ChatGPT

# Ответы на вопросы

## 1. Определение, виды и компоненты ЭИС

**Определение:** ЭИС (экономическая информационная система) – это информационная система, предназначенная для управления предприятиями и бизнеса [1]. Обычно под ЭИС понимают совокупность организационных, технических, программных и информационных средств, объединённых единой системой для сбора, хранения, обработки и выдачи необходимых данных [2].
**Виды:** Чаще всего выделяют шесть главных типов ЭИС: *ESS* (Executive Support Systems), *MIS* (Management Information Systems), *DSS* (Decision Support Systems), *KWS* (Knowledge Work Systems), *OAS* (Office Automation Systems) и *TPS* (Transaction Processing Systems) [3] [4]. Эти типы различаются по уровню управления и функционалу (стратегический, управленческий, операционный и т.д.).
**Компоненты:** В состав ЭИС входят четыре группы компонентов: организационные (методы и процедуры управления), технические (аппаратное обеспечение, сети), программные (операционные системы, прикладные программы) и информационные (базы данных) средства [2]. Иногда выделяют также человеческий фактор – персонал, обслуживающий систему.

## 2. Понятие и состав обеспечивающих подсистем ЭИС

Обеспечивающие подсистемы ЭИС – это общесистемные подсистемы, которые одинаковы для всех функциональных частей ЭИС и не зависят от предметной области. К ним относят подсистемы организационного, правового, технического, математического, программного, информационного, лингвистического и технологического обеспечения [5]. То есть это общие модули и службы: нормативно-методические документы, законодательство, техническая инфраструктура, прикладное ПО (утилиты), алгоритмические библиотеки, методики организации данных, языковые словари и пр., которые поддерживают работу всей ЭИС [5].

## 3. Классификация программного обеспечения

Программное обеспечение классифицируется по назначению и по областям применения. **По назначению** выделяют: системное ПО (ОС, драйверы, утилиты), прикладное ПО (приложения для решения задач пользователей) и инструментальное (средства разработки – компиляторы, CASE, библиотеки) [6]. **По области применения** – отраслевое, универсальное, встраиваемое и т.д. Можно также разделить по лицензии (проприетарное, свободное), уровню (низкоуровневое, высокоуровневое) и иным критериям. Но базовая классификация – на системное, прикладное и программные инструменты [6].

## 4. Эволюция и основные характеристики технологий разработки ПО

Классическими технологическими моделями развития ПО являются: линейная («водопадная»), инкрементная (итерационная с последовательной поставкой частей) и спиральная модели. **Линейная модель** предполагает последовательные фазы (анализ требований → проектирование → реализация → тестирование → внедрение) [7]. **Инкрементная модель** разбивает проект на серии «инкрементов», каждый из которых проходит полный цикл разработки и выпуска функционального фрагмента [8]. **Спиральная модель** сочетает итеративный подход с управлением рисками: каждый цикл (виток спирали) включает этапы планирования, анализа рисков, разработки и оценки результатов. Со временем развились гибкие (agile) методологии, но классические модели послужили базой. Основные характеристики: линейность и строгость фаз (водопад), итеративность и ранний выпуск (инкремент), ориентация на управление рисками и адаптивность (спиральная) [7] [8].

## 5. Классификация стандартов программной инженерии и основные международные стандарты

Стандарты можно классифицировать по уровню: *корпоративные* (внутренние документы компании), *отраслевые* (например, отраслевые ГОСТы), *государственные* (ГОСТ) и *международные/межгосударственные* (ISO/IEC, IEEE и др.) [9] [10]. Международные стандарты охватывают все ключевые области ПО: процессы жизненного цикла, качество, управление проектами, конфигурацию и т.д. Так, **ISO/IEC 12207** регламентирует процессы жизненного цикла ПО (приобретение, поставка, разработка, эксплуатация, сопровождение и завершение) [11]. **ISO/IEC 15288** – аналогичная модель процессов для систем (включая ПО) [12]. Для качества есть серия ISO/IEC 25000 (ранее 9126), для оценки процессов – ISO/IEC 15504 (SPICE) и модель CMMI. Существуют также **ISO 9001** (СМК), ISO/IEC 90003 (применение ISO9001 к ИТ), стандарты IEEE/ГОСТ на документацию (например, требования, тестирование, конфигурирование) и прочие. Международные стандарты обеспечивают унификацию подходов к жизненному циклу ПО и покрывают такие области как разработка, тестирование, сопровождение, менеджмент, безопасность, надежность и др.

