Вот перефразированные определения первых 5 пунктов с примерами:

---

1. Две схемы организации коллективов специалистов
а) Единая проектная команда: Формируется цельная команда под конкретный проект, включающая всех необходимых специалистов под руководством лидера проекта.
Пример: Компания создаёт отдельную команду из разработчиков, тестировщиков и дизайнеров для создания нового мобильного приложения.

б) Децентрализованная модель с интеграторами: Руководитель и небольшая группа интеграторов координируют работу узких специалистов, которые формально не входят в единую команду проекта.
Пример: Над проектом работают внешние фрилансеры (программист, копирайтер), которых координирует внутренний менеджер-интегратор.

---

2. Четыре стратегии руководства
а) Директивное управление: Руководитель чётко ставит задачи, контролирует сроки и результаты.
Пример: Менеджер назначает разработчику задачу с жёстким дедлайном и регулярно проверяет прогресс.

б) Объяснения: Лидер не только даёт указания, но и разъясняет причины решений, сочетая директивный и коллективный подход.
Пример: Руководитель проводит встречу, где объясняет, почему выбран именно этот стек технологий для проекта.

в) Участие: Лидер вовлекает команду в принятие решений, поощряет инициативу.
Пример: Перед запуском нового функционала проводится мозговой штурм с участием всех разработчиков.

г) Делегирование: Лидер передаёт задачи подчинённым, обеспечивает ресурсы и наблюдает за процессом.
Пример: Тимлид поручает senior-разработчику самостоятельно спроектировать архитектуру нового модуля.

---

3. Четыре стадии развития группы в команду
а) Формирование: Начальный этап, когда участники знакомятся, проявляют энтузиазм, но ещё нет чёткого взаимодействия.
Пример: Первое собрание новой команды, где все представляются и обсуждают общие цели.

б) Разногласия и конфликты: Этап споров и поиска эффективных способов работы, часто вызванный трудностями.
Пример: Разработчики спорят о выборе инструментов для проекта, что приводит к конфликту.

в) Становление: Устанавливаются доверие, нормы сотрудничества и чёткое разделение обязанностей.
Пример: Команда договорилась о единых стандартах кодирования и регулярных код-ревью.

г) Отдача: Команда работает слаженно, эффективно использует сильные стороны каждого.
Пример: Команда быстро выпускает новые версии продукта, так как каждый знает свою роль и доверяет коллегам.

---

4. Аспекты управления командой
а) Ролевая модель: Определение и распределение ролей в команде.
Пример: В проекте есть аналитик, разработчик, тестировщик и дизайнер, каждый отвечает за свою зону.

б) Модель организации: Структура команды (иерархия, самоорганизация и т.д.).
Пример: Команда организована по принципу Scrum с ролями Scrum-мастера, владельца продукта и разработчиков.

в) Общение в команде: Эффективная коммуникация между участниками.
Пример: Ежедневные стендапы для обмена статусом задач и проблемами.

---

5. Административная модель организации команды
Характерные черты: Жёсткая иерархия, чёткие роли и обязанности, следование инструкциям, менеджер контролирует и принимает ключевые решения.
Пример: Государственная организация, где каждый сотрудник выполняет строго определённые функции согласно должностной инструкции.

Преимущества: Ясность, предсказуемость, простота управления.
Недостатки: Медленная реакция на изменения, подавление инициативы.
Пример: Внедрение новой технологии в такой структуре требует долгого согласования и часто встречает сопротивление.

---

6. Модель хаоса
Характерные черты: Отсутствие чёткой иерархии, формальных инструкций и процедур. Менеджер лишь ставит задачу и обеспечивает ресурсами, не вмешиваясь в процесс. Решения принимаются на месте обнаружения проблемы, основа — дружеское соревнование и инициатива.
Пример: Стартап на ранней стадии, где разработчики сами выбирают инструменты и методы решения задач, а менеджер только следит за общим прогрессом.

Преимущества: Полная свобода творчества, раскрытие потенциала участников.
Недостатки: Высокий риск неудачи при отсутствии координации.

