**Вопросы к экзамену по ПМ 02 «Осуществление интеграции программных модулей»**

• **Дайте определение понятия репозитория проекта. Опишите классы уровней репозиториев.**

Репозиторий – хранилище информации (место где мы храним наш код проекта и всю информацию по файлам, их изменения), связанной с проектом разработки программного продукта в течение всего его жизненного цикла.

Классы:

1 Уровень.

Репозиторий - обеспечивает логическое деление на группы проектов. Пользователь может выбирать репозитории из меню “Репозиторий/выбрать репозиторий”.

2 Уровень.

Проекты - содержат всю информацию в рамках отдельных разработок.

3 Уровень.

Конфигурации. Конфигурации являются удобным инструментом для управления версиями выпускаемых моделей или ПО и тому подобных задач.

4 Уровень.

Стадии. Служат для управления жизненным циклом проекта.

5 Уровень.

Системы. Система является самым нижним уровнем в иерархии проекта и содержит информацию непосредственно о модели: диаграммы в нотации UML, и другие объекты.

• **Расскажите об инструментарии анализа качества программных продуктов в среде разработки**

дебагер

• **Дайте определение понятия структура проекта. Назовите основные задачи структуризации.**

Сущность структуризации проекта заключается в разбивке проекта на иерархические подсистемы и компоненты, которые необходимы для того, чтобы проектом можно было управлять.

Основные задачи структуризации проекта:

· разбивка проекта на поддающиеся управлению блоки;

· распределение ответственности за различные элементы проекта и увязка работ со структурой организации (ресурсами);

· точная оценка необходимых затрат - средств, времени и материальных ресурсов;

· создание единой базы для планирования, составления смет и контроля за затратами;

· увязка работ по проекту с системой ведения бухгалтерских счетов в компании;

· переход от общих, не всегда конкретно выражаемых целей к определенным заданиям, выполняемым подразделениями компании;

· определение комплексов работ (подрядов).

• **Дайте определение свойств качественного программного обеспечения: мобильность, полезность, машино-независимость. Поясните их назначение.**

Мобильность — это способность ПП быть перенесенным из одной вычислительной среды (окружения) в другую, в частности, с одной ЭВМ на другую (применяют термин “перенос с одной платформы на другую”

Полезность - удобность практического применения.

Машино-независимость - программа может выполняться на вычислительной машине иной конфигурации, чем та, для которой они непосредственно предназначены.

• **Дайте определение понятия структура проекта. опишите виды и классификацию проектов.**

Сущность структуризации проекта заключается в разбивке проекта на иерархические подсистемы и компоненты, которые необходимы для того, чтобы проектом можно было управлять.

(Возможно)

Веб проекты, десктопные приложения, мобилки…

• **Дайте определение свойств качественного программного обеспечения: точность, доступность, модифицируемость. Поясните их назначение**

Точность - Программный продукт обладает свойством точности, если выдаваемые им результаты имеют точность, достаточную с точки зрения основного их назначения.

Доступность - Программный продукт обладает свойством доступности, если он допускает селективное использование отдельных его компонент.

Модифицируемость – совокупность свойств программного средства, характеризующая усилия, необходимые для внесения в него изменений

• **Сформулируйте определение интеграции программных модулей. Опишите виды и цели интеграции программных модулей.**

Интеграцией называется элемент процесса разработки программного обеспечения, в ходе которого отдельные компоненты программного продукта объединяются в единое целое

Цели интеграции

- уменьшить стоимость эксплуатации совокупности приложений предприятия;

- увеличить скорость выполнения типичных задач или гарантировать сроки их выполнения;

- поднять качество выполнения задач за счет формализации процессов и минимизации человеческого фактора, как основного источника ошибок.

• **Перечислите и охарактеризуйте виды тестирования производительности.**

1. Нагрузочное тестирование — основной вид, используется для измерения различных параметров сервера при подаче нагрузки от нагрузочного инструмента, имитирующего работу клиентов.

2. Стрессовое тестирование — в этом виде требуется нагрузить систему по максимуму ресурсов, а затем снять нагрузку и наблюдать — сможет ли восстановиться эта система.

3. Тестирование на большом количестве данных — либо заранее, либо в процессе тестов требуется сгенерировать настолько много данных, насколько хватит ресурсов, а затем снимать метрики. В какой-то степени — подвид предыдущих тестов, но обычно их разделяют.

4. Тестирование стабильности — самое долгое, ставит перед собой цель — выявить различные утечки ресурсов и деградацию производительности при средних нагрузках, но за большее количество времени.

5. Тесты на масштабируемость — нужны, чтобы понять, будет ли увеличиваться производительность с увеличением количества ресурсов, и если будет, то насколько пропорционально.

6. Конфигурационное тестирование — проверка работы продукта на различных конфигурациях оборудования, стендов

• **Дайте определение понятия интеграции. Опишите современные технологии и инструменты интеграции.**

Интеграцией называется элемент процесса разработки программного обеспечения, в ходе которого отдельные компоненты программного продукта объединяются в единое целое

Инструменты:

Jenkins — открытое серверное приложение, которое позволяет разработчикам быстрее строить, автоматизировать и тестировать любой проект

Buddy — веб-инструмент для непрерывной интеграции и доставки с дружественным интерфейсом. Известен в качестве одного из самых надежных и простых инструментов CI/CD.

• **Расскажите о графическом интерфейсе пользователя. Особенности тестирования, требования.**

UI — это User Interface (дословно «пользовательский интерфейс») — то, как выглядит интерфейс и то, какие физические характеристики приобретает. Определяет, какого цвета будет ваше «изделие», удобно ли будет человеку попадать пальцем в кнопочки, читабельным ли будет текст и тому подобное

UI — тестирования самого интерфейса приложения. Заключается в том, что мы имитируем действия пользователя — клики, переходы по ссылкам, и другие действия подобного плана. Смысл его — в проверке взаимодействия компонентов друг с другом.

