Теоретические вопросы к экзамену по МДК 06.02 Инженерно-техническая поддержка сопровождения информационных систем[1]

1. Задачи сопровождения информационной системы. Сценарий сопровождения

Сопровождение информационной системы включает в себя мероприятия, направленные на обеспечение ее работоспособности, актуальности и соответствия потребностям пользователей.
Основные задачи:

- **Исправление ошибок**: устранение багов, возникающих при работе системы.
- **Обновление функционала**: добавление новых возможностей или улучшение существующих.
- **Адаптация**: настройка системы под новые требования бизнеса, изменения в законодательстве или инфраструктуре.
- **Техническая поддержка**: помощь пользователям при возникновении сложностей.

Адаптивное сопровождение – это доработка программного продукта после поставки, позволяющее адаптировать его к новым условиям эксплуатации.
Полная модернизация – является наиболее дорогостоящим этапом в развитии ИС. Основным фактором, объясняющим стремление сменить функционирующую ИС является моральное устаревание системы.
Разработка дополнительного ПО предполагает доработку функционал под нужды персонала.
Корректирующее сопровождение (направлено на выявление и устранение несоответствий и ошибок после поставки программного продукта.)
Сопровождение данных включает: Контроль целостности данных
Поддержка актуальности данных
Резервное копирование и восстановление
Перенос данных в другую систему
Целостность информации определяется как состояние информации, при котором её изменение осуществляется только преднамеренно субъектами.
Сценарий сопровождения – это некоторая последовательность действий, иллюстрирующая поведение системы. Сценарии использования описывают то, кто и что может сделать с рассматриваемой системой. Методика сценариев использования применяется для выявления требований поведению системы.

2. Договор на сопровождение

Договор на сопровождение – это официальный документ, который описывает отношения между заказчиком и исполнителем по поддержке информационной системы.
Основные пункты договора:

1. **Предмет договора**: что именно сопровождается (система, ПО).
2. **Обязанности сторон**:
   - Исполнитель: устраняет ошибки, проводит обновления, консультирует.
   - Заказчик: предоставляет доступ к системе, своевременно оплачивает услуги.
3. **Уровень сервиса (SLA)**: сроки решения проблем (например, критическая ошибка – до 4 часов).
4. **Оплата**: фиксированная сумма или оплата по факту выполненных работ.
5. **Ответственность сторон**: за задержки, ошибки или неисполнение условий.

6. **Срок действия и расторжение**: сколько действует договор, как его расторгнуть.

- ответственность сторон
- конфиденциальность
- интеллектуальная собственность
- обстоятельства непреодолимой силы
- разрешение споров
- юридические адреса и банковские реквизиты сторон

3. Структура ИТ-сопровождения. Центр компетенции

Первая линия (пользовательская поддержка): операторы принимают обращения, фиксируют проблемы и передают сложные случаи выше.
Вторая линия (техническая поддержка): специалисты решают технические задачи, связанные с настройкой и мелким ремонтом системы.
Третья линия (разработчики или эксперты): работают над сложными ошибками, доработками и обновлениями системы.
Центр компетенции – это команда экспертов, которая:
Анализирует систему, предлагает улучшения.
Поддерживает разработку новых функций.
Обучает пользователей.
Контролирует внедрение обновлений.
Центр компетенции осуществляет:

- Функциональная поддержка
- Техническая поддержка
- Управление договорами и лицензиями
- Аудит (оценка, проверка) и масштабируемость системы.

4. Анализ исходных программ и компонентов программного средства. Принцип модульности процессов и продуктов

Анализ исходных программ включает:

1. **Проверка структуры кода**: соответствует ли код стандартам.
2. **Выявление уязвимостей**: ищутся ошибки, влияющие на безопасность.
3. **Оптимизация**: анализ на избыточность и производительность.
4. **Совместимость**: насколько код подходит для работы с другими компонентами.

Принцип модульности:

- Программа делится на независимые модули.
- Каждый модуль выполняет отдельную функцию (например, авторизация, обработка данных).
- Модули взаимодействуют через четко определенные интерфейсы.

Преимущества модульности:

- Легче дорабатывать и тестировать.
- Снижается вероятность ошибок в одной части системы, влияющих на другие.
- Проще внедрять изменения.

