонент различных информационных систем, а также средств передачи данных. Задача сводится к максимально возможному повторному использованию разработанных и апробированных программных и информационных компонент при изменении вычислительных аппаратных платформ, операционных систем и процессов взаимодействия.

С практической точки зрения обеспечения безопасности наиболее важными являются следующие принципы построения архитектуры ИС:

* проектирование ИС на принципах "открытых систем", следование признанным стандартам, использование апробированных решений, иерархическая организация ИС с небольшим числом сущностей на каждом уровне - все это способствует "прозрачности" и хорошей управляемости ИС;
* непрерывность защиты в пространстве и времени, невозможность миновать защитные средства, невозможность спонтанного или вызванного перехода в небезопасное состояние - при любых обстоятельствах, в том числе нештатных, защитное средство либо полностью выполняет свои функции, либо полностью блокирует доступ в систему или ее часть;
* усиление самого слабого звена, минимизация привилегий доступа, разделение функций обслуживающих сервисов и обязанностей персонала. Предполагается такое распределение ролей и ответственности, чтобы один человек не мог нарушить критически важный для организации процесс или создать брешь в защите по неведению или по заказу злоумышленников. Применительно к программно-техническому уровню принцип минимизации привилегий предписывает выделять пользователям и администраторам только те права доступа, которые необходимы им для выполнения служебных обязанностей. Это позволяет уменьшить ущерб от случайных или умышленных некорректных действий пользователей и администраторов;
* эшелонирование обороны, разнообразие защитных средств, простота и управляемость информационной системы и системы ее безопасности. Принцип эшелонирования обороны предписывает не полагаться на один защитный рубеж, каким бы надежным он ни казался. За средствами физической защиты должны следовать программно-технические средства, за идентификацией и аутентификацией - управление доступом, протоколирование и аудит. Эшелонированная оборона способна не только не пропустить злоумышленника, но и в в некоторых случаях идентифицировать его, благодаря протоколированию и аудиту. Принцип разнообразия защитных средств предполагает создание различных по своему характеру оборонительных рубежей, чтобы от потенциального злоумышленника требовалось овладение разнообразными и, по возможности, несовместимыми между собой навыками.

За средствами физической защиты должны следовать программно-технические средства, за идентификацией и аутентификацией - управление доступом, протоколирование и аудит. Эшелонированная оборона способна не только не пропустить злоумышленника, но и в в некоторых случаях идентифицировать его, благодаря протоколированию и аудиту.

 При строгом соблюдении правил структурного построения значительно облегчается достижение высоких показателей качества и безопасности, так как сокращается число возможных ошибок в реализующих программах, отказов и сбоев оборудования, упрощается их диагностика и локализация. В хорошо структурированной системе с четко выделенными компонентами (клиент, сервер приложений, ресурсный сервер) контрольные точки выделяются достаточно четко, что решает задачу доказательства достаточности применяемых средств защиты и обеспечения невозможности обхода этих средств потенциальным нарушителем.

Концепция информационной безопасности определяет этапы построения системы информационной безопасности в соответствии со стандартизованным жизненным циклом ИС: аудит безопасности (обследование) существующей системы защиты ИС, анализ рисков, формирование требований и выработка первоочередных мер защиты, проектирование, внедрение и аттестация, сопровождение системы.

Аудит безопасности. Аудит безопасности может включать по крайней мере четыре различных группы работ.

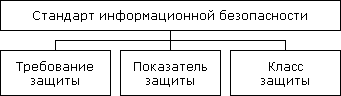
К первой группе относятся так называемые "тестовые взломы" ИС. Эти тесты применяются, как правило, на начальных стадиях обследования защищенности ИС. Причина малой эффективности "тестовых взломов" скрывается в самой постановке задачи. Действительно, основной задачей "взломщика" является обнаружение нескольких уязвимостей и их использование для доступа в систему. Если тест оказался успешным, то, предотвратив потенциальное развитие возможных сценариев "взлома", работу надо начинать сначала и искать следующие. Неуспех "взлома" может означать в равной мере как защищенность системы, так и недостаточность тестов.

Вторая группа - экспресс-обследование. В рамках этой, обычно непродолжительной, работы оценивается общее состояние механизмов безопасности в обследуемой ИС на основе стандартизованных проверок. Экспресс-обследование обычно проводится в случае, когда необходимо определить приоритетные направления, позволяющие обеспечить минимальный уровень защиты информационных ресурсов. Основу для него составляют списки контрольных вопросов, заполняемые как в результате интервьюирования, так и в результате тестовой работы автоматизированных сканеров защищенности.

Третья группа работ по аудиту - аттестация систем на соответствие требованиям защищенности информационных ресурсов. При этом происходит формальная проверка набора требований как организационного, так и технического аспектов, рассматриваются полнота и достаточность реализации механизмов безопасности. Типовая методика анализа защищенности корпоративной информационной системы состоит в совокупности следующих методов:

Четвертая группа - предпроектное обследование - самый трудоемкий вариант аудита. Такой аудит предполагает анализ организационной структуры предприятия в приложении к информационным ресурсам, правила доступа сотрудников к тем или иным приложениям. Затем выполняется анализ самих приложений. После этого должны учитываться конкретные службы доступа с одного уровня на другой, а также службы, необходимые для информационного обмена. Затем картина дополняется встроенными механизмами безопасности, что в сочетании с оценками потерь в случае нарушения ИБ дает основания для ранжирования рисков, существующих в ИС, и выработки адекватных контрмер. Успешное проведение предпроектного обследования, последующего анализа рисков и формирования требований определяют, насколько принятые меры будут адекватны угрозам, эффективны и экономически оправданы.

1. **Стандартизация подходов к обеспечению информационной безопасности**



Стандарт информационной безопасности – нормативный документ, определяющий порядок и правила взаимодействия субъектов информационных отношений, а также требования к инфраструктуре информационной системы, обеспечивающие необходимый уровень информационной безопасности.

Согласно указу Президента N 1085, ФСТЭК России является федеральным органом исполнительной власти, уполномоченным в области противодействия техническим разведкам и технической защиты информации, а также специально уполномоченным органом в области экспортного контроля.

В Российской Федерации информационная безопасность обеспечивается соблюдением указов Президента, федеральных законов, постановлений Правительства Российской Федерации, руководящих документов Федеральной службы по техническому и экспортному контролю (до 16 августа 2004 года ФСТЭК носила название - Государственная техническая комиссия при Президенте РФ, а в августе, в рамках административной реформы, комиссия была переименована в Федеральную службу и подчинена Министерству обороны) и других нормативных документов.

За 10 лет своего существования ФСТЭК разработала и довела до уровня национальных стандартов десятки документов, среди которых: Руководящий документ "Положение по аттестации объектов информатизации по требованиям безопасности информации" (Утверждено Председателем ФСТЭК России 25.11.1994 г.). Руководящий документ "Автоматизированные системы (АС). Защита от несанкционированного доступа (НСД) к информации. Классификация АС и требования к защите информации" (ФСТЭК России, 1997 г.). Руководящий документ "Средства вычислительной техники. Защита от НСД к информации. Показатели защищенности от НСД к информации" (ФСТЭК России, 1992 г.). Руководящий документ "Концепция защиты средств вычислительной техники от НСД к информации" (ФСТЭК России, 1992 г.). Руководящий документ "Защита от НСД к информации. Термины и определения" (ФСТЭК России, 1992 г.). Руководящий документ "Средства вычислительной техники (СВТ). Межсетевые экраны. Защита от НСД к информации. Показатели защищенности от НСД к информации" (ФСТЭК России, 1997 г.). Руководящий документ "Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей" (ФСТЭК России, 1999 г.). Руководящий документ "Специальные требования и рекомендации по технической защите конфиденциальной информации" (ФСТЭК России, 2001 г.). Руководящий документ "СВТ. Защита от НСД к информации. Показатели защищенности от НСД к информации" устанавливает классификацию СВТ по уровню защищенности от НСД к информации на базе перечня показателей защищенности и совокупности описывающих их требований. Основой для разработки этого документа явилась "Оранжевая книга". Этот оценочный стандарт устанавливается семь классов защищенности СВТ от НСД к информации. Самый низкий класс – седьмой, самый высокий – первый. Классы подразделяются на четыре группы, отличающиеся уровнем защиты: I. первая группа содержит только один седьмой класс, к которому относят все СВТ, не удовлетворяющие требованиям более высоких классов; II. вторая группа характеризуется дискреционной защитой и содержит шестой и пятый классы; III. третья группа характеризуется мандатной защитой и содержит четвертый, третий и второй классы; IV. четвертая группа характеризуется верифицированной защитой и включает только первый класс. Руководящий документ "АС. Защита от НСД к информации. Классификация АС и требования по защите информации" устанавливает классификацию автоматизированных систем, подлежащих защите от несанкционированного доступа к информации, и требования по защите информации в АС различных классов. К числу определяющих признаков, по которым производится группировка АС в различные классы, относятся: I. наличие в АС информации различного уровня конфиденциальности; II. уровень полномочий субъектов доступа АС на доступ к конфиденциальной информации; III. режим обработки данных в АС – коллективный или индивидуальный. В документе определены девять классов защищенности АС от НСД к информации. Каждый класс характеризуется определенной минимальной совокупностью требований по защите. Классы подразделяются на три группы, отличающиеся особенностями обработки информации в АС. В пределах каждой группы соблюдается иерархия требований по защите в зависимости от ценности и конфиденциальности информации и, следовательно, иерархия классов защищенности АС. Руководящий документ "СВТ. Межсетевые экраны. Защита от НСД к информации. Показатели защищенности от НСД к информации" является основным документом для анализа системы защиты внешнего периметра корпоративной сети. Данный документ определяет показатели защищенности межсетевых экранов (МЭ). Каждый показатель защищенности представляет собой набор требований безопасности, характеризующих определенную область функционирования МЭ. Всего выделяется пять показателей защищенности: управление доступом; идентификация и аутентификация; регистрация событий и оповещение; контроль целостности; восстановление работоспособности. На основании показателей защищенности определяются следующие пять классов защищенности МЭ: простейшие фильтрующие маршрутизаторы – 5 класс; пакетные фильтры сетевого уровня – 4 класс; простейшие МЭ прикладного уровня – 3 класс; мЭ базового уровня – 2 класс; продвинутые МЭ – 1 класс.

1. **Организация доступа пользователей к информационной системе. Регистрация пользователей**

Принципы создания информационных систем обеспечение общения конечного пользователя (исследователя, проектировщика, плановика) с системой автоматизации на профессионально-ограниченном естественном языке, представление входной и результирующей информации в привычной и удобной пользователю форме;

обеспечение возможности решения задач планирования, управления, проектирования, подготовки производства и научных исследований по их постановкам и исходным данным независимо от сложности и наличия формальных математических моделей этих задач;

создание конечному пользователю таких условий работы, при которых он осуществляет процессы управления, планирования, проектирования и поиска новых решений в режиме активного, расширяющегося диалога с ИС, оперируя понятиями своей предметной области, используя профессиональный опыт и навыки и принимая решения одновременно по множеству критериев.

– интеллектуальные диалоговые (вопросноответные);

– расчетно-логические, или системы поддержки принятия решений (для решений различных задач проектирования, организационного управления);

– экспертные системы.

Каждую информационную систему оценивают двумя критериями: точностью и удобством.

Понятие точности и удобства точность означает, что при поступлении на вход системы заданных значений на ее выходе получаются ожидаемые результаты.

Критерий удобства означает, что при работе с информационной системой пользователь не должен существенно менять стиль своей работы.

Обеспечивается логикой работы ИС Обеспечивается интерфейсом ИС – Функции интерфейса ИС

• организация диалога, обеспечивающего интеграцию профессиональных потребностей и интересов пользователей с ресурсами информационных систем

• обеспечивает связь между пользователем и процессом, выполняющим некоторое задание. Это дает возможность определять, какие процессы сделать активными в данный момент, и как передавать им данные для обработки и принимать результаты обработки.

• через интерфейс осуществляется ввод/вывод данных, в том числе и для управления работой ИС.

Интерфейс – как средство диалога Процесс диалога – это механизм обмена информацией, который можно рассматривать как оболочку, включающую все входящие в систему процессы по выполнению определенных заданий. Процессы ввода-вывода обеспечивают обмен на самом верхнем уровне; на этом уровне диалоговый процесс должен правильно интерпретировать каждое слово и звук.

Задачи диалогового процесса:

– определение задания, которое пользователь возлагает на систему;

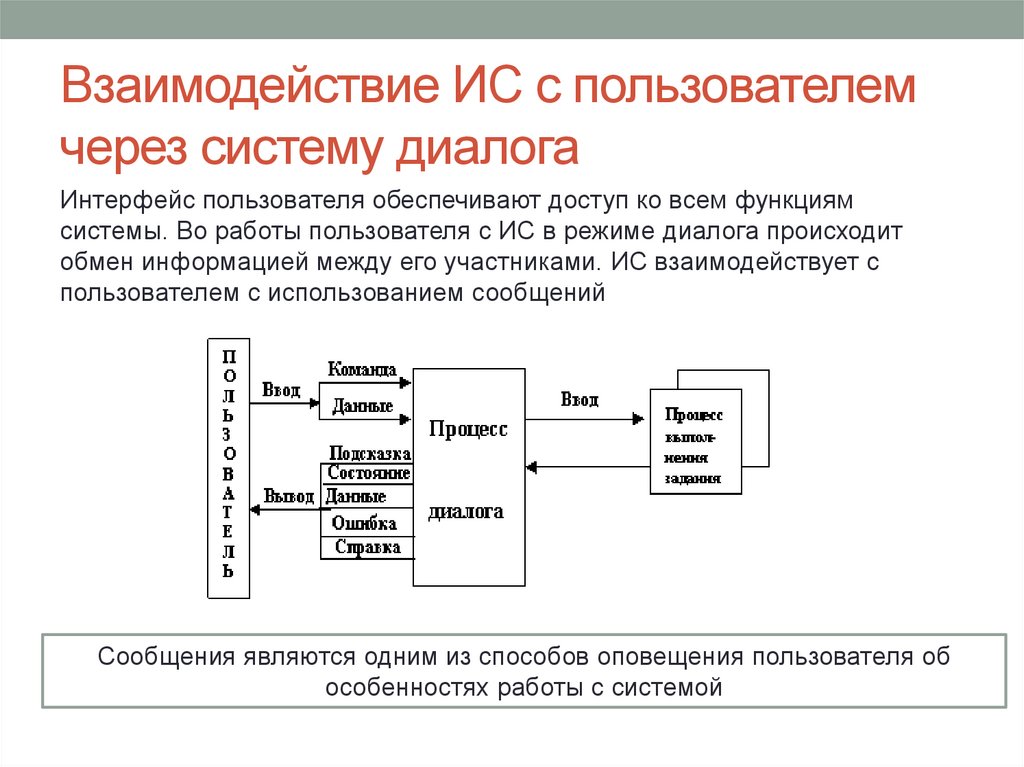
– прием логически связанных входных данных и размещение их в переменных соответствующего процесса и в нужном формате;

– вызов процесса выполнения требуемого задания;

– вывод результатов обработки по окончании процесса в подходящем для пользователя формате.