---

7. Модель «открытая архитектура»
Характерные черты: Гибкая адаптация к условиям, коллективное обсуждение и принятие решений, распределённая ответственность, динамичный состав групп, отсутствие жёсткой специализации (участники могут менять роли). Менеджер участвует в обсуждениях, но не диктует решения.
Пример: Команда в IT-компании, работающая по Agile, где на разных этапах разработчики, аналитики и тестировщики совместно планируют спринты и могут подменять друг друга.

Преимущества: Высокая адаптивность, подходит как для индивидуалистов, так и для коллективистов.

---

8. Отличие компромисса от консенсуса
Компромисс — соглашение, достигнутое путём взаимных уступок; часто является усреднённым, неидеальным решением.
Пример: Команда выбирает язык программирования: часть хотела Python, часть — Java. В итоге соглашаются на Kotlin как на условный «средний» вариант.

Консенсус — общее мнение, выработанное через обсуждение и поиск оптимального решения, сочетающего лучшие идеи.

Пример: После анализа команда решает использовать микросервисную архитектуру, так как это удовлетворяет всех и объединяет преимущества разных предложений.

---

9. Инициация проекта
Инициация — это процессы формального запуска проекта, часто выполняемые до его начала. Включает уточнение целей, ресурсов, назначение менеджера, фиксацию допущений и ограничений в Уставе проекта.
Пример: Перед разработкой новой CRM-системы проводится встреча с заказчиком, определяются основные требования, бюджет и сроки, после чего выпускается Устав проекта.

---

10. Критерии определения приоритета проекта
Приоритет проекта оценивается по трём критериям:
1) Финансовая ценность — ожидаемая окупаемость и доходность.
Пример: Проект с окупаемостью до года и доходом в 1.5 раза выше расходов имеет высокий приоритет.

2) Стратегическая ценность — влияние на позиции компании на рынке.
Пример: Запуск уникального продукта, который выведет компанию на новый рынок и даст устойчивое преимущество.

3) Уровень рисков — степень неопределённости в целях, ресурсах и технологиях.
Пример: Проект с чёткими требованиями, известными технологиями и доступными ресурсами имеет низкий уровень риска.

---

11. Концепция проекта
Концепция проекта — это основной документ, который определяет ключевые параметры проекта и используется для принятия решений на всех этапах, включая приёмку результатов.
Пример: Перед созданием нового веб-сайта документ "Концепция" описывает цели (увеличить продажи), сроки (3 месяца), бюджет, риски и критерии приёмки (скорость загрузки страниц не более 2 секунд).

Разделы концепции проекта:

- Название проекта
- Цели и ожидаемые результаты

- Допущения и ограничения
- Участники и заинтересованные стороны
- Ресурсы
- Сроки
- Риски
- Критерии приёмки
- Обоснование полезности проекта

---

12. Цель проекта и примеры
Цель проекта отвечает на вопрос, зачем проект нужен, описывая решаемые бизнес-задачи.
Примеры:

1. **Автоматизация процессов:** Внедрение CRM-системы для ускорения обработки заявок клиентов.
2. **Реализация стратегии:** Разработка мобильного приложения для выхода на рынок онлайн-образования.

---

13. Критерии правильной цели проекта
Цели должны быть:

- **Значимыми** (соответствовать стратегии компании)
- **Конкретными** (чётко определёнными для данного проекта)
- **Измеримыми** (иметь количественные показатели)
- **Достижимыми** (реалистичными)

Пример: Увеличить конверсию на сайте с 3% до 5% за 6 месяцев через оптимизацию UX — это конкретная, измеримая и достижимая цель.

---

14. Ключевые участники программного проекта

- **Спонсор** — финансирует проект.
  Пример: Директор компании, выделяющий бюджет на разработку.
- **Заказчик** — будущий пользователь продукта.
  Пример: Отдел маркетинга, который будет использовать аналитическую систему.