5. Программная инженерия. Методы программной инженерии

Программная инженерия – это системный подход к разработке, эксплуатации и сопровождению программного обеспечения (ПО).
Основные методы:

1. **Каскадная модель (водопад)**: этапы разработки выполняются последовательно (анализ, проектирование, реализация, тестирование, внедрение). Подходит для простых проектов.
2. **Гибкие методологии (Agile)**: разработка идет небольшими итерациями, активно учитываются пожелания заказчика. Пример – Scrum.
3. **Инкрементная разработка**: ПО создается по частям, каждая из которых проходит полный цикл разработки.
4. **Экстремальное программирование (XP)**: упор на простоту, тесное сотрудничество с заказчиком и частые релизы.
5. **DevOps**: объединяет процессы разработки и эксплуатации, акцент на автоматизацию и непрерывное обновление.

6. Классификация программ

Программы можно классифицировать по различным критериям, но вот основные типы:

- **Системные программы**: Это основа для функционирования компьютера. Сюда относятся операционные системы (Windows, Linux, macOS), системные утилиты (антивирусы, драйверы устройств) и сервисы, которые обеспечивают базовые функции компьютера.
- **Прикладные программы**: Используются для выполнения конкретных задач пользователя. Примерами являются офисные приложения (Microsoft Office), браузеры (Chrome, Firefox), графические редакторы (Photoshop) и множество других специализированных программ.
- **Инструментальные программы**: Созданы для разработки и обслуживания других программ. Это компиляторы, интерпретаторы, отладчики, среды разработки (IDE как Visual Studio или Eclipse).

- **Первый класс** оставляют относительно небольшие программы, создаваемые для получения конкретных результатов автоматизации или для анализа относительно простых процессов самими разработчиками программ

- **Второй класс** составляют крупномасштабные комплексы программ для сложных систем управления и обработки информации, оформляемые в виде программных продуктов.

7. Оценка качества программных средств

Качество программного обеспечения оценивается по нескольким ключевым аспектам:

- **Функциональность**: Насколько хорошо программа выполняет заявленные функции. Тестирование на соответствие требованиям.
- **Производительность**: Скорость работы, использование ресурсов (памяти, процессора), оптимизация кода.
- **Надежность**: Стабильность работы, низкая вероятность сбоев, хорошая обработка ошибок.
- **Удобство использования (юзабилити)**: Интуитивно понятный интерфейс, простота использования, качественная документация.
- **Поддержка и обновления**: Регулярные обновления, исправление багов, добавление новых функций, техподдержка.

Критерии качества ПО:
- надёжность ПО
- сопровождение
- удобство применения

- эффективность
- универсальность
- корректность

6. **Функциональная пригодность (функциональность)** в модели качества продукта относится к способности продукта выполнять заданные функции, которые соответствует установленным требованиям и ожиданиям пользователей.
7. **Уровень производительности** в модели качества продукта выполнять свои функции эффективно и с удовлетворительными временными и ресурсными характеристиками: скорость, эффективность, использования ресурсов, нагрузочная способность, масштабируемость, время безотказной работы.
8. **Совместимость** в модели качества продукта относится к способности продукта работать и взаимодействовать с другими системами, приложениями, устройствами и средами: аппаратная и программная совместимость.
9. **Удобство использования** является критическим аспектом модели качества продукта, который определяет, насколько легко и приятно пользователю взаимодействовать с продуктом.
10. **Надежность** в модели качества продукта относится к способности продукта функционировать корректно и стабильно в течении заданного периода времени и при определенных условиях использования (отказоустойчивость, устойчивость к сбоям, поддержка целостности данных, валидация)
11. **Защищенность** в модели качества продукта относится к способности продукта защищать данные и информацию пользователей от несанкционированного доступа, потери или повреждения.
12. **Сопровождаемость** в модели качества продукта относится к легкости, с которой продукт может быть поддерживаем, обновляем и модифицирован после его запуска.
13**. Переносимость** в модели качества продукта относится к способности продукта функционировать в различных средах, системах или на различных платформах с минимальными изменениями или усилиями.