Взаимодействие ИС с пользователем через систему диалога

Интерфейс пользователя обеспечивают доступ ко всем функциям системы. Во работы пользователя с ИС в режиме диалога происходит обмен информацией между его участниками. ИС взаимодействует с пользователем с использованием сообщений Сообщения являются одним из способов оповещения пользователя об особенностях работы с системой



Виды сообщений в информационной системе

• Подсказка – это выходное сообщение системы, побуждающее пользователя вводить данные.

• Команда – входное управляющее сообщение, предназначенное для управления ходом диалога.

• Входные данные – данные, необходимые для выполнения процесса.

• Сообщение об ошибке – это сигнал диалогового процесса о том, что невозможно дальнейшее выполнение работы, потому что вызванный процесс выполнения задания не может обработать введенное сообщение.

• Выходные данные – это данные, которые возвращает процесс по окончании обработки задания. Процесс выполнения задания передает выходные данные в стандартной форме на вход диалогового процесса, который преобразует их в подходящий для пользователя формат.

• Сообщение о состоянии системы – это информация для пользователя о том, что произошло или происходит в системе.

• Справочная информация – это информация, поясняющая, как должен поступить пользователь в той или иной ситуации.

Интеллектуальный интерфейс интерфейс пользователя, дополнительно снабженный программным обеспечением, способным выполнять элементарные функции анализа, синтеза, сравнения, обобщения, накопления, обучения всех составных элементов, участвующих в процессе взаимодействия с пользователем, делая обычный интерфейс пользователя разумным, т.е. интеллектуальным.



ДРУЖЕСТВЕННОСТЬ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Обеспечение общения пользователя с системой на языке, близком к естественному. Данная возможность позволяет вести взаимодействие с компьютером при минимальной подготовке пользователя, без необходимости предварительного обращения к инструкциям и запоминания различных правил построения высказываний, на языке, близком к языку профессиональной деятельности пользователя.

Однозначность понимания. Возможность неоднозначного толкования заставляет пользователя тратить время на размышления о том, какие именно действия системы последуют в том или ином случае. Сообщения системы должны быть точны, корректны и недвусмысленны. Этот принцип можно рассматривать как лингвистический аспект дружественности.

Деликатность. Поскольку многие пользователи осваивают систему методом проб и ошибок, наличие в ее интерфейсе хороших возможностей по обработке и диагностике ошибок, механизмов отката и средств снижения их вероятности (например, путем предоставления пользователю понятных вариантов выбора) позволяют создать более дружественный пользователю интерфейс. Помимо этого, необходимо предусмотреть возможность выхода (отказа) для любой опции системы.

Привлекательность и визуальная ясность. Эстетический аспект имеет большое значение для создания положительного эмоционального настроя в работе. Поэтому необходимо при разработке интерфейса обращать внимание и на такие вещи, как группирование и пропорциональность объектов, использование цветового и яркостного контраста, трехмерное представление, размер текста, зазоры между кнопками и др. Принцип привлекательности отвечает визуальному аспекту дружественности.

Концентрация внимания. Элементы интерфейса, призванные повысить привлекательность интерфейса, не должны отвлекать внимание пользователя от главного - той работы, которую ему необходимо выполнить. Помимо этого, интерфейс должен помогать пользователю сконцентрироваться именно на том действии, которое он намерен выполнить в данный момент.

Простота чтения. В это понятие включаются не только выбор шрифтов, но и стиль изложения материала, независимость от разрешения и типа монитора, привычное (общепринятое) расположение стандартных объектов в окне диалога.

Предвосхищение событий. В интерфейс могут включаться средства запоминания предыдущих действий пользователя и логические механизмы обработки стандартных последовательностей действий, что позволяет системе спрогнозировать следующий шаг пользователя. Помимо этого, интерфейс должен информировать пользователя о текущем состоянии, обеспечивать функцию отклика для указания на ошибки, отображения степени выполнения процесса и др., что соответствует функциональному аспекту свойства дружественности.

Снижение нагрузки на память. Для пользователя не просто запоминать сложные последовательности опций и команд, поэтому все действия в системе должны быть вынесены в ее меню, через которое пользователь будет работать в режиме распознавания команд, а не их вспоминания.

Использование вычислительных способностей ЭВМ. Пользователь должен быть освобожден от выполнения действий, которые могут быть выполнены ЭВМ, например, вычисления дня недели по заданной дате.

1. **Стандарт управления правами доступа к информационным системам**

Существует два направления контроля и управления доступом в ИС: физическое и логическое. Физическое управление доступом применяется к техническим и аппаратным средствам ИС, а также к информации, представленной в печатной, визуальной и аудиоформе. Логическое управление доступом – к программным средствам и информации, представленной в электронной форме. Оно реализуется программными средствами. Логическое управление доступом – это основной механизм многопользовательских систем, призванный обеспечить конфиденциальность и целостность объектов и, в некоторой степени, их доступность (путем запрещения обслуживания неавторизованных пользователей).

В основе управления доступом лежит идентификация и аутентификация. Если субъект и СИБ территориально разнесены, то с точки зрения безопасности необходимо рассмотреть два аспекта: что служит аутентификатором; как организован (и защищен) обмен данными идентификации и аутентификации.

Имеется совокупность субъектов и набор объектов. Задача логического управления доступом состоит в том, чтобы для каждой пары "субъект-объект" определить множество допустимых операций (зависящее, быть может, от некоторых дополнительных условий) и контролировать выполнение установленного порядка. Отношение "субъекты-объекты" можно представить в виде матрицы доступа, в строках которой перечислены субъекты, в столбцах – объекты, а в клетках, расположенных на пересечении строк и столбцов, записаны дополнительные условия (например, время и место действия) и разрешенные виды доступа.

Существуют следующие концептуальные механизмы ИБ: Идентификация и аутентификация; Контроль и управление доступом; Протоколирование и аудит; Шифрование: Контроль целостности; Экранирование.

Для надежной ЗИ необходима комплексная реализация всех перечисленных механизмов. Некоторые из них могут быть реализованы в более полной мере, другие – нет. Защита ИС в первую очередь зависит от реализации механизма идентификации и аутентификации. Идентификатор –уникальный набор символов, однозначно соответствующий объекту или субъекту в данной системе. Идентификация – распознавание участника процесса информационного взаимодействия (ИВ) перед тем, как к нему будут применены какие-либо аспекты ИБ. Пароль – секретный набор символов, позволяющий подтвердить соответствие субъекта предъявленному им идентификатору. Аутентификация – обеспечение уверенности в том, что участник ИВ идентифицирован верно. Профиль – набор установок и конфигураций для данного субъекта или объекта и определяющий его работу в ИС. Авторизация – формирование профиля прав для конкретного участника ИВ.

1. **Организация сбора данных об ошибках в информационных системах**

Ошибки сбора данных

Ошибки, возникающие при сборе информации, можно разделить на два основных типа: ошибки в выборке (выборочные) и систематические ошибки (вневыборочные).

Ошибки в выборке - разность между наблюдаемыми значениями количественного признака и средним значением генеральной совокупности. Ошибка в выборке может быть уменьшена путем увеличения объема выборки; при этом возрастает концентрация распределения выборочной статистики возле долгосрочного среднего значения, а выбо­рочная статистика выравнивается.

Систематические ошибки могут возникать и при сплошных исследованиях. Они могут являться следствием концептуальных или логических ошибок, неправильной ин­терпретации вопросов и ответов, а также статистических, арифметических, табуляционных, кодовых или отчетных ошибок.

Вневыборочная ошибка включает в свой состав:

1. Все типы ошибок, обусловленные тем, что не все респонденты дали ответы.
2. Ошибки сбора данных.
3. Ошибки обращения с полученными данными.
4. Ошибки анализа сбора данных.
5. Ошибки интерпретации полученных результатов.

Вневыборочные ошибки можно классифицировать на ошибки лиц, осуществляющих сбор данных, и респондентов.

Вневыборочные ошибки делятся на:

1. Преднамеренные. Ошибки лица, собирающего информацию, когда оно сознательно нарушает установленные исследователем требования к сбору данных. Такое нарушение может носить характер обмана и стремление склонить респондента к определенному ответу путем использования особых слов, интонаций, мимики, жестов, подсказок.
2. Непреднамеренные. Ошибки лица, собирающего информацию. Главным образом определяется неправильным пониманием со стороны опрашиваемого отдельных аспектов сбора данных, изложенных в различных инструкциях, хотя ему и кажется, что он все делает правильно. Причиной непреднамеренной ошибки могут быть усталость лица, собирающего информацию, особенно, когда в течении рабочего дня было опрошено достаточно большое количество респондентов. По этой причине может быть ослаблен контроль заполнения анкет, приглашение принять участие в опросе произносится усталым, раздражительным голосом, в результате чего, потенциальный респондент откажется принять участие в опросе.

Существует 2 вида преднамеренных ошибок респондента:

1. Обусловлен стремлением респондента фальсифицировать свои ответы в следствии определенного замешательства, нежелания отвечать на персональные вопросы из-за подозрения, что интервьюер преследует какие-то свои цели.
2. Обусловлен отказом респондента отвечать на вопросы из-за своей занятости, нежелания открывать личные аспекты своей жизни, предубежденности к опросу.

Непреднамеренная ошибка респондента возникает, когда респондент, думая, что говорит правду дает ошибочный ответ. Это обусловлено плохим пониманием вопросов и/или инструкции по заполнению анкет, использования предположений вместо точных знаний.

1. **Источники сведений в информационной системе**

# **Информационные ресурсы корпоративных информационных систем**

1. Источники информации в информационной системе. Информационные модели объекта управления. Информационные массивы и потоки.

2. Информационное обеспечение корпоративных информационных систем.

3. Информационные ресурсы. Роль информационных ресурсов в управлении экономикой.

Можно классифицировать всю бизнес-информацию двумя способами. Классификация первым способом предполагает, что информация может быть либо первичной, полученной в результате исследования или анализа, инициированного фирмой, либо вторичной, полученной из каких-то уже имеющихся источников. Второй способ подразделяет информацию на внутреннюю и внешнюю по отношению к фирме. Вторичные источники можно классифицирует по характеру доступа к ним. Эти источники могут быть:

публичными, доступными для любого исследователя;

частными, составляющими собственность определенной компании или института (но с возможностью доступа к ним за плату);

подписными, представляющими собой гибрид публичных и частных источников, когда информация находится в чьей-то собственности, но постоянно обновляется и становится доступной для определенного ограниченного круга подписчиков.

При решении задач в различных областях деятельности приходится строить различные модели. В информатике рассматриваются в основном *информационные* и *математические модели***.** Рассмотрим примеры информационных моделей.

*Информационная модель личности.*На каждого сотрудника какого-либо предприятия или учреждения в отделе кадров заводится личное дело, в котором, в частности, находится личный листок по учету кадров. В этом документе отражаются такие атрибуты сотрудников, как фамилия, имя и отчество, дата рождения, пол, образование, домашний адрес и т. д. А такие атрибуты, как цвет глаз, рост, вес, в личном листке никак не отражаются. Можно считать, что этот документ представляет собой информационную модель личности сотрудника учреждения. В этой модели отражены только значимые для отдела кадров атрибуты сотрудников. Если же рассмотреть ситуацию, когда создается информационная модель личности преступника, разыскиваемого органами правопорядка, то незначимые ранее атрибуты - цвет глаз, рост и вес — теперь могут стать существенными, а, скажем, образование и точная дата рождения — несущественными.

*Информационная модель печатного издания.*В библиотеке на каждое печатное издание заводится библиографическая карточка. В ней отражаются инвентарный и каталожные номера, название, фамилия автора или авторов, год и место издания, том, номер и т. д. Все это вместе взятое является информационной моделью печатного издания. Наличие или отсутствие суперобложки, формат (размер) издания, качество бумаги в этой модели — несущественные атрибуты, и поэтому они отбрасываются. С другой стороны, при определении цены издания для его реализации через торговую сеть эти атрибуты становятся важными, и нужно строить другую модель печатного издания.

*Информационные потоки*— это физическое перемещение информации от одного сотрудника предприятия к другому или от одного подразделения к другому. Система информационных потоков — совокупность физических перемещений информации, дающая возможность осуществить какой-либо процесс, реализовать какое-либо решение. Наиболее общая система информационных потоков — это сумма потоков информации, которая позволяет вести предприятию финансово-хозяйственную деятельность.

Информация в информационных потоках должна отвечать следующим требованиям:

своевременность, т.е. информация по затратам, выручке, прибыли должна поступать тогда, когда еще имеет смысл ее анализировать;

достоверность;

релевантность, т.е. информация должна помогать принимать решения;

полезность (эффект от использования информации должен перекрывать затраты на ее получение);

полнота, т.е. не должно быть упущений;

понятность, т.е. информация не должна требовать «расшифровки»;

регулярность поступления.

Информационный *массив*- совокупность зафиксированной информации, предназначенная для хранения и использования и рассматриваемая как единое целое. Информация может быть зафиксирована в виде публикаций, отчетов, электронных записей, микрокопий и т.д. Обычно на предприятиях и в учреждениях информационные массивы формируются по функциональному признаку.

*Информационное обеспечение*– совокупность единой системы классификации и кодирования информации, унифицированных систем документации, схем информационных потоков, циркулирующих в организации, а также методология построения баз данных.

Информационное обеспечение ИС является средством для решения следующих задач:

однозначного и экономичного представления информации в системе (на основе кодирования объектов);

организации процедур анализа и обработки информации с учетом характера связей между объектами (на основе классификации объектов);

организации взаимодействия пользователей с системой (на основе экранных форм ввода-вывода данных);

обеспечения эффективного использования информации в контуре управления деятельностью объекта автоматизации (на основе унифицированной системы документации).