- **Пользователи** — те, кто непосредственно работает с результатом проекта.
- **Куратор** — представитель исполнителя, отвечающий за ресурсы и изменения.
- **Руководитель проекта** — управляет реализацией (сроки, бюджет, качество).
- **Соисполнители, поставщики** — внешние подрядчики.

---

15. Риски проекта и критерии приёмки
Риск — неопределённое событие, которое может негативно или позитивно повлиять на проект.
Пример: Риск задержки поставки серверов от внешнего поставщика.

Критерии приёмки — чёткие, измеримые требования, по которым определяется успешность проекта.
Пример: Система должна обрабатывать 1000 транзакций в секунду без сбоев — это критерий приёмки для тестирования производительности.

---

16. Планирование проекта
Планирование проекта — это непрерывный процесс определения оптимальной последовательности действий для достижения целей проекта с учётом текущих условий.
Пример: Перед запуском разработки мобильного приложения менеджер составляет план, включающий этапы, сроки, бюджет и распределение ресурсов.

Цель планирования: Создать модель реализации проекта.
Результат: Сводный план проекта, который служит основной моделью действий и прогнозом состояния проекта.

---

17. Предметная область проекта
Предметная область проекта — это совокупность продуктов, услуг и результатов, которые должны быть получены по завершении проекта.
Пример: Для проекта «Разработка сайта интернет-магазина» предметная область включает: каталог товаров, корзину, систему оплаты, личный кабинет и документацию.

Задачи планирования предметной области:

- Уточнение целей и результатов проекта
- Определение характеристик проекта

- Корректировка ограничений и допущений
- Выбор критериев оценки результатов
- Построение структурной декомпозиции работ

---

18. Сетевая диаграмма проекта
Сетевая диаграмма — это графическое представление работ проекта и зависимостей между ними в виде графа, где вершины — это работы, а рёбра — связи.
Пример: При разработке ПО сначала идёт «Проектирование», затем «Кодирование», потом «Тестирование» — эти этапы отображаются в виде связанных блоков на диаграмме.

Как составляется:

1. Определяются все работы проекта
2. Устанавливаются последовательности и зависимости между работами
3. Работы обозначаются прямоугольниками с информацией (название, длительность)
4. Связи показываются стрелками

---

19. Диаграмма Ганта
Диаграмма Ганта — это горизонтальная линейная диаграмма, где задачи проекта представлены в виде отрезков, длина которых соответствует продолжительности задач.
Пример: В проекте «Запуск рекламной кампании» задача «Написание текстов» длится 5 дней и отображается горизонтальной полосой в календаре.

Отличие от сетевой диаграммы:

- В диаграмме Ганта работы визуализируются на временной шкале, а в сетевой — как граф зависимостей
- Диаграмму Ганта можно дополнять информацией о стоимости, исполнителях и прогрессе

Пример схемы диаграммы Ганта:

```
Задача           | Янв | Фев | Мар | Апр
-----------------------------------------
Анализ           |=====|     |     |
Проектирование   |     |=====|     |
```

```
Разработка       |     |     |=====|
Тестирование     |     |     |     |=====
```

---

20. Этапы планирования трудовых ресурсов

1. **Определение доступных ресурсов:** Составление списка исполнителей с указанием их занятости
   Пример: В проекте доступны 3 разработчика, каждый может выделить 20 часов в неделю
2. **Назначение исполнителей:** Распределение конкретных людей по задачам проекта
   Пример: Senior-разработчик назначается на создание архитектуры, junior — на написание тестов
3. **Разрешение конфликтов:** Анализ и устранение перегрузок или накладок в расписании
   Пример: Если два важных этапа требуют одного специалиста одновременно, сроки одного этапа сдвигаются

---

21. Организационная структура проекта
Организационная структура — это утверждённое распределение ролей, обязанностей и целей между ключевыми участниками проекта.
Пример: В проекте по разработке ПО есть роли: руководитель проекта, архитектор, разработчики, тестировщики — каждый имеет свои задачи и отчёты.