8. Реинжиниринг. Цель и задачи реинжиниринга. Этапы реинжиниринга ПО

Реинжиниринг – повторная реализация наследуемой системы с целью повышения удобства её эксплуатации и сопровождения (функциональность и архитектура не изменяется)
Детальная оценка и перестройка ПО для формирования понимания, воссоздания (на уровне моделей или требований) и дальнейшей реализации его функциональности в новой форме.
Реинжиниринг подразделения (организационные изменения)
Виды реинжиниринга ИС:
 Модернизация ИС (прямой инжиниринг)
 Редокументирование
 Рефакторинг ИС
 Редизайн ИС
 Реверс – инжиниринг
 Трансляция исходного кода
 Реинжиниринг бизнес – процессов (переориентация)
Причины реинжиниринга:
- моральное устаревание ИС:
Более эффективных ИТ
Новых способов организации пользовательского интерфейса

Новых решений в области архитектуры ИС
Вычислительных устройств с более высокой производительностью
Новых носителей информации
- физическое устаревание
Физический износ используемого аппаратного обеспечения (снижение надёжности, увеличением количества сбоев)
Ухудшение характеристик производительности аппаратного обеспечения.
-причины организационного характера
Основные пути реинжиниринга ИС:
Создание новой ИС взамен существующей
- модификация существующей ИС
Три идеологии реинжиниринга:
- западная (проведения реинжиниринга по необходимости)
- восточная (профилактическое проведение реинжиниринга)
- отечественная
Этапы реинжиниринга:
1) Формирование команды реинжиниринга
2) Сбор претензий к системе
3) Создание спецификации требований к системе
4) Актуализация структурных моделей системы
5) Генерация альтернатив реинжиниринга системы
6) Выбор оптимальной альтернативы
7) Реализация выбранной альтернативы

9. Система резервного копирования. Полное, выборочное, разностное добавочное (инкрементное) резервирование

Резервное копирование данных – процесс создания копии данных на носителе, предназначенном для восстановления данных в оригинальном месте их расположения в случае их повреждения или разрушения.

Классификация резервного копирования

1. По полноте сохраняемой информации

a. **Полное резервирование** (Full backup) – создание резервного архива всех системных файлов, обычно включающего состояние системы, реестр и другую информацию
b. **Выборочное резервирование** (Selective backup) – создание резервного архива только из отобранных файлов.
c. **Разностное резервирование** (Differential backup) – тип резервного копирования файлов, при котором копируются не все исходные файлы, а только новые и измененные с момента создания предыдущей полной копии.

2. **По способу доступа к носителю**

Оперативное резервирование (online backup) копирование журналов повторов без отключения пользователей от БД.
Автономное резервирование (offline backup) аналогично полному, отключение

- **Инкрементное резервирование**: Копируются только изменения с момента последнего резервирования (будь то полное, разностное или инкрементное). Восстановление требует всех инкрементных копий, сделанных после последнего полного резервирования.

10. Децентрализованная схема резервного копирования

Её суть в том, что на каждом сервере и рабочей станции может быть собственное ПО для резервного копирования, работающее независимо от других узлов сети. Все данные выгружаются на какой-либо общий сетевой ресурс, откуда потом попадают в архив или восстанавливаются, при необходимости.

В этой схеме данные резервируются на различные устройства или серверы, не зависящие от одного центрального хранилища. Это может включать:

- **Распределенные сети**, где каждый участник хранит части данных других участников.
- **Облачные сервисы**, где данные хранятся в разных географических локациях для обеспечения доступности и безопасности.
- **Блокчейн-технологии**, использующие распределенные системы для хранения данных с высоким уровнем защиты и шифрования.

Такой подход уменьшает риски потери данных из-за одной точки отказа, повышает конфиденциальность и может обеспечить более быстрое восстановление.