Информационное обеспечение ИС включает два комплекса: внемашинное информационное обеспечение (классификаторы технико-экономической информации, документы, методические инструктивные материалы) и внутримашинное информационное обеспечение (макеты/экранные формы для ввода первичных данных в ЭВМ или вывода результатной информации, структуры информационной базы: входных, выходных файлов, базы данных). Под электронными формами документов понимается не изображение бумажного документа, а изначально электронная (безбумажная) технология работы; она предполагает появление бумажной формы только в качестве твердой копии документа.

К недостаткам электронных документов можно отнести неполную юридическую проработку процесса их утверждения или подписания. Технология обработки электронных документов требует использования специализированного программного обеспечения — программ управления документооборотом, которые зачастую встраиваются в корпоративные ИС.

К информационному обеспечению предъявляются следующие общие требования:

информационное обеспечение должно быть достаточным для поддержания всех автоматизируемых функций объекта;

для кодирования информации должны использоваться принятые у заказчика классификаторы;

для кодирования входной и выходной информации, которая используется на высшем уровне управления, должны быть использованы классификаторы этого уровня;

должна быть обеспечена совместимость с информационным обеспечением систем, взаимодействующих с разрабатываемой системой;

формы документов должны отвечать требованиям корпоративных стандартов заказчика (или унифицированной системы документации);

структура документов и экранных форм должна соответствовать характеристикам терминалов на рабочих местах конечных пользователей;

графики формирования и содержание информационных сообщений, а также используемые аббревиатуры должны быть общеприняты в этой предметной области и согласованы с заказчиком;

в ИС должны быть предусмотрены средства контроля входной и результатной информации, обновления данных в информационных массивах, контроля целостности информационной базы, защиты от несанкционированного доступа.

Информационные ресурсы – это документы и массивы документов в информационных системах (библиотеках, архивах, фондах, банках данных, депозитариях, музейных храненьях и др.).

Разработаны стандарты, где устанавливаются требования:

к унифицированным системам документации;

к унифицированным формам документов различных уровней управления;

к составу и структуре реквизитов и показателей;

к порядку внедрения, ведения и регистрации унифицированных форм документов.

Информационные ресурсы, выступающие на рынке как товар в виде информационной продукции, могут быть объектами товарных отношений.

Информационные ресурсы могут иметь государственное значение или относиться к категории, имеющей значение только для юридических и физических лиц.

Активными информационными ресурсами является та часть национальных ресурсов, которую составляет информация, доступная для автоматизированного поиска, хранения и обработки: формализованные и законсервированные на машинных носителях в виде работающих программ профессиональные знания и навыки, текстовые и графические документы.

1. **Системы управления производительностью приложений**

**Системы управления производительностью приложений (APM)** — это инструменты, которые помогают мониторить, анализировать и улучшать производительность приложений в реальном времени. Их цель — быстро выявлять и устранять проблемы, которые могут замедлять работу или вызывать сбои в приложениях.

Основные функции APM:

1. **Мониторинг производительности** — отслеживание работы приложения, включая скорость отклика, время загрузки и использование ресурсов.
2. **Анализ проблем** — выявление узких мест, таких как медленные запросы, ошибки или сбои, которые могут негативно повлиять на работу приложения.
3. **Трассировка запросов** — отслеживание путей запросов через различные компоненты системы (например, серверы, базы данных), чтобы понять, где происходит замедление.
4. **Оптимизация** — рекомендации по улучшению производительности и устранению проблем, таких как утечки памяти или неэффективные алгоритмы.
5. **Аналитика и отчеты** — создание отчетов о производительности для оценки эффективности и планирования улучшений.

Системы APM важны для поддержания стабильности приложений, их быстродействия и удобства для пользователей.

1. **Средства управления приложениями**

**Средства управления приложениями (Application Management Tools)** — это набор программных инструментов и технологий, используемых для контроля, поддержки, обслуживания и оптимизации приложений на протяжении всего их жизненного цикла. Эти средства помогают обеспечить стабильную работу приложений, минимизировать сбои и ускорить их разработку.

Основные функции средств управления приложениями:

1. **Установка и развертывание**:
   * Автоматизация процессов установки и обновлений приложений на серверах или устройствах пользователей.
2. **Мониторинг**:
   * Постоянный мониторинг состояния и производительности приложений, включая использование ресурсов (ЦП, память, дисковое пространство) и время отклика.
3. **Производительность**:
   * Анализ и оптимизация работы приложений для предотвращения замедлений, ошибок и сбоев. Включает инструменты для отслеживания и повышения производительности.
4. **Управление инцидентами**:
   * Обнаружение и обработка ошибок, сбоев и других проблем в приложениях. Включает системы для регистрации и отслеживания
5. **Обновление и патчи**:
   * Управление процессом установки патчей, обновлений и новых версий приложений, чтобы гарантировать их безопасность и актуальность.
6. **Безопасность**:
   * Обеспечение защиты данных и предотвращение уязвимостей, например, с помощью инструментов для управления правами доступа, шифрования и аудита безопасности.
7. **Аналитика и отчетность**:
   * Сбор статистики и создание отчётов об использовании и проблемах приложений для улучшения принятия решений и планирования.
8. **Управление конфигурациями**:
   * Контроль

**Примеры средств управления приложениями**:

* **Jira, ServiceNow** — для управления инцидентами и задачами.
* **Nagios, Zabbix, Prometheus** — для мониторинга и производительности.
* **Chef, Puppet, Ansible** — для автоматизации развёртывания и управления конфигурациями.
* **New Relic, AppDynamics** — для мониторинга и оптимизации производительности приложений.

Эти средства помогают упростить задачи по администрированию приложений, повысить их стабильность и эффективность работы.

1. **Мониторинг сетевых ресурсов**

Одно из самых часто используемых и наиболее важных средств мониторинга системы — это регистрация различных событий в журналах операционной системы Windows. Регистрацию событий в системе Windows осуществляет служба " *Журнал событий* " ( *Event Log* ). В любой системе семейства Windows всегда присутствуют 3 журнала:

* журнал " **Система** " ( **System** ) — события, записанные в журнал компонентами операционной системы (например, сбой в запуске службы при перезагрузке); расположение журнала по умолчанию — в папке " %SystemRoot%\system32\config\SysEvent.Evt ";
* журнал " **Безопасность** " ( **Security** ) — регистрация событий, относящихся к системе безопасности (например, попытки регистрации пользователей, изменения в политиках безопасности, попытки доступа к различным ресурсам); набор событий, регистрируемых в журнале " *Безопасность* ", настраивается локальной или групповых политик (об управлении аудитом событий безопасности — в следующем подразделе); расположение по умолчанию — " %SystemRoot%\system 32\соnfig\SecEvent.Evt ";
* журнал " **Приложение** " ( **Application** ) — события, порожденные различными приложениями (например, сбой MS SQL при доступе к базе данных); набор событий, регистрируемых в журнале " *Приложения* ", определяется разработчиками приложений; расположение по умолчанию — " %SystemRoot%\system32\config\AppEvent.Evt ".

При установке в системе каких-либо компонент могут появиться журналы, регистрирующие события, относящиеся к работе данных компонент.

При установке службы DNS появляется журнал " **DNS-сервер** " ( **DNS Server** ), регистрирующий события, связанные с работой службы DNS (расположение по умолчанию — " %SystemRoot%\system32\config\DNSEvent.Evt ").

При создании контроллера домена в системе появляются журналы:

* журнал " **Служба каталогов** " ( **Directory Service** ) — события, порожденные службой каталогов Active Directory;. расположение по умолчанию — " %SystemRoot%\system32\config\NTDS.Evt ";
* журнал " **Служба репликации файлов** " ( **File Replication Service** ) — события, связанные с репликацией файлов (в первую очередь файлы в папке SYSVOL и файлы в сетевых папках, управляемых рапределенной файловой системой DFS); расположение по умолчанию — " %SystemRoot%\system32\config\NtFrs.Evt ".

##### Работа с журналами

Открыть системные журналы можно следующими способами:

* открыть консоль " *Управление компьютером* " и в разделе " *Служебные программы* " открыть оснастку " *Просмотр событий* ";
* открыть отдельную консоль " *Просмотр событий* " в разделе " *Администрирование* " Главного меню системы Windows

В левой части консоли будет список имеющихся на данном компьютере журналов, в правой части — список событий для выбранного журнала.

В большинстве журналов события бывают трех видов:

* **Уведомление** — информация о событии, связанным с успешным действием (например, успешный запуск или останов службы, успешное завершение операции какой-либо службы);
* **Предупреждение** — информация о событиях, которые в будущем могут вызвать проблемы в работе системы;
* **Ошибка** — сообщение об ошибке (например, сбой при запуске службы).

В журнале " *Безопасность* " — 2 типа событий:

* **Аудит успехов** — событие, связанное с успешным выполнением действия, связанного с системой безопасности (например, успешный вход в систему или успешный доступ к сетевому ресурсу);
* **Аудит отказов** — событие, связанное со сбоем в выполнении действия, связанного с системой безопасности (например, отказ в аутентификации пользователя при входе в систему по причине ввода неверного пароля, блокировка учетной записи после нескольких неудачных попыток входа в систему, отказ в доступе к сетевому ресурсу).

В столбцах журнала, кроме типа события, содержатся следующие данные:

* **Дата и время** регистрации события;
* **Источник** — приложение, служба или системная компонента, записавшие событие в журнал;
* **Категория** — категория события, иногда используемая для его более подробного описания;
* **Событие** — код события;
* **Пользователь** — учетная запись пользователя, действовавшая в момент события;
* **Компьютер** — имя компьютера, на котором произошло событие.

Если открыть какое-либо событие, то можно получить более детальную информацию о нем ([рис. 16.2](https://intuit.ru/studies/courses/991/216/lecture/5582?page=1#image.16.2)):

* **Описание** — текстовое описание события;
* **Данные** — любые данные, сгенерированные событием, или связанный с ним код ошибки.

Размер и способ ведения журналов событий можно настраивать. Для настройки параметров надо щелкнуть правой кнопкой на нужном журнале событий и выбрать в контекстном меню команду *Свойства*.

Системы семейства Windows Server позволяют осуществлять мониторинг сетевой активности на низком уровне — на уровне сетевых фреймов, передаваемых и принимаемых сетевыми адаптерами компьютера. Для выполнения этой задачи служит компонента системы " *Сетевой монитор* " (*Network Monitor*). Правда, штатная компонента " *Сетевой монитор* " имеет ограничение — она захватывает только те сетевые пакеты, которые передаются данному компьютеру (на котором запущен " *Сетевой монитор* ") или отправляются с данного компьютера, в том числе широковещательные пакеты. " *Сетевой монитор* " — мощный инструмент для обнаружения различных сетевых проблем и неполадок, позволяющий детально изучать содержимое передаваемых по сети пакетов.

Все данные, пересылаемые по сети от одного сетевого адаптера другому, состоят из фреймов (кадров). Каждый фрейм содержит следующую информацию:

* *Адрес источника (отправителя)* — MAC-адрес сетевого адаптера, с которого отправлен кадр;
* *Адрес назначения (получателя)* — MAC-адрес сетевого адаптера, которому предназначался кадр (этот адрес может также определять группу сетевых адаптеров);
* *Данные заголовка* — информация, содержащая описание для каждого протокола, используемого при передаче кадра;
* *Данные* — передаваемые данные (или часть данных).

*Установка компонентов Windows* " — " *Средства управления и наблюдения* " — кнопка " *Состав* " — " *Средства сетевого монитора* ". После установки " *Сетевого монитора* " в разделе " *Администрирование* " *Главного меню* системы появляется соответствующий ярлык.

1. **Схемы и алгоритмы анализа ошибок**

Неотъемлемой частью процесса сопровождения программного обеспечения (ПО) является сбор и консолидация данных об ошибках, возникающих в процессе его работы.

 Источниками данных об ошибках могут быть как пользователи ПО, так и средства мониторинга вычислительных систем.

Всё множество программных ошибок, приводящих к некорректному поведению программы во время исполнения, можно разделить на следующие классы по видам вредоносного воздействия:

• ошибки, приводящие к порче пользовательских данных в процессе обработки: целочисленное переполнение, порча данных в оперативной памяти, обращение к неинициализированному блоку памяти, обращение к памяти по неинициализированному или висящему указателю (англ. – dangling pointer), фальсификация данных (англ. – request forgery) и др.;

• ошибки, приводящие к неавторизованному доступу к пользовательским данным: получение неавторизованного доступа к базе данных, получение неавторизованного доступа к информации в оперативной или постоянной памяти вычислительного устройства, получение повышенного уровня привилегий доступа к данным и др.;

• ошибки, приводящие к исчерпанию системных ресурсов, таких как память на куче, файлы, сокеты и др.;

• ошибки, приводящие к аварийному завершению исполнения программы: доступ к области памяти, не принадлежащей программе, деление на ноль и др.;

• ошибки, приводящие к аварийному завершению исполнения программы: доступ к области памяти, не принадлежащей программе, деление на ноль и др.;

• Влияние квалификации команды разработчиков программы: плохой или недостаточный уровень проработки архитектуры программы: плохо спроектированная архитектура программы может привести к внесению ошибок в процессе развития программной системы;

Причины появления ошибок в программах • Уделяется мало внимания различным уровням тестирования программы в процессе разработки: \*недостаточность или отсутствие интеграционных тестов, что приводит к интеграционным ошибкам в процессе изменения кода компонентов, в том числе разделяемых компонентов операционной системы, не учитывающих контекст их использования; \*недостаточное количество и качество автоматических тестов на компоненты (модульных тестов), что приводит к позднему обнаружению нарушения контракта использования модуля и функции; \*недостаточный уровень тестирования программы на устойчивость при обработке некорректных внешних данных

 Влияние изменений, вносимых в среду исполнения программы. Многие современные программы используют разделяемые библиотеки, как правило, входящие в комплект поставки операционных систем или используемые совместно различными программами. Изменение в коде разделяемых библиотек при недостатке интеграционного тестирования может привести к неправильной работе программы.

Всё множество причин появления ошибок в программах в итоге сводятся к одной – недостаточной квалификации разработчиков программного обеспечения. В связи с этим становится очевидной необходимость создания автоматических средств обнаружения ошибок в программном обеспечении, которые позволят обнаруживать ошибки на различных этапах жизненного цикла ПО.

На ранних этапах развития вычислительной техники применялся процесс «Кодирование и исправление», который заключался в кодировании программы с последующим этапом избавления программы от ошибок (англ. debugging) программистом, написавшим программу.