## 6. Определение стандарта и сертификации организации по стандарту

**Стандарт** – это нормативный документ, созданный на основе согласия экспертов и утверждённый признанным органом (ISO, ГОСТ и др.), устанавливающий правила, требования или характеристики для обеспечения оптимального порядка в конкретной области [13]. Он описывает «что» и «как» должно делаться. **Сертификация** организации на соответствие стандарту означает, что независимый сертифицирующий орган проверил (аудит) и подтвердил, что процессы и продукты данной организации удовлетворяют требованиям этого стандарта [14] [15]. Итогом является выданный сертификат, который даёт уверенность заинтересованным сторонам (заказчикам, регуляторам и т.д.) в том, что организация следует установленным стандартам качества или управления [14] [15].

## 7. Международная практика разработки стандартов в программной инженерии

Международные стандарты ПО обычно разрабатываются комитетами ISO/IEC (например, JTC1/SC7 – «Системная и программная инженерия») и техническими обществами (IEEE, IEC). Так, **IEEE** начал выпускать стандарты с 1972 г. (первый – IEEE 730 «Критерии обеспечения качества ПО») и создал свод правил терминологии и процессов (IEEE Std 1012, 1220 и др.) [16]. В 1990–95 гг. проведена работа по созданию единого международного стандарта для жизненного цикла ПО: ISO/IEC 12207:1995 [17]. А уже в 1995–2000-х ISO/IEC и IEEE разработали и выпустили уточняющие стандарты (например, 15288 для систем, 15504/SPICE для оценки процессов). Международная практика – это консенсусный процесс: стандарты пишутся рабочими группами (участники – эксперты разных стран), обсуждаются и периодически пересматриваются [16] [17]. Кроме того, ACM и IEEE подготовили SWEBOK (Software Engineering Body of Knowledge) – справочник по дисциплине, а INCOSE – аналогичный для системной инженерии.

## 8. Определение и содержание CASE-технологий разработки ПО

**CASE-технологии** – это совокупность инструментов и методов автоматизации процессов проектирования и разработки информационных систем [18] [19]. Под CASE понимают программные средства, поддерживающие этапы жизненного цикла ПО (анализ требований, моделирование, проектирование, программирование, тестирование и др.). Так, CASE-инструменты позволяют строить различные модели (IDEF0, DFD, UML-диаграммы, ER-диаграммы БД, сценарии, документацию), генерировать код или проекты БД, управлять конфигурацией и т.д. [18] [19]. В зависимости от охвата они делятся на локальные (для решения одной задачи), частично интегрированные (объединяют несколько этапов) и полностью интегрированные системы, поддерживающие полный цикл разработки [20].

## 9. Элементы функциональных схем (блок-схем)

Функциональная схема (алгоритма) состоит из стандартных графических элементов. Основные элементы блок-схем: **«Действие»** (прямоугольник) – выполнение операции, **«Ввод/Вывод»** (параллелограмм) – прием или вывод данных, **«Предопределенный процесс»** (прямоугольник с двойными границами) – вызов подпрограммы, **«Условие/Вопрос»** (ромб) – точка ветвления по условию, **«Начало/Конец»** (овал) – начало или завершение алгоритма, **«Цикл»** (обычно повторяющаяся последовательность) и **«Соединитель»** (кружок) – склейка частей схемы. Эти элементы соединяются стрелками, показывающими порядок выполнения [21].

## 10. Определение и содержание технологического процесса проектирования ЭИС

**Технологический процесс проектирования ЭИС** – это регламентированная последовательность проектных действий (этапов и операций) с заранее определенными исполнителями, методами, средствами и ресурсами [22]. Проще говоря, это упорядоченный набор этапов (анализ требований, технико-экономическое обоснование, техническое и рабочее проектирование, внедрение, сопровождение), на каждом из которых выполняются проектные (вырабатывающие результаты) и

оценочные операции [23] [24]. Процесс включает планирование работ, сбор требований, разработку архитектуры и детальной проектной документации, проверку и согласование результатов, а также подготовку к вводу системы в эксплуатацию.

## 11. Классификация методов проектирования ЭИС

Методы проектирования ЭИС можно классифицировать по различным признакам:
- **По использованию типовых решений:** *оригинальные* (с нуля) и *типовые* (на базе готовых шаблонов, стандартных архитектур) [25].
- **По характеру адаптации готовых продуктов:** *перепрограммирование* (адаптация ПО путём модификации), *параметризация* (настройка конфигурации), *моделирование* (использование шаблонов с дальнейшим преобразованием) [26].
- **По степени автоматизации:** *с универсальными средствами* (общего назначения, например, Visual Studio, Eclipse) и *специализированными CASE-системами* (AllFusion, IBM Rational и др.) [27].
- **По основному подходу:** функционально-ориентированные (структурные) методы против объектно-ориентированных [28]. Структурные строят ИС через функциональную декомпозицию задач (например, SADT/IDEF), объектные – через выявление объектов и классов.