Что включает:

- Систему рабочих взаимоотношений между группами
- Систему отчётности
- Систему оценки прогресса
- Систему принятия решений

---

22. Управление рисками и факторы риска
Управление рисками — это часть управления проектами, которая включает процессы по выявлению, анализу и реагированию на риски.
Пример: При запуске нового сервиса команда заранее оценивает риск сбоя в пиковую нагрузку и готовит план масштабирования инфраструктуры.

Факторы риска (три параметра):

1. **Рисковое событие** — что может произойти
   Пример: Срыв сроков поставки оборудования
2. **Вероятность наступления** — какова вероятность события
   Пример: Вероятность срыва поставки — 30%
3. **Размер потерь** — последствия события
   Пример: Задержка проекта на 2 недели, убыток 500 000 рублей

---

23. Модель технологической зрелости
Модель технологической зрелости — это шкала из пяти уровней, оценивающая, насколько последовательно компания применяет повторяемые процессы в работе.
Пример: Уровень 1 — хаотичная разработка, уровень 5 — оптимизированные процессы с непрерывным улучшением (как в компаниях, внедривших CMMI).

Уровни:

1. **Начальный** — процессы хаотичны
2. **Повторяемый** — есть базовые процессы управления
3. **Определённый** — процессы стандартизированы
4. **Управляемый** — процессы измеряются и контролируются
5. **Оптимизируемый** — процессы постоянно улучшаются

---

24. Уровни значимости зрелости

1. **Беспорядок/кризис:** Успех зависит от отдельных людей, процессы не систематизированы
   Пример: Стартап, где каждый разработчик работает как хочет, документация ведётся от случая к случаю
2. **Стандартное управление проектами:** Внедрены базовые процессы управления
   Пример: Компания начинает использовать методологии Scrum и вести регулярное планирование
3. **Стандартные процессы организации:** Стандартизация распространяется на производственную деятельность
   Пример: Все проекты используют единые инструменты разработки и регламенты тестирования
4. **Управляемая обратная связь:** Собираются метрики, создаётся база знаний
   Пример: После каждого проекта анализируются ошибки, данные используются для

улучшения процессов

5. **Оптимизация/непрерывное совершенствование:** Замкнутый цикл улучшений на основе измерений
   Пример: Автоматизированная система анализа качества кода и постоянная адаптация процессов под меняющиеся требования

---

25. SWOT-анализ

SWOT-анализ — метод стратегического планирования, при котором выявляются внутренние (сильные и слабые стороны) и внешние (возможности и угрозы) факторы.
Пример: Компания-разработчик проводит SWOT-анализ перед выходом на новый рынок: сильная сторона — опытная команда, слабая — малый бюджет, возможность — растущий спрос, угроза — сильные конкуренты.

Компоненты:

- **S (Strengths)** — внутренние преимущества
- **W (Weaknesses)** — внутренние недостатки
- **O (Opportunities)** — внешние благоприятные условия
- **T (Threats)** — внешние риски и угрозы

---

26. Термины: событие риска, величина риска, управленческий резерв

Событие риска — потенциальное событие, которое может негативно или позитивно повлиять на ход проекта.
Пример: Возможный сбой в работе облачного сервера, ведущий к недоступности сервиса.

Величина риска — числовой показатель, объединяющий вероятность наступления риска и масштаб его последствий (рассчитывается как: Вероятность × Последствия).
Пример: Риск задержки поставки оборудования с вероятностью 20% и убытком в 1 млн рублей имеет величину риска = 0,2 × 1 000 000 = 200 000 рублей.

Управленческий резерв — резерв средств или времени, не включённый в основной план проекта, который используется руководством для реагирования на непредсказуемые события.
Пример: В бюджет проекта заложен управленческий резерв 10% на случай непредвиденных расходов, например, внезапного роста цен на лицензии ПО.