Современные методы решения ошибок • Позже стали применяться методы тестирования программ для подтверждения того, что программа делает именно то, что требуется, и не делает ничего, для чего она не предназначена

1. **Методы контроля и диагностики систем и сетей связи**

Методы контроля и диагностики систем и сетей связи включают в себя:

1. **Мониторинг состояния сети**: Регулярный сбор и анализ данных о состоянии компонентов сети (серверов, маршрутизаторов, каналов связи) для выявления потенциальных неисправностей.
2. **Тестирование производительности**: Измерение пропускной способности, задержек, потерь пакетов и других характеристик сети с помощью различных инструментов, таких как iperf, ping, traceroute.
3. **Анализ журналов событий**: Системы сбора и анализа логов (например, syslog) помогают выявлять ошибки и сбои в работе оборудования и приложений.
4. **Диагностика с помощью протоколов связи**: Использование протоколов диагностики, таких как ICMP, SNMP, и другие для тестирования соединений и мониторинга состояния сетевых устройств.
5. **Методы визуализации и картографирования сети**: Использование программных средств для создания карт сети, которые позволяют оперативно выявлять проблемы и управлять инфраструктурой.
6. **Использование средств сетевого анализа**: Программы вроде Wireshark позволяют анализировать трафик, выявлять аномалии и проблемы в сети, такие как перегрузки или утечки данных.
7. **Методы диагностики на основе искусственного интеллекта**: Современные системы могут использовать машинное обучение для предсказания отказов и оптимизации работы сети.

Эти методы позволяют своевременно обнаруживать и устранять неисправности, обеспечивая стабильную и эффективную работу сетей связи.

1. **Использование баз знаний в информационных системах**

*Знание* - это закономерности *предметной области* (принципы, связи, законы), полученные в результате практической деятельности и профессионального опыта, позволяющие специалистам ставить и решать задачи в этой области

При обработке на ЭВМ знания трансформируются аналогично данным.

Z1 — знания в памяти человека как результат мышления;

Z2 — материальные носители знаний (учебники, методические пособия);

Z3 — *поле знаний* — условное описание основных объектов предметной области, их атрибутов и закономерностей, их связывающих;

Z4 — знания, описанные на языках представления знаний (продукционные языки, семантические сети, фреймы — см. далее);

Z5 — *база знаний на машинных носителях информации.*

1. **Хранилища данных и базы знаний**

Причины возникновения технологии DW

Потребовался совокупный анализ информации, который не могли осуществить БД Для работы с информацией возникла необходимость хранить её в интегрированном виде

Принципы работы БД не подходили для этих целей

Аналитическая информационная система

Для достижения этих целей была создана аналитическая информационная система. Её компоненты: - Оперативные БД как источники информации - Система аналитической обработки данных для их анализа - И, самое главное, аналитическая БД для сбора, хранения и предоставления необходимой информации

Понятие хранилищ данных

Хранилище данных (Data Warehouse): это предметно-ориентированный, интегрированный, неизменчивый, поддерживающий хронологию набор данных, организованный для целей поддержки управления..

Концепция хранилищ данных по С.Кузнецову

Интеграция ранее разъединенных детализированных данных в едином DW, их согласование и агрегация Разделение наборов данных для операционной обработки и наборов данных для решения задач анализа

Свойства хранилищ данных

Предметная ориентация Интегрированность данных Инвариантность во времени Стабильность информации Минимизация избыточности информации

Модели хранилищ данных

Схема «звезда» Схема «снежинка» ER-диаграмма

Основные компоненты хранилищ данных

Оперативные источники данных Средства переноса и трансформации данных Метаданные Реляционное хранилище OLAP-хранилище и OLAP-клиенты Средства доступа и анализа данных

Применение хранилищ данных

Финансовые DW DW в области страхования DW для управления людскими ресурсами DW в области телекоммуникаций и прочие

Инструментальные средства: решения в области хранилищ данных

IBM: A Data Warehouse Plus Oracle: Warehouse Technology Initiative Hewlett Packard: OpenWarehouse NCR: Enterprise Information Factory Informix Software: On-Line Dinamic Parallel Server SAS Institute: ряд решений Sybase: Warehouse WORKS

Примеры успешного внедрения хранилищ данных

Сеть супермаркетов Hyundai Интеграция сведений о клиентах, продажах и спросе на товары для выявления тенденций, предпочтений и формирования стратегии Корпорация NCR Для повышения эффективности деятельности, управления компанией и для обеспечения общего представления об эффективности собственной работы Компания Sekisui DW упростило учёт финансовой деятельности, объединило базы данных управления и бухгалтерии, позволило анализировать текущее положение компании гораздо оперативнее, чем раньше.

Особенности, тенденции и проблемы: технологии хранилищ данных

Технологии хранилищ данных «Взрывание» объемов данных Активные DW Распространение источников данных Новые требования к DW Интегрирование представления о клиенте Потребность в более сложных запросах Слияние с CRM

1. **Экспертные системы база знаний формализованных эмпирических знаний**

**Организация знаний**

Большая часть задач инженерии знаний включает получение знаний от экспертов в виде фактов и правил и преобразование этой информации в форму, которая может быть эффективно использована машинной программой. Рассмотрение экспертных знаний как обязательного элемента, определяющего успех функционирования экспертной системы, приводит к тому, что процесс приобретения и представления знаний становится решающим аспектом разработки этих систем.

**Процесс создания, ведения и модификации баз знаний экспертных систем**включает:

* определение состава представляемых знаний;
* организацию знаний;
* представление знаний, т.е. выбор или создание модели представления;
* использование модели представления знаний.

Определение **состава представляемых знаний** производится в зависимости от предметной области и структуры экспертной системы. При этом учитывается:

* тип решаемых задач,
* статический или динамический характер данных, а также такие параметры экспертных знаний, как
* точность,
* ошибочность,
* многозначность,
* полнота или неполнота знаний конкретной предметной области.

База знаний экспертной системы может быть представлена как:

* база фактов,
* база правил и
* база процедур.

Это разбиение отражает **различные уровни представления знаний**.

Усложнение функциональных возможностей экспертной системы происходит за счет того, что она должна уметь не только использовать свои **знания о проблемной области (объектный уровень знаний)**, но и обладает способностью исследовать их – экспертная система должна иметь **знания о том, как представлены ее знания** о проблемной области **(метауровневые знания)**.

Метазнания (знания метауровня) не содержат ссылок к знаниям объектного уровня и не зависят от проблемной области. Основная **цель организации метазнаний** (многоуровневая организация знаний) заключается в следующем:

* разработка стратегий доказательств в ЭС;
* управление выводом результатов поиска решений;
* увеличение выразительной мощности языков представления знаний.

Использование метазнаний при разработке и выборе стратегий доказательств связано с **возможностью** **определения** **на метауровне новых правил поиска результатов объектного уровня**. Метазнания в виде стратегических метауровневых правил:

* рассматриваются как высокоуровневый метод построения формальных доказательств;
* используются для выбора релевантных правил;
* позволяют системе адаптироваться путем перестройки правил и функций объектного уровня; а также
* явно указать возможности и ограничения системы.

База знаний - важная компонента экспертной системы, она предназначена для хранения долгосрочных , знаний, описывающих рассматриваемую предметную область (а не текущих данных), и правил, описывающих целесообразные преобразования знаний этой области. В качестве предметной области выбирается узкая (специальная) прикладная область.

1. **Отчет об ошибках системы: содержание, использование информации**

Отчёт об ошибке (англ. error report или crash report) – это файл, содержащий техническую информацию об исключительной ситуации (исключении), произошедшей в программе на компьютере пользователя.Отчеты об ошибках создаются, когда в системе возникают неполадки с аппаратным или программным обеспечением. Отчеты об ошибках содержат следующие разделы: сведения о состоянии компьютера при возникновении ошибки, версия используемой операционной системы и аппаратного обеспечения, а также цифровой код продукта, используемый для определения лицензии. Также передается IP-адрес компьютера, поскольку для отправки отчета необходимо подключиться к сетевой службе. Отчеты об ошибках могут содержать данные файлов журналов, например, имена пользователей, IP-адреса, URL-адреса, имена файлов и пути к ним, а также адреса электронной почты. Отчеты об ошибках отправляются с использованием шифрования в базу данных с ограниченным доступом и не используются в каких-либо коммерческих целях.

Структура отчета об ошибке

ID (Идентификатор, Номер). Каждой записи в системе учета ошибок присваивается уникальный идентификатор или номер. Как правило, он задается самой системой по определенному шаблону. (1,2,3,…3487), а может быть идентификатор вида ПРОЕКТ-НОМЕР, например, TPO-2367, REG-335 и так далее.

Тип (Трекер). Системы управления задачами, как правило, содержат в себе записи различных типов – задача (Task), улучшение (Feature), баг (Bug), пользовательская история (User Story) и так далее. Для каждого типа может быть свой набор полей и свой жизненный цикл.

Заголовок (Тема, Title). Краткое описание проблемы. Оно отображается в списках, результатах поиска, фильтрах и позволяет быстро понять, в чем суть проблемы. Оно не должно быть слишком коротким и общим, но одновременно и не должно быть слишком длинным. Существует мнемоническое правило для грамотного составления описания «Что? Где? Когда?»: описание должно описывать, что именно сломалось, где сломалось и при каких условиях. Например, «Не работает поиск» – плохое описание ошибки, а «В форме поиска после отправки запроса выдается ошибка «Internal Error» вместо результатов»

– уже лучше.

Проект. Как правило, один большой продукт подразделяется на несколько проектов для более удобного управления, и в системе учета задач необходимо указать, к какому именно проекту относится данная ошибка.

Версия. Версия продукта, в которой ошибка была обнаружена.

Компонент. Компонент продукта, где была обнаружена ошибка. Как правило, список доступных компонентов ограничен и создается администратором системы.

Приоритет (Priority). Приоритет показывает, с какой срочностью должен быть исправлен дефект.

Серьезность (Важность, Severity). Серьезность показывает степень влияния дефекта на систему.

Окружение. Описание системы – программного и аппаратного обеспечения, на котором воспроизводится данный дефект.

На кого назначен. Кто будет ответственен за решение данного дефекта. В зависимости от принятых правил и процессов в компании, это может быть конкретный разработчик, руководитель группы разработчиков, или же поле по умолчанию остается пустым, а разработчики сами решают, кто будет править этот дефект.

Описание. Подробное описание проблемы. Иногда бывает одно поле для описания проблемы, и тогда в нем нужно указать шаги для воспроизведения, фактический результат, ожидаемый результат. Иногда вместо этого поля могут быть 3 разных – для шагов, фактического и ожидаемого результатов.

Шаги для воспроизведения. Не всегда этот пункт выделяется как отдельное поле.

Если его нет, то нужно шаги для воспроизведения написать в поле «Описание».

Фактический результат. Должен быть обязательно – нужно указать, что мы получили в результате выполнения указанных шагов.

Ожидаемый результат. Необходимо указать, чтобы было понятно, как система должна работать. Если есть возможность, то необходимо давать ссылку на документацию, требования или иные документы, в которых описывается ожидаемое поведение системы.

Вложения (скриншоты, файлы журнала и пр.). Файлы, которые могут помочь для понимания, воспроизведения, локализации и исправления дефекта.

Протоколы ошибок

Процесс регистрации ошибки начинается с момента, когда ошибка обнаружена модулем операционной системы. Сегмент кода, отвечающий за обнаружение ошибок, передает сведения об ошибке либо в службы ядра и либо в функцию errlog. В обоих случаях данные заносятся в особый файл. Вместе с данными об ошибке записывается время ее обнаружения. Происходит постоянная проверка на наличие новых записей в этом особом файле, а при поступлении новых данных выполняет стандартную процедуру обработки.

Прежде чем добавить запись в протокол ошибок, демон ошибок сравнивает метку, полученную от ядра или приложения, с содержимым реестра шаблонов ошибок. Если в реестре есть запись, соответствующая метке, демон начинает сбор данных из других областей системы.

Для создания записи в протоколе ошибок демон ошибок считывает шаблон из реестра, имя ресурса блока, обнаружившего ошибку, и сведения об ошибке. Если ошибка свидетельствует об аппаратной неполадке, то для нее предусмотрены специальные данные в реестре аппаратного обеспечения. Большинство записей в протоколе ошибок связано с программными и аппаратными неполадками, однако в нем могут быть и информационные сообщения.

Команда diag (Linux) применяется для диагностики аппаратных неполадок на основе содержимого протокола ошибок. Для правильной диагностики новых неполадок система удаляет из протокола записи об аппаратных ошибках старше 90 дней. Записи о программных ошибках удаляются через 30 дней после занесения в протокол.

Передача протокола ошибок в другую систему

Существует несколько способов передать файл в другую систему. Например, файл можно скопировать в смонтированную файловую систему из удаленной системы. Файл можно передать по сетевому соединению. Кроме того, файл можно скопировать на съемный носитель, а затем восстановить его в другой системе.

Удаление записей из протокола ошибок Записи удаляются из протокола ошибок либо пользователем, мощи специальной команды ежедневно выполняемым заданием, либо при начале нового цикла записи в файл протокола ошибок после того, как был достигнут максимальный размер файла. После того как размер файла протокола ошибок достигает ограничения, указанного в базе данных конфигурации протокола ошибок, самые старые записи протокола начинают заменяться на новые записи.

Команда errclear (Linux)

позволяет выборочно удалить записи из протокола ошибок. В качестве критерия выбора записей можно указать ИД ошибки, порядковый номер, метку ошибки, имя ресурса, класс ресурса, класс ошибки и тип ошибки. Кроме того, необходимо указать минимальное время создания записей. Команда удалит все записи, соответствующие заданному критерию и созданные позже указанного времени.

Занесение в протокол информации об обслуживании

С помощью команды errlogger (Linux) системный администратор может добавлять записи в протокол ошибок. При выполнении обслуживания системы рекомендуется заносить в системный протокол ошибок информацию о выполненных действиях, например, об очистке протокола ошибок, замене аппаратного компонента или применении исправления.

1. **Методы и инструменты тестирования приложений**

Целью тестирования - проверка работы реализованных функций в соответствии с их спецификацией. На основе внешних спецификаций функций и проектной информации на процессах ЖЦ создаются функциональные тесты, с помощью которых проводится тестирование с учетом требований, сформулированных на этапе анализа *предметной области*. Методы *функционального тестирования* подразделяются на статические и динамические.

*Статические методы*используются при проведении инспекций и рассмотрении спецификаций компонентов без их выполнения. Техника статического анализа заключается в методическом просмотре (или обзоре) и анализе структуры программ, а также в доказательстве их правильности. Статический анализ направлен на анализ документов, разработанных на всех этапах ЖЦ и заключается в инспекции исходного кода и сквозного контроля программы.

