ChatGPT

1. **Понятие жизненного цикла ИС.** — Жизненный цикл информационной системы – это период времени от момента решения создать систему до полного прекращения её эксплуатации. В него обычно входят последовательные стадии: планирование (предпроектное исследование), анализ и определение требований, проектирование, разработка и тестирование, внедрение системы и её дальнейшая эксплуатация.

2. **Жизненный цикл в методологии проектирования ИС.** — В методологии проектирования информационных систем процесс создания описывается как жизненный цикл, разбитый на упорядоченные стадии. Для каждой стадии определены свои задачи, результаты, методы и исполнители, что позволяет планировать и организовывать коллективную работу над проектом и эффективно управлять разработкой.

3. **Основные стандарты жизненного цикла ИС.** — К основным стандартам и методологиям жизненного цикла можно отнести:

4. **ГОСТ 34.601-90** – отечественный стандарт, устанавливающий стадии и этапы создания автоматизированных систем (фактически соответствует каскадной модели).
5. **ISO/IEC 12207:1995** – международный стандарт процессов жизненного цикла программного обеспечения (определяет состав основных, вспомогательных и организационных процессов без жёсткого деления на фазы).
6. **Rational Unified Process (RUP)** и **Microsoft Solution Framework (MSF)** – методологии, предлагающие итерационный жизненный цикл с четырьмя фазами разработки.

7. **Extreme Programming (XP)** – гибкая методология экстремального программирования, предполагающая командную работу, постоянное взаимодействие с заказчиком и непрерывное прототипирование.

8. **Что включает в себя полный жизненный цикл ИС?** — Полный жизненный цикл информационной системы включает совокупность стадий, через которые проходит система от начала до окончания её использования:

9. **Планирование** – предварительное обследование, анализ целей и обоснование создания системы.
10. **Анализ требований** – сбор и формализация функциональных требований, составление технического задания.
11. **Проектирование** – разработка архитектуры и детализация решений системы.
12. **Реализация и тестирование** – программирование системы, отладка и проверка её работоспособности.

13. **Внедрение и эксплуатация** – установка системы в рабочую среду, обучение пользователей, сопровождение и использование системы в повседневной деятельности.

14. **CASE-средство как организация процессов ЖЦ. Что такое CASE-средство? Какие процессы ЖЦ включает в себя CASE-средство?** — CASE-средства (Computer-Aided Software Engineering) — это программные инструменты для автоматизации разработки и сопровождения программных систем. Современное CASE-средство охватывает многие процессы жизненного цикла ИС, включая:

15. анализ и моделирование системы (построение диаграмм требований и архитектуры);
16. проектирование и генерацию исходного кода приложений;
17. тестирование и отладку программных компонентов;
18. автоматическое документирование проекта;

19. сопровождение, внесение изменений и управление конфигурацией системы.

20. **Четыре стадии ЖЦ согласно методологии Rational Software.** — В методологии Rational Unified Process жизненный цикл разделён на 4 фазы: **начало** (inception), **исследование** (elaboration), **построение** (construction) и **внедрение** (transition). Эти стадии итерационно повторяются при разработке, проходя цикл за циклом для выпуска новых версий системы.

21. **Понятие модели жизненного цикла.** — Модель жизненного цикла – это абстрактное описание процесса разработки и сопровождения системы, представленное в виде упорядоченной совокупности этапов. Иными словами, модель ЖЦ определяет, какие стадии и в каком порядке проходит проект, и какие работы выполняются на каждом этапе создания программной системы.

22. **Каскадная модель жизненного цикла. Понятие, особенности и основные этапы.** — Каскадная (водопадная) модель жизненного цикла – это последовательный способ разработки, в котором каждая фаза проекта выполняется один раз и полностью завершается перед началом следующей. Особенность каскадной модели в том, что возврат к предыдущим этапам затруднён, поэтому требования должны быть тщательно собраны и зафиксированы в начале. Основные этапы каскадной модели совпадают с типовыми стадиями разработки: анализ требований, проектирование системы, реализация (кодирование), тестирование, внедрение и последующее сопровождение программного продукта.