## 12. Цели, задачи и содержание моделирования предметной области при проектировании ЭИС

**Цель моделирования предметной области (ПОД)** – формализовать и наглядно представить ключевые объекты и процессы реальной бизнес-среды, чтобы обеспечить понимание требований к системе [29]. Это позволяет получить единую терминологию и концептуальную основу для дальнейшего проектирования. **Задачи:** собрать и упорядочить термины предметной области (словарь терминов), выявить доменные сущности (объекты, роли, события), их свойства и связи; смоделировать бизнес-процессы и взаимодействия. **Содержание:** построение *концептуальной модели* ПОД – обычно диаграмм классов UML или ER-диаграмм, где отображаются **концептуальные классы** (доменные объекты), их **атрибуты** и **ассоциации** [30]; а также описание сценариев (кейсов) взаимодействия объектов. Результат – «схема» предметной области и словарь терминов, на основе которых затем разрабатывается техническое задание и проект ИС [29] [30].

## 13. Основные инструментальные средства структурного анализа и проектирования

Для структурного анализа и проектирования применяются CASE-средства, поддерживающие методологии вроде IDEF0, DFD, IDEF3 и т.п. Например, **AllFusion Process Modeler** (ранее BPwin) позволяет строить IDEF0-, DFD- и IDEF3-диаграммы процессов [31]. Для моделирования данных по стандарту IDEF1X (ER-модель) используют **AllFusion Data Modeler** [31]. Кроме того, широко применяются: Microsoft Visio (универсальный редактор диаграмм), IBM/Rational System Architect, Sparx Enterprise Architect (с поддержкой UML и потоков данных), ERwin Data Modeler, Oracle Designer и др. Эти инструменты позволяют визуально конструировать функциональные схемы, диаграммы потоков данных, структуры баз данных и автоматически управлять связями между ними [31].

## 14. Базовые принципы методологии SADT

SADT (Structured Analysis and Design Technique) – методология функционального анализа. Её **базовые принципы** сводятся к тому, что: (1) все функции (процессы) системы изображаются в виде блоков (прямоугольников), (2) взаимодействия между функциями показываются стрелками, (3) интерфейсы системы также изображаются стрелками [32]. Иными словами, модель строится как «сетевой график» функций: каждая функция – это черный ящик, а стрелки задают входы, выходы и информационные связи. При этом используется иерархическая декомпозиция: сложные функции разбиваются на простые, чтобы обеспечить ясность описания системы [32] [33].

## 15. Назначение, основные понятия и структурные элементы стандарта IDEF0

**IDEF0** – язык функционального моделирования. Его предназначение – описывать функции (процессы) системы и потоки данных/управления между ними. **Основные понятия:** функция (деятельность) представляется прямоугольным блоком; её связи с окружающим миром показываются стрелками- *ICOM* (Input, Control, Output, Mechanism). Стрелка слева – *вход* данных, справа – *выход*, сверху – *управление* (параметры или указания), снизу – *механизм/ресурс* (средства выполнения) [34] [35]. **Структурные элементы:** иерархия диаграмм – сверху строится контекстная диаграмма (уровень 0, показывающая всю систему одной функцией), затем декомпозиция: каждый блок разворачивается в более детальную диаграмму. Важно ограничивать число блоков на диаграмме (обычно 3–6) [33]. Таким образом, модель IDEF0 – это контекстная диаграмма + серия подробных функциональных схем с соответствующими стрелками ICOM [34] [35].

## 16. Назначение, основные понятия и структурные элементы методики диаграмм потоков данных (DFD)

**DFD (Data Flow Diagram)** – нотация для моделирования потоков данных в системе. Ее задача – показать, как информационные потоки проходят через процессы и хранилища. **Основные понятия:** *Процесс* (обработка данных) изображается в виде круга или прямоугольника с закругленными углами; *поток данных* – стрелка; *внешняя сущность* (внешний источник/получатель) – прямоугольник; *хранилище данных* – две параллельные линии [36] [37]. Поток связывает либо внешнюю сущность с процессом, процесс с процессом, либо процесс с хранилищем. **Структурные элементы:** моделирование ведут по уровням. Уровень 0 (контекстная диаграмма) отображает систему как один процесс с внешними источниками/приёмниками [38]. Затем строятся диаграммы более высокого уровня (1, 2 и т.д.), где основной процесс дробится на подпроцессы (например, 1.1, 1.2,...), уточняя потоки данных между ними.

