Вопросы к экзамену по «Управлению проектами»

I. История развития программной инженериии основные понятия.

1. Определения проекта и проектного управления. Жизненный цикл проекта, процессы жизненного цикла проекта и их категории.

`Проект` — уникальная деятельность, имеющая начало и конец во времени, направленная на создание заранее определенного нового продукта (услуги, результата) при заданных ограничениях (бюджет, время, ресурсы, требования качества, допустимый уровень риска).

`Управление проектом` — деятельность, использующая способности, специальные инструменты и процессы для достижения поставленной цели, балансируя с ограничениями в рамках конкретного проекта (объемами работ, сроками, ресурсами, качеством, рисками).

`Жизненный цикл проекта`
Инициация (Initiation). Идея, концепция.
Планирование (Planning). Самое сложное — написать реалистичный план.
Выполнение (Execution). Мониторинг и контроль отклонений от плана, его корректирование.
Закрытие (Closing). Аналитический отчёт и презентация.

Каждая фаза заканчивается аналитической запиской (ошибки, сложности и т.д.)

`Процессы жизненного цикла`

- *Основные*:
  - Заказ (acquisition).
  - Поставка (supply).
  - Разработка (development).
  - Эксплуатация (operation).
  - Сопровождение (maintenance).
- *Вспомогательные*:
  - Документация (documentation).
  - Управление конфигурацией (configuration management).
  - Качество (QA).

    - Верификация (verification).
    - Аттестация.
    - Анализ (joint analysis).
    - Аудит.
    - Анализ проблем и их решение.
- *Организационные*:
    - Управление (management).
    - Инфраструктура проекта (infrastructure).
    - Усовершенствование проекта (improvement).
    - Обучение (training) (своих специалистов/заказчиков).

2. Стадии развития продукта, жизненный цикл продукта. Виды защиты интеллектуальной собственности.

Стадии развития продукта

1. *Концепция продукта* — идея с доказанной выполнимостью. Обоснование идеи. Схема, алгоритм, средства выполнения.
2. *Рабочая модель* — реализованная концепция, рабочий макет основного функционала. Демоверсия проекта. Возможность оценить функционал (основной) приложения, например, консольное приложение.
3. *Прототип* — все физические свойства итогового продукта. Например, демонстрационная модель. Что-то может работать с ошибками, но основной функционал работает.
4. *Инженерный прототип* — прототип + технология его производства. Технология может быть на бумаге. Например, альфа-версия ПО.
5. *Предпродажный прототип* — бета-версия, инженерный прототип + дизайн. Есть несколько экземпляров, есть поставщики, имеем опытный образец.
6. *Продукт*.

Жизненный цикл продукта

1. *Разработка*.
2. *Изучение рынка*.
3. *Рост*.
4. *Стадия насыщения*.
5. *Уход с рынка*.

Длительности стадий для разных продуктов разные.

Виды защиты интеллектуальной собственности

1. *Know-how*. Коммерческая тайна. Документация содержится в секрете.
2. *Торговая марка* (знак).
    - Логотип, бренд, слоган.
    - Средство идентификации продукта.
    - Можно свободно ставить ™ или ®, но защиты это не даёт.

    - Нужно регистрировать торговую марку.
    - Нужно отправить туда образец продукта и доказательство, что продукт продаётся (например, фото полки в магазине).
3. *Copyright* ©. Авторские права. Могут быть переданы кому-либо.
4. *Патент* — решение технической задачи: новое, выполнимое, полезное, отличающееся от других, неочевидное для специалистов в этой области.
    - Не дает права другим производить и продавать на рынке.
    - Необходимо раскрыть в определенной степени идею, которую предлагаете реализовать (~80% идеи).
    - В основном это Utility patents — методы, приборы, т.п.
    - В основном, истекает через 20 лет.
      Структура патента:
    - Аннотация (abstract).
    - Обзор всех разработок.
    - Детальное описание продукта (диаграммы, рисунки и текст).
    - Формула патента (claims) — фактически положение, которое вы защищаете.

3. Бизнес-план start-up компании (бизнес-проекта).