23. **Основные достоинства каскадной модели.** — Достоинства каскадной модели:

24. Простота и чёткость процесса – последовательные шаги легко планировать и контролировать.
25. Требования фиксируются в начале и остаются стабильными в течение разработки, что упрощает управление.
26. Поэтапная документация – на каждом этапе формируется полный пакет документов, облегчающий понимание проекта и передачу его между командами.

27. Структура модели понятна заказчику: наличие формальных этапов с результатами позволяет отслеживать прогресс разработки.

28. **Основные недостатки каскадной модели.** — Недостатки каскадной модели:

29. Жёсткая последовательность этапов затрудняет внесение изменений: если требования поменялись, вернуться назад очень сложно.
30. Заказчик получает готовую систему только в конце проекта, поэтому обратная связь приходит слишком поздно.
31. Риски обнаруживаются на поздних стадиях (например, проблемы выявляются лишь на этапе тестирования).

32. Не подходит для нечетко определённых или изменчивых требований – отсутствует гибкость при изменении условий.

33. **Спиральная модель жизненного цикла. Понятие, особенности и основные этапы.** — Спиральная модель – итеративная модель разработки, изображающая процесс в виде спирали, где каждый виток соответствует одному циклу создания продукта с уточнением требований. Особенность этой модели в том, что на каждом витке последовательно выполняются этапы планирования, анализа рисков, разработки (реализации) и оценки результатов, после чего принимается решение о целесообразности продолжения проекта. Основные этапы (анализ, разработка, тестирование и т.д.) повторяются на каждой итерации, что позволяет постепенно улучшать проект, учитывая новые требования и устраняя выявленные риски.

34. **Основные достоинства спиральной модели.** — Достоинства спиральной модели:

35. Гибкость и адаптивность: итерационный процесс позволяет вносить изменения в требования или дизайн на каждом новом витке разработки.
36. Управление рисками: на каждой итерации проводится анализ и оценка рисков, что помогает выявлять и решать проблемы на ранних этапах.

37. Постепенное построение системы: заказчик может видеть прототипы на промежуточных этапах, давая обратную связь и уменьшая вероятность несоответствия конечного продукта ожиданиям.

38. **Основные недостатки спиральной модели.** — Недостатки спиральной модели:

39. Сложность и затратность: требуются опыт в анализе рисков и тщательное планирование, что увеличивает расходы и длительность работ для небольших проектов.
40. Трудность определения границ проекта: заранее не всегда понятно, сколько потребуется итераций, поэтому сложно точно спланировать сроки и бюджет.

41. Возможна избыточность процесса для простых задач: для маленьких проектов использование спиральной модели может быть чрезмерно сложным и неоправданным по затратам.

42. **Основные процессы жизненного цикла и их спецификация.** — К основным процессам жизненного цикла информационной системы относятся ключевые стадии разработки и эксплуатации, каждый из которых имеет свои задачи и результаты. В их число входят:

43. **Формирование требований и технического задания** – сбор и анализ требований, формулирование целей проекта, составление официального ТЗ.
44. **Проектирование** – разработка проектных решений: архитектура системы, моделирование данных и процессов, подготовка дизайна.
45. **Разработка и тестирование** – непосредственное создание программного кода, интеграция модулей и проверка работоспособности системы.
46. **Внедрение и эксплуатация** – установка системы, обучение пользователей, начало промышленной эксплуатации.

47. **Сопровождение** – поддержка работоспособности, исправление ошибок, модификация и обновление системы в ходе её эксплуатации.