## 17. Сравнение технологий IDEF0 и DFD

IDEF0 и DFD – это разные подходы к функциональному анализу. **IDEF0** ориентирован на *функции (процессы)* и их связи: каждый процесс – прямоугольник с входными, управляющими, выходными и ресурсными стрелками [34] [35]. Модель строится сверху вниз по иерархии. **DFD** фокусируется на *потоках данных*: процессы (более грубо – кружки), внешние сущности и хранилища соединяются стрелками данных [36] [37]. Важное отличие: в IDEF0 выделены управляющие воздействия и

механизмы, в DFD их нет (только данные). При IDEF0 главный акцент – *что делает система*, при DFD – *как данные перемещаются и преобразуются*. IDEF0 предъявляет жесткие правила декомпозиции (по контекстной диаграмме) [33], DFD более гибок в делении на уровни. Наконец, как отмечено: в IDEF0 «прямоугольники изображают функции или процессы» [35], а в DFD – «блоки, в которых происходит преобразование данных» [37].

## 18. Понятие, использование и критерии окончания декомпозиции при моделировании процессов

**Декомпозиция** – это процесс детального разбиения функции (процесса) на более мелкие подпроцессы. Она используется для упрощения сложной системы: каждый процесс моделируют сначала в общих чертах, а затем уточняют, пока детальность не будет достаточной. Критерии окончания декомпозиции обычно определяются здравым смыслом и требованиями: дальнейшее разделение прекращают, когда нижестоящие процессы слишком «примитивны» или «тривиальны» (далее нет новой существенной информации) [39]. Другой признак: на диаграмме уже показаны все ключевые аспекты родительского процесса, и если разделить далее, получится избыточное дробление, а не дополнительные сведения [39] [40]. Например, если процесс выполняется одним человеком за очень короткое время или представляет собой «атомарную» операцию, то его нецелесообразно дробить.

## 19. Назначение, основные понятия и структурные элементы стандарта IDEF3

**IDEF3** – методика описания процессов («описание сценариев»). Она предназначена для документирования порядка выполнения процессов и их причинно-следственных связей. **Основные понятия:** *Элемент работы (Work)* – выполняемая операция/событие (блок), *единство поведения (Unit of Behavior, UOB)* – совокупность совместно происходящих работ, *объект* – участник процесса, *ситуация/событие*. IDEF3 поддерживает два вида диаграмм: **описание процесса** (Process Flow Description, PFD) – где строится цепочка событий/действий, и **сеть состояний объектов** (Object State Transition Network, OSTN) – где рассматривается изменение состояний объектов. **Структурные элементы PFD:** блоки «Work» (работы), стрелки-предшественники и наследующие связи, **перекрёстки** (junctions) для развилки/слияния потоков, а также **объекты-ссылки** (object references) – пометки важных данных/ресурсов, не являющихся отдельным процессом [41] [42].

## 20. Назначение и использование перекрёстков и объектов-ссылок на диаграммах IDEF3

На диаграммах IDEF3 **перекрёстки (junctions)** служат для отображения логики разветвления или слияния потоков работ. Перекрёсток показывает, как один процесс может запускать несколько последующих (Fan-out) или как несколько процессов объединяются перед выполнением следующего (Fan-in) [43]. Существуют разновидности перекрёстков с «AND»/«OR» и синхронной/асинхронной логикой, чтобы задать условия одновременного ожидания или последовательного выполнения [43] [44]. **Объекты-ссылки (object references)** – это вспомогательные пометки (ярлыки) на диаграмме, которые используются для указания объектов, идей или данных, не представленных в виде отдельных блоков или стрелок. Они нужны, чтобы привлечь внимание к важному элементу модели

(например, к общему ресурсу, условию или событию) и обычно содержат имя и тип ссылочного объекта [45].

## 21. Назначение AllFusion Process Modeler

AllFusion Process Modeler (ранее BPwin) – это CASE-средство для графического моделирования и анализа бизнес-процессов и ИС. Оно предназначено для построения функциональных моделей с использованием методологий SADT/IDEF: поддерживает IDEF0 (функциональные схемы), IDEF3 (описания процессов) и DFD (диаграммы потоков данных) [46]. С помощью этого инструмента специалисты могут рисовать иерархические диаграммы процессов, определять стрелки ICOM, устанавливать связи между уровнями и генерировать отчеты по модели. Таким образом, AllFusion Process Modeler упрощает создание и верификацию проектной документации ЭИС [46].

## 22. Поддерживаемая методология в AllFusion Process Modeler

AllFusion Process Modeler реализует **SADT/IDEF**-методологию. Он поддерживает функциональное моделирование по стандарту IDEF0, динамическое описание процессов IDEF3 и моделирование потоков данных DFD [47]. Это значит, что в этой среде можно строить диаграммы IDEF0 (функции с входами/управлениями/выходами/механизмами), IDEF3 (сценарии процессов) и DFD (потоки данных).