*Инспекция ПО*- это статическая проверка соответствия программы заданным спецификациями, проводится путем анализа различных представлений результатов проектирования (документации, требований, спецификаций, схем или исходного кода программ) на процессах ЖЦ. Просмотры и инспекции результатов проектирования и соответствия их требованиям заказчика обеспечивают более высокое качество создаваемых ПС.

При инспекции программ рассматриваются документы рабочего проектирования на этапах ЖЦ совместно с независимыми экспертами и участниками разработки ПС. На начальном этапе проектирования инспекция предполагает проверку полноты, целостности, однозначности, непротиворечивости и совместимости документов с исходными требованиями к программной системе. На этапе реализации системы под *инспекцией*понимается анализ текстов программ на соблюдение требований стандартов и принятых руководящих документов технологии программирования.

Эффективность такой проверки заключается в том, что привлекаемые эксперты пытаются взглянуть на проблему "со стороны" и подвергают ее всестороннему критическому анализу.

Эти приемы позволяют на более ранних этапах проектирования обнаружить ошибки или дефекты путем многократного просмотра исходных кодов. Символьное тестирование применяется для проверки отдельных участков программы на входных символьных значениях.

Кроме того, разрабатывается множество новых способов автоматизации символьного выполнения программ. Например, автоматизированное средство статического контроля для языков ориентированной разработки, инструменты автоматизации *доказательства корректности* и автоматизированный аппарат сетей Петри.

#### 7.2.2. Динамические методы тестирования

*Динамические методы тестирования*используются в процессе выполнения программ. Они базируются на графе, связывающем причины ошибок с ожидаемыми реакциями на эти ошибки. В процессе тестирования накапливается информация об ошибках, которая используется при оценке надежности и качества ПС.

Динамическое тестирование ориентировано на проверку корректности ПС на множестве тестов, прогоняемых по ПС, в целях проверки и сбора данных на этапах ЖЦ и проведения измерения отдельных показателей (число отказов, сбоев) тестирования для оценки характеристик качества, указанных в требованиях, посредством выполнения системы на ЭВМ. Тестирование основывается на систематических, статистических, (вероятностных) и имитационных методах.

Дадим краткую их характеристику.

Систематические методы тестирования делятся на методы, в которых программы рассматриваются как "черный ящик" (используется информация о решаемой задаче), и методы, в которых программа рассматривается как "белый ящик" (используется структура программы). Этот вид называют тестированием с управлением по данным или управлением по входу выходу. Цель - выяснение обстоятельств, при которых поведение программы не соответствует ее спецификации. При этом количество обнаруженных ошибок в программе является критерием качества входного тестирования.

Цель динамического тестирования программ по принципу "черного ящика" - выявление одним тестом максимального числа ошибок с использованием небольшого подмножества возможных входных данных.

*Методы "черного ящика"*обеспечивают:

* эквивалентное разбиение;
* анализ граничных значений;
* применение функциональных диаграмм, которые в соединении с реверсивным анализом дают достаточно полную информацию о функционировании тестируемой программы.

Эквивалентное разбиение состоит в разбиении входной области данных программы на конечное число классов эквивалентности так, чтобы каждый тест, являющийся представителем некоторого класса, был эквивалентен любому другому тесту этого класса.

Классы эквивалентности выделяются путем перебора входных условий и разбиения их на две или более групп. При этом различают два типа классов эквивалентности: правильные, задающие входные данные для программы, и неправильные, основанные на задании ошибочных входных значений. Разработка тестов методом эквивалентного разбиения осуществляется в два этапа: выделение классов эквивалентности и построение тестов. При построении тестов, основанных на выборе входных данных, проводится символическое выполнение программы.

Итак, методы тестирования по принципу "черного ящика" используются для тестирования функций, реализованных в программе, путем проверки несоответствия между реальным поведением функций и ожидаемым поведением с учетом спецификаций требований. Во время подготовки к этому тестированию строятся таблицы условий, причинно-следственные графы и области разбивки. Кроме того, подготавливаются тестовые наборы, учитывающие параметры и условия среды, которые влияют на поведение функций. Для каждого условия определяется множество значений и ограничений предикатов, которые тестируются.

*Метод "белого ящика"*позволяет исследовать внутреннюю структуру программы, причем обнаружение всех ошибок в программе является критерием исчерпывающего тестирования маршрутов потоков (графа) передач управления, среди которых рассматриваются:

* (а) критерий покрытия операторов - набор тестов в совокупности должен обеспечить прохождение каждого оператора не менее одного раза;
* (б) критерий тестирования ветвей (известный как покрытие решений или покрытие переходов) - набор тестов в совокупности должен обеспечить прохождение каждой ветви и выхода, по крайней мере, один раз.

Критерий (б) соответствует простому структурному тесту и наиболее распространен на практике. Для удовлетворения этого критерия необходимо построить систему путей, содержащую все ветви программы. Нахождение такого оптимального покрытия в некоторых случаях осуществляется просто, а в других является более сложной задачей.

Тестирование по принципу "белого ящика" ориентировано на проверку прохождения всех путей программ посредством применения путевого и имитационного тестирования.

*Путевое тестирование*применяется на уровне модулей и графовой *модели программы* путем выбора тестовых ситуаций, подготовки данных и включает тестирование следующих элементов:

* операторов, которые должны быть выполнены хотя бы один раз, без учета ошибок, которые могут остаться в программе из-за большого количества логических путей и необходимости прохождения подмножеств этих путей;
* путей по заданному графу потоков управления для выявления разных маршрутов передачи управления с помощью путевых предикатов, для вычисления которого создается набор тестовых данных, гарантирующих прохождение всех путей. Однако все пути протестировать бывает невозможно, поэтому остаются не выявленные ошибки, которые могут проявиться в процессе эксплуатации;
* блоков, разделяющих программы на отдельные части блоки, которые выполняются один раз или многократно при нахождении путей в программе, включающих совокупность блоков реализации одной функции либо нахождения входного множества данных, которое будет использоваться для выполнения указанного пути.

"Белый ящик" базируется на структуре программы, в случае "черного ящика", о структуре программы ничего неизвестно. Для выполнения тестирования с помощью этих "ящиков" известными считаются выполняемые функции, входы (входные данные) и выходы (выходные данные), а также логика обработки, представленные в документации.

1. **Методы проектирования тестовых наборов данных при тестировании приложений**

**Решение о проведении приемочного тестирования** принимается, когда:

* продукт достиг необходимого уровня качества;
* заказчик ознакомлен с **Планом Приемочных Работ** (**Product Acceptance Plan**) или иным документом, где описан набор действий, связанных с проведением приемочного тестирования, дата проведения, ответственные и т.д.

**Фаза приемочного тестирования** длится до тех пор, пока заказчик не выносит решение об отправлении приложения на доработку или выдаче приложения.

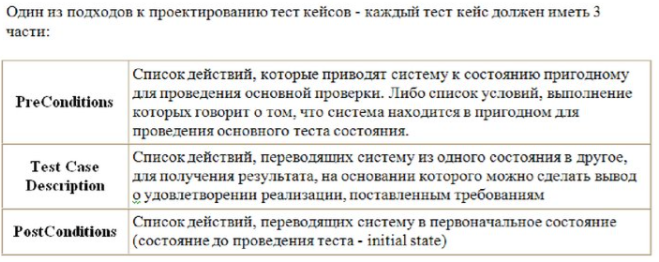
Тестовые Артефакты В соответствие с процессами или методологиями разработки ПО, во время проведения тестирования создается и используется определенное количество тестовых артефактов (документы, модели и т.д.). Наиболее распространенными тестовыми артефактами являются: • Тест План (План тестирования) • Набор тест кейсов и тестов • Баг (дефект) репорт. Тест План (План тестирования)

Тест план (Test Plan) - это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения. Тест план – это документ, отвечающий на следующие вопросы: • Что надо тестировать? • описание объекта тестирования: системы, приложения, оборудование • Что будете тестировать? • список функций и описание тестируемой системы и её компонент в отдельности • Как будете тестировать? • стратегия тестирования, а именно: виды тестирования и их применение по отношению к тестируемому объекту • Когда будете тестировать? • последовательность проведения работ: подготовка (Test Preparation), тестирование (Testing), анализ результатов (Test Result Analisys ) в разрезе запланированных фаз разработки

Тест План (План тестирования) • Критерии начала тестирования: • готовность тестовой платформы (тестового стенда) • законченность разработки требуемого функционала • наличие всей необходимой документации • ... • Критерии окончания тестирования: • результаты тестирования удовлетворяют критериям качества продукта • требования к количеству открытых багов выполнены • выдержка определенного периода без изменения исходного кода приложения Code Freeze (CF) • выдержка определенного периода без открытия новых багов Zero Bug Bounce (ZBB)

Можно дополнить тест план следующими пунктами: • Окружение тестируемой системы • Необходимое для тестирования оборудование и программные средства • Риски и их разрешение Чаще всего на практике приходится сталкиваться со следующими видами тест планов: • Мастер Тест План (Master Plan or Master Test Plan) • Тест План (Test Plan, назовем его детальный тест план) • План Приемочных Испытаний (Product Acceptance Plan) - документ, описывающий набор действий, связанных с приемочным тестированием: стратегия, дата проведения, ответственные работники и т.д.

Набор тест кейсов и тестов (Test Case & Test suite) Тестовый случай (Test Case) - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. Под тест кейсом понимается структура вида: Action > Expected Result > Test Result Тест кейсы разделяются по ожидаемому результату на позитивные и негативные: • Позитивный тест кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию. • Негативный тест кейс оперирует как корректными, так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.



1. **Список вопросов для выявления ошибок при инспекции**

Важной частью процесса инспектирования является проверка программы на наличие общих ошибок с помощью списка вопросов для выявления ошибок. Концентрация внимания в предлагаемом списке на рассмотрении стиля, а не ошибок (вопросы типа «Являются ли комментарии точными и информативными?» и «Располагаются ли операторы THEN/ELSE и DO/END по одной вертикали друг под другом?») представляется неудачной. Также неудачным представляется и наличие неопределенности в списке, уменьшающее его полезность (вопросы типа «Соответствует ли текст про­грам­мы требованиям, выдвигаемым при проектировании?»). Список, приведенный в данном разделе, был составлен после многолетнего изучения ошибок программного обеспечения. В значительной мере он является независимым от языка; это означает, что большинство ошибок встречается в любом языке программирования.

## Ошибки обращения к данным

1. Существуют ли обращения к переменным, значения которым не присвоены или не инициализированы? Наличие переменных с не установленными значениями — наиболее часто встречающаяся программная ошибка, она возникает при различных обстоятельствах. Для каждого обращения к единице данных (например, к переменной, элементу массива, полю в структуре) попытайтесь неформально «доказать», что ей присвоено значение в проверяемой точке.

2. Не выходит ли значение каждого из индексов за границы, определенные для соответствующего измерения при всех обращениях к массиву?

3. Принимает ли каждый индекс целые значения при всех обращениях к массиву? Нецелые индексы не обязательно являются ошибкой для всех языков программирования, но представляют практическую опасность.

4. Для всех обращений с помощью указателей или переменных-ссы­лок память, к которой производится обращение, уже распределена? Наличие переменных-ссы­лок представляет собой ошибку типа «подвешенного обращения». Она возникает в ситуациях, когда время жизни указателя больше, чем время жизни памяти, к которой производится обращение. Например, к такому результату приводит ситуация, когда указатель задает локальную переменную в теле процедуры, значение указателя присваивается выходному параметру или глобальной переменной, процедура завершается (освобождая адресуемую память), а программа затем пытается использовать значение указателя. Как и при поиске ошибок первых трех типов, попытайтесь неформально доказать, что для каждого обращения, использующего переменную-ука­за­тель, адресуемая память существует.

5. Если одна и та же область памяти имеет несколько псевдонимов (имен) с различными атрибутами, то имеют ли значения данных в этой области корректные атрибуты при обращении по одному из этих псевдонимов? Ошибки типа некорректных атрибутов у псевдонимов могут возникнуть при использовании атрибута DEFINED или базированной памяти в PL/1, оператора EQUIVALENCE в Фортране, глагола REDEFINES в Коболе, записей с вариантами в Паскале или объединений (UNION) в языке Си. В качестве примера можно привести программу на языке Си, содержащую вещественную переменную *A*и целую переменную *B.*Обе величины размещены на одном и том же месте памяти (т.е. помещены в одно и то же объединение). Если программа записывает величину *A,* а обращается к переменной *B,* то, вероятно, произойдет ошибка, поскольку машина будет использовать битовое представление числа с плавающей точкой в данной области памяти как целое.

6. Отличаются ли типы или атрибуты переменных величин от тех, которые предполагались компилятором? Это может произойти в том случае, когда программа на PL/1 или Коболе считывает записи из памяти и обращается к ним как к структурам, но физическое представление записей отлично от описания структуры. Или программа на языке Си, допускающем произвольные преобразования типов, содержит переменную-ука­за­тель на структуру типа *T*1, инициализированную указателем на структуру типа *T*2 (не наследуемую от *T*1).

7. Существуют ли явные или неявные проблемы адресации, если в машине будут использованы единицы распределения памяти, меньшие, чем единицы адресации памяти? Например, в PL/1 в системе IBM S/370 битовые строки фиксированной длины не обязательно начинаются с границ байтов, а адресами могут быть только границы байтов. Аналогично, в языке Си размер отдельных полей в структуре может задаваться в битах. Если программа вычисляет адрес битового поля и впоследствии обращается к нему по этому адресу, то может произойти ошибочное обращение к памяти. Такое же положение может сложиться при передаче подпрограмме (процедуре) битового поля в качестве аргумента.

8. Если используются указатели или переменные-ссыл­ки, то имеет ли адресуемая память атрибуты, предполагаемые компилятором? Примером несоответствия атрибутов может служить случай, когда указатель PL/1, по которому базируется структура данных, имеет в качестве значения адрес другой структуры.

9. Если к структуре данных обращаются из нескольких процедур или подпрограмм, то определена ли эта структура одинаково в каждой процедуре?

10. Не превышены ли границы строки при индексации в ней?

11. Существуют ли какие-ни­будь другие ошибки в операциях с индексацией или при обращении к массивам по индексу?

# **Ошибки описания данных**

1. Все ли переменные описаны явно? Отсутствие явного описания не обязательно является ошибкой, но служит общим источником беспокойства. Так, если в подпрограмме на Фортране используется элемент массива и отсутствует его описание (например, в операторе DIMENSION), то обращение к массиву (например, Х = А(1)), будет интерпретироваться как вызов функции. Последнее приведет к тому, что машина будет пытаться обработать массив как программу. Если отсутствует явное описание переменной во внутренней процедуре или блоке, следует ли понимать это так, что описание данной переменной совпадает с описанием во внешнем блоке?