48. **Вспомогательные и организационные процессы жизненного цикла.** — Вспомогательные процессы ЖЦ обеспечивают поддержку основных этапов разработки. К ним относят, например: документирование проекта, управление конфигурацией

(версиями и изменениями), контроль качества (тестирование, верификация, аудит), управление проблемами и поддержку пользователей. Организационные процессы связаны с управлением самой разработкой: это **управление проектом** (планирование, контроль сроков и ресурсов), создание и обслуживание инфраструктуры проекта, совершенствование процессов разработки, обучение команды и другие процессы, направленные на эффективную организацию работы и улучшение самого жизненного цикла.

49. **Стадии жизненного цикла.** — В жизненном цикле информационной системы традиционно выделяют следующие стадии:

50. **Предпроектная** – исследование текущей ситуации, анализ предметной области, обоснование необходимости новой системы.
51. **Техническое задание** – формулирование требований к системе, утверждение целей и условий разработки в официальном документе.
52. **Проектирование** – разработка архитектуры и деталей системы (концептуальный и технический проекты, создание схем и моделей).
53. **Реализация и тестирование** – написание программного кода, сборка системы и проверка её корректной работы на соответствие требованиям.

54. **Внедрение и сопровождение** – установка системы у заказчика, опытная и промышленная эксплуатация, поддержка пользователями и дальнейшее обслуживание системы.

55. **Управление качеством информационных систем и парадигма «философия качества информационных систем». Составляющие категории парадигмы.** — Управление качеством ИС – это процесс обеспечения того, чтобы информационная система удовлетворяла установленным требованиям и ожиданиям по надежности, функциональности и другим показателям на всех этапах её жизненного цикла. В рамках парадигмы «философия качества ИС» выделяют три ключевые категории понятий:

56. **Свойства ИС** – характеристики, отражающие сущность, содержание и ценность системы (признаки, параметры качества системы).
57. **Структура ИС** – способ организации элементов системы и их взаимодействия (архитектура, набор компонентов, методов и средств, определяющих внутреннее устройство системы).

58. **Закономерности процессов** – принципы и закономерности создания, функционирования и эволюции системы во времени, от которых зависит изменение её качества (влияние внешних и внутренних факторов на качество ИС).

59. **Понятие «качество информации» и понятие «качество ИС».** — *Качество информации* характеризует сами данные и степень их полезности: это совокупность свойств информации, таких как точность, достоверность, актуальность, полнота, понятность и ценность для пользователя. *Качество информационной системы* относится к системе в целом и отражает её способность эффективно решать поставленные задачи: сюда входят показатели надежности, функциональности, производительности, удобства использования, безопасности и другие свойства системы, влияющие на удовлетворенность пользователей.

60. **Процедура оценки качества ИС.** — Оценка качества информационной системы проводится с помощью специальных методик (например, квалиметрии) по набору критериев. Сначала определяют важные показатели качества (функциональность, производительность, надежность, удобство и др.), затем измеряют их для данной системы (через тестирование, анализ метрик). Полученные значения сравниваются с требуемыми (нормативами, стандартами или ожиданиями заказчика), и на основе этого выносится заключение о степени соответствия системы заданным требованиям качества.

61. **Понятие проекта и особенности проектной работы.** — Проект представляет собой комплекс взаимосвязанных работ, ограниченных по времени и ресурсам, направленных на создание уникального продукта или услуги. Проектная работа характеризуется четко определенными целями, уникальностью результата, наличием календарного начала и конца, а также ограничениями по бюджету и срокам. В проекте задействована команда специалистов, и успех достигается за счет планирования, координации действий участников и эффективного управления ресурсами.

62. **Процесс управления проектами.** — Управление проектом включает планирование и организацию всех этапов работ, координацию команды и контроль над исполнением задач. Проект-менеджер определяет цели и scope проекта, разрабатывает план (график работ, бюджет), распределяет обязанности между исполнителями, следит за прогрессом и качеством, управляет рисками и при необходимости вносит коррективы, чтобы проект был выполнен в срок и удовлетворял требованиям заказчика.