Особенности

Тест планы в виде сценариев работы пользователя

Сценарии на естественном языке или в виде скриптов

Выполнение тестов в ручном режиме или с помощью

эмулятора

Анализ экранных форм и видимых элементов а не

внутренних переменных

Покрытие — участие интерфейсных элементов в тестах

Найденный дефект: несоответствие реального

поведения требованиям или проблемы в требованиях

Проблема отчета об ошибке: расплывчатость

формулировок

Требования

Требования к внешнему виду пользовательского

интерфейса и формам взаимодействия с пользователем

– Требования к размещению элементов управления на

экранных формах

– Требования к содержанию и оформлению выводимых

сообщений

– Требования к форматам ввода

Требования по доступу к внутренней функциональности

системы при помощи пользовательского интерфейса

– Требования к реакции системы на ввод пользователя

– Требования к времени отклика на команды

пользователя

• **Сформулируйте цель и задачи автоматизация бизнес-процессов. Опишите процессы хаотичной автоматизации, автоматизации по участкам, автоматизация по направлениям и комплексной автоматизации деятельности организации.**

• **Перечислите и поясните принципы отладки программного обеспечения.**

· на каждую используемую функцию или возможность - хотя бы один тест,

· на каждую область и на каждую границу изменения какой-либо входной величины - хотя бы один тест,

· на каждую особую (исключительную) ситуацию, указанную в спецификациях, - хотя бы один тест.

• **Опишите процесс выбора и настройки работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий).**

система контроля версий – это система, записывающая изменения

в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии.

Что не должно попасть в удаленный репозиторий:

Лишние файлы конфигурации, исполняемые файлы, временные файлы, ключи к репозиторию, платежные ключи.

• **Перечислите и опишите методы оценки качества.**

Под *качеством программного обеспечения* будем понимать совокупность свойств ПО, обусловливающих его пригодность удовлетворять определенные потребности пользователей и специалистов, участвующих в создании и сопровождении ПО.

методы

– оценить качество программных комплексов, используя различные системы показателей качества;

– закладывать качество программ при разработке технического задания и контролировать его на всех этапах жизненного цикла, т.е. оценивать минимальный уровень качества при неполной информации о программных системах, который достигнут при уже полученных значениях показателей качества;

– основываясь на установленной системе показателей качества, проводить оценку разных программ одинакового назначения в целях выявления лучшего их них.

• **Опишите процесс разработки модульной структуры проекта (диаграммы модулей).**

Диаграмма модулей показывает распределение классов и объектов по модулям в физическом проектировании системы. Каждая отдельная диаграмма модулей представляет некоторый ракурс структуры модулей системы. При разработке мы используем диаграмму модулей, чтобы показать физическое деление нашей архитектуры по слоям и разделам.

Основными элементами диаграммы модулей являются модули и их зависимости.

• **Дайте определение понятия «Качество продукции», перечислите показатели качества.**

Надежность (Reliability) – способность ПО выполнять требуемые задачи в обозначенных условиях на протяжении заданного промежутка времени или указанное количество операций. Удобство использования программного средства (usability) – совокупность свойств ПС, характеризующая усилия, необходимые для его использования, и оценку результатов его использования заданным кругом пользователей ПС. Эффективность (Efficiency) – способность ПО обеспечивать требуемый уровень производительности в соответствии с выделенными ресурсами, временем и другими обозначенными условиями. Удобство сопровождения (Maintainability) – легкость, с которой ПО может анализироваться, тестироваться, изменяться для исправления дефектов, для реализации новых требований, для облегчения дальнейшего обслуживания и адаптироваться к именующемуся окружению. Портативность (Portability) – совокупность свойств ПС, характеризующая приспособленность для переноса из одной среды функционирования в другие.

Надежность

Завершенность (вероятность отказа)

Устойчивость к отказам (работоспособность)

Восстанавливаемость

Наличие системы резервного копирования

Сохранение тестов в отдельном файле

Удобство использования

Легкость освоения

Наличие методических указаний по изучению

Понятность

Удобство и простота использования

Функциональность

Наличие средств защиты (например, шифрование тестов)

Возможность работы локальной компьютерной сети

Работа в сети Internet

Удобство сопровождения

Наличие отдельных модулей

Портативность

Занимаемый объем

• **Опишите принцип работы протоколов транспортного уровня.**

• **Дайте определение свойств качественного программного обеспечения: надежность, структурированность, эффективность. Поясните их назначение.**

Надежность программного обеспечения есть вероятность его работы без отказов в течение определенного периода времени, рассчитанная с учетом стоимости для пользователя каждого отказа.

Структурированность определяет наличие связей и отношений между элементами системы, распределение элементов системы по уровням иерахии.

Эффективность — Набор атрибутов, относящихся к соотношению между уровнем качества функционирования ПО и объемом используемых ресурсов при установленных условиях.

• **Дайте определение системы управления версиями. Сформулируйте основные принципы организации работы команды в системе контроля версий.**

Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией

• **Перечислите и охарактеризуйте функциональные виды тестирования.**

Функциональные тесты основываются на функциях, выполняемых системой, и могут проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы (use cases).

Тестирование в перспективе «требования» использует спецификацию функциональных требований к системе как основу для дизайна тестовых случаев (Test Cases). В этом случае необходимо сделать список того, что будет тестироваться, а что нет, приоритезировать требования на основе рисков (если это не сделано в документе с требованиями), а на основе этого приоритезировать тестовые сценарии (test cases). Это позволит сфокусироваться и не упустить при тестировании наиболее важный функционал.

Тестирование в перспективе «бизнес-процессы» использует знание этих самых бизнес-процессов, которые описывают сценарии ежедневного использования системы. В этой перспективе тестовые сценарии (test scripts), как правило, основываются на случаях использования системы (use cases).