## 23. Назначение элементов управления и настройки системы AllF PM

В AllFusion Process Modeler на панели инструментов (Toolbar) есть стандартные элементы управления: кнопки создания/открытия/сохранения модели, печати, настройки масштаба и проверки орфографии [48] [49]. Имеются переключатели для отображения вспомогательных окон (Model Explorer – навигатор модели, ModelMart – словарь элементов) [50]. Через контекстное меню объектов можно вызвать редакторы свойств: **Font Editor** и **Color Editor** – для задания шрифтов и цветов элементов модели [51]. В меню *Tools* также можно установить *Default Fonts* (шрифты по умолчанию) для разных типов объектов [52]. Эти настройки позволяют адаптировать внешний вид и поведение системы под требования проекта и пользователя.

## 24. Диалоги «Свойство активностей» и «Свойство стрелок»

Диалог **«Свойство активностей»** открывается при редактировании процесса (блока работы) на диаграмме IDEF. В нём задаются характеристики операции: уникальный идентификатор, наименование, подробное описание, ответственный исполнитель, иные атрибуты (например, метрики или стоимость). Аналогично, диалог **«Свойство стрелок»** служит для задания свойств стрелочного соединения: имени потока данных, его типа (вход, выход, управление, ресурс), описания и связанных элементов. Эти диалоги позволяют документировать семантику блоков и потоков модели, связывая графические обозначения с конкретными данными модели. Например, через контекстное меню объектов можно вызвать эти диалоги и указать детальные параметры активности или потока [51].

## 25. Работа со стрелками в IDEF0: туннели

При построении IDEF0-модели стрелки можно «назначать» типом (I, C, O или M) – вход, управление, выход или механизм [53]. Также в интерфейсе AllFusion PM есть функция *Arrow Tunnel*: если стрелка нижнего уровня должна продолжить движение на уровень выше, её можно превратить в «туннельную» (Tunnel) или «разрешить» (Resolve) [54]. **Туннели** обозначаются круглыми скобками: они показывают, что поток данных передаётся между уровнями, но не изображается явно на диаграмме-родителе [54]. Таким образом, туннель позволяет сохранить связь между входом/выходом родительской функции и деталями нижестоящих диаграмм (без визуального повторения стрелки). После развертки (Resolve) стрелка появляется на родительской диаграмме, а при переводе в Tunnel – скрывается на верхнем уровне [54].

## 26. Назначение и использование словаря стрелок

**Словарь стрелок** – это централизованное хранилище определений всех потоков данных и сигналов, используемых в модели IDEF0/IDEF3. В AllFusion PM словарь редактируется через специальный диалог *Arrow Dictionary Editor*, где каждому потоку даётся имя и, при необходимости, комментарий [55]. Такая практика позволяет закрепить единую терминологию: каждой концепции потока соответствует четкое определение, что исключает неоднозначность профессионального жаргона при обсуждении модели [55] [56]. Кроме того, из словаря можно формировать отчёт (Arrow Report) – своеобразный глоссарий используемых потоков, удобный для проверки и представления результатов экспертам [55] [56].

## 27. Назначение и использование ICOM-кодов

**ICOM-коды** – это буквенные обозначения типов стрелок в IDEF0: *I* (Input), *C* (Control), *O* (Output), *M* (Mechanism). Они проставляются автоматически для каждой стрелки в модели, чтобы упростить её анализ. В AllFusion PM включение отображения ICOM-кодов (в *Model Properties*) добавляет в углу каждого блока соответствующую метку типа входа/управления/выхода/ресурса [57]. Это облегчает понимание функций диаграммы и проверку того, что каждая стрелка имеет правильную роль.

## 28. Диаграммы дерева узлов и FEO: назначение и создание

**Диаграммы дерева узлов (Node Tree)** показывают иерархию функций (этапов) модели без указания потоков данных [58]. Они представляют «развертку» декомпозиции – список функций и их подфункций в виде дерева, независимо от связей. В AllFusion PM можно создавать неограниченное число таких деревьев произвольной глубины (не обязательно из корня модели) [58]. **FEO-диаграммы** (Function-Entity-Organization) используются для иллюстрации отдельных фрагментов или альтернативных видов модели [59]. FEO строится для выделения специфических компонентов системы с точки зрения функций, сущностей и организационных ролей; его создают, чтобы представить иной ракурс или дополнительно пояснить проект [59]. В инструменте AllFusion PM эти диаграммы создаются отдельной командой («Add Node Tree» или «Add IDEF0 View») и заполняются соответствующими элементами модели.