---

27. Четыре стратегии реагирования на негативные риски

1. **Уклонение от риска:** Изменение плана проекта для полного исключения риска.
   Пример: Отказ от использования новой нестабильной технологии в пользу проверенной, чтобы избежать риска сбоев.
2. **Передача риска:** Передача ответственности за риск третьей стороне (например, через страхование или аутсорсинг).
   Пример: Заключение договора с хостинг-провайдером с гарантией компенсации при простое сервиса.
3. **Принятие риска:** Осознанное принятие риска без активных действий до его наступления.
   Пример: Команда понимает риск задержки из-за сложной интеграции, но не меняет план, готовая работать сверхурочно в случае необходимости.
4. **Снижение риска:** Уменьшение вероятности или последствий риска.
   Пример: Проведение дополнительного тестирования на ранних этапах, чтобы снизить риск критических ошибок перед релизом.

---

28. Мониторинг управления рисками

Мониторинг управления рисками — процесс отслеживания идентифицированных рисков, контроля остаточных рисков, выявления новых рисков и оценки эффективности планов реагирования в течение всего проекта.
Пример: Регулярные еженедельные встречи команды, на которых пересматривается реестр рисков и корректируются планы действий.

Цель мониторинга: Контроль выполнения планов по рискам и своевременное оповещение команды о наступлении рисковых событий.

Исходные данные для мониторинга:

- План управления рисками
- Реестр рисков
- Одобренные запросы на изменения
- Отчёты о выполнении работ

---

29. Принципы фон Неймана

1. **Принцип программного управления:** Программа состоит из набора команд, выполняемых процессором последовательно.

Пример: Процессор выполняет инструкции программы «сложение двух чисел» строго по порядку.

2. **Принцип однородности памяти:** Программы и данные хранятся в одной памяти, над командами можно выполнять операции как над данными.
   Пример: Компилятор может модифицировать код программы во время её выполнения, так как код хранится в общей памяти.
3. **Принцип адресности:** Память состоит из пронумерованных ячеек, доступных процессору в любой момент.
   Пример: Процессор обращается к ячейке памяти с адресом 0x1000, чтобы прочитать данные.

---

30. Программа

Программа — программный код, который может быть выполнен компьютером для решения определённой задачи.
Пример: Скрипт на Python, который автоматически создаёт резервные копии файлов каждый день.

---

31. Программирование
Программирование — это процесс создания компьютерных программ путём написания кода на специальных языках, позволяющих компьютеру выполнять определённые задачи.
Пример: Разработчик пишет код на Python для автоматического анализа данных из таблицы Excel.

---

32. Кризис программного обеспечения
Кризис программного обеспечения — это ситуация, когда сложность и масштаб программных проектов опережают возможности разработчиков эффективно управлять ими, что приводит к срывам сроков, перерасходу бюджета и низкому качеству продукта.
Пример: Проект по созданию государственной информационной системы, который несколько раз пересматривался, вышел далеко за рамки бюджета и в итоге устарел ещё до запуска.

---

33. Программная инженерия
Программная инженерия — это систематический, дисциплинированный подход к

разработке, эксплуатации и сопровождению программного обеспечения, основанный на инженерных методах и принципах.
Пример: Использование методологии Scrum, автоматизированного тестирования и CI/CD для создания и поддержки интернет-банка.

---

34. Жизненный цикл
Жизненный цикл — это последовательность этапов существования любого объекта от возникновения до прекращения использования. Применительно к ПО — это период от зарождения идеи до окончательного вывода продукта из эксплуатации.
Пример: Жизненный цикл компьютерной игры включает концепцию, разработку, тестирование, выпуск, выпуск обновлений и, наконец, закрытие серверов.

---

35. Расшифровка аббревиатур ISO и IEC
ISO (International Organization for Standardization) — Международная организация по стандартизации, разрабатывающая стандарты для различных отраслей.
Пример: Стандарт ISO 27001 определяет требования к системе управления информационной безопасностью.

IEC (International Electrotechnical Commission) — Международная электротехническая комиссия, отвечающая за стандарты в области электротехники и электроники.
Пример: Стандарт IEC 62368 устанавливает требования безопасности для аудио-, видео- и IT-оборудования.