• **Дайте определение понятия проект. Охарактеризуйте состав и структуру коллектива разработчиков, их функции.**

Проект – взаимосвязанный набор мероприятий и процессов, основная цель которого – реализация масштабной задачи. Его главные атрибуты – наличие сроков исполнения, ресурсы, миссия.

менеджеры проекта – руководитель управляет такой всеми делятся на тим лидов, ПМов

дизайнеры – думают как все должно выглядеть

программисты – работяги которые пишут код делятся на бэк и фронт

тестировщики – разрушители и проверяльщики того что сделали программисты и разработчики документации делятся на мануальщиков и автоматизаторов

разработчики документации – составляет проектную документацию

• **Перечислите и охарактеризуйте связанные с изменениями виды тестирования.**

Регресс и тестирование новых фич

• **Сформулируйте понятие и принципы работы с инструментальными средствами разработки ПО.**

Инструментальные средства разработки ПО– это совокупность аппаратно-программных средств, позволяющих осуществить написание и отладку программ для микропроцессорных систем с большой степенью достоверности их работоспособности в реальных системах.

• **Дайте определение понятий «Отладка», «Локализация Ошибки». Какие виды ошибок существуют? Охарактеризуйте их.**

Отла́дка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

Локализация - это нахождение места ошибки в программе.

- Ошибки пользовательского интерфейса.

- Ошибки вычислений.

- Ошибки управления потоком.

- Ошибки передачи или интерпретации данных.

- Перегрузки.

• **Опишите инструментальные средства создания Windows-приложений.**

Языки программирования

Система программирования — это совокупность программного обеспечения, включающая: транслятор, интерфейс разработчика, набор библиотек подпрограмм, редактор связей, отладчик, сервисные средства, справочную систему

Транслятор (англ. translator — переводчик) — это средство преобразования исходного текста программы на алгоритмическом языке в машинный код. Трансляторы реализуются в виде компиляторов или интерпретаторов, которые существенно различаются по принципам работы.

Компилятор (англ. compiler — составитель, собиратель) читает всю программу целиком, делает ее перевод и создает объектный модуль на машинном языке (из него потом создается законченный вариант программы).

Интерпретатор (англ. interpreter — истолкователь, устный переводчик) переводит и выполняет программу построчно. Программа, обрабатываемая интерпретатором, должна заново переводиться на машинный язык при каждом очередном ее запуске.

Откомпилированные программы работают быстрее, но интерпретируемые проще исправлять и изменять.

Редактор связей — осуществляет перевод объектного модуля в загрузочный (исполняемую программу) с подключение всех используемых подпрограмм. Он включает в себя: отладчик; набор библиотек; средства оптимизации кода программ; справочные системы; сервисные средства для работы с библиотеками и др.

СУБД

• **Опишите процесс разработки тестовых модулей проекта для тестирования отдельных модулей.**

Модульное тестирование проверяет функциональность конкретного куска кода, обычно по одной функции за раз.

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода. Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

Выберите логическое расположение тестов в вашей VCS

Выберите способ именования проектов с тестами

Используйте такой же способ именования для тестовых классов

Выберите «говорящий» способ именования методов тестирующих классов

Выберите тестовый фреймворк, который подходит вам

Выбрать то что нужно тестировать и чем можно пренебречь

Придерживайтесь единого стиля написания тела теста

Тестируйте одну вещь за один раз

• **Опишите процесс разработка приложений Windows.Forms в среде программирования Microsoft Visual Studio.**

Создание БД

Создание проекта

Добавление элементов на форму

Добавление кода на форму

Подключение БД

Запуск приложения

• **Перечислите и охарактеризуйте нефункциональные виды тестирования.**

нагрузочное тестирование (Performance and Load Testing)

стрессовое тестирование (Stress Testing)

тестирование стабильности или надежности (Stability / Reliability Testing)

объемное тестирование (Volume Testing)

Тестирование установки (Installation testing)

Тестирование удобства пользования (Usability Testing)

Тестирование на отказ и восстановление (Failover and Recovery Testing)

Конфигурационное тестирование (Configuration Testing)

• **Расскажите про инструменты разработки программных средств. Перечислите и охарактеризуйте группы инструментов ПС.**

Инструментарий программирования — это совокупность программных продуктов, обеспечивающих технологию разработки, отладки и внедрения создаваемых новых программных продуктов.

Языки программирования

Среды разработки

Инструментальные среды пользователя — специальные средства, которые встроены в пакеты прикладных программ

БД

• **Сформулируйте определение понятия тестирование. Опишите методы и средства организации тестирования.**

Тестирование – процесс проверки программы на соответствие требованиям, поиск багов

1) Модульное тестирование

2) Интеграционное тестирование

3) Системное тестирование

4) Приемочные испытания

• **Перечислите и охарактеризуйте основные классы инструментальных сред разработки и сопровождения ПС.**

Инструментальные среды программирования - *Инструментальная среда программирования* предназначена в основном для поддержки процессов программирования (кодирования), тестирования и отладки ПС.

Рабочие места компьютерной технологии - Оно существенно использует свойства специализированности, ориентированности на конкретную технологию программирования и, как правило, интегрированности.

Инструментальные системы технологии программирования - Инструментальная система технологии программирования предназначена для поддержки всех процессов разработки и сопровождения в течение всего жизненного цикла ПС и ориентирована на коллективную разработку больших программных систем с продолжительным жизненным циклом.

• **Расскажите о методах проведения тестирования пользовательского интерфейса.**

// Тестирование бумажных прототипов

// Ui-тестирование интерактивного прототипа

Функциональное тестирование UI

Ручное тестирование

Автоматическое тестирование

Автоматизация тестирования UI

Координатный метод

Распознавание образов

Accessibility метод

Гибридный подход

Usability testing

Тестирование специальных возможностей

• **Опишите методы организации коллективной разработки ПО.**

Закон Брукса

Ни одно обсуждение команд разработчиков не проходит без упоминания данного принципа:

«Добавляя людских ресурсов, мы задерживаем окончание программного проекта» (Брукс, 1975).