2. Если не все атрибуты переменной явно присутствуют в описании, то понятно ли их отсутствие? Например, отсутствие атрибутов, считающееся общепринятым в PL/1, часто является источником неожиданных осложнений.

3. Если начальные значения присваиваются переменным в операторах описания, то правильно ли инициализируются эти значения? Во многих языках программирования присвоение начальных значений массивам и строкам представляется довольно сложным и, следовательно, является возможным источником ошибок.

4. Правильно ли для каждой переменной определены длина, тип и класс памяти (например, STATIC, AUTOMATIC, BASED или CONTROLLED в PL/1; AUTO, CONST, VOLATILE, REGISTER, STATIC в Си и т.д.)?

5. Согласуется ли инициализация переменной с ее типом памяти? Например, если в подпрограмме на Фортране необходимо устанавливать начальные значения переменной каждый раз при вызове подпрограммы, переменная должна быть инициализирована в операторе описания, отличном от оператора DATA. Если в PL/1 описывается инициализация величины и начальное значение необходимо устанавливать каждый раз при вызове процедуры, то для этой переменной должен быть определен класс памяти АUTОМАТIС, а не STATIC.

6. Есть ли переменные со сходными именами (например, VOLT и VOLTS)? Наличие сходных имен не обязательно является ошибкой, но служит признаком того, что имена могут быть перепутаны где-ни­будь внутри программы.

## 6.3.3.3 Ошибки вычислений

1. Имеются ли вычисления, использующие переменные недопустимых (например, неарифметических) типов данных?

2. Существуют ли вычисления, использующие данные разного вида? Например, сложение переменной с плавающей точкой и целой переменной. Такие случаи не обязательно являются ошибочными, но они должны быть тщательно проверены для обеспечения гарантии того, что правила преобразования, принятые в языке, понятны. Это особенно важно для языков со сложными правилами преобразования (например, для PL/1, Си). Например, следующий фрагмент программы на PL/1:

DECLARE A BIT(1);

A=1;

определяет значение A рав­ным битовому 0, а не 1. Или в языке Си попытка присвоить переменной с плавающей точкой значение 1/2 даст значение 0 (т.к. 1 и 2 имеют целый тип).

3. Существуют ли вычисления, использующие переменные, имеющие одинаковый тип данных, но разную длину? Такой вопрос справедлив для PL/1, и возник он из этого языка. Например, в PL/1 результатом вычисления выражения 25 + 1/3 будет 5.333..., а не 25.333… Аналогично, в языке Си присутствуют множество данных одного типа, но имеющих разную длину. Например, это целые типы (CHAR, INT, LONG, SHORT, INT64 и т.д.), вещественные типы (FLOAT, DOUBLE, LONG DOUBLE) и т.д. К тому же, эти типы могут быть как знаковыми (SIGNED), так и беззнаковыми (UNSIGNED).

4. Имеет ли результирующая переменная оператора присваивания атрибуты, описывающие ее с меньшей длиной, чем в атрибутах выражения в правой части?

5. Возможны ли переполнение или потеря результата во время вычисления выражения? Это означает, что конечный результат может казаться правильным, но промежуточный результат может быть слишком большим или слишком малым для машинного представления данных.

6. Возможно ли, чтобы делитель в операторе деления был равен нулю?

7. Если величины представлены в машине в двоичной форме, получаются ли какие-ни­будь результаты неточными? Так, 10×0,1 редко равно 1.0 в двоичной машине.

8. Может ли значение переменной выходить за пределы установленного для нее диапазона? Например, для операторов, присваивающих значение переменной PROB (имеющей смысл вероятности какого-либо события), может быть произведена проверка, будет ли полученное значение всегда положительным и не превышающим 1,0.

9. Верны ли предположения о порядке оценки и следования операторов для выражений, содержащих более чем один оператор?

10. Встречается ли неверное использование целой арифметики, особенно деления? Например, если I — целая величина, то выражение 2\*I/2 = I зависит от того, является значение I четным или нечетным, и от того, какое действие — умножение или деление — выполняется первым.

# **Ошибки при сравнениях**

1. Сравниваются ли в программе величины, имеющие несовместимые типы данных (например, строка символов с адресом)?

2. Сравниваются ли величины различных типов или величины различной длины? Если да, то проверьте, правильно ли интерпретируются (поняты) правила преобразования.

3. Корректны ли операторы сравнения? Программисты часто путают такие отношения, как *наибольший, наименьший, больше чем, не меньше чем, меньше или равно*.

4. Каждое ли булевское выражение сформулировано так, как это предполагалось? Программисты часто делают ошибки при написании логических выражений, содержащих операции «И», «ИЛИ», «НЕ».

5. Являются ли операнды булевских выражений булевскими? Существуют ли ошибочные объединения сравнений и булевских выражений? Они представляют другой часто встречающийся класс ошибок. Примеры нескольких типичных ошибок приведены ниже. Если величина L определена как лежащая в интервале между 2 и 10, то выражение 2<L<10 является неверным. Вместо него должно быть написано выражение (2 < L)&(L < 10). Если же величина L определена как большая, чем Х или Y, то выражение L>X|Y является неверным; оно должно быть записано в виде (L>Х) | (L>Y). При сравнении трех чисел на равенство выражение IF (A=B=C) означает совсем другое. Например, в языке Си произойдет присвоение переменным A и B значения переменной C. А условие будет истинным, если это значение ненулевое. В случае необходимости проверить математическое отношение X=Y=Z правильным будет выражение (X=Y)&(Y=Z). Также в языке Си следует различать булевские и битовые операторы. Например, если A = 1 и B = 2, то условие IF(A && B) будет истинно, а IF(A & B) — ложно.

6. Сравниваются ли в программе мантиссы или числа с плавающей запятой, которые представлены в машине в двоичной форме? Это является иногда источником ошибок из-за усечения младших разрядов. Или из-за неточного равенства чисел в двоичной и десятичной формах представления.

7. Верны ли предположения о порядке оценки и следовании операторов для выражений, содержащих более одного булевского оператора? Иными словами, если задано выражение (А==2)&(В==2)|(С==3), понятно ли, какая из операций выполняется первой: И или ИЛИ?

8. Влияет ли на результат выполнения программы способ, которым конкретный компилятор выполняет булевские выражения? Например, оператор

IF (Х≠О)&((Y/X)>Z)

является приемлемым для некоторых компиляторов PL/1 (т.е. компиляторов, которые заканчивают проверку, как только одно из выражений оператора «И» окажется ложным), но приведет к делению на 0 при использовании других компиляторов.

## Ошибки в передачах управления

1. Если в программе содержится переключатель (например, вычисляемый оператор GO TO в Фортране или его аналог ON… GOTO в Бейсике), то может ли значение индекса когда-ли­бо превысить число возможных переходов? Например, всегда ли L будет принимать значение 1, 2 или 3 в операторе Фортрана GO TO (200, 300, 400), L или операторе Бейсика ON L GOTO 200, 300, 400?

2. Будет ли каждый цикл, в конце концов, завершен? Придумайте неформальное доказательство или аргументы, подтверждающие их завершение.

3. Будут ли программа, модуль или подпрограмма, в конечном счете, завершены?

4. Возможно ли, что из-за входных условий цикл никогда не сможет выполняться? Если это так, то является ли это оплошностью? Например, что произойдет для циклов, начинающихся операторами:

DO WHILE (NOTFOUND)

DO I=X ТО Z

если первоначальное значение NOTFOUND — *ложь* или если Х больше Z?

5. Для циклов, управляемых как числом итераций, так и булевским условием (например, цикл для организации поиска), какова последовательность «погружения в тело цикла»? Например, что произойдет с циклом, имеющим заголовок

DO I=1 ТО TABLESIZE WHILE (NOTFOUND)

если NOTFOUND никогда не принимает значение «ложь»?

6. Существуют ли какие-ни­будь ошибки «отклонения от нормы» (например, слишком большое или слишком малое число итераций)?

7. Если язык программирования содержит понятие группы операторов (например, DO-груп­пы в PL/1, ограниченные операторами DO-END), то имеется ли явный оператор END для каждой группы и соответствуют ли операторы END своим группам?

8. Существуют ли решения, подразумеваемые по умолчанию? Например, пусть ожидается, что входной параметр X принимает значения 1, 2 или 3. Логично ли тогда предположить, что он должен быть равен 3, если он не равен 1 или 2? Например, рассмотрим программу на языке Си:

switch(X)

{case 1: printf(“1!!!”); break;

case 2: printf(“2!!!”); break;

default: printf(“3!!!”);}

Коль скоро это так, то является ли предположение правильным?

1. Равно ли число параметров, получаемых рассматриваемым модулем, числу аргументов, передаваемых каждым из вызывающих модулей? Правилен ли порядок их следования?

2. Совпадают ли атрибуты (например, тип и размер) каждого параметра с атрибутами соответствующего ему аргумента?

3. Совпадают ли единицы измерения каждого параметра с единицами измерения соответствующих аргументов? Например, нет ли случаев, когда значение параметров выражено в градусах, а аргумента — в радианах?

4. Равно ли число аргументов, передаваемых из рассматриваемого модуля другому модулю, числу параметров, ожидаемых в вызываемом модуле?

5. Соответствуют ли атрибуты каждого аргумента, передаваемого другому модулю, атрибутам соответствующего параметра в рассматриваемом модуле?

6. Совпадают ли единицы измерения каждого аргумента, передаваемого другому модулю, с единицами измерения соответствующего параметра в рассматриваемом модуле?

7. Если вызываются встроенные функции, правильно ли заданы число, атрибуты и порядок следования аргументов?

8. Если модуль имеет несколько точек входа, передается ли параметр всегда вне зависимости от точки входа? Такая ошибка присутствует во втором операторе присваивания следующей программы на PL/1:

A: PROCEDURE (W,X);

W=X+1;

RETURN;

В: ENTRY (Y,Z);

Y=X+Z;

END;

9. Не изменяет ли подпрограмма параметр, который должен использоваться только как входная величина?

10. Если имеются глобальные переменные (например, переменные в PL/1 с атрибутом EXTERNAL или в Си с атрибутом EXTERN, переменные, указанные в операторах COMMON Фортрана), имеют ли они одинаковые определения и атрибуты во всех модулях, которые к ним обращаются?

11. Передаются ли в качестве аргументов констан­ты? В некоторых реализациях Фортрана такие операто­ры, как

CALL SUBX (J,3)

являются опасными, поскольку, если подпрограмма SUBX присвоит значение второму параметру, значение константы 3 будет изменено.

## 6.3.3.7 Ошибки ввода-вывода

1. Являются ли правильными атрибуты файлов, описанных явно?

2. Являются ли правильными атрибуты оператора OPEN?

3. Согласуется ли спецификация формата с информацией в операторах ввода-вы­во­да? Например, согласуется ли каждый оператор FORMAT (с точки зрения числа элементарных данных и их атрибутов) с соответствующими операторами READ и WRITE в программе, написанной на Фортране? То же самое применимо к проверке соответствия между списком данных и списком форматов в операторах ввода-вы­во­да PL/1 и Си.

4. Равен ли размеру записи размер области памяти для ввода-вы­во­да? Это может быть важно при блочном вводе-вы­во­де (функции BLOCKREAD и BLOCKWRITE в Паскале, FREAD и FWRITE в Си).

5. Все ли файлы открыты перед их использованием?

6. Правильно ли обнаруживаются и трактуются признаки конца файла?

7. Правильно ли трактуются ошибочные состояния ввода-вы­во­да?

8. Существуют ли смысловые или грамматические ошибки в тексте, выводимом программой на печать или экран дисплея?

# **Другие виды контроля**

1. Если компилятор выдает таблицу перекрестных ссылок идентификаторов, проверьте величины, на которые в этом списке нет ссылок или есть только одна ссылка.

2. Если компилятор выдает список атрибутов, проверьте атрибуты каждой величины для обеспечения гарантии того, что в программе нет никаких неожиданных и отсутствующих атрибутов.

3. Если программа оттранслирована успешно, но компилятор выдает одно или несколько «предупреждений» или «информационных» сообщений, внимательно проверьте каждое из них. Предупреждение свидетельствует о «подозрениях» компилятора в отношении правильности ваших действий. Все эти «подозрения» должны быть рассмотрены. В информационных сообщениях могут перечисляться неописанные переменные или конструкции языка, которые препятствуют оптимизации кода.

4. Является ли программа (или модуль) достаточно устойчивой? Иными словами, проверяет ли она правильность своих входных данных?

5. Не пропущена ли в программе какая-ни­будь функция?

1. **Пользовательская документация: «Руководство программиста»**

# **Гост 19.504-79 еспд. Руководство программиста. Требования к содержанию и оформлению**

Руководство программиста должно содержать разделы:

1. Назначение и условия применения программы.
2. Характеристики программы.
3. Обращение к программе.
4. Входные и выходные данные.
5. Сообщения.

При описании *назначения и условий применения программы*необходимо указать назначение и функции, выполняемые программой; условия, необходимые для выполнения программы: объем оперативной памяти, требования к составу и параметрам периферийных устройств; требования к ПО и т.д.

В разделе *Характеристики программы*необходимо привести описание основных характеристик и особенностей программы: временных характеристик, режима работы, средств контроля правильности выполнения и самовосстанавливаемости программы и т.д.

Раздел *Обращение к программе*представляет собой описание процедур вызова программы (способов передачи управления и параметров данных и др.).

Раздел *Входные и выходные данные*должен содержать описание организации, используемой входной и выходной информации и при необходимости ее кодирования.

При описании *сообщений*необходимо привести тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действия, которые необходимо предпринять по этим сообщениям.

# **Гост 19.504-79 еспд. Руководство программиста. Требования к содержанию и оформлению**

Руководство программиста должно содержать разделы:

1. Назначение и условия применения программы.
2. Характеристики программы.
3. Обращение к программе.
4. Входные и выходные данные.
5. Сообщения.

При описании *назначения и условий применения программы*необходимо указать назначение и функции, выполняемые программой; условия, необходимые для выполнения программы: объем оперативной памяти, требования к составу и параметрам периферийных устройств; требования к ПО и т.д.

В разделе *Характеристики программы*необходимо привести описание основных характеристик и особенностей программы: временных характеристик, режима работы, средств контроля правильности выполнения и самовосстанавливаемости программы и т.д.