Достоинства схемы в том, что она чрезвычайно простая, легко реализуется и обычно не требует дополнительного ПО, копирование выполняется штатными средствами операционной системы или СУБД. Есть и недостатки – сложно установить общую политику резервного копирования и защиты информации, общее для всех программ расписание бэкапов, настраивать и мониторить деятельность каждой из программ придется отдельно, что усложняет администрирование

11. Централизованная схема резервного копирования

для ее реализации необходимо специализированное клиент-серверное ПО. Серверная часть устанавливается на сервер резервного копирования и централизованно управляет установленными у пользователей программными агентами, которые собирают, копируют информацию о системе или восстанавливают ее из копии. В таком варианте легко настраивать общие политики создания резервных копий, расписание бэкапов, все участники могут работать согласно с общей для компании инструкцией по резервному копированию информации;

В централизованной схеме резервного копирования все данные копируются на один или несколько центральных серверов или хранилищ. Это может быть:

Сервер резервного копирования, к которому подключаются все рабочие станции и серверы для выполнения резервирования.
Центральное хранилище данных (например, NAS - Network Attached Storage или SAN - Storage Area Network), где данные хранятся в одном месте.

Преимущества включают централизованное управление, упрощённое восстановление и экономию на ресурсах, но существует риск потери всех данных при сбое центрального хранилища.

12. Технологии резервного копирования (LAN, SAN и другие)

- **LAN (Local Area Network)**: Резервное копирование через локальную сеть. Данные передаются по сети на сервер резервного копирования. Это удобно для офисных сетей, но скорость может быть ограничена пропускной способностью сети.
- **SAN (Storage Area Network)**: Выделенная сеть для хранения данных, обеспечивающая высокую скорость и пропускную способность. SAN часто используется для резервирования больших объемов данных благодаря своей производительности и надежности.
- **NAS (Network Attached Storage)**: Устройство, подключенное к сети, предоставляющее файловое хранилище. Подходит для резервирования, так как позволяет легко расширить объем хранилища.
- **Облачное резервное копирование**: Использование удалённых серверов для хранения данных. Предоставляет гибкость, масштабируемость и доступность данных из любой точки мира, но зависит от интернет-соединения.

13. Причины разрушения информации

 Физические повреждения: Поломка носителей данных, воздействие воды, огня, магнитных полей.
 Логические ошибки: вирусы, вредоносное ПО.
 Человеческий фактор: Ошибки при работе с данными, случайное удаление, неправильное управление.
 Износ оборудования: Со временем носители могут деградировать.
 Сбои системы: Программные ошибки, сбои в работе серверов или баз данных.

14. Сохранение информации

Сохранение информации – это процедура получения резервной копии с целью её последующего использования при ликвидации возможных разрушений информации.

- **Регулярное резервное копирование**: Создание копий данных на регулярной основе.
- **Использование различных носителей**: Комбинация жестких дисков, SSD, лент, облачного хранилища для защиты от различных угроз.
- **Шифрование данных**: Защита от несанкционированного доступа.
- **Правила 3-2-1**: 3 копии данных, на 2 разных типах носителей, 1 из которых хранится вне основного места хранения.
- **Автоматизация**: Использование специального ПО для автоматического резервирования.

15. Восстановление информации

Восстановление информации – это процедура ликвидации разрушений данных с использованием сохранённой информации на некоторый момент времени (копии) и возможной корректуры с момента создания копии.

- **Определение проблемы**: Анализ причин потери данных и выбор стратегии восстановления.
- **Использование резервных копий**: Восстановление с последней доступной резервной копии. Важно иметь актуальные копии.
- **Средства восстановления данных**: Программное обеспечение для восстановления данных с поврежденных носителей, иногда с участием специалистов.
- **Проверка целостности**: После восстановления необходимо проверить, что все данные восстановлены корректно и функциональны.

- **План восстановления**: Создание и регулярное обновление плана восстановления данных, чтобы минимизировать время простоя.

16. Идентификация и аутентификация

Идентификация - присвоение субъектам и объектам доступа личного идентификатора и сравнение его с заданным.
Аутентификация - проверка принадлежности субъекту доступа предъявленного им идентификатора и подтверждение его подлинности. Аутентификация заключается в проверке: является ли подключающийся субъект тем, за кого он себя выдает.
Авторизация - это предоставление доступа к данным, закрытым от публичного просмотра.
В качестве идентификаторов обычно используют набор символов, который пользователь запоминает или для их запоминания использует специальные средства хранения (эл. ключи)
Используют также физиологические параметры человека или особенности поведения.
Парольные методы аутентификации по степени изменяемости паролей делятся на:
методы, использующие постоянные (многократно используемые) пароли;
методы, использующие одноразовые (динамично изменяющиеся) пароли.