Бесчисленные команды разработчиков подтвердили постулат. Законы Брукса и Конвея составляют базу.

Присоединяя нового человека, команда тратит усилия на введение в курс дела, на объяснение используемых трюков и устройства. Участники расходуют время на информирование и синхронизацию с новобранцем, на обучение труду в команде и передачу знаний. Работа замедляется.

Закон Конвея — «Организации проектируют системы, которые копируют структуру коммуникаций в этой организации». Заключается в том, что при декомпозиции инженерами крупных задач на более мелкие для передачи части работ коллегам, возникает сложность координации между функциональными подразделениями. Во многих организациях команды разделены в соответствии с тем функционалом, который они выполняют, и существуют в отрыве от других команд. Поэтому структура программного интерфейса системы будет отражать социальные границы организации (организаций), которые её создали, что затрудняет общение. В рамках своего функционала команды могут отлично справляться с выполнением своих задач. Однако для того, чтобы создать что-то новое (функцию, продукт и т. д.), командам необходимо кросс-функциональное взаимодействие и совместная работа.

Число Данбара. Природные ориентиры

Частый вопрос от групп разработчиков: «Насколько большой должна быть команда?» Работа антрополога Робина Данбара дает интересные идеи при попытке ответить на вопрос.

Автор предоставляет убедительные доводы, что 150 является верхним пределом для организациилюдей. Данбар заметил указанное число в воинских формированиях со времён Древнего Рима, в неолитических поселениях, в общинах амишей и в современных научно-исследовательских группах.

Магическая семерка Миллера (кошелёк Миллера)

Принято считать мудрым выбором размер команды из 7 человек (± 2). Практического смысла в утверждении нет. Доказательства тезиса об оптимальном количестве членов команды в 5–9 человек отсутствуют.

Сторонники размера апеллируют к знаменитой статье Миллера 1956 года «Магическое число семь плюс минус два: некоторые ограничения нашей способности обработки информации».

Размер команды в методологии Scrum

Как статья Миллера об объемах обработки информации человеческим мозгом может применяться к определению размера команды разработчиков ПО? Обратимся к методологии Scrum. В учебнике говорится: «Команда в Scrum должна быть семь плюс или минус два человека» (Димер и др., 2008). Одновременно в руководстве по Scrum за 2011 утверждается: «Команды из более девяти членов вызывают слишком много проблем в координации. Большие команды разработчиков заметно усложняют весь процесс» (Сазерленд и Швабер, 2013).

Закон Паркинсона и Закон Хофштадтера

Работа заполняет все время, выделенное для ее выполнения,

— закон Паркинсона

Всегда потребуется больше времени, чем вы ожидаете, даже если вы знаете закон Хофштадтера,

— закон Хофштадтера, 1980

Закон Голла. Парнас и Александер

Сложная рабочая система неизменно получается из простой рабочей системы. Сложная система, разработанная с нуля, никогда не работает. И никакие улучшения не заставят ее работать. Начинать следует с простой рабочей системы,

— закон Голла, 1986

Законы Келли

Следует добавить пару полезных постулатов от автора статьи.

Масштаб ПО всегда будет увеличиваться пропорционально имеющимся ресурсам,

— первый закон Келли

Внутри каждого большого проекта в области разработки есть маленький побочный проект вне основной задачи

— второй закон Келли

• **Перечислите и охарактеризуйте методы отладки программного обеспечения.**

Статические

- ручную прокрутку программы;

- прокрутку программы программными анализаторами ( например, компилятором); автоматизированный анализ программы в этом случае проводится без выполнения ее на ЭВМ и поэтому попадает в категорию «статических»;

- коллективную проверку программ;

- проверку программы программистом-технологом с целью выявления и исправления в ней технологических ошибок.

Динамические

- тестирование;

- поиск ошибок с использованием системных средств;

- отладка программы в интерактивном режиме.

• **Дайте определение понятию отладки программного средства.**

этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки.

• **Опишите методы и способы идентификации сбоев и ошибок.**

· идентификация множества функциональных требований;

· идентификация внешних функций и построение последовательностей функций в соответствии с их использованием в ПС;- идентификация множества входных данных каждой функции и определение областей их изменения;

· построение тестовых наборов и сценариев тестирования функций;

· выявление и представление всех функциональных требований с помощью тестовых наборов и проведение тестирования ошибок в программе и при взаимодействии со средой.

Стратегическая цель тестирования состоит в том, чтобы убедиться, что каждый рассматриваемый входной набор данных соответствует ожидаемым выходным выходных данным.

• **Дайте определение понятия и опишите особенности разработки программного модуля.**

Программный модуль (program module) — программа или отдельная ее функциональная часть, рассматриваемая как единое целое в контекстах хранения, замены, трансляции, объединения с другими программными модулями и ее загрузки в оперативную память ЭВМ.

1. изучение и проверка спецификации модуля, выбор языка программирования; (т.е. разработчик, изучая спецификацию, выясняет, понятна она ему или нет, достаточно ли полно она описывает модуль; затем он выбирает язык программирования, на котором будет написан модуль, хотя язык программирования может быть единым для всего ПС)

2. выбор алгоритма и структуры данных (здесь выясняется не известны ли какие-либо алгоритмы для решения поставленной задачи и если есть, то воспользоваться им)

3. программирование модуля (написание кода программы)

4. шлифовка текста модуля (редактирование имеющихся комментариев, добавление дополнительных комментариев, для того чтобы обеспечить требуемое качество)

5. проверка модуля (проверяется логика работы модуля, отлаживается его работа)

• **Опишите инструментальные средства поддержки процесса документирования.**

К данным средствам относятся средства документирования, анализаторы программ, средства управления изменениями и конфигурацией ПС и систем, средства реструктурирования и реинженении, средства обеспечения мобильности.