1. *executive summary*
    - 1-ая страница — аннотация (abstract). Кто что для кого будет производить, краткое описание продукта, расположение офиса (контакты).
    - 2-ая страница.
        - Конкретные цели по годам (3 предложения/абзаца — основные цели, вехи и результаты).
        - Миссия (что вы принесете в мир?).
        - Ключи к успеху (маркетерские ходы), ключ к тому, что ваш продукт будет успешным.
2. *О компании* (company).
    - 2.1. Юридическая организация компании (ООО, инд. предприниматель и пр.)
    - 2.2. Расположение участников (фрилансеры, аутсорс и всё прочее).
    - 2.3. Затраты на старте.
3. *Продукт*.
    - 3.1. В какую категорию попадает ваш продукт.
    - 3.2. Описание продукта.
    - 3.3. Производство, себестоимость.
    - 3.4. Безопасность продукта.
    - 3.5. Планы по развитию.
4. *Маркетинг*.
    - 4.1. Общее описание подобных продуктов на рынке и состояние рынка

(например сколько аналогичных товаров такого типа продаётся).
    - 4.2. Сегментация — определяем целевую группу.
    - 4.3. Стратегия, вид рекламы.
    - 4.4. Анализ индустрии.
5. *Продажи*.
    - 5.1. Конкуренты.
    - 5.2. Ценовая политика.
    - 5.3. Каналы продаж.
    - 5.4. Вехи продаж по годам (1 г. — поквартально).
6. *Менеджмент*.
    - 6.1. Владельцы.
    - 6.2. Должности (в том числе вспомогательные, например, бухгалтер).
    - 6.3. Зарплаты (в т.ч. мб оценка роста по годам).
7. *Финансы*.
    - 7.1. Затраты на стартап, кредиты, если взяты.
    - 7.2. Точка безубыточности (когда можно точно работать на собственные средства).

4. Области знаний необходимые в УП. Отличия программной инженерии от других отраслей. Эволюция подходов к управлению программными проектами.

`Программная инженерия` (ПИ) — это применение системного и измеримого подхода к разработке, эксплуатации и поддержке.

`Основные области знаний`

1. Программные требования.
2. Проектирование ПО.
3. Разработка ПО.
4. Тестирование ПО.
5. Эксплуатация и поддержка.
6. Конфигурационное управление.
7. Процессы ПИ (состыковывают части).
8. Инструменты и методы (поддерживают современные технологии).
9. Качество ПО.
10. Управление в ПИ.

`Дополнительные области знаний`

1. Разработка hardware.
2. Теоретические основы (Computer Science).
3. Системное проектирование (поддерживает инфраструктуру предприятия).
4. Управление качеством.
5. Управление проектами.
6. Общий менеджмент.

Отличия программной инженерии от других отраслей

- Неуспешные проекты — 45%.
- Успешные проекты — 35% (не прошли по срокам/средствам).
- Провальные проекты — 20% (были закрыты до выпуска).
- Виноват менеджмент. Все проблемы должны быть решены на этапе препроектной подготовки.
- Разработка ПО ближе к НИР. Это проект в нематериальной сфере.

Эволюция подходов к управлению программными проектами

- «Как получится». Разомкнутая система управления. Полное доверие техническим лидерам. Представители бизнеса практически не участвуют в проекте. Планирование, если оно и есть, то неформальное и словесное. Время и бюджет, как правило, не контролируются.
- «Водопад» или каскадная модель. Жесткое управление с обратной связью. Расчет опорной траектории (план проекта), измерение отклонений, коррекция и возврат на опорную траекторию. Лучше, но не эффективно.
- «Гибкое управление». Расчет опорной траектории, измерение отклонений, расчет новой попадающей траектории и коррекция для выхода на нее. «Планы — ничто, планирование — всё».
- «Метод частых поставок». Самонаведение. Расчет опорной траектории, измерение отклонений, уточнение цели, расчёт новой попадающей траектории и коррекция для выхода на нее.

5. Модели процесса разработки ПО. Закон четырех «П».

Модель (методология) — система принципов, понятий, методов, способов и средств, определяющие стиль разработки ПО. Их классифицируют по весу — количеству формализованных процессов.

Выбор методологии зависит:

- От самого проекта.
- От размера и профессионализма команды.
- Стабильность и зрелость процессов компании.

Модели процесса разработки ПО

0. *«Как получится»*.
1. *«Водопад» или каскадная модель* — жёсткое управление с обратной связью.
    - Следование чёткому регламенту.
    - Нельзя откатиться назад.
    - Обязательна документация.
    - Каскады обеспечивают безопасность.
    - Для военных, секретных производств и защищённых объектов.
2. *Software Capability Mature Model* (SWCMM). Имеет уровни:
    - 2.1. Начальный — когда определены немногие процессы и успех во