63. **Классификация проектов.** — Проекты можно классифицировать по разным признакам, например:

64. **По сфере и назначению:** производственные, информационные (IT-проекты), научно-исследовательские, строительные, организационные и т.д.
65. **По масштабу и сложности:** небольшие (локальные) проекты, крупномасштабные (корпоративные или межорганизационные), типовые либо инновационные проекты.
66. **По длительности:** краткосрочные (несколько недель или месяцев), среднесрочные, долгосрочные проекты (несколько лет).
67. **По значимости и эффекту:** рядовые/операционные проекты, стратегические проекты (влияющие на развитие компании).

68. **По составу и охвату:** монопроекты (в рамках одной организации) vs. мульти-проекты/программы (состоящие из множества взаимосвязанных проектов), внутренние vs. внешние проекты.

69. **Основные фазы проектирования информационной системы и их особенности.** — Процесс проектирования ИС подразделяется на несколько фаз, каждая из которых имеет свои результаты:

70. **Техническое задание (ТЗ):** формулирование целей и требований к системе совместно с заказчиком, документальное закрепление этих требований.
71. **Эскизное проектирование:** разработка концепции системы – общей структуры, основных функциональных решений, выбор платформ и технологий.
72. **Техническое проектирование:** детальное проектирование системы – создание подробных схем, моделей данных, алгоритмов, подготовка технической документации для реализации.

73. **Рабочее проектирование (рабочая документация):** выпуск конечных рабочих чертежей, спецификаций и инструкций, по которым будут вести программирование и внедрение.
    (На каждой фазе разрабатываются свои документы, и по результатам фаз проходит согласование с заказчиком перед переходом к следующему этапу.)

74. **Задачи методологии проектирования ИС.** — Методология проектирования корпоративных информационных систем призвана решить несколько основных задач:

75. Обеспечить соответствие создаваемой системы целям предприятия и требованиям автоматизации бизнес-процессов (то есть, чтобы функционал системы покрывал нужды организации).
76. Гарантировать достижение результатов проекта в заданные сроки и бюджет с требуемым качеством.
77. Облегчить сопровождение, модификацию и наращивание системы в будущем, чтобы она могла адаптироваться к изменениям в деятельности предприятия.
78. Обеспечить открытость, переносимость и масштабируемость создаваемой системы (совместимость с другими системами, возможность работы на разных платформах, рост вместе с бизнесом).

79. Позволить повторно использовать ранее внедрённые на предприятии программные и технические решения (интеграция с существующими базами данных, приложениями, оборудованием и сетями).

80. **Технология проектирования ИС: составляющие части, ресурсы, требования к технологии.** — Технология проектирования информационных систем включает в себя:

81. **Составные части процесса:** определённую последовательность этапов и операций проектирования, а также правила, методы и нотации (графические и текстовые), применяемые для описания системы.
82. **Ресурсы проектирования:** входные данные (результаты предыдущих этапов, требования), методические материалы и стандарты, инструментальные средства (CASE-программы, средства моделирования), технические средства и команда исполнителей.

83. **Основные требования к технологии:** поддерживать полный жизненный цикл системы (от разработки до сопровождения), обеспечивать достижение целей проекта с требуемым качеством в установленные сроки, а также допускать декомпозицию крупного проекта на подсистемы с параллельной работой команд над частями. Иными словами, технология должна быть эффективной, масштабируемой и стандартизированной, чтобы повышать продуктивность разработки.

84. **Методология RAD и ее основные особенности.** — Методология Rapid Application Development (RAD) ориентирована на максимально быструю разработку приложений за счёт использования инструментальных средств и особой организации процесса. Характерные особенности RAD:

85. Итеративная модель разработки с короткими циклами (спринтами), похожая на спиральный подход.
86. Активное участие заказчика и будущих пользователей на всех этапах (постоянная обратная связь).