• **Опишите процесс тестирования интерфейса пользователя средствами инструментальной среды разработки.**

• **Дайте определение понятия обработка исключительных ситуаций. Опишите инструменты среды разработки для обработки исключительных ситуаций.**

Обработка исключительных ситуаций — механизм языков программирования, предназначенный для описания реакции программы на ошибки времени выполнения и другие возможные проблемы (исключения), которые могут возникнуть при выполнении программы и приводят к невозможности (бессмысленности) дальнейшей отработки программой её базового алгоритма.

Для обработки исключений в С# используется оператор try-catch. Он имеет следующую структуру: try { блок кода, в котором возможно исключение } catch ([тип исключения] [имя]) { блок кода – обработка исключения }.

• **Опишите методические аспекты проектирования ПО. Общие принципы проектирования систем.**

Метод декомпозиции

Взаимодействие между модулями

Принципы модульности

• **Сформулируйте основные этапы документирования результатов тестирования.**

• **Перечислите стандарты качества программных средств.**

Стандарт ISO 9126:1991 - Оценка программного продукта. Основные факторы, определяющие качество сложных программных средств. Внутренние метрики. Внешние метрики. Метрики качества в использовании

Российский аналог стандарта ГОСТ Р ИСО/МЭК 9126-93. В 2011 году был заменён стандартом ISO/IEC 25010:2011. Состоит из 4 частей, описывающие следующие аспекты: модель качества; внешние метрики качества; внутренние метрики качества; метрики качества в использовании.

• **Опишите процесс выявление ошибок системных компонентов. (ХЗ)**

СИСТЕМНОЕ ТЕСТИРОВАНИЕ — это уровень тестирования, который проверяет законченный и полностью интегрированный программный продукт. Целью системного теста является оценка сквозных технических характеристик системы.

Тестирование полностью интегрированных приложений, включая внешние периферийные устройства, с целью проверки взаимодействия компонентов друг с другом и с системой в целом. Это также называется сквозным сценарием тестирования.

Проверьте тщательное тестирование каждого входа в приложении, чтобы проверить желаемые результаты.

Тестирование опыта пользователя с приложением.

• **Дайте определение понятия «Качество программного обеспечения». Перечислите критерии оценки качества ПО.**

Качество программного обеспечения — способность программного продукта при заданных условиях удовлетворять установленным или предполагаемым потребностям

адекватность, доступность, преемственность, безопасность, своевременность, эффективность (в том числе экономическая), ориентированность на клиента

• **Перечислите основные средства проектирования интерфейса пользователя и опишите принцип их работы.**

Для создания у пользователя такого ощущения «внутренней свободы» интерфейс должен обладать целым рядом свойств :

Естественность интерфейса.

Согласованность интерфейса.

Дружественность интерфейса (Принцип «прощения пользователя»)

Принцип «обратной связи».

Простота интерфейса.

Гибкость интерфейса.

Эстетическая привлекательность.

• **Дайте определение свойств качественного программного обеспечения: понятность, осмысленность, завершенность. Поясните их назначение.**

Понятность системы — это то ее свойство, благодаря которому пользователю не составит труда в ней разобраться

Программный продукт обладает свойством осмысленности,если его документация не содержит избыточной информации

завершенность программного средства — Совокупность свойств программного средства, характеризующая частоту отказов, обусловленных дефектами программного средства.

• **Дайте определение понятий ручное и автоматизированное тестирование. Расскажите об их преимуществах и недостатках.**

Ручное тестирование (manual testing) — часть процесса тестирования на этапе контроля качества в процессе разработки программного обеспечения. Оно производится тестировщиком без использования программных средств

Автоматизированное тестирование - выполнение тестов, реализуемое при помощи заранее записанной последовательности тестов. Тест-кейсы частично или полностью выполняет специальное инструментальное средство, однако разработка тест-кейсов, подготовка данных, оценка результатов выполнения, написания отчётов об обнаруженных дефектах — всё это и многое другое делает человек.

• **Перечислите и поясните принципы отладки программного обеспечения.**

· на каждую используемую функцию или возможность - хотя бы один тест,

· на каждую область и на каждую границу изменения какой-либо входной величины - хотя бы один тест,

· на каждую особую (исключительную) ситуацию, указанную в спецификациях, - хотя бы один тест.

Различаются два основных вида отладки (включая тестирование): автономную и комплексную отладку ПС. Автономная отладка ПС означает последовательное раздельное тестирование различных частей программ, входящих в ПС, с поиском и исправлением в них фиксируемых при тестировании ошибок. Она фактически включает отладку каждого программного модуля и отладку сопряжения модулей. Комплексная отладка означает тестирование ПС в целом с поиском и исправлением фиксируемых при тестировании ошибок во всех документах (включая тексты программ ПС), относящихся к ПС в целом.

• **Опишите современные принципы и методы разработки программных приложений**

Метод нисходящего проектирования (метод пошаговой детализации, метод иерархического проектирования, top-down-подход)

Суть метода заключается в определении спецификаций компонентов системы путем последовательного выделения в ее составе отдельных составляющих и их постепенной детализации до уровня, обеспечивающего однозначное понимание того, что и как необходимо разрабатывать и реализовывать.

Модульное проектирование

+Реализация метода нисходящего проектирования тесно связана с другим понятием программирования - модульным проектированием, так как на практике при декомпозиции сложной программы возникает вопрос о разумном пределе ее дробления на составные части. Вместе с тем понятие модульности нельзя сводить только к представлению сложных программных комплексов в виде набора отдельных функциональных блоков.