Раздел *Обращение к программе*представляет собой описание процедур вызова программы (способов передачи управления и параметров данных и др.).

Раздел *Входные и выходные данные*должен содержать описание организации, используемой входной и выходной информации и при необходимости ее кодирования.

При описании *сообщений*необходимо привести тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действия, которые необходимо предпринять по этим сообщениям.

1. **Пользовательская документация: «Руководство системного администратора»**

Руководство системного администратора составляют на основании ГОСТ 19.503-79

Руководство системного администратора — программный документ, предоставляющий специалисту информацию, необходимую для выполнения работы по установке программ и поддержанию работоспособности всего программного обеспечения, используемого в рамках тех или иных систем.

Руководство системного администратора — вспомогательный документ для прикладных программных продуктов и основной для серверных и системных, которые не имеют непосредственных пользователей.

В руководстве системного администратора должны быть изложены:

– назначение и область применения программы;

– состав программы, основные принципы ее функционирования;

– комплект поставки (если он не указан в отдельном документе);

– системные требования для программы или ее компонентов;

– предпочтительная очередность установки компонентов;

– процедура установки программы или каждого ее компонента;

– порядок обязательной первоначальной настройки программы;

– способы интеграции установленных копий компонентов между собой;

– интегрирование программы со сторонним ПО, например, с сервером БД;

– способы и периодичность контроля правильности работы программы;

– порядок текущего обслуживания работающих копий программы;

– порядок решения всевозможных вспомогательных задач;

– аварийные ситуации и способы их устранения.

Дополнительно в руководстве системного администратора можно описать:

– [пользовательский интерфейс](https://studopedia.ru/5_54877_graficheskiy-interfeys-Windows.html) административной консоли;

– утилиты командной строки и синтаксис их запуска;

– конфигурационные файлы и правила их написания;

– язык для составления управляющих скриптов.

[Структура руководства](https://studopedia.ru/4_91122_ponyatie-rukovodstva-i-liderstva-tipi-rukovodstva-tipi-liderstva.html) системного программиста имеет вид:

1. Общие сведения о программе.

2. Архитектура и [принципы функционирования](https://studopedia.ru/3_42078_printsipi-organizatsii-vichislitelnih-mashin-i-sistem.html).

3. Системные требования.

4. Установка программы.

5. Административная консоль и работа с ней.

6. Файл конфигурации. Составление и правка.

7. Обязательная начальная настройка программы.

8. Проверка правильности функционирования программы.

9. Мероприятия по текущему обслуживанию программы.

10. Оптимизация работы программы.

11. Аварийные ситуации и способы их устранения.

В качестве примера приведено содержание руководства системного администратора для программы ТурбоБухгалтер 6.5. Сетевая версия.

ТурбоБухгалтер — одна из наиболее популярных в нашей стране программ автоматизации [бухгалтерского учета](https://studopedia.ru/3_91501_formi-buhgalterskogo-ucheta.html). Сетевая версия программы позволяет организовать совместную работу нескольких бухгалтеров с общим набором данных. [Клиент-серверная архитектура](https://studopedia.ru/19_17145_klient-servernaya-sistema-SQL-Server.html) и применение протокола TCP/IP для обмена данными между клиентской и серверной частями делает возможной работу бухгалтеров в режиме удаленного доступа (например, из дома). Программа ТурбоБухгалтер разработана компанией ДИЦ. Руководство системного администратора на серверную часть разработано «Философтом».

ВВЕДЕНИЕ

Краткое содержание

Терминология и [условные обозначения](https://studopedia.ru/3_70916_uslovnie-graficheskie-oboznacheniya-tsifrovih-mikroshem.html)

[Техническая поддержка](https://studopedia.ru/11_38363_tehnicheskoe-obespechenie-is.html)

Требования к конфигурации аппаратуры и программного обеспечения

Требования к квалификации администратора

1 Сетевая версия программы и принципы ее работы

1.1 Назначение сетевой версии программы

1.2 Базовые [понятия и определения](https://studopedia.ru/2_12468_osnovnie-ponyatiya-i-opredeleniya-v-oblasti-gostinichnogo-hozyaystva.html)

1.2 План сервера

1.2.2 Базы данных и табличные журналы

1.2.3 [Структура баз данных](https://studopedia.ru/4_82766_ponyatie-bazi-dannih-i-etapi-ee-proektirovaniya.html)

1.2.4 Схемы доступа

1.2.5 Пользователи

1.3 Порядок работы с сетевой версией программы

1.4 Необходимое условие корректной работы сетевой версии программы

2 Установка сервера турбо бухгалтер 6.5

3 [Пользовательский интерфейс](https://studopedia.ru/5_54877_graficheskiy-interfeys-Windows.html) сервера

3.1 Запуск программы Турбо Бухгалтер Сервер

3.2 Главное окно

3.3 Меню

3.4 Панель инструментов

3.5 Строка состояния

3.6 Список пользователей

4 Настройка плана сервера

4.1 Порядок выполнения настройки. Редактор плана сервера

4.2 Настройка баз данных, табличных журналов и их разделов

4.3 [Структура баз данных](https://studopedia.ru/4_82766_ponyatie-bazi-dannih-i-etapi-ee-proektirovaniya.html): редактирование и реорганизация

4.4 Настройка схем доступа

4.5 Настройка перечня пользователей

4.6 Завершение работы с редактором плана сервера

5 Установка связи сервера с клиентскими рабочими местами

5.1Подключение клиентских мест к Серверу Турбо Бухгалтер 6.5

5.2. Отладка связи между клиентскими местами и программой-сервером.

5.3 Основные неполадки и способы их устранения

6 Управление работой клиентских модулей с сервера

6.1 Просмотр изменений, вносимых пользователями в данные

6.2 Отправка сообщений пользователям

6.3 Закрытие клиентских мест

6.4 Отключение клиентских мест

6.5 Потеря связи с клиентским местом

7 Взаимодействие пользователей с программой-сервером

7.1 Отключение от программы-сервера

7.2.Типичные неполадки при взаимодействии пользователя с программой-сервером

8 Оптимизация и настройка программы-сервера

8.1 Подготовительные и заключительные операции

8.2 Проверка наличия связи

8.3 Режим регистрации изменений записей общих данных

8.4 Пароль администратора сервера

8.5 Дополнительные параметры

9.Ведение резервных копий (зеркалирование) баз данных

9.1 Понятие и порядок зеркалирования

9.2 Настройка зеркалирования

9.3 Экспорт зеркалирования баз данных

9.4 Импорт зеркалирования баз данных

ПРИЛОЖЕНИЯ

Приложение A. Перенос баз данных и табличных журналов на сервер

Приложение B. Возможности работы в глобальных сетях

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

УКАЗАТЕЛЬ

ГЛОССАРИЙ

1. **Выявление аппаратных ошибок информационной системы**

Под *контролем ИС понимаются процессы, обеспечивающие обнаружение ошибок*в работе информационной системы, вызванных отказами или сбоями аппаратуры, программного обеспечения или ошибками человека-оператора [5.1].

Важным показателем качества контроля является *полнота контроля.*Количественно полнота контроля оценивается отношением, показывающим относительное число элементов схемы (количество команд программы), охваченных данным способом контроля.

По *типу применяемых средств*контроль ИС разделяется на аппаратный, программный и смешенный, осуществляемый совместно аппаратными и программными средствами.

*Аппаратный контроль* отличается большим быстродействием, но требует дополнительной аппаратуры.

*Оперативный контроль* осуществляется в ходе решения функциональных задач ИС и позволяет немедленно обнаруживать ошибку в ее работе. Оперативный контроль в принципе не является полным.

*Тестовый контроль* осуществляется в специально отведенные промежутки времени при помощи специальных тестовых задач. Он основан на тестах, которые могут обеспечить полный контроль объекта (элементов аппаратуры или команд программы) за минимальное время. Недостаток тестового контроля – потеря процессорного времени, расходуемого на тесты. Тестовый контроль, как правило, не позволяет обнаруживать сбои аппаратуры в процессе ее эксплуатации, так как обычно к моменту проведения теста имевший место сбой спонтанно исчезает. Тестовый контроль программ рассматривается в 6.5.1.

*По способу организации* различают контроль прямой, обратный, смешанный.

При прямом контроле основной вычислительный процесс **О**с определенными исходными данными **Х**сопровождается параллельным процессом **П.**В случае безошибочной работы системы результаты этих процессов должны совпадать (это определяется сравнивающим устройством **М**). Метод позволяет выявить отказы и сбои аппаратуры, если процессы **О** и **П** выполняются по одной и той же программе. Если эти процессы выполняются с помощью разных программ, то можно определить ошибки и в программах (см. п. 6.5.2). Основной недостаток прямого контроля – большая трата аппаратных средств (обычно его проводят на резервированной аппаратуре). Если же процессы **О** и **П**выполнять последовательно, то потребуется избыток времени, при этом будут выявлены только сбои аппаратуры.

Для некоторых задач, характеризуемых взаимно однозначным соответствием между исходными данными и результатами, эффективнее применять *обратный контроль.* При таком контроле параллельный процесс**П1**с исходными данными**У**ирезультатом **Х**осуществляют обратное преобразование контролирующего процесса **О**. Например, если осуществляется операция извлечения квадратного корня, то проще осуществить контроль возведением полученного результата в квадрат и сопоставлением его с исходной величиной. Недостаток обратного контроля, кроме ограниченности класса решаемых задач, расход дополнительного времени.

Иногда используется *смешанный контроль,*при котором как исходные данные, таки результаты подвергаютсянекоторым преобразованиям**П2**и**П3 ,**подобранных так, чтобы в случае безошибочных результатов процессов получались сопоставимые данные.

*По объекту контроля*различают контроль аппаратуры, программного обеспечения и человека-оператора.

В качестве оперативных методов контроля аппаратурыиспользуется **кодовый контроль**, который основан на том факте, что коды, циркулирующие в ЭВМ, во многих случаях могут быть разделены на допустимые (правильные) и неправильные. Так можно выявить недопустимые коды операций, адресов. Для несложных цифровых устройств применяется также *контроль по модулю*[5.1]:*числовой и цифровой*, которые позволяют выявить ошибки в работе аппаратуры преобразования данных, например, сумматоров и ошибки аппаратуры передачи и хранения данных (при этом методе на уровне интегральных схем вводится дополнительные схемы контроля).

1. **Программно-аппаратная диагностика**

Различают аппаратные и программные неисправности компьютерной техники.

Аппаратные неисправности – это неисправности устройств, комплектующих, механизмов и электронно-цифровых компонентов, а также отсутствие контактов в разъемах и сокетах и несовместимость комплектующих. К программным неисправностям относятся: ошибки в файловой системе, ошибки в ОС и программах, несовместимость приложений, ошибки и недоработки в драйверах и их несовместимость, и, конечно, вирусы и неправильное пользование программами.

Рассмотрим неисправности более подробно.

1 Компьютер не включается. В этом случае нужно: проверить подключение к сети, включен или отключен блок питания, подключения всех кабелей питания и кабелей от корпуса к материнской плате; проверить индикатором питания материнскую плату (может отсутствовать), если индикатор не горит, значит, материнская плата или блок питания неисправны; подключить другой блок питания и включить компьютер, если компьютер не включается, следовательно, материнская плата неисправна; подключить блок питания к стендовому компьютеру.

2 Компьютер включается, а на экране нет информации, но горят индикаторы, работает система охлаждения, у приводов работают двигатели. Нужно выполнить следующие действия: проверить индикатор монитора, правильность подключения кабеля монитора к видеокарте и кабеля питания; переподключить кабель к различным видеопортам; извлечь видеокарту при наличии интегрированной и снова включить компьютер; при наличии стендового компьютера проверить работоспособность видеокарты; установить работоспособную видеокарту и подключить дополнительное питание от блока питания, если нужно; при наличии только интегрированной видеокарты (встроенной в материнскую плату) это говорит о том, что материнская плата неисправна; посмотреть инструкцию по подключению видеокарты, может нужно проверить джамперы или заглушки; если при подключении работоспособной карты не появилось изображение, значить, материнская плата не исправна.

3 Компьютер включается и издает звук. Это означает, что в компьютере произошел конфликт или неработоспособность одного из устройств: материнской платы, процессора, видеокарты, оперативной памяти. Следует проверить правильность подключения комплектующих и дополнительного питания.

4 Не происходит загрузка ОС после UEFI. Рекомендуется проверить подключения клавиатуры, шлейфов и питания накопителей, установлен ли накопитель в UEFI или изменен его контроллер.

5 Комплектующие неисправны. Лучше проверить их отдельно на стендовом (работоспособном) компьютере.

6 Компьютер перезагружается или зависает. Причины: несовместимость отдельных комплектующих, тогда нужно проверить их характеристики; не хватать мощности блока питания или оперативной памяти недостаточно; перегрев комплектующих (южный мосты, процессор); комплектующие разогнаны, нужно проверить частоты процессора и видеоадаптера; неправильно установлены комплектующие; неправильно работает накопитель или оперативная память.

7 Компьютер перегрелся. Перегрев вызывают следующие причины: система охлаждения работает неправильно; не хватает мощности системы охлаждения; системный блок установлен неправильно, т.е. не хватает оборота воздуха (установлен близко к стене), или находится рядом с системой отопления; системный блок требует чистки от пыли; отсутствует термопаста; неправильно подобрана система охлаждения по мощности (например, мощность процессорного кулера меньше CPU) или по размеру (например, охлаждающая подставка не соответствует размерам ноутбука); неправильно установлены кулеры системного блока.

8 Произошла аппаратная несовместимость оборудования. Причины: старое оборудование конфликтует с новым – при обновлении драйверов, комплектующих произошел сбой. Нужно проверить работоспособность их и оборудования на стендовом компьютере и переустановить ОС.

Неисправности материнской платы. Неисправности материнской платы можно разделить на несколько групп.

1 Механические повреждения. Обычно они появляются при монтаже материнской платы. Иногда помогут исправить паяльные работы.

2 Выход из строя каких-либо компонентов материнской платы, например, сетевой, звуковой и видео карты (встроенные в материнскую карту). Можно установить новую плату расширения.

3.Повреждение, неправильная настройка или заводской брак UEFI.

4 Установка неправильных драйверов материнской платы или их отсутствие.

5 Неисправности или повреждение портов ввода-вывода.

6 Неисправности питания.

7 Недостаток охлаждения.

8 Ресурсные возможности иссякли: морально и физически устарела, износ.