Идентификация: Это процесс, при котором пользователь заявляет о своей личности, обычно предоставляя уникальный идентификатор, например, логин или имя пользователя. Это первая ступень доступа к системе или ресурсу.
Аутентификация: Следующий шаг после идентификации, где система проверяет, действительно ли пользователь является тем, за кого себя выдаёт. Это может быть выполнено через:
Что вы знаете? (пароли, ПИН-коды)
Что вы имеете? (смарт-карты, USB-токены)
Кем вы являетесь? (биометрия: отпечатки пальцев, распознавание лица)
Что вы делаете? (узоры движения, анализ поведения)

17. Криптография и шифрование. Симметричное и асимметричное шифрование

- **Криптография**: Наука о методах защиты информации посредством кодирования данных для обеспечения конфиденциальности, целостности и аутентичности.

Шифрование - процесс кодирования информации с целью предотвращения несанкционированного доступа.
В симметричных методах для шифрования и расшифрования используется один и тот же секретный ключ.
Ассиметричные методы используют два взаимосвязанных ключа: для шифрования и расшифрования.
Для контроля целостности передаваемых по сетям данных используется эл. цифровая подпись, которая реализуется по методу шифрования с открытым ключом.

- **Симметричное шифрование**: Используется один и тот же секретный ключ для шифрования и дешифрования данных. Примеры алгоритмов: AES, DES. Преимущество в скорости, но ключ необходимо безопасно передавать между сторонами.

- **Асимметричное шифрование**: Включает использование пары ключей - открытого (публичного) и закрытого (приватного). Данные, зашифрованные с использованием публичного ключа, могут быть расшифрованы только соответствующим приватным ключом. Примеры: RSA, ECC. Обеспечивает безопасную передачу данных без предварительного обмена ключами, но процесс медленнее, чем при симметричном шифровании.

18. Методы разграничение доступа

После выполнения идентификации и аутентификации подсистема защиты устанавливает полномочия (совокупность прав) субъекта для последующего контроля санкционированного использования объектов информационной системы.

 Дискреционный контроль доступа (DAC): Владелец ресурса контролирует, кто и как может получить к нему доступ.

 Обязательный контроль доступа (MAC): Доступ определяется на основе политик безопасности, часто используется в системах с высоким уровнем защиты (например, военные системы), где уровень доступа зависит от уровня секретности.

 Контроль доступа на основе ролей (RBAC): Доступ предоставляется на основе роли пользователя в системе, что упрощает управление правами.

 Атрибутный контроль доступа (ABAC): Принятие решения о доступе основано на атрибутах пользователя, ресурса и окружения.

19. Регистрация как механизм обеспечения защищенности ИС

Регистрация является еще одним механизмом обеспечения защищенности информационной системы. Этот механизм основан на подотчетности системы обеспечения безопасности, фиксирует все события, касающиеся безопасности.

вход и выход субъектов доступа;
запуск и завершение программ;
выдача печатных документов;
попытки доступа к защищаемым ресурсам;
изменение полномочий доступа;

 Фиксация событий: Запись всех действий, выполняемых пользователями или системами, чтобы можно было отследить любые аномалии или нарушения безопасности.

 Логирование: Создание логов, которые используются для анализа активности, аудита и расследования инцидентов безопасности.

 Управление учетными записями: Регистрация новых пользователей с присвоением им уникальных идентификаторов и паролей, обеспечение корректного удаления учетных записей.

20. Аудит информационной системы

Аудит - это анализ накопленной информации, проводимый оперативно в реальном времени или периодически (например, раз в день). Оперативный аудит с автоматическим реагированием на выявленные нештатные ситуации называется активным.

Позволяет:
обеспечить подотчетность пользователей;

реконструировать последовательность событий;
обнаружить попытки нарушения ИБ;
предоставить информацию для выявления и анализа проблем.
Статистические методы аудита основаны на накоплении среднестатистических параметров функционирования подсистем и сравнении текущих параметров с ними.
Эвристические методы аудита используют модели сценариев несанкционированных действий, которые описываются логическими правилами или модели действий, по совокупности приводящие к несанкционированным действиям.

- **Цель аудита**: Проверка соответствия информационной системы политикам, стандартам и регуляциям безопасности, оценка эффективности мер безопасности и обнаружение уязвимостей.
- **Типы аудита**: Может быть внутренним (проводится сотрудниками компании) или внешним (независимыми аудиторами).