Структурное программирование

+Актуальная для начального периода развития и использования ЭВМ проблема разработки программ, занимающих минимум основной памяти и выполняющихся за кратчайшее время, в последующем в связи резким падением стоимости аппаратной части ЭВМ, значительным возрастанием их быстродействия и объемов памяти сменилась необходимостью разработки и применения принципиально новых методов составления программ. Все это нашло свое воплощение в разработке принципа структурного программирования. Одной из целей структурного программирования было стремление облегчить разработку и отладку программных модулей, а главное - их последующее сопровождение и модификацию.

CASE-технологии

За последнее десятилетие в области средств автоматизации программирования сформировалось новое направление под общим названием CASE-технологии (Computer Aided Software Engineering).

+CASE-технология представляет собой совокупность средств системного анализа, проектирования, разработки и сопровождения сложных программных систем, поддерживаемых комплексом взаимоувязанных инструментальных средств автоматизации всех этапов разработки программ. Благодаря структурным методам CASE-технология на стадиях анализа и проектирования обеспечивает разработчиков широкими возможностями для различного рода моделирования, а централизованное хранение всей необходимой для проектирования информации и контроль за целостностью данных гарантируют согласованность взаимодействия всех специалистов, занятых в разработке ПО.

Технологии RAD

В начале 80-х годов появилась методология, по которой разработка программы начиналась не после завершения процесса выработки окончательных требований к ней, а как только устанавливались требования на первый, “стартовый” (пилотный) вариант прикладной программы, позволяющий начать содержательную работу по ее реализации на компьютере.

Data Warehouse

Другое направление разработки прикладных программных средств, олицетворяющее собой современный подход к реализации широкого круга задач для принятия управленческих решений, базируется на концепции создания специального хранилища данных (Data Warehouse). Основное отличие концепции Data Warehouse от традиционного представления баз данных заключается в следующем:

во-первых, в том, что актуализация данных в Data Warehouse означает не обновление элементов информации, а добавление новых элементов к уже имеющимся (что расширяет возможности проведения различного рода сравнительного анализа);

во-вторых, в том, что наряду с информацией, непосредственно отражающей состояние системы управления, в Data Warehouse аккумулируются и метаданные.

• **Опишите работу с системой контроля версий**

Основной принцип работы Git на удивление прост – его только нужно усвоить. Во-первых, Git манипулирует информацией в форме снимков текущего состояния, по одному на каждую фиксацию, или «коммит», и «знает», как применить набор изменений для перехода к следующему состоянию или возврата к предыдущему.

Создание репозитория

Определение состояния

Подготовка файлов

Коммит

Пуш

Создание веток

Клонирование удаленного репозитория

Отслеживание изменений

Разрешение конфликтов

• **Расскажите про основные подходы к интегрированию программных модулей.**

• **Расскажите про стандарты кодирования.**

• **Расскажите про описание требований: унифицированный язык моделирования - краткий словарь. Диаграммы UML.**

Требования к программному обеспечению являются описанием функций и функциональных возможностей целевой системы. Требования передают ожидания пользователей от программного продукта. Требования могут быть очевидными или скрытыми, известными или неизвестными, ожидаемыми или неожиданными с точки зрения клиента.

Унифицированный язык объектно-ориентированного моделирования Unified Modeling Language (UML) явился средством достижения компромисса между этими подходами. Существует достаточное количество инструментальных средств, поддерживающих с помощью UML жизненный цикл информационных систем, и, одновременно, UML является достаточно гибким для настройки и поддержки специфики деятельности различных команд разработчиков.

UML — это сокращение от Unified Modeling Language, и, как мы знаем, он является стандартизированным языком моделирования, состоящим из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем, а также, к примеру, для бизнес-моделирования.

• **Расскажите про описание и оформление требований (спецификация). Анализ требований и стратегии выбора решения**

Если коротко, то основные этапы разработки требований — это:

Зачем нам что-то делать? (нужно больше золота)

Что мы будем делать? (все как у людей, но дешевле)

Как мы это сделаем? (с блокчейном и датасаентистами, естественно)

Когда мы это сделаем? (вчера, а отрефакторим «потом»)

Анализ требований — часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование.

Перед специалистом IT-отдела, либо независимым консультантом, которому поручен выбор решения в области автоматизации предприятия, стоит нелегкая задача. Ведь на рынке представлены сотни решений в области автоматизации и чтобы хотя бы бегло ознакомиться с каждым из них может потребоваться несколько человеко-лет.

На практике, конечно, вряд ли путь подробного изучения всех известных на рынке решений можно рассматривать всерьез. В простейшем случае рассматриваются аналитические обзоры, подготовленные независимыми экспертами, оцениваются финансовые возможности предприятия внедрения и на третейский суд инвестора предоставляются 2-3 решения.

Чтобы сделать выбор обоснованным, используются стратегии выбора решения. Основные из них:

анализ требований,

анализ несоответствия,

подход на основе "лучших практик".

• **Расскажите про стандарты качества программной документации.**

• **Расскажите про анализ спецификаций, верификацию и аттестацию программного обеспечения.**

Результатом **анализа** требований является получение **спецификации** разрабатываемого ПО, т.е. выполняют декомпозицию и содержательную постановку решаемых задач, уточняют их взаимодействие и эксплуатационные ограничения. ... 4.1. **Спецификации** ПО при структурном подходе. **Спецификация** – это полное и точное описание функций и ограничений разрабатываемого ПО.

Верификацией и аттестацией называются процессы проверки и анализа, в ходе которых проверяется соответствие программного обеспечения своей спецификации и требованиям заказчиков. Верификация и аттестация охватывают весь цикл жизни ПО – они начинаются на этапе анализа требований и завершаются проверкой программного кода на этапе тестирования программной системы.

