Вопросы к экзамену «Основы инженерии программного обеспечения»

1. **Что такое программное обеспечение, какая цель программного обеспечения**.

   Программное обеспечение – продукт деятельности команды программистов, который разрабатывался как часть другой системы. Цель: создание массового программного продукта.

2. **Что такое программная инженерия**.

   Программная инженерия – форма инженерии, которая применяет принципы информатики и математики для получения рентабельных решений в области ПО. Программная инженерия – инженерная дисциплина, охватывающая все принципы разработки ПО.

3. **Информатика, системотехника, бизнес-реинжиниринг**.

   Информа́тика — наука о методах и процессах сбора, хранения, обработки, передачи, анализа и оценки информации с применением компьютерных технологий, обеспечивающих возможность её использования для принятия решений.

   Системотехника — советская инженерная дисциплина, появившаяся как аналог системной инженерии — направления науки и техники, охватывающего проектирование, создание, испытание и эксплуатацию сложных систем технического и социально-технического характера.

   Реинжиниринг бизнес-процессов — фундаментальное переосмысление и радикальное перепроектирование бизнес-процессов для достижения максимального эффекта производственно-хозяйственной и финансово-экономической деятельности, оформленное соответствующими организационно-распорядительными и нормативными документами. Реинжиниринг использует специфические средства представления и обработки проблемной информации, понятные как менеджерам, так и разработчикам информационных систем.

4. **Что такое проектирование программного обеспечения**.

   Проектирование программного обеспечения — процесс создания проекта программного обеспечения (ПО), а также дисциплина, изучающая методы проектирования. Проектирование ПО является частным случаем проектирования продуктов и процессов. Проектирование ПО включает следующие основные виды деятельности:

- выбор метода и стратегии решения;
- выбор представления внутренних данных;
- разработка основного алгоритма;
- документирование ПО;
- тестирование и подбор тестов;
- выбор представления входных данных.

5. **Как происходит процесс создания ПО. Этапы проекта в соответствии с каскадной моделью**:

- Анализ требований → Спецификация программного обеспечения
- Проектирование программного обеспечения
- Программирование
- Тестирование программного обеспечения
- Системная интеграция
- Внедрение программного обеспечения (или Установка программного обеспечения)
- Сопровождение программного обеспечения
-

6. **Назовите этапы (фазы) создания программного обеспечения**.(+)

7. **Что такое программный продукт, что включает в себя программный продукт.**

   Программный продукт – программное обеспечение и соответствующая документация, предназначенные для поставки массовому пользователю. Совокупность компьютерных программ, процедур и, возможно, связанных с ними документации и данных

8. **Три фактора успешного программного проекта**.

   Уложились вовремя, Выполнили требования, Уложились в бюджет

9. **Что такое «рефакторинг»**.

   Рефáкторинг, или перепроектирование кода, переработка кода, равносильное преобразование алгоритмов — процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

10. **Почему программное обеспечение считается одним из самых сложных продуктов в современном мире**.

    Высокая атомарность элементов системы, сложность прогнозирования результата, сложность планирования результата, высокая степень творчества в отличии от иных отраслей.

11. **Что такое методологии проектирования ПО, какие бывают методы.**

    Методология проектирования представляет собой концепцию / принципы проектирования, реализуемые набором методов проектирования, поддерживаемых средствами проектирования. Объектно-ориентированный, структурный, автоматизированный неавтоматизированный.

12. **Что такое требования к ПО**.

    Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований.

13. **Какие проблемы возникают при создании интернациональных команд.**

    Отличия в часовых зонах, механизмы взаимодействия в команде (прим. Кастовая система в Индии)

14. **Что такое водопадный подход к проектированию.**

    Модель процесса разработки программного обеспечения, в которой процесс разработки выглядит как поток, последовательно проходящий фазы анализа требований, проектирования, реализации, тестирования, интеграции и поддержки.

15. **Для каких задач подходит водопадный подход**.

    Задачи с определенными, неизменяемыми требованиями, одним циклом конструирования и без промежуточных версий.

16. **Достоинства и недостатки водопадной модели.**

    Достоинства: полное документирование каждого этапа, четкое планирование сроков и затрат, прозрачность для заказчика. Недостатки: Необходимость утверждения полного объема требований к системе еще на первом этапе; возврат к первой стадии и переделка заново всей проделанной работы для внесения изменений; Увеличение затрат средств и времени в случае необходимости изменения требований.