Аудит обеспечивает прозрачность и подотчетность в управлении информационной безопасностью, способствует улучшению систем и поддержанию доверия.

21. Межсетевое экранирование

Экранирование. Одним из эффективных механизмом обеспечения информационной безопасности распределенных вычислительных сетях является экранирование. **Межсетевое экранирование повышает** безопасность объектов внутренней сети за счет игнорирования неавторизованных запросов из внешней среды.
Межсетевое экранирование (файервол или брандмауэр) – это система безопасности, которая контролирует и фильтрует входящий и исходящий трафик между компьютерными сетями, обычно между частной сетью и Интернетом, основываясь на предварительно установленных правилах безопасности. Основные функции:

- **Фильтрация пакетов**: Блокировка или разрешение передачи данных на основе таких параметров, как IP-адреса, порты, протоколы.
- **Прокси-серверы**: Обеспечение дополнительного уровня контроля и безопасности через посредничество в передаче данных.
- **Состоящая фильтрация**: Анализ текущего состояния соединений для принятия решений о пропуске трафика.

Брандмауэр – это средство защиты, которое может быть программой или аппаратным устройством. Его функции включают фильтрацию входящего и исходящего трафика на основе заданных правил, предотвращение несанкционированного доступа к сети, защиту от вредоносных атак и обеспечение безопасности данных.

22. Технология виртуальных частных сетей (VPN)

Туннелирование. Для передачи данных VPN-агенты создают виртуальные каналы между защищаемыми локальными сетями или компьютерами

VPN – это технология, которая предоставляет защищённый туннель для передачи данных через публичную сеть, например, Интернет. Основные аспекты:

- **Шифрование данных**: Информация шифруется перед передачей, что предотвращает перехват данных.
- **Маскировка IP-адреса**: Пользователь может казаться подключённым из другого места, что обеспечивает анонимность и доступ к геоблокированному контенту.

- **Аутентификация**: Убеждается, что только авторизованные пользователи могут подключиться к VPN.

23. Сбои информационных систем. Происхождение (генезис) ошибок в ИС

Сбои в информационных системах могут возникать из-за:

- **Аппаратных неисправностей**: Поломка оборудования, износ компонентов.
- **Программных ошибок**: Баги в коде, несовместимость программного обеспечения.
- **Человеческого фактора**: Неправильное использование, ошибки администрирования, вредоносные действия.
- **Внешних факторов**: Атака вирусов, кибератаки, физические повреждения от стихийных бедствий.

• Природные явления – нарушение работоспособности системы без вмешательства человека;
• Атаки – вид умышленных угроз со стороны «чужих» (повреждение оборудования, коммуникации);
• Фальсификация – умышленная угроза со стороны «своих» (ввод ложных данных);
• Злоумышленное кодирование – нелегальные программы и фрагменты, выполняемые на системных компьютерах (вирусы, трояны).
• Шпионаж – получение информации, не подлежащей огласке;
• Взлом и хакерство – проникновение в компьютерную систему или программу путём разгадывания или расшифровки кодов доступа, номеров счетов, паролей и файлов;
• Мошенничество – использование ресурсов путем умышленного обмана.

24. Организация сбора данных об ошибках. Анализ ошибок в ИС. Отчеты об ошибках системы. Обзор программ-сборщиков отчетов об ошибках

Источниками данных об ошибках (или то, как мы узнаём об ошибках могут быть как пользователи ПО, так и средства мониторинга вычислительных систем.

- **Сбор данных об ошибках**: Включает автоматическое логирование событий, создание дампов памяти при сбоях, использование инструментов мониторинга.
- **Анализ ошибок**: Определение причин и последствий ошибок, их классификация, поиск шаблонов для предотвращения повторений.
- **Отчеты об ошибках**:
  - **Внутренние отчеты**: Помогают разработчикам и администраторам улучшать систему.
  - **Отчеты для пользователей**: Информируют о временном недоступности сервисов и планируемых действиях.
- **Программы-сборщики отчетов об ошибках**:
  - **Windows Error Reporting (WER)**: Собирает данные о сбоях в Windows.
  - **Sentry**: Используется для мониторинга и сбора информации об ошибках в веб-приложениях.
  - **Crashlytics** (Google): Обеспечивает анализ аварийных ситуаций в мобильных приложениях.