**Вопросы к экзамену по ПМ 11 «Разработка, администрирование и защита баз данных»**

• **Приведите классификацию баз данных. Определения, основные функции, виды.**

По типу хранимой информации БД делятся на

документальные - Среди документальных баз различают библиографические, реферативные и полнотекстовые,

фактографические - В системах фактографического типа в БД хранится информация об интересующих пользователя объектах предметной области в виде «фактов» (например, биографические данные о сотрудниках, данные о выпуске продукции производителями и т.п.);

лексикографические - К лексикографическим базам данных относятся различные словари (классификаторы, многоязычные словари, словари основ слов и т. п.).

По характеру организации хранения данных и обращения к ним различают

локальные (персональные) - Персональная база данных - это база данных, предназначенная для локального использования одним пользователем. Локальные БД могут создаваться каждым пользователем самостоятельно, а могут извлекаться из общей БД.,

общие (интегрированные, централизованные) , распределенные базы данных - Интегрированные и распределенные БД предполагают возможность одновременного обращения нескольких пользователей к одной и той же информации (многопользовательский, параллельный режим доступа).

По характеру организации данных БД могут быть разделены на

неструктурированные,

частично структурированные и

структурированные.

Структурированные БД, в свою очередь, по типу используемой модели делятся на

иерархические,

сетевые,

реляционные,

• **Основы реляционной алгебры. Определения высказываний, запись, примеры.**

Реляционной базой данных называется совокупность отношений, содержащих всю информацию, которая должна хранится в базе.

Запись - строка в таблице

Объединение

Пересечение

Вычитание

Декартово произведение

Выборка

Проекция

Соединение

Деление

Проекция является операцией, при которой из отношения выделяются атрибуты только из указанных доменов, то есть из таблицы выбираются только нужные столбцы, при этом, если получится несколько одинаковых кортежей, то в результирующем отношении остается только по одному экземпляру подобного кортежа.

Выборка — это операция, которая выделяет множество строк в таблице, удовлетворяющих заданным условиям.

Умножение или декартово произведение является операцией, производимой над двумя отношениями, в результате которой мы получаем отношение со всеми доменами из двух начальных отношений.

Операция соединения обратна операции проекции и создает новое отношение из двух уже существующих. Новое отношение получается конкатенацией кортежей первого и второго отношений, при этом конкатенации подвергаются отношения, в которых совпадают значения заданных атрибутов.

Результатом операции пересечения будет отношение, состоящее из кортежей, полностью входящих в состав обоих отношений.

• **Иерархическая модель данных. Основные понятия, графическое изображение, примеры.**

Иерархическая модель является самой ранней моделью баз данных. Для понимания иерархической модели необходимо запомнить следующие термины:

Атрибут (или поле)– минимальный элемент данных. Атрибут имеет уникальное имя, по которому к нему можно обратиться из программного кода.

Запись – логически связанная совокупность атрибутов. Запись имеет уникальное имя, которое позволяет обращаться к ней из программного кода. Записи можно добавлять, изменять, удалять.

Экземпляр записи – конкретная запись с конкретными значениями атрибутов.

Групповое отношение - иерархическое отношение между записями двух разных типов. Запись, которая, находится выше по иерархии, называется родительской. Записи, которые, расположены ниже по иерархии называются дочерними.

Над данными в иерархической модели возможны следующие операции:

Добавление в базу новой записи.

Изменение значений атрибутов (кроме ключевых) отдельной записи.

Удаление записи со всеми дочерними записями.

Извлечение записи.

• **Сетевая модель данных. Основные понятия, графическое изображение, примеры.**

Сетевая модель данных — логическая модель данных, являющаяся расширением иерархического подхода, строгая математическая теория, описывающая структурный аспект, аспект целостности и аспект обработки данных в сетевых базах данных.

Примерный набор операций манипулирования данными:

найти конкретную запись в наборе однотипных записей;

перейти от предка к первому потомку по некоторой связи;

перейти к следующему потомку в некоторой связи;

перейти от потомка к предку по некоторой связи;

создать новую запись;

уничтожить запись;

модифицировать запись;

включить в связь;

исключить из связи;

переставить в другую связь

• **Реляционная модель данных. Основные понятия, графическое изображение, примеры.**

Реляционная модель данных (РМД) — логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

На реляционной модели данных строятся реляционные базы данных.

Реляционная модель данных включает следующие компоненты:

Структурный аспект (составляющая) — данные в базе данных представляют собой набор отношений.

Аспект (составляющая) целостности — отношения отвечают определённым условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.

Аспект (составляющая) обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

Кроме того, в состав реляционной модели данных включают теорию нормализации.

Для лучшего понимания РМД следует отметить три важных обстоятельства:

модель является логической, то есть отношения являются логическими (абстрактными), а не физическими (хранимыми) структурами;

для реляционных баз данных верен информационный принцип: всё информационное наполнение базы данных представлено одним и только одним способом, а именно — явным заданием значений атрибутов в кортежах отношений; в частности, нет никаких указателей (адресов), связывающих одно значение с другим;

наличие реляционной алгебры позволяет реализовать декларативное программирование и декларативное описание ограничений целостности, в дополнение к навигационному (процедурному) программированию и процедурной проверке условий.

• **Первая, вторая, третья нормальные формы реляционной модели данных. Определение, требования, примеры.**

Первая нормальная форма (1NF)

Основная статья: Первая нормальная форма

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

В реляционной модели отношение всегда находится в первой нормальной форме по определению понятия отношение. Что же касается различных таблиц, то они могут не быть правильными представлениями отношений и, соответственно, могут не находиться в 1НФ.

Вторая нормальная форма (2NF)

Основная статья: Вторая нормальная форма

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый неключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Функционально полная зависимость означает, что если потенциальный ключ является составным, то атрибут зависит от всего ключа и не зависит от его частей.

Третья нормальная форма (3NF)

Основная статья: Третья нормальная форма

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых.