17. **Что такое «рабочий продукт» и чем он отличается от «компонентов ПО».**

    Компоненты — это ипостаси системы, в которых она выполняет какую-то функцию (имеет какое-то назначение, предназначение) в составе

надсистемы/использующей системы. *Рабочие продукты* — это спецификации, тесты, диаграммы и какие-то модели, базы данных и физические объекты, промежуточные результаты разработки ПО.

18. **Что такое «нематериальный продукт».**

    Продукт, который человек может воспринимать лишь косвенно, который может не иметь физического воплощения в реальном мире.

19. **Что обычно регламентирует дисциплина обязательств.**

    обязательства, которые:

- даются добровольно;
- не даются легко – работа, ресурсы, расписание должны быть тщательно учтены;
- между сторонами включает в себя то, что будет сделано, кем и в какие сроки ;
- открыто и публично сформулированы (то есть это не "тайное знание"). Кроме того:
- ответственная сторона стремится выполнить обязательства, даже если нужна помощь;
- до наступления deadline, как только становится очевидно, что работа не может быть закончена в срок, обсуждаются новые обязательства.

20. **Что такое «корпоративная культура компании».**

    Корпоративная культура — это модель поведения внутри организации, сформированная в процессе функционирования компании и разделяемая всеми членами коллектива. Это некая система ценностей, нормы, правила, традиции и принципы, по которым живут сотрудники.

21. **Что такое проектная деятельность**.

    Процесс решения проблем и задач, поставленных в проекте.

22. **Что такое проект.**

    Проект – это деятельность по достижению нового результата в рамках установленного времени с учетом определенных ресурсов. Описание конкретной ситуации, которая должна быть улучшена, и конкретных методов по ее улучшению. Описание конкретной ситуации, которая должна быть улучшена, и конкретных шагов по её реализации.

23. **Управление проектами – стейкхолдеры**.

    Стейкхолдеры (или заинтересованные лица) – это группы людей или отдельные люди, которых проект как-то затрагивает (как в хорошем, так и в плохом смысле) либо (и это важно, про это почему-то часто забывают!) те, кого проект не затрагивает, но они сами могут его «затронуть» или как-то на него повлиять, используя имеющиеся у них возможности. (Спонсоры, инвесторы, подрядчики, менеджеры, конечный потребитель)

24. **Управление проектами – границы проектов**.

    Границы Проекта - данные о работах или событиях, являющихся началом и окончанием Проекта, а также о работах, входящих и не входящих в Проект. Границы Проекта определяют рамки ответственности Участников Проекта

25. **Управление проектами – чем можно управлять при создании ПО**.

    Среда проекта, формулировка проекта, планирование проекта, техническое исполнение проекта, контроль над выполнением проекта.

26. **Фронтэнд, бекэнд**.

    Front-end — интерфейс взаимодействия между пользователем и основной программно-аппаратной частью (back-end).

27. **Что такое «архитектура ПО».**

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
- соединение выбранных элементов структуры и поведения во всё более крупные системы;
- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

28. **Почему при разработке ПО важно учитывать множественность точек зрения**.

Поскольку при разработке ПО на разных этапах на передний план выходят разные аспекты конечного продукта, важно принимать во внимание многие точки зрения для достижения наиболее оптимального результата как со стороны функционала, так и со стороны реализации.

29. ***Кто участвует в разработке ПО, кроме программистов, кто участвует на стороне заказчик***.

со стороны разработчика: тестировщики, дизайнеры, PM, руководитель группы, аналитики, архитекторы, билд-инженеры, проектировщики. Со стороны заказчика: тендер-менеджер, контрактный управляющий либо целая контрактная служба

30. ***Моделирование в процессе создания ПО***

UML (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. UML является языком широкого профиля, это — открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем. UML не является языком программирования, но на основании UML-моделей возможна генерация кода.

31. **В чем проблемы во взаимодействии заказчика и разработчика**.