Отчёт об ошибке (англ. Error report или crash report) – это файл, содержащий техническую информацию об исключительной ситуации (исключении), произошедшей в программе на компьютере пользователя.
Отчёт об ошибке обычно создаётся специальной программой (англ. crash reporter). Целью такой программы является сбор данных о произошедшем крэше и отправка этих данных по

сети Интернет некой третьей стороне, обычно этой третьей стороной является производитель ПО.

25. Производительность приложений

Производительность приложений зависит от многих взаимосвязанных факторов, таких как:

- Системные ресурсы
- Архитектура баз данных и приложений
- Эффективность программного кода
- Сетевая инфраструктура.

Производительность приложений включает:

- **Отзывчивость**: Время отклика на действия пользователя.
- **Скорость обработки**: Эффективность выполнения задач, включая обработку данных и выполнение запросов.
- **Использование ресурсов**: Оптимальное потребление CPU, памяти, сетевых ресурсов.
- **Масштабируемость**: Способность приложения справляться с увеличением нагрузки без значительной потери производительности.
- **Методы улучшения производительности**:
  - **Оптимизация кода**: Удаление узких мест, кэширование, асинхронные операции.
  - **Использование профилировщиков**: Инструменты для анализа работы приложения и выявления проблем.
  - **Мониторинг и тестирование нагрузки**: Для предсказания поведения при различных условиях нагрузки.

26. Оптимизация приложений

Оптимизация приложения – это процесс модификации программной системы с целью повышения эффективности ее работы или использования меньшего количества ресурсов. Оптимизация приложений направлена на улучшение их работы в различных аспектах:

- **Скорость выполнения**: Снижение времени отклика через оптимизацию алгоритмов, использование кэширования.
- **Эффективное использование ресурсов**: Оптимизация потребления памяти, процессорного времени, сетевых ресурсов.
- **Масштабируемость**: Обеспечение возможности роста приложения без потери производительности.
- **Оптимизация базы данных**: Улучшение запросов, индексация, нормализация данных.
- **Методы оптимизации**:
  - **Профилирование кода** для выявления узких мест.
  - **Рефакторинг** для улучшения структуры кода.
  - **Использование асинхронных операций** и многопоточности.
  - **Компрессия и минификация** для веб-приложений.

27. Тестирование приложений. Методы тестирования

Тестирование программ и систем – это способ проверки программы, который заключается в обработке программой последовательности разнообразных контрольных наборов тестов с известными результатами.

Тест – это совокупность входных данных и/или действий пользователя с указанием ожидаемых результатов или соответствующих реакций программы

Тестовые данные – особый тип данных, которые предназначены для проверки работы системы, создаются по-разному: генератором тестовых данных, проектной группой на основе документов или файлов, пользователем из спецификации требований и т.д

Тестирование приложений – это процесс проверки соответствия продукта требованиям и ожиданиям пользователей. Основные методы:

- **Функциональное тестирование**: Проверка на соответствие функциональным требованиям.
  - **Модульное тестирование**: Проверка отдельных компонентов.
  - **Интеграционное тестирование**: Проверка взаимодействия компонентов.
  - **Системное тестирование**: Оценка всей системы в целом.
- **Нефункциональное тестирование**:
  - **Производительность**: Тесты на нагрузку, стресс-тесты.
  - **Безопасность**: Поиск уязвимостей, тестирование защиты.
  - **Удобство использования (Usability)**: Тестирование интерфейса и опыта пользователя.
- **Автоматизация тестирования**: Использование инструментов для автоматического выполнения тестов, ускоряя процесс и повышая охват.