---

36. Стандарт ISO/IEC 12207 “Information Technology – Software Life Cycle Process”
Данный стандарт определяет структуру жизненного цикла программного обеспечения, устанавливая процессы, действия и задачи, которые должны выполняться при создании ПО.
Пример: Компания-разработчик использует этот стандарт как основу для построения своих внутренних регламентов, например, для чёткого разделения этапов: анализ требований → проектирование → разработка → тестирование → внедрение.

---

37. Процесс
Процесс — это совокупность взаимосвязанных действий, которые преобразуют входные

данные в выходные результаты. Каждый процесс включает задачи, методы их решения и делится на действия.
Пример: Процесс тестирования ПО принимает на вход готовый код, выполняет тесты и выдаёт отчёт об ошибках — это выходные данные.

---

38. Три группы процессов жизненного цикла по ISO/IEC 12207

1) Основные процессы:

- Приобретение
- Поставка
- Разработка
- Эксплуатация
- Сопровождение
  Пример: Процесс разработки включает этапы от проектирования до выпуска готового продукта.

2) Вспомогательные процессы:

- Документирование
- Управление конфигурацией
- Обеспечение качества
- Верификация
- Совместная оценка
- Аттестация
- Аудит
- Разрешение проблем
  Пример: Процесс управления конфигурацией помогает отслеживать версии компонентов ПО.

3) Организационные процессы:

- Управление
- Создание инфраструктуры
- Обучение
- Совершенствование
  Пример: Процесс обучения — проведение тренингов для новых разработчиков по стандартам компании.

---

39. Действия заказчика в процессе приобретения

1. Инициирование приобретения
2. Подготовка заявочных предложений
3. Подготовка и корректировка договора
4. Надзор за деятельностью поставщика
5. Приёмка и завершение работ
   Пример: Компания-заказчик формирует техническое задание на разработку сайта, выбирает подрядчика, заключает договор, контролирует ход работ и принимает готовый продукт.

---

40. Действия поставщика в процессе поставки

1. Инициирование поставки
2. Подготовка ответа на заявочные предложения
3. Подготовка договора
4. Планирование работ по договору
5. Выполнение, контроль и оценка работ
6. Поставка и завершение работ
   Пример: IT-студия получает запрос от клиента, составляет коммерческое предложение, подписывает договор, разрабатывает сайт, тестирует его и передаёт заказчику.

---

41. Действия разработчика в процессе разработки
Разработчик выполняет последовательность действий для создания программного продукта, включая подготовку, проектирование, кодирование, тестирование и интеграцию.
Пример: При разработке мобильного приложения команда сначала анализирует требования (что должно делать приложение), затем проектирует интерфейс, пишет код, тестирует модули, объединяет их в единую систему и проводит итоговое тестирование перед выпуском.

Ключевые действия:

- Подготовительная работа
- Анализ требований к системе и ПО
- Проектирование архитектуры системы и ПО
- Детальное проектирование ПО

- Кодирование и тестирование компонентов
- Интеграция ПО и системы
- Квалификационное тестирование
- Установка и приёмка ПО

---

42. Действия оператора в процессе эксплуатации
Оператор обеспечивает штатную работу системы в соответствии с документацией, оказывая поддержку пользователям.
Пример: Оператор банковской системы следит за работоспособностью серверов, проводит плановые обновления и помогает сотрудникам банка решать проблемы с ПО.

Основные действия:

- Планирование операционных задач и установка стандартов
- Определение процедур решения проблем
- Эксплуатационное тестирование новых версий
- Работа системы в целевой среде
- Консультация и помощь пользователям

---

43. Действия организации в процессе сопровождения
Сопровождение включает поддержку, модификацию и обновление ПО после его выпуска.
Пример: После выпуска CRM-системы команда сопровождения исправляет найденные ошибки, добавляет новые функции по запросам клиентов и обновляет документацию.

Этапы сопровождения:

- Планирование работ и процедур решения проблем
- Анализ проблем и запросов на изменения
- Внесение изменений в ПО и документацию
- Проверка целостности системы после изменений

---

44. Конфигурация ПО и управление конфигурацией
Конфигурация ПО — это совокупность характеристик ПО, описанных в документации и реализованных в продукте.