Ожидания от процесса взаимодействия. Когда будет оплата, когда будет возможно проверить продукт. Критерии качества проекта – понимание заказчиком о том, что является качественным продуктом и правильное донесение этой информации разработчику. Понимание цели создания ПО, заказчик должен дать четко понять зачем ему нужен данный продукт. Совпадение плоскостей навыков, терминов и понятий общения. Понимание между людьми.

32. **Основные задачи аналитика требований ПО**.

Коммуникация с заказчиком, приведение разговора в нужное русло, формализация требований и пожеланий клиента, предложение оптимального решения в случае растерянности заказчика, анализ новых требований на существующие архитектуры и функционал, документация требований, их согласование и утверждение, отслеживание выполнения требований, верхнеуровневый контроль соответствия реализованного функционала требованиям, управление изменениями требований, участие в сдаче продукта.

33. **Для чего нужно проектировать базы данных. О: Основные задачи проектирования баз данных:**
    - Обеспечение хранения в БД всей необходимой информации.
    - Обеспечение возможности получения данных по всем необходимым запросам.
    - Сокращение избыточности и дублирования данных.
    - Обеспечение целостности базы данных.
34. **Что происходит на этапе сбора требований**.

    Сбор требований — общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области.
35. **Что происходит на этапе анализа и планирования. Каков результат этого этапа**.

    Анализ требований — определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем; выявление взаимосвязи требований.
36. **Что проектируют на этапе проектирования. О: В рамках данного этапа стороны должны осуществить**:
    - оценку результатов проведенного первоначально анализа и выявленных ограничений;
    - поиск критических участков проекта;
    - формирование окончательной архитектуры создаваемой системы;
    - анализ необходимости использования программных модулей или готовых решений сторонних разработчиков;
    - проектирование основных элементов продукта — модели базы данных, процессов и кода;
    - выбор среды программирование и инструментов разработки, утверждение интерфейса программы, включая элементы графического отображения данных;
    - определение основных требований к безопасности разрабатываемого ПО.
37. **Что проверяют на этапе тестирования.**

    Процесс тестирования позволяет смоделировать ситуации, при которых программный продукт перестает функционировать.
38. **Что такое сопровождение системы. Что исправляется на этом этапе, а что не исправляется.**

    Сопровожде́ние (поддержка) программного обеспечения — процесс улучшения, оптимизации и устранения дефектов программного обеспечения(ПО) после передачи в эксплуатацию. В общем случае процесс сопровождения состоит из следующих задач:
    - устранение сбоев;
    - улучшение дизайна;
    - расширение функциональных возможностей;
    - создание интерфейсов взаимодействия с другими (внешними) системами;
    - адаптация (например, портирование) для возможности работы на другой (или обновленной) аппаратной платформе, применение новых системных возможностей, функционирование в среде обновленной телекоммуникационной инфраструктуры и т.п.;
    - миграция унаследованного (legacy) программного обеспечения; вывод программного обеспечения из эксплуатации.

39. **Чем отличаются итерационные и каскадные модели от водопадных. Какие недостатки этих подходов**.

    Итерационная – наличие обратной связи с прошлыми этапами производства, снижается прогнозируемость. Каскадная – подтверждение каждого этапа на соответствие требованиям, сохраняются этапы возврата.

40. **Что такое верификация (требований, архитектуры)**.

    Верификация – проверка на совпадение требований, архитектуры и технического задания.

41. **Что такое методология прототипирования**.

    Прототипи́рование программного обеспечения (от англ. prototyping) — этап разработки программного обеспечения (ПО), процесс создания *прототи́па программы* — макета (черновой, пробной версии) программы, обычно — с целью проверки пригодности предлагаемых для применения концепций, архитектурных и/или технологических решений, а также для представления программы заказчику на ранних стадиях процесса разработки.

42. **Достоинства и недостатки методологии прототипирования**.

    Основными преимуществами прототипирования являются сокращение времени и стоимости разработки за счёт того, что оценка прототипа позволяет на более ранних стадиях обнаружить недостаточность или несоответствие требований. Чем позднее проводятся изменения в спецификации, тем они дороже, поэтому уточнение «чего же пользователи/заказчики хотят *на самом деле*» на ранних стадиях разработки снижает общую стоимость. Психологически важную роль играет также вовлечение заказчика в процесс разработки. Работа с прототипом позволяет будущим пользователям увидеть, как будет выглядеть будущая программа, и повлиять на её поведение, что уменьшает расхождения в представлении о программе между разработчиками и пользователями. Снижается и эффект почти неизбежного отторжения новой системы при внедрении, особенно когда она внедряется на место ранее использовавшейся. Тем не менее, использование прототипирования создаёт ряд дополнительных рисков.