многом зависит от конкретных исполнителей.
    - 2.2. Повторяемый — срок, бюджет и функциональность.
    - 2.3. Определённый — когда повторяемые процессы объединены в общую систему компании.
    - 2.4. Управляемый — когда не просто используем систему, но и анализируем статистику по использованию.
    - 2.5. Оптимизируемый — когда стараемся оптимизировать процессы.
3. *RUP* (Rational Unifed Process) — универсальная система ⇒ можно работать и по гибкой системе, и по водопадной.
4. *Microsoft Solution Framework* — использует итеративную модель разработки, меньше бюрократии.
5. *Институт ПИ* (PSP/TSP) — определяет требования/компетенции.
    - *PSP*.
        - (1) Программист должен уметь оценивать объем задачи.
        - (2) Разбивать на подзадачи.
        - (3) Распределить задачи по времени и последовательности.
        - (4) Выполнять сверху собственные разработки с движением архитектуры проекта (движение синхронно с общей архитектурой, на совещаниях и пр.)
        - (5) Индивидуальная проверка кода.
        - (6) Регрессивное тестирование.
        - (7) Учитывать найденные дефекты (помечать их).
        - (8) Классифицировать найденные дефекты.
        - (9) Описывать результат тестирования.
        - (10) Учитывать своё время на разработку.
    - *TSP*.
        - (1) Команда должна иметь четкие цели.
        - (2) Четкий план и процессы взаимодействия.
        - (3) Отслеживать выполнение работы.
        - (4) Максимальная мотивация и производительность.
6. *Гибкие модели*.
    - Процесс долбжен быть адаптивным в управлении и ориентирован на управление людьми.
    - 4 важных пункта:
        - (1) Интерактивонсть.
        - (2) Инкрементальность.
        - (3) Самоуправляемость команды.
        - (4) Адаптивность.
    - Пример — SCRUM.

SCRUM (Основывается на эмпирическом подходе — знание «как управлять» приходит с опытом), для управления надо понимать три принципа:

1. Прозрачность — значимые аспекты и результаты процесса разработки должны быть доступны, все участники должны видеть как движется команда (у всех

одинаковая картина перед глазами).
2. Инспекция – выявление нежелательных отклонений от плана.
3. Адаптация – если находим отклонения от плана, то мы их корректируем.

`Спринт` – подпроект большого проекта (обычно не больше месяца), каждый такой спринт обеспечивать инкрементальность разработки.

Важно: процессы планирования спринта, процессы мониторинга отклонения от плана, разработка и обзор всего спринта с ретроспективой.

1. Продолжительность спринта короткая
2. Принятый план спринта не подлежит никакому изменению

`Scrum-команда` – аналог руководителя продукта, состоит из руководителя проекта, scrum-master (руководитель проекта на спринт), команда разработки.

В обязанности скрам-мастера входят организация проведение и подведение результатов ежедневных рабочих встреч.

Каждый на летучке говорит, что сделал с момента предыдущей встречи, что делает сегодня и какие видятся проблемы.

`Отмена спринта` – если вдруг есть неразрешимая проблема (например, изменение условий рынка, заказчика, технологий), то он отменяется владельцем продукта.

`Закон трёх «П»` (в конспекте трёх, а не четырёх, как в вопросе)

1. Сам проект.
    - По масштабу.
        - Малые (< 6 месяцев, до 50 чел. мес.)
        - Средние (от 6 до 12 месяцев, 50-100 чел. мес.)
        - Крупные (> 1 года, > 100 чел. мес.),
    - Трудоёмкость (человекомесяцы).
2. Сам продукт. Сложность продукта, его риски.
3. Персонал.
    - Уровень профессионализма проектной команды.
    - Эффективность коммуникаций.
    - Мотивация команды.
    - Сплочённость и стабильность команды.

6. Действия для успешности программного проекта.

7. Проект – основа стратегического развития компании. Критерии успешности проекта. Железный треугольник.

8. Проект и организационная структура компании, виды матричного управления.

9. Организационная структура проектной команды.

II. Фазы проекта. Инициация проекта.

10. Управление приоритетами проектов, определение ценности проекта.

11. Концепция проекта.

III. Планирование проекта.

12. Анализ содержания и состава работ. Декомпозиция и иерархическая структура работ (ИСР). Базовый план проекта.

13. Общий план проекта, виды проектных планов. Рабочий план проекта.

14. Стадии разработки ПО. Трудоемкость и сроки выполнения проекта.

IV. Реализация проекта.

15. Управление рисками проекта.

16. Управление командой проекта.

17. Инструменты количественного управления проектом.

Измерения по проекту необходимо выполнять не реже одного раза в 1-2 недели.

Совещания

1. Анализ результата за неделю
2. Проблемы (которые возникли или вот-вот возникнут)
3. Уточнение приоритетов

Показатели:

1. Показатели отклонений по объёму и по затратам
   Метод освоенного объёма:
   $\text{ОГ} = \text{ОО} - \text{ПО}$ или $SV = EV - PV$,
   где ОГ - отклонения от графика (Schedule variance)
   ОО - Освоенный объём (Earned value) - то, сколько фич сделано (напр-р, за неделю)
   ПО - Плановый объём (Planned value) - то, сколько фич запланировано