Нужно обращать внимание на внешний вид плат расширения. Визуальный осмотр может сказать о многом, не только о механических повреждениях, но и о сгоревших элементов: дорожки, конденсаторы, транзисторы, мост, чип и т.д. Некоторые сгоревшие элементы можно заменить, припаять другой, а иногда нужно полностью поменять плату, так как невозможно отремонтировать.

Неполадки, связанные с оперативной памятью. Компьютер может не включаться по разным причинам, в том числе и по причине неисправности памяти. Если во время включения компьютер выдает серию звуковых сигналов, это может быть связано с неисправностью или отсутствием оперативной памяти. В руководстве к материнской плате и приложении есть расшифровка звуковых сообщений об ошибках.

Самопроизвольная перезагрузка компьютера чаще всего свидетельствует о неисправности блока питания, но может быть вызвана и неисправностью ОЗУ. Чаще всего по причине неисправности памяти компьютер зависает при выполнении таких сложных задач, как архивирование или обработка трехмерной графики и видео. В большинстве случаев зависание компьютера сопровождается появлением «синего экрана смерти».

Сбои и зависания компьютера в процессе установки ОС указывают на неисправности ОЗУ.

Появление сообщений об ошибке «Incorrect Memory Size» (неправильный объем памяти) происходит из-за неправильных установленных параметров системы в UEFI, а может, и из-за ошибки в самой работе памяти. Другое сообщение, например, «general RAM error», говорит об ошибки в оперативной памяти с указанным адресом. Вариант сообщения: «Memory address line failure at <XXXX>, read <YYYY>, expecting <ZZZZ>».

Система не может распознать или загрузит новый модуль памяти. Такое может произойти из-за несовместимости оборудования или старая версия BIOS не поддерживает новые модули оперативной памяти. Если установлены модули памяти разных производителей, то появляются сбои в оперативной памяти, а иногда и отказ в работе. После выхода из программы памяти, занимаемая ею, может не возвратиться в расположение ОС. Такая ситуация называется утечкой памяти. При перезагрузке компьютера работа памяти восстанавливается. При неисправности оперативной памяти нужно выполнить следующие действия: заменить оперативную память по параметрам конфигурации; проверить правильность определение модуля BIOS/UEFI, очисть оперативную память от загруженных программ, проверить характеристики памяти и её соответствие данной конфигурации программного обеспечения.

1. **Аппаратная диагностика**

Для поиска и прогр неисправностей в компьютере используются диагностические программы. Они позволяют проверить работу всей в системы и отдельных компонентов. Проверочные тесты предназначены для периодической проверки работоспособности системы и обнаружении неисправностей в процессе эксплуатации. Диагностические тесты служат не только для обнаружения ошибки, но и для локализации местных неисправностей. Для поиска и устран. Прогр. неисправностей можно использовать методы:

1. анализ сообщ. Об ошибках. Изучив жур. Событий можно вычислить причину сбоев и своевременно их устранить

2. использование специализированных утилей. С их помощью можно запустить низкоуровневое тестирование устройств

3. анализ информации из средств «просмотр событий». Для этого нужно открыть меню просмотр событий в разделе панель управления и раздел журналы виндовс для категории «приложением». Здесь хранятся сообщ которые возникали при работе установленного ПО.

4. использование цетра поддержки. Список отчётов об ошибках можно посмотреть через центр поддержки на панели задач. В отделе обслуживания нужно нажать кнопку отчёт о проблемах, если отсортировать список по источнику, то можно увидеть какие программы чаще всего дают сбои

Тестирование и диагностика сети

Под диагн подразумевают изменение характеристик и мониторинг показателей работы сети в процессе её эксплуатации без остановки работы пользователей

Диагн сети является измерения числа ошибок передачи данных степени загрузки её ресурсов или времени реакции прикладного ПО.

Тестирование — это процесс активного воздействия на сеть с целью проверки её работоспособ и определение потенциальных возможностей по передаче сетевого графика

Тест сети включает след этапы:

1 проверка кабелей каналов (целостность, расположение относительно источников помех и соответствия стандартов)

2 осмотр рабочих узлов (правильность прокладки, наличие маркировки)

3 тест коммутационного оборудования

Условно оборуд для диагностики, поиска неисправностей и сертификации можно поделить на:

1. приборы для сертификации кабельных систем

2. сетевые анализаторы

3. кабельные сканеры

4. тестеры

1. **Модели процесса сопровождения и планирование деятельности людей**

**Модели процесса сопровождения и планирования деятельности людей** включают:

1. **Модель жизненного цикла проекта**:
   * Планы для каждой стадии проекта (от старта до завершения).
   * Сопровождение включает мониторинг, анализ и корректировки на всех этапах.
2. **Модель ресурсного планирования**:
   * Определение нужных ресурсов (человеческих, материальных, временных) для выполнения задач.
   * Планирование их использования с учетом времени и нагрузки.
3. **Модель управления персоналом**:
   * Постоянное оценивание и развитие сотрудников.
   * Планирование обучения, повышение квалификации, карьерный рост.
4. **Модель коммуникаций**:
   * Определение каналов и методов коммуникации внутри команды.
   * Сопровождение процесса через регулярные совещания и отчеты.
5. **Модель прогнозирования и оценки рисков**:
   * Анализ возможных рисков в процессе работы.
   * Составление плана по минимизации негативных последствий для людей и ресурсов.

Каждая модель ориентирована на обеспечение эффективного функционирования команды и достижения целей проекта.

1. **Варианты сопровождения информационной системы: по заявкам, локальное обслуживание по договору, регламентированное сопровождение, аудит/консультирование, полное сопровождение**

Сопровождение информационной системы (ИС) может осуществляться разными способами в зависимости от потребностей организации и уровня поддержки. Вот краткое описание каждого варианта:

1. **По заявкам**
   * Реактивный подход: обслуживание только при возникновении проблем или необходимости изменений.
   * Преимущества: экономия на регулярных платежах.
   * Недостатки: возможные простои системы из-за времени отклика.
2. **Локальное обслуживание по договору**
   * Предусматривает фиксированный договор с определенным перечнем услуг (например, установка обновлений, исправление ошибок).
   * Преимущества: стабильность и понятность затрат.
   * Недостатки: ограниченный объем услуг.
3. **Регламентированное сопровождение**
   * Регулярное обслуживание по заранее утвержденному графику (мониторинг, профилактика, обновления).
   * Преимущества: минимизация рисков сбоев.
   * Недостатки: может быть дороже, чем обслуживание "по заявкам".
4. **Аудит/консультирование**
   * Анализ текущего состояния ИС с рекомендациями по улучшению и устранению недостатков.
   * Преимущества: независимая оценка, повышение качества системы.
   * Недостатки: не решает текущих проблем, только рекомендации.
5. **Полное сопровождение**
   * Комплексное обслуживание, включающее техническую поддержку, мониторинг, обновления, аудит и консультирование.
   * Преимущества: высокий уровень надежности и минимизация простоев.
   * Недостатки: наиболее затратный вариант.

Каждый из этих подходов может быть оптимальным в зависимости от требований бизнеса и критичности работы ИС

1. **Культура и этика программной инженерии.**

Культура и этика в программной инженерии формируют профессиональные нормы и ценности, направленные на обеспечение качества разработки, безопасности пользователей и соблюдение моральных принципов. Основные аспекты:

Культурная программа инженерии

1. **Коллективная работа**
   * Уважение к коллегам, умение работать в команде.
   * Обмен знаниями и поддержка совместного роста.
2. **Стремление к качеству**
   * Создание надежного, удобного и эффективного ПО.
   * Следование принципам разработки (например, SOLID, DRY).
3. **Инновации и обучение**
   * Постоянное изучение новых технологий и инструментов.
   * Готовность к изменениям и внедрению передовых решений.
4. **Четкая документация**
   * Поддержка прозрачности в процессе разработки.
   * Обеспечение доступности информации для коллег и заказчиков.

Этика программной инженерии

1. **Ответственность**
   * Учет возможных последствий работы программы на пользователей и общество.
   * Минимизация вреда и защита данных пользователей.
2. **Честность**
   * Корректное представление возможностей ПО заказчику.
   * Недопустимость обмана, плагиата и нарушения интеллектуальной собственности.
3. **Конфиденциальность**
   * Защита личной и корпоративной информации пользователей.
   * Соблюдение стандартов безопасности.
4. **Справедливость**
   * Недопущение дискриминации (например, через алгоритмы).
   * Обеспечение равного доступа к разработанным продуктам.
5. **Следование законам и стандартам**
   * Соблюдение международных норм, локальных законов и профессиональных стандартов.

Эти аспекты помогают не только повышать профессиональный уровень инженеров, но и укреплять доверие общества к информационным технологиям.

1. **Модели и характеристики качества**

**Модели качества**

1. **McCall**: функциональность, использование, поддержка.
2. **Boehm**: корректность, удобство, надежность, гибкость.
3. **ISO/IEC 25010**: функциональность, надежность, удобство, производительность, совместимость, безопасность, сопровождаемость, переносимость.

**Характеристики качества**

* **Функциональность**: соответствие задачам.
* **Надежность**: устойчивость к сбоям.
* **Удобство**: простота использования.
* **Производительность**: скорость и ресурсы.
* **Совместимость**: интеграция.
* **Безопасность**: защита данных.
* **Сопровождаемость**: легкость изменений.
* **Переносимость**: адаптация к платформам

1. **Верификация, тестирование и оценивание корректности компонентов**

**Верификация, тестирование и оценка корректности** — это процессы проверки качества и соответствия компонентов ПО требованиям.

Верификация

* Цель: Проверка соответствия продукта спецификациям (делаем правильно?).
* Методы:
  + **Анализ** (код-ревью, инспекции).
  + **Моделирование** (проверка архитектуры).

**Тестирование**

* Цель: Проверка работы продукта на реальных данных (делаем то, что нужно?).
* Типы:
  + **Модульное**: проверка отдельных компонентов.
  + **Интеграционное**: взаимодействие модулей.
  + **Системное**: тестирование всей системы.
  + **Приемочное**: проверка готовности для пользователя.

**Оценка корректности**

* Цель: Подтверждение, что программа работает правильно.
* Методы:
  + **Формальная проверка**: математический анализ.
  + **Динамическая проверка**: выполнение тестов.
  + **Статическая проверка**: анализ кода без запуска.

Все этапы работают вместе для повышения надежности и качества ПО.

1. **Требования к качеству программного обеспечения**

**Требования к качеству ПО**:

1. **Функциональность**: выполнение заданных функций.
2. **Надежность**: устойчивость к сбоям.
3. **Удобство использования**: простота и интуитивность.
4. **Производительность**: скорость работы, минимизация ресурсов.
5. **Совместимость**: взаимодействие с другими системами.
6. **Безопасность**: защита данных и предотвращение утечек.
7. **Сопровождаемость**: легкость внесения изменений.
8. **Переносимость**: адаптация к другим платформам.

Эти требования формируют основу качества ПО.

1. **Вневыборочная ошибка, ее состав. Виды вневыборочных ошибок**

**Вневыборочная ошибка** — ошибка, возникающая из-за внешних факторов, не связанных с выборкой.

Состав вневыборочной ошибки:

1. **Ошибки измерения** — погрешности в процессе сбора данных.
2. **Ошибки обработки** — ошибки при анализе данных.
3. **Ошибки выборки** — отсутствие некоторых групп в выборке.
4. **Невозможность контакта** — неудачные попытки связаться с участниками.
5. **Ошибки репрезентативности** — выборка не отражает популяцию.

**Виды вневыборочных ошибок**:

1. **Ошибка измерения** — погрешности инструментов.
2. **Ошибка категории** — неправильное определение категорий.
3. **Ошибка отклика** — неполные или игнорируемые ответы.
4. **Ошибка системы** — технические или логистические проблемы.
5. **Архитектура средств управления приложениями.**

**Архитектура средств управления приложениями** включает компоненты для эффективного управления жизненным циклом приложений:

1. **Интерфейс управления** — взаимодействие с пользователями через графические или командные интерфейсы.
2. **Средства автоматизации** — автоматическое развертывание и обновление приложений.
3. **Мониторинг и логирование** — отслеживание работы приложений и анализ логов.
4. **Управление конфигурацией** — централизованное управление настройками.
5. **Управление производительностью** — анализ и оптимизация работы приложений.
6. **Интеграция с другими системами** — взаимодействие с корпоративными сервисами.
7. **Безопасность** — защита данных, аутентификация и авторизация.

Эта архитектура помогает управлять, поддерживать и оптимизировать приложения.

1. **Комплекс мероприятий сопровождения информационной системы**

**Комплекс мероприятий сопровождения ИС** включает действия для поддержания ее работоспособности:

1. **Мониторинг и диагностика** — контроль работы системы и устранение проблем.
2. **Обновление и модернизация** — регулярные обновления и улучшения.
3. **Резервное копирование** — создание копий данных и восстановление при сбоях.
4. **Техническая поддержка** — помощь пользователям при возникновении проблем.
5. **Обеспечение безопасности** — защита от угроз и несанкционированного доступа.
6. **Документация и обучение** — создание документации и обучение пользователей.
7. **Аудит и оценка** — регулярные проверки для оптимизации работы системы.

Эти мероприятия обеспечивают стабильность и безопасность системы.

1. **Процедуры сохранения и восстановления информации как защитная мера от разрушения (потери) данных**

**Процедуры сохранения и восстановления информации**:

1. **Резервное копирование** — регулярное создание копий данных (полное, инкрементальное, дифференциальное).
2. **Хранение копий** — безопасное хранение бэкапов (внешние хранилища, облако).
3. **Мониторинг** — проверка успешности и целостности резервных копий.
4. **Восстановление** — восстановление данных из резервных копий после сбоя.
5. **Шифрование** — защита резервных копий от несанкционированного доступа.
6. **Политика хранения** — определение сроков хранения и удаления резервных копий.

Эти меры защищают данные и обеспечивают их восстановление в случае потери.

1. **Управление производительностью приложений**

**Управление производительностью приложений** включает:

1. **Мониторинг** — отслеживание показателей (время отклика, использование ресурсов).
2. **Оптимизация кода** — улучшение алгоритмов и структуры данных.
3. **Кеширование** — ускорение доступа к часто запрашиваемым данным.
4. **Масштабирование** — увеличение ресурсов для обработки большего объема запросов.
5. **Нагрузочное тестирование** — выявление узких мест при высокой нагрузке.
6. **Профилирование** — анализ производительности и «тяжелых» участков кода.
7. **Оптимизация базы данных** — индексация и улучшение запросов.

Эти меры обеспечивают высокую эффективность и стабильную работу приложений.