## 29. Возможные нотации при построении диаграмм в AllFusion PM

AllFusion Process Modeler поддерживает несколько нотаций моделирования процессов. В первую очередь это **IDEF0, IDEF3 и DFD** (стандарты SADT) [47]. То есть можно строить функциональные схемы (IDEF0), диаграммы потоков данных (DFD) и сценарии процессов (IDEF3). Кроме того, инструмент позволяет делать деревья узлов и FEO-диаграммы. При необходимости DFD можно рисовать в стилистике Гейн–Сарсона или Юордана. Таким образом, AllFusion PM заточен под классические структурные нотации, ориентированные на описание функций и информационных потоков [47] [60].

## 30. Нормативно-методическое обеспечение создания ПО и стандарты его жизненного цикла

Нормативно-методическое обеспечение разработки ПО включает совокупность стандартов, правил и методических материалов, регламентирующих процессы жизненного цикла ПО. К ним относятся международные и национальные стандарты по программной инженерии (см. выше), а также отраслевые ГОСТы и корпоративные методики. Стандарты жизненного цикла (например, ГОСТ Р ИСО/МЭК 12207-99) определяют основные процессы разработки (планирование, управление, техническое обеспечение, контроль качества, конфигурирование и др.), и устанавливают требования к документации, верификации, коммуникациям [61]. Методическая база формируется на основании этих процессов, определяя, какие работы следует выполнять на каждом этапе и какие роли за них отвечают [61]. Таким образом, нормативы задают «правила игры» в создании ПО – от управления рисками и ресурсами до процедур тестирования и приёмки, обеспечивая единообразие разработки и контроля качества.

## 31. Этапы создания ИС и содержание основных процессов по стандартам

По большинству стандартов и нормативов **жизненный цикл ПО** включает следующие стадии: - **Анализ требований** – сбор, формализация и согласование требований заказчика; - **Проектирование** – разработка архитектуры и проектной документации (технико-экономическое и детальное проектирование); - **Разработка (кодирование)** – непосредственная реализация системы (программирование, настройка компонентов); - **Тестирование** – проверка разработанного ПО (модульное, интеграционное, системное тестирование); - **Внедрение** – ввод системы в эксплуатацию (установка, обучение пользователей, передача в промышленную эксплуатацию); - **Сопровождение (эксплуатация)** – поддержка, обновление и модификация ПО после ввода в эксплуатацию.
Так, например, ГОСТ Р ИСО/МЭК 12207-2010 формулирует аналогичные этапы (анализ, проект, программирование, тестирование, поставка/ввод, сопровождение) [62]. В рамках каждого этапа выполняются процессы менеджмента проекта, обеспечения качества, конфигурационного управления и др. (это детально описано в стандартах процесса).

## 32. Определение технологической операции и технологической сети проектирования ЭИС

**Технологическая операция проектирования** – это наименьшая элементарная часть процесса разработки ЭИС, совокупность действий, выполняемых за одну задачу на одном рабочем месте. Ей соответствует «шаблон» (V, P, W, R, S), где *V* – входные исходные данные, *П* – собственно действие (преобразователь), *W* – выходной результат, *R* – затраты (ресурсы), *S* – применяемые средства [63]. Проще говоря, Т.О. описывается: что мы делаем (П), с чем (V), что получаем (W), на что (R), с помощью чего (S) [63]. **Технологическая сеть проектирования** – это граф последовательности таких операций. Она строится объединением операций в сеть (обычно сетевой график), где выходы одних операций становятся входами других. Сеть отражает логическую и временную связь операций: порядок выполнения, параллелизм, зависимости. (Например, по ней можно определить критический путь проекта.) Технологическая сеть получается после сгруппировки всех технологических операций согласно их зависимостям – это база для планирования и управления проектом.

## 33. Различия, достоинства, недостатки и применимость стилей проектирования ПО

Стили проектирования различаются подходом к организации кода и данных. **Процедурно-структурный стиль:** фокус на последовательных алгоритмах и функциях; хорошо подходит для невозрастающих простых задач. *Плюсы:* высокая производительность и простота кода для небольших проектов [64]. *Минусы:* затруднено сопровождение больших систем – библиотеки быстро разрастаются, сильные связи и неявные зависимости усложняют изменения [64].
**Объектно-ориентированный стиль:** упор на объекты/классы и их взаимодействие. *Плюсы:* высокая модульность, повторное использование компонентов, ясное разделение функциональности (методы разделены по классам), что упрощает поддержку и расширение системы [65] [66]. *Минусы:* накладные расходы (иерархии классов, виртуализация) снижают эффективность, сложность разработки выше (переход к ООП требует смены парадигмы) [67] [64].
**Компонентно-ориентированный стиль:** система собирается из заранее готовых программных компонентов (модулей). *Плюсы:* ускоряет разработку за счёт повторного использования готовых блоков, упрощает обновления (смена компонента не затрагивает другие части). *Минусы:* сильно зависит от наличия качественной библиотеки компонентов, сложен вопрос интеграции и совместимости.
**Границы применимости:** процедурный стиль эффективен при ограниченном наборе операций (например, низкоуровневые модули, простые утилиты); ООП – для крупных приложений с ясно выраженными сущностями и долгим сроком поддержки (например, сложные корпоративные системы); компонентный – при постройке систем на основе стандартизированных сервисов (например, микросервисы, плагинные архитектуры).