• **НФБК, четвертая нормальная форма реляционной модели данных. Определение, требования, примеры.**

Нормальная форма Бойса — Кодда (BCNF)

Основная статья: Нормальная форма Бойса — Кодда

Переменная отношения находится в нормальной форме Бойса — Кодда (иначе — в усиленной третьей нормальной форме) тогда и только тогда, когда каждая её нетривиальная и неприводимая слева функциональная зависимость имеет в качестве своего детерминанта некоторый потенциальный ключ.

Четвёртая нормальная форма (4NF)

Основная статья: Четвёртая нормальная форма

Переменная отношения находится в четвёртой нормальной форме, если она находится в нормальной форме Бойса — Кодда и не содержит нетривиальных многозначных зависимостей.

• **Модель сервера приложений. Архитектура, компоненты, преимущества.**

Модель сервера приложений позволяет помещать прикладные программы на отдельные серверы приложений. Программа, выполняемая на компьютере-клиенте, решает задачу ввода и отображения данных, т. е. реализует операции первой группы. Прикладной компонент реализован как группа процессов, выполняющих прикладные функции, и называется сервером приложения. Доступ к информационным ресурсам, необходимым для решения прикладных задач, обеспечивается так же, как в модели доступа к удаленным данным, т. е. прикладные программы обращаются к серверу базы данных с помощью SQL-запросов.

Технологии «клиент-сервер» имеют следующие преимущества:

• позволяют организовывать сети с большим количеством рабочих станций;

• обеспечивают централизованное управление учетными записями пользователей, безопасностью и доступом, что упрощает сетевое администрирование ;

• предоставляют эффективный доступ к сетевым ресурсам.

• **Модель клиент. Архитектура, компоненты, преимущества.**

Модели клиент—сервер в технологии распределенных баз данных.

Термин клиент-сервер исходно применялся к архитектуре, при которой клиентский процесс запрашивает некоторые услуги, а серверный процесс обеспечивает их выполнение. Реализация архитектуры клиент - сервер, применительно к разработке БД позволяет более полно использовать ресурсы сети.

Модель клиент-сервер предоставляет возможность разграничить поставленные задачи и работу над вычислениями между теми, кто заказывает услуги и теми, кто их поставляет.

Пользователь БД — программа или человек, обращающийся к базе данных. Запрос — процесс обращения пользователя к БД с целью ввода, получения или изменения информации в БД. Транзакция — последовательность операций модификации данных в БД, переводящая БД из одного непротиворечивого состояния в другое непротиворечивое состояние.

• **Создание структуры таблицы. Типы данных таблиц базы данных. Ключевое поле таблиц баз данных.**

Cтруктура таблицы: названия полей, их тип, ключи, значения по умолчанию, прочее…

Рассмотрим типы данных, которые есть в языке SQL. У каждого столбца таблицы (у атрибута) должен быть тип данных для значений, которые хранятся в столбце. Тип данных для столбца определяется при создании таблицы, а еще лучше, когда он определяется на этапе проектирования баз данных.

Язык SQL делит данные на пять типов:

Целочисленный тип данных SQL.

Вещественный тип данных SQL.

Типы данных даты и времени SQL.

Строковый тип данных SQL.

Строковый тип данных SQL в кодировки Юникод.

Первичный ключ – это значение, которое уникально для каждой записи в таблице.

• **Заполнение таблиц данными. Установление связей между таблицами. Назначение, главная и подчиненная таблица, последовательность действий.**

для заполнения таблицы данными используется команда INSERT INTO имя\_таблицы VALUES ('значение\_первого\_столбца','значение\_второго\_столбца', ..., 'значение\_последнего\_столбца');

Связи между таблицами устанавливаются с помощью внешних ключей FOREIGN KEY (имя\_столбца\_которое\_является\_внешним\_ключом) REFERENCES имя\_таблицы\_родителя (имя\_столбца\_родителя);

Внешний ключ представляет один или несколько столбцов из одной таблицы, который одновременно является потенциальным ключом из другой таблицы. Внешний ключ необязательно должен соответствовать первичному ключу из главной таблицы. Хотя, как правило, внешний ключ из зависимой таблицы указывает на первичный ключ из главной таблицы.

Связи между таблицами бывают следующих типов:

Один к одному (One to one)

Один к многим (One to many)

Многие ко многим (Many to many)

• **Запрос на выборку. Назначение, создание, примеры. Запрос с параметром. Назначение, создание, примеры. Создание запроса на выборку с логическими операциями в условиях отбора.**

select

• **Запрос на создание таблицы. Запрос на удаление. Назначение, создание, примеры. Запрос на обновление. Запрос на добавление. Назначение, создание, примеры.**

Create database

• **Опишите основные принципы построения концептуальной, логической и физической модели данных.**

• **Расскажите про структуры данных СУБД, общий подход к организации представлений, таблиц, индексов и кластеров**

• **Расскажите про основные принципы структуризации и нормализации базы данных.**

• **Опишите методы описания схем баз данных в современных СУБД. Расскажите про структуры данных СУБД.**

• **Расскажите про методы организации целостности данных в БД.**

• **Опишите современные инструментальные средства проектирования схемы базы данных.**

• **Введение в SQL и его инструментарий. Подготовка систем для установки SQL-сервера. Расскажите про установку и настройку SQL-сервера, автоматизация управления SQL**

• **Расскажите про выполнение мониторинга SQLServer с использование оповещений и предупреждений.**

• **Расскажите про поиск и решение типичных ошибок, связанных с администрированием.**

• **Расскажите про способы контроля доступа к данным и управления привилегиями.**

• **Опишите алгоритм проведения процедуры резервного копирования.**

• **Опишите модели восстановления SQL-сервера.**

• **Расскажите про резервное копирование баз данных, восстановление баз данных**

• **Расскажите про аутентификацию и авторизацию пользователей, назначение серверных ролей и ролей баз данных, авторизацию пользователей при получении доступа к ресурсам.**