87. Использование эффективных инструментов разработки и CASE-средств для автоматизации большинства этапов.
88. Широкое прототипирование: быстрое создание рабочих прототипов, позволяющее уточнять требования.

89. Небольшая и высококвалифицированная команда, сжатые сроки проекта и интенсивное управление, нацеленное на быстрый результат.

90. **RAD как инструмент разработки приложений.** — В методологии RAD выделяются три ключевых элемента организации разработки приложения:

91. **Небольшая команда** разработчиков (примерно 2–10 человек), обладающих достаточным опытом.
92. **Короткие сроки** и жёсткий график проекта (обычно от 2 до 6 месяцев), рассчитанные на быстрое достижение результата.

93. **Итерационный процесс с тесным участием заказчика:** требования уточняются по ходу разработки, заказчик регулярно взаимодействует с командой, проверяя прототипы и направляя развитие системы.

94. **Объектно-ориентированный подход к созданию приложений.** — Объектно-ориентированный подход предполагает построение приложения из совокупности объектов, каждый из которых объединяет данные и методы для их обработки. В контексте RAD это означает, что система проектируется как набор взаимодействующих компонентов (объектов), моделирующих реальные сущности. Такой подход упрощает разработку: объекты можно повторно использовать, легко модифицировать, а сходство с реальными объектами бизнеса облегчает понимание системы и уменьшает разрыв между предметной областью и кодом.

95. **Визуальное программирование.** — Визуальное программирование – способ разработки, при котором программист создаёт интерфейсы и логику приложения с помощью графических инструментов, а не только кода. Разработчик буквально «собирает» программу из визуальных компонентов (окон, кнопок, форм, блок-схем), задаёт их свойства и связи, а среда разработки автоматически генерирует за кулисами необходимый исходный код. Это делает процесс создания приложений более наглядным и быстрым, снижая порог вхождения и вероятность ошибок.

96. **Событийное программирование.** — Событийно-ориентированное (event-driven) программирование – парадигма, в которой выполнение программы управляется событиями. Логика приложения разбивается на обработчики событий: программа реагирует на действия пользователя или другие события (например, нажатие кнопки, ввод данных, наступление таймера) вызовом соответствующих функций-обработчиков. Такой подход особенно характерен для графических интерфейсов и позволяет создавать интерактивные приложения, где компоненты работают автономно и активируются по мере событий.

97. **Фазы жизненного цикла в рамках методологии RAD.** — Жизненный цикл проекта в методологии RAD делится на **четыре фазы**:
    1) **Анализ и планирование требований** – определение функций будущей системы, сбор приоритетных требований совместно с пользователями, оценка возможности реализации в заданных ресурсных рамках.

2) **Проектирование** – быстрая разработка архитектуры и функциональности, создание и улучшение прототипов приложения с использованием CASE-средств при активном участии пользователей.
3) **Построение (разработка)** –31. **Фазы жизненного цикла в рамках методологии RAD.** — Жизненный цикл проекта в методологии RAD делится на **четыре фазы**:
1) **Анализ и планирование требований** – определение функций и требований будущей системы, совместная с пользователями проработка бизнес-потребностей, оценка масштаба проекта и возможностей реализации в заданных ресурсах.
2) **Проектирование** – быстрая разработка архитектуры и функционала: создание прототипов приложения с использованием CASE-средств, анализ и при необходимости корректировка моделей, уточнение требований на основе обратной связи от пользователей.
3) **Построение (разработка)** – непосредственное создание приложения: программирование модулей, интеграция компонентов, параллельное тестирование и доработка системы в итеративном режиме.
4) **Внедрение** – установка готовой системы в рабочую среду, перенос данных, обучение пользователей, начало опытной эксплуатации и дальнейшая передача системы в промышленную эксплуатацию.