## 34. Содержание этапов классического жизненного цикла (водопадная модель), её достоинства и недостатки

Классическая (каскадная) модель ЖЦ включает последовательно следующие этапы: **1)** определение требований, **2)** проектирование (архитектурное и детализация), **3)** разработка (программирование), **4)** интеграция и тестирование, **5)** внедрение, **6)** эксплуатация и сопровождение [7]. На первом этапе

формируют полное ТЗ, далее готовят технический и рабочий проект, затем пишут код и проводят тесты, после чего передают систему заказчику и обеспечивают её поддержку. *Достоинства:* простота понимания (чёткие линейные фазы) [7], удобство планирования и контроля (каждый этап закрывается документами), раннее выявление рисков за счёт подробного проектирования и согласования на старте [7]. *Недостатки:* крайне низкая гибкость – изменения требований на ходу практически не предусмотрены [68]. Любая ошибка или неточность выявляется слишком поздно (после этапа тестирования), что может привести к большим доработкам. Длительный цикл и обязательная последовательность (нельзя перейти к следующей фазе, пока не завершена текущая) затрудняют адаптацию к изменяющимся условиям [68].

## 35. Характеристика инкрементной модели ЖЦ: достоинства и недостатки

Инкрементная модель строит систему итеративно: проект разбивается на несколько **инкрементов**, каждый из которых проходит полный цикл разработки (анализ, проектирование, реализацию, тестирование) и выпускается как работающая часть системы [8]. *Достоинства:* после каждого инкремента заказчик получает функциональный продукт (часть системы) [8]. Благодаря коротким итерациям требования стабилизируются (пользователи участвуют в процессе и несущественные изменения переносятся на будущие итерации) [69]. Это улучшает понимание требований в целом и снижает риски: после каждого инкремента проходит ревью, можно скорректировать дальнейшие планы [8] [70]. Кроме того, упрощается тестирование и ввод критичных функций производится раньше. *Недостатки:* модель подходит не для всех проектов. Для планирования инкрементов необходимо изначально зафиксировать общую структуру системы, что требует полного представления о системе в начале разработки [71]. Необходимо чётко определять интерфейсы между модулями, иначе интеграция разных инкрементов может оказаться сложной [71]. Иногда крупные или сложные задачи переносятся на поздние инкременты, что может нарушить сроки. Также сама организация работы более сложна и требует тщательного управления.

## 36. Особенности спиральной модели ЖЦ

Спиральная модель представляет собой итеративный, риско-ориентированный подход. Процесс разработки разбит на циклы (витки спирали). Каждый цикл включает: определение целей, анализ рисков (в т.ч. прототипирование спорных частей), разработку и тестирование, а затем планирование следующего витка. Таким образом, особенность модели – фокус на **управлении рисками**. При каждой итерации уточняются требования и архитектура, что делает модель гибкой к изменениям и пригодной для проектов с неопределёнными требованиями. *Достоинства:* раннее выявление и устранение рисков, постепенно разрабатывается работоспособная система. *Недостатки:* высокая стоимость и сложность реализации (необходим опыт в анализе рисков), большая нагрузка на управление процессом. Модель целесообразна в крупных долгосрочных проектах; в небольших или с фиксированными требованиями она избыточна.

# 37. Отличие компонентно-ориентированной модели от спиральной и классической моделей ЖЦ

Компонентно-ориентированная модель основывается на сборке системы из готовых программных модулей (компонентов) с четко определённым интерфейсом. Это отличается от классической модели (которая предполагает разработку всего кода «с нуля») и от спиральной (которая ориентирована на итерации и управление рисками). *В компонентной модели* акцент на повторном использовании существующих компонентов: требования трансформируются в список необходимых модулей, которые интегрируются и настраиваются. *Плюсы:* значительная экономия времени и ресурсов за счёт готовых блоков, повышение качества (используются отлаженные решения), легкость модификации путем замены компонентов. *Минусы:* зависимость от доступности качественных компонентов и сложности интеграции. Компонентная модель особенно эффективна при создании систем на платформе сервисов/плагинов. В отличие от неё *классическая модель* игнорирует переиспользование (разработка «с нуля»), а *спиральная* не фокусируется на переиспользовании, а на управлении проектом и рисками.