Приемочное тестирование - формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приемки.
Альфа-тестирование - моделируемое или действительное эксплуатационное тестирование потенциальными пользователями / заказчиками в качестве внутреннего приемочного тестирования.
Бета-тестирование - эксплуатационное тестирование потенциальными и/или существующими клиентами/заказчиками на внешней стороне с целью получить отзывы рынка.
Интеграционные тесты - тесты, которые проверяют интеграцию различных частей системы. Они позволяют убедиться, что все компоненты системы запущены, каналы работоспособны, форматы взаимодействия соответствует спецификациям.
Компонентное интеграционное тестирование проверяет взаимодействие между программными компонентами и производится после компонентного тестирования.
Модульные тесты - тесты которые проверяют поведение отдельного класса (или несколько тесно взаимосвязанных классов). Они не взаимодействуют с какими-либо внешними интерфейсами.
Компонентное тестирование занимается поиском дефектов и верификацией функционирования программных модулей, программ, объектов, классов и т.п., которые можно протестировать изолированно

28. Верификация и валидация информационных систем

Верификация означает проверку того, что ПО разработано в соответствии со всеми требованиями к нему.
Валидация – проверка того, что сам продукт правилен, т.е подтверждение того, что он действительно удовлетворяет потребностям и ожиданиям пользователей, заказчиков и других заинтересованных сторон.

Верификация в тестировании ПО – процесс просмотра документации, дизайна, кода и программы для того, чтобы проверить, было ли программное обеспечение создано в соответствии с требованиями или нет. Основная цель процесса верификации – обеспечить качество приложения, дизайна, архитектуры и т.д. Процесс верификации включает в себя такие действия, как ревью, пошаговое руководство и инспекция.

Валидация в разработке ПО – динамический механизм тестирования и проверки того, действительно ли программный продукт соответствует точным потребностям заказчика или нет. Этот процесс помогает гарантировать, что ПО выполняет желаемое использование в подходящей среде. Процесс валидации включает в себя такие действия, как модульное тестирование, интеграционное тестирование, системное тестирование и пользовательское приемочное тестирование.

29. Пользовательская документация ПО

Документация входит в состав проекта по созданию, внедрению, сопровождению, модернизации и ликвидации ИС на протяжении полного жизненного цикла этой ИС.

К такой документации относятся документы, которыми должен руководствоваться пользователь при инсталляции ПО, при применении ПО для решения своих задач и при управлении ПО.

Эксплуатационные документы содержат сведения необходимые для обеспечения функционирования и эксплуатации системы.

Пользовательская документация предназначена для конечных пользователей программного обеспечения и включает:

- **Руководство пользователя**: Объясняет, как использовать функции программы, включает скриншоты, пошаговые инструкции, FAQ.
- **Справочное руководство**: Более детальное описание функций, может включать справочные таблицы, описание команд.
- **Видеоуроки**: Особенно полезны для обучения сложным операциям или интерфейсу.
-

Документация должна быть ясной, доступной и регулярно обновляться по мере развития ПО.

30. Документация администрирования ИС. Руководство системного администратора. Руководство программиста

- **Документация администрирования ИС**:
  - **Руководство системного администратора**: Включает инструкции по установке, настройке, обслуживанию и мониторингу системы, управлению пользователями, резервному копированию, обновлению ПО.
  - **Руководства по безопасности**: Описание политик безопасности, настройки firewall, аудит и мониторинг безопасности.
- **Руководство программиста**: Предназначено для разработчиков, работающих с системой:
  - **API документация**: Описание интерфейсов программирования для взаимодействия с системой.
  - **Архитектура системы**: Объяснение структуры, компонентов, взаимодействия между ними.
  - **Стандарты кодирования**: Нормы и правила написания кода для поддержания консистентности.
  - **Инструменты и окружение разработки**: Описание используемых средств разработки, инструментов тестирования, процессов деплоя.

РУКОВОДСТВО СИСТЕМНОГО ПРОГРАММИСТА (АДМИНИСТРАТОРА)

- общие сведения о ПК, реализующем функции сбора и обработки государственной статистической отчетности по форме 1-ду;
- структура ПК и условия его применения;
- установка ПК: установка серверной и клиентской частей;
- обновление версий ПК;
- настройка программ ПК;
- управление пользователями ПК;
- рекомендации по защите и сохранности баз данных.

Руководство программиста.

Руководство программиста предназначено для разработчиков программного обеспечения и специалистов, которые будут его сопровождать.

Это руководство в качестве основных документов включает:

1) задание на разработку программного обеспечения (техническое задание);
2) спецификацию;
3) прокомментированные исходные тексты (листинги) модулей программы и управляющего модуля;