Пример: Версия 2.0 текстового редактора включает функции: автоперевод, проверку орфографии и облачное сохранение — это и есть его конфигурация.

Управление конфигурацией — это процесс контроля изменений в ПО на всех этапах жизненного цикла.
Пример: Использование Git для управления версиями кода, где каждая правка фиксируется и может быть отслежена.

---

45. Качество ПО и вспомогательные процессы
Качество ПО — это совокупность свойств, определяющих способность ПО соответствовать установленным требованиям.
Пример: Качественное мобильное приложение должно быть быстрым, стабильным, безопасным и удобным для пользователя.

Вспомогательные процессы, используемые для обеспечения качества:

- Верификация (проверка соответствия требованиям)
- Аттестация (оценка соответствия назначению)
- Совместная оценка
- Аудит
- Разрешение проблем
  Пример: Перед выпуском приложения проводится аудит кода и тестирование (верификация), чтобы убедиться в отсутствии критических ошибок.

---

46. Верификация и аттестация

Верификация — это процесс проверки соответствия программного обеспечения установленным требованиям, чтобы убедиться, что продукт разработан правильно.
Пример: Тестирование модуля расчёта налогов в бухгалтерской программе на соответствие техническому заданию.

Аттестация — это оценка того, насколько ПО соответствует своему целевому назначению и потребностям пользователя.
Пример: Проверка мобильного приложения на удобство использования и функциональность в реальных условиях клиентом перед приёмкой.

---

47. Аудит и совместная оценка

Аудит — независимая проверка, проводимая для оценки соответствия ПО или процессов установленным стандартам, планам или договорам.
Пример: Внешний аудитор проверяет, соблюдает ли компания стандарты безопасности при разработке банковского ПО.

Совместная оценка — процесс, в ходе которого команда проекта и заказчик совместно оценивают состояние работ и создаваемого ПО.
Пример: На еженедельных встречах разработчики демонстрируют клиенту текущий прогресс по проекту, обсуждают возможные изменения и корректируют планы.

Процесс совместной оценки включает:

- Подготовительную работу
- Оценку управления проектом
- Техническую оценку

---

48. Процесс усовершенствования и процесс создания инфраструктуры

Процесс усовершенствования — это непрерывный цикл оценки, измерения, контроля и улучшения процессов жизненного цикла ПО на основе анализа опыта предыдущих проектов.
Пример: Команда после каждого релиза проводит ретроспективу, чтобы выявить слабые места в процессе разработки и внести улучшения на следующий цикл.

Основа процесса: анализ преимуществ и недостатков каждого процесса, сбор исторических и технических данных по завершённым проектам.

Процесс создания инфраструктуры — это выбор, внедрение и поддержка технологий, стандартов и инструментов, необходимых для разработки, эксплуатации и сопровождения ПО.
Пример: Компания выбирает и настраивает единый набор инструментов: Git для контроля версий, Jira для управления задачами, Jenkins для автоматической сборки.

---

49. Малые программы

Малые программы — это компактные программы, которые создаются одним специалистом или небольшой группой для решения конкретных, часто узких задач, не предназначенные для массового рынка.
Пример: Скрипт на Python, написанный аналитиком для автоматической выгрузки и предварительной обработки данных из Excel-отчётов каждый день.

Характеристики малых программ:

- Не предназначены для массового распространения
- Не имеют внешнего заказчика с чёткими требованиями и бюджетом
- Не имеют строгих ограничений по срокам, качеству и документации
- Их жизненный цикл непредсказуем

---

50. Модель жизненного цикла программного обеспечения

Модель ЖЦ ПО — это структура, определяющая последовательность и взаимосвязь процессов, действий и задач на протяжении всего жизненного цикла ПО. Модель зависит от специфики проекта.
Пример: Для простого, чётко определённого проекта (например, создание лендинга) может использоваться каскадная модель, а для сложного продукта с меняющимися требованиями (например, мобильное приложение) — гибкая модель (Agile).