98. **Недостатки и ограничения методологии RAD.** — Ограничения методологии RAD:

99. Не подходит для очень крупных или критически важных проектов: при большом масштабе сложно уложиться в короткие циклы, а также тяжело обеспечить требуемую надёжность и безопасность.
100. Требует постоянной вовлечённости заказчика и конечных пользователей; если обратная связь затруднена, эффективность RAD снижается.
101. Опирается на высококвалифицированную, небольшую команду разработчиков — недостаток опыта или дисциплины у участников может привести к снижению качества продукта.

102. Акцент на быстрых результатах иногда приводит к упрощению архитектуры и недостаточному вниманию к документации, что может осложнить поддержку системы в будущем.

103. **Понятие и применение открытых ИС. — Открытая информационная система** – это система, построенная на основе открытых стандартов и протоколов, обеспечивающая совместимость с другими программными и техническими платформами. Применение открытых ИС позволяет легко интегрировать систему с внешними компонентами и другими приложениями, независимо от производителя, и упрощает масштабирование и модернизацию: новую функциональность можно добавлять без кардинальной переделки системы благодаря унифицированным интерфейсам и форматам данных.

104. **Понятие профиля ИС. — Профиль информационной системы** – это согласованный набор стандартов и технических решений, выбранных для данного типа информационных систем или конкретного проекта. Проще говоря, профиль ИС определяет специфику платформы и технологий: какие стандарты, протоколы, программные и аппаратные требования используются при создании и внедрении системы, чтобы обеспечить совместимость и соблюдение нормативов.

105. **Классификация профилей ИС.** — Профили информационных систем обычно делят на два основных вида:

106. **Профиль конкретной системы** – набор стандартизованных решений для реализации одной информационной системы в рамках определённого проекта (все стандарты и технологии, принятые внутри данного проекта).

107. **Отраслевой (типовой) профиль** – унифицированный набор стандартов для информационных систем определённого класса или прикладной области (решения, применимые ко всем системам данного класса задач).
(Также профили могут различаться по охвату: например, профиль аппаратно-программной платформы, профиль безопасности, профиль взаимодействия и т.д., но базовое разделение – по индивидуальному или типовому применению.)

108. **Принципы формирования профиля ИС.** — При формировании профиля информационной системы преследуются следующие основные цели:

109. **Снижение трудоёмкости** разработки и внедрения за счёт использования готовых стандартизованных решений.
110. **Повышение качества** компонентов и решений системы благодаря опоре на проверенные стандарты и лучшие практики.
111. **Масштабируемость системы** – заложенные в профиле стандарты должны обеспечивать возможность расширения и наращивания функциональности ИС.
112. **Интегрируемость** – профиль предусматривает совместимость и обмен данными между различными модулями и подсистемами, в том числе ранее независимыми.

113. **Переносимость ПО** – выбор стандартов должен обеспечивать работу системы на разных платформах и облегчать переход на новые технологии.

114. **Актуальность использования профилей ИС.** — В современных условиях профили ИС становятся актуальными по ряду причин:

115. Существующие международные и национальные стандарты в области ИТ разрознены и не полностью покрывают потребности сложных интегрированных систем. Профиль позволяет объединить нужные стандарты для конкретной системы.
116. Разработка и утверждение новых стандартов занимает годы, технологии за это время успевают уйти вперёд. Применение профиля помогает использовать уже имеющиеся стандарты гибко, не дожидаясь появления идеального нового стандарта.
117. Многие стандарты описывают только отдельные аспекты (например, технические протоколы, форматы данных), тогда как комплексные процессы создания и внедрения ИС остаются недостаточно формализованными. Профиль восполняет пробел, задавая единый каркас из разных стандартов под потребности проекта.
118. Благодаря профилям удаётся создавать совместимые, унифицированные решения: организации могут опираться на базовые профили (например, профиль открытых систем) при построении своих ИС, что снижает затраты и риски при интеграции и развитии информационных систем.