    - Риск недостаточного анализа. Концентрация усилий на ограниченном прототипе может отвлекать разработчиков от надлежащего анализа требований на полную систему.
    - Риск смешения прототипа и готовой системы в представлении пользователей. Пользователи могут подумать, что прототип, который в действительности предполагается «выбросить», и есть основа будущей системы. Это может привести к двум противоположным по сути негативным эффектам. Во-первых, пользователи могут ожидать от прототипа более точного поведения и, не обнаружив его, разочароваться в возможностях разработчиков. Во-вторых, наличие быстро созданного прототипа может создать впечатление того, что «почти вся работа уже сделана», что может стимулировать попытки неоправданно сократить время и/или бюджет разработки.
    - Риск потери времени на создание прототипа. Ключевое свойство прототипа — то, что он создается быстро. Если разработчики тратят время на создание слишком сложного и функционального прототипа, они теряют преимущества от применения прототипирования вообще.

43. **Что такое инкрементная модель проектирования**.

    Итеративный инкрементный подход основывается на базовом формальном описании системы, дающем возможность создать первую

исполняемую функциональную модель. Полученная модель проверяется на соответствие описанию системы, а затем расширяется далее, последовательно преобразуясь в новые модели, в которых отражается увеличение требований к системе и уточнение деталей их реализации. Процесс продолжается до трансформации модели в реальную программную систему.

44. **Что такое риски программных продуктов, какие бывают риски**.

Риски программных продуктов – опасности, которые могут помешать соответствию продукта требованиям заказчика, а также его успешной реализации. Риски программных проектов можно разделить на четыре основные категории:

- Риски, связанные с требованиями
- Технологические риски
- Риски, связанные с квалификацией персонала
- Политические риски

45. **Что такое быстрая разработка приложений. Когда они используются?**

RAD — концепция организации технологического процесса разработки программных продуктов, ориентированная на максимально быстрое получение качественного результата в условиях сильных ограничений по срокам и бюджету и нечётко определённых требований к продукту. Эффект ускорения разработки достигается путём использования соответствующих технических средств и непрерывного, параллельного с ходом разработки, уточнения требований и оценки текущих результатов с привлечением заказчика. Применяется при сжатых сроках на выполнение, нечетких требованиях к ПО, при ограниченности бюджета, если GUI – главный фактор, если есть необходимость в разбиении на функциональные компоненты, при нихкой вычислительной сложности ПО.

46. **Что такое RUP – унифицированный процесс проектирования.**

Rational Unified Process – это методология создания программного обеспечения, оформленная в виде размещаемой на Web базы знаний, которая снабжена поисковой системой. Особенностью RUP является то, что в результате работы над проектом создаются и совершенствуются модели. Вместо создания громадного количества бумажных документов, RUP опирается на разработку и развитие семантически обогащенных моделей, всесторонне представляющих разрабатываемую систему. RUP – это руководство по тому, как эффективно использовать UML. Стандартный язык моделирования, используемый всеми членами группы, делает понятными для всех описания требований, проектирование и архитектуру системы.

47. **Какие бывают артефакты в RUP.**

Основные артефакты в RUP – модель, элемент модели, документ, исходный код, исполняемая программа.

48. **Что такое риски проекта, какие риски проекта Вы можете назвать**.

Риск проекта – это неопределенное событие или условие, которое в случае возникновения имеет воздействие (позитивное или негативное) по меньшей мере на одну из целей **проекта**, например, сроки, стоимость, содержание или качество.

49. **Время, стоимость и функционал – почему эти требования противоречат друг другу**.

Чем больше времени длится разработка, тем больше денег требуется на её поддержание, чем больше требуемый функционал, тем больше денег и

времени требуется на его разработку, чем ниже бюджет, тем меньший функционал возможно сделать.

50. **Что такое «словарь» проекта**. -
51. **Что описывает язык UML**.

Параметры и модель будущего продукта

52. **Что такое требования, кто их формирует**.