---

51. Стадии жизненного цикла программной системы
Жизненный цикл программной системы включает последовательные этапы от зарождения идеи до полного вывода продукта из эксплуатации.
Пример: Разработка мессенджера проходит через стадии: сбор требований, проектирование, кодирование, тестирование, внедрение, эксплуатация, поддержка и в конце — закрытие серверов.

Основные стадии:

1. Формирование требований
2. Исследование и описание концепций
3. Проектирование и разработка
4. Испытания системы
5. Внедрение
6. Распространение и продажа
7. Эксплуатация
8. Сопровождение и мониторинг
9. Снятие с эксплуатации (утилизация)

---

52. Особенности, преимущества и недостатки каскадной (водопадной) модели
Особенность: Чёткая последовательность этапов, переход к следующему только после полного завершения предыдущего, возвраты не предусмотрены.
Пример: Строительство дома по готовому проекту, где нельзя начать отделку, пока не возведены стены.

Преимущества:

- На каждом этапе создаётся полный комплект документации
- Легко планировать сроки и бюджет благодаря линейной структуре

Недостатки:

- Ошибки обнаруживаются поздно, на этапе тестирования
- Требования должны быть чётко определены в начале
- Заказчик видит результат только в конце проекта

---

53. Итерационная модель жизненного цикла
Итерационная модель предполагает разработку ПО циклами (итерациями), позволяя возвращаться к предыдущим этапам для исправления ошибок и уточнения требований.
Пример: Разработка сайта интернет-магазина: сначала создаётся базовая версия с каталогом, затем в следующей итерации добавляется корзина, потом — система оплаты.

Особенность: Возможность вносить изменения после каждой итерации, что снижает риски и повышает качество продукта.

---

54. Стратегии конструирования ПО

1. **Однократный проход (каскадная стратегия):** Линейная последовательность этапов без возвратов.
   Пример: Разработка простого калькулятора по заранее утверждённому ТЗ.
2. **Инкрементная стратегия:** Система создаётся версиями, каждая добавляет новые функции, но все требования известны с начала.
   Пример: Создание текстового редактора: версия 1 — базовое редактирование, версия 2 — форматирование, версия 3 — проверка орфографии.
3. **Эволюционная стратегия:** Система также строится версиями, но требования уточняются в процессе разработки.

Пример: Разработка стартап-проекта, где финальный функционал определяется по мере получения обратной связи от первых пользователей.

---

55. Инкрементная модель жизненного цикла (пример)
Инкрементная модель объединяет линейные этапы каскадной модели с итерационным подходом: каждый цикл выпускает рабочую версию продукта с новыми функциями.
Пример: Разработка облачного хранилища:

- **1-й инкремент:** Загрузка и скачивание файлов
- **2-й инкремент:** Совместный доступ к файлам
- **3-й инкремент:** Шифрование и резервное копирование

Каждый инкремент проходит этапы: анализ → проектирование → кодирование → тестирование → поставка.

---

56. Спиральная модель жизненного цикла (пример)
Спиральная модель основывается на циклах (витках спирали), каждый из которых включает четыре действия: планирование, анализ рисков, конструирование и оценку заказчиком.
Пример: Разработка системы автоматического трейдинга:

- **Виток 1:** Сбор требований, анализ рисков (например, волатильность рынка), создание прототипа, оценка заказчиком
- **Виток 2:** Уточнение требований, анализ новых рисков, разработка ядра системы, повторная оценка
- **Виток 3:** Добавление сложных алгоритмов, тестирование в реальных условиях, финальная приёмка

---

57. Преимущества и недостатки спиральной модели
Преимущества:

- Реалистично отражает разработку ПО через эволюцию и итерации
- Позволяет явно учитывать риски на каждом этапе
- Сочетает гибкость и системный подход

Недостатки:

- Требует высокой вовлечённости и компетентности заказчика
- Сложно контролировать сроки и бюджет из-за цикличности

Пример использования: Создание сложного медицинского ПО для диагностики, где требования и риски (например, точность алгоритмов) уточняются в процессе тестирования с врачами.