**Источники:** официальные определения и описания взяты из учебных материалов и стандартов (см. цитирования) [2] [3] [5] [6] [18] [34] [36] [55] [61] [64] [65] [7] [8] [71] .

---


[1] Информационная система — Википедия
https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81

[2] ЭИС | это... Что такое ЭИС?
https://abbr_rus.academic.ru/25081/%D0%AD%D0%98%D0%A1

[3] [4] 7. Виды, категории эис. Принципы построения и функционирования эис.
https://studfile.net/preview/8968936/page:6/

[5] Обеспечивающие подсистемы эис
https://studfile.net/preview/7099696/page:10/

[6] Классификация программного обеспечения компьютера
https://unix-it.ru/information/klassifikatsiya-programmnogo-obespecheniya-kompyutera/

[7] [68] Waterfall, Agile, Scrumban — плюсы и минусы, или Что не так с эталонными подходами к разработке / Хабр
https://habr.com/ru/companies/magnus-tech/articles/785638/

[8] [69] [70] [71] Преимущества инкрементной модели жизненного цикла
https://studfile.net/preview/1444533/page:9/

[9] [10] Стандарты программной инженерии - презентация онлайн
https://ppt-online.org/691808

[11] ISO/IEC 12207:2008 — Википедия
https://ru.wikipedia.org/wiki/ISO/IEC_12207:2008

12 Стандарты системной инженерии
https://systems-analysis.ru/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D1%8B_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE

13 1.2. Нормативные документы по стандартизации и виды стандартов — Стандартизация и сертификация
https://xumuk.ru/ssm/2.html

14 15 Сертификация — Википедия
https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F

16 17 Знакомьтесь: SWEBOK | Открытые системы. СУБД | Издательство «Открытые системы»
https://www.osp.ru/os/2006/07/3290839

18 19 20 CASE — Википедия
https://ru.wikipedia.org/wiki/CASE

21 Блок-схема — Википедия
https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D0%BA-%D1%81%D1%85%D0%B5%D0%BC%D0%B0

22 23 24 3.1. Технология проектирования эис
https://studfile.net/preview/10678630/page:13/

25 26 27 28 10.2. Методы проектирования информационных систем
https://studfile.net/preview/9057964/page:87/

29 30 26. Моделирование предметной области и основные понятия модели предметной области
https://studfile.net/preview/21785343/page:9/

31 Учреждение образования
https://www.bsuir.by/m/12_100229_1_149101.pdf

32 33 Функциональное моделирование на примере sadt
https://studfile.net/preview/10161837/page:5/

34 35 Базовые знания по IDEF0 — Блог Comindware
https://www.comindware.ru/blog/%D0%B0%D0%B7%D1%8B-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D0%B2-idef0/

36 37 38 Диаграмма потоков данных (DFD) для чайников: что это такое, как сделать и какой бывает / Хабр
https://habr.com/ru/companies/kaiten/articles/925044/

39 40 1.5. Критерий окончания декомпозиции диаграмм и меры по управлению проектом; цикл папки
https://studfile.net/preview/8847890/page:6/

41 42 16. Основные элементы idef3-диаграмм
https://studfile.net/preview/1746604/page:12/

43 44 45 Технология разработки ПО: Методология IDEF3
https://technologiarpo.blogspot.com/p/idef3.html

46 47 AllFusion Process Modeler BPwin: Описание, Функции и Интерфейс – 2025
https://soware.ru/products/allfusion-process-modeler-bpwin

48 49 50 51 52 58 59 60 BPwin уроки страница 1
https://specialf.narod.ru/bpwin/urok.html

53 54 34. Пакет bpWin. Туннелирование стрелок.
https://studfile.net/preview/12644898/page:11/

55 56 57 НОУ ИНТУИТ | Проектирование информационных систем. Лекция 7: Моделирование бизнес-процессов средствами BPwin
https://intuit.ru/studies/courses/2195/55/lecture/1630?page=3

61 Стандарты и методологии в жизненном цикле программного обеспечения информационных систем | Директор информационной службы | Издательство «Открытые системы»
https://www.osp.ru/cio/2001/10/171950

62 Описание процессов жизненного цикла
https://get-lab.ru/upload/docs/AMS-Software_lifecycle.pdf

63 Технологические операции проектирования
https://studfile.net/preview/2566872/page:8/

64 66 Ответы Mail: Преимущества и недостатки процедурного программирования? Также можно привести плюсы\минусы относительно ООП
https://otvet.mail.ru/question/178951910

65 67 Основные положения объектной модели и ее преимущества и недостатки
https://studfile.net/preview/16447867/page:3/