Требования к программному обеспечению — совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований. Требования могут выражаться в виде текстовых утверждений и графических моделей. *Источники требований:*

- Федеральное и муниципальное отраслевое законодательство (конституция, законы, распоряжения)
- Нормативное обеспечение организации (регламенты, положения, уставы, приказы)
- Текущая организация деятельности объекта автоматизации
- Модели деятельности (диаграммы бизнес-процессов)
- Представления и ожидания потребителей и пользователей системы
- Журналы использования существующих программно-аппаратных систем
- Конкурирующие программные продукты

53. **Что такое функциональные требования, приведите примеры**.

Функциональные требования объясняют, что должно быть сделано. Они идентифицируют задачи или действия, которые должны быть выполнены. Функциональные требования определяют действия, которые система должна быть способной выполнить, связь входа/выхода в поведении системы.

54. **Что такое нефункциональные требования, приведите примеры.**

Нефункциональные требования — требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы. Например, производительность, удобство сопровождения, расширяемость, надежность, факторы эксплуатации.

55. **Приведите примеры управления техническими ресурсами в процессе создания ПО**.

Обновление и заказ нового оборудования для рендера и тд.

56. **Как происходит внедрение проекта – альфа- и бета- версии**.

Бета - версия подукта, которую мы можем отдать внешним тестерам (заказчик) альфа - можем использовать только во внутреннем тестировании

57. **Гибкие методологии проектирования – что это**?

Гибкая методология разработки , agile-методы — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля.

58. **Что такое экстремальное программирование**.

Экстремальное программирование (XP) – это упрощенная методология организации разработки программ для небольших и средних по размеру

команд разработчиков, занимающихся созданием программного продукта в условиях неясных или быстро меняющихся требований.

59. **За какое максимальное время должна появляться готовая версия в экстремальном программировании.**

    Один месяц

60. **Что такое «парное программирование».**

    Одновременная работа двух программистов над кодом, один пишет, второй проверяет написанный код

61. **Что такое бэклог**.

    Бэклог - это журнал оставшейся работы, которую необходимо выполнить команде. Термин пришел из семейства методологий Agile, в частности из Scrum, где он является одним из основных артефактов - источником пользовательских историй.

62. **Что такое спринт (в скраме).**

    Спринт — это временной отрезок длительностью месяц или меньше, в течение которого создается «готовый», то есть пригодный к использованию и выпуску Инкремент продукта.

63. **Сколько времени длится спринт, что является завершением спринта**. *Месяц или менее.*

64. **Как часто происходит встреча разработчиков (в скраме)**.

    Ежедневный –ти минутный SCRUM митинг. -

65. **Роли участников (в скраме).**

    В Скраме всего три роли, но они не согласуются с привычными для нас должностями.

    - Владелец Продукта — носитель вижена (общей картины) продукта.
    - Скрам Мастер — помогает команде эффективно использовать Скрам в разработке продукта.
    - Команда разработки — разрабатывает продукт.

66. **Как организован процесс работы в скраме. -**

67. **Приведите примеры некорректных требований к ПО**.

    Требования не дают в достаточной степени понятие о конечной цели проекта, плохо описаны условия и возможности, которые должны быть доступны конечному пользователю, плохо сформированы задачи проекта…

68. **Что такое «качество» при создании требований**.

    Степень соответствия требований критериям качества требований.-

69. Какие бывают ограничения при создании требований.

70. Назовите характеристики качества (их всего 6).

- Функциональность (Functionality)
- Надежность (Reliability)
- Практичность (Usability)
- Эффективность (Efficiences)
- Сопровождаемость (Maintainability)
- Мобильность (Portability

71. **Кто такие заказчики и пользователи систем.**

Заказчик:

- Инициирует разработку
- Участвует в сборе требований
- Участвует в разработке спецификации требований
- Принимает результаты разработки

Пользователь:

- Не является заказчиком проекта
- Может являться, а может и не являться сотрудником проекта
- Является главным потребителем проекта
- Обычно существуют группы пользователей проекта

72. **Насколько большой документ – документация требований. Какие программные продукты считаются наиболее сложными**.
73. **Способы сбора требований**.
    Исследования, интервью, Семинар, Прототипирование, use case сценарии.
74. **Что такое use case диаграммы**.
75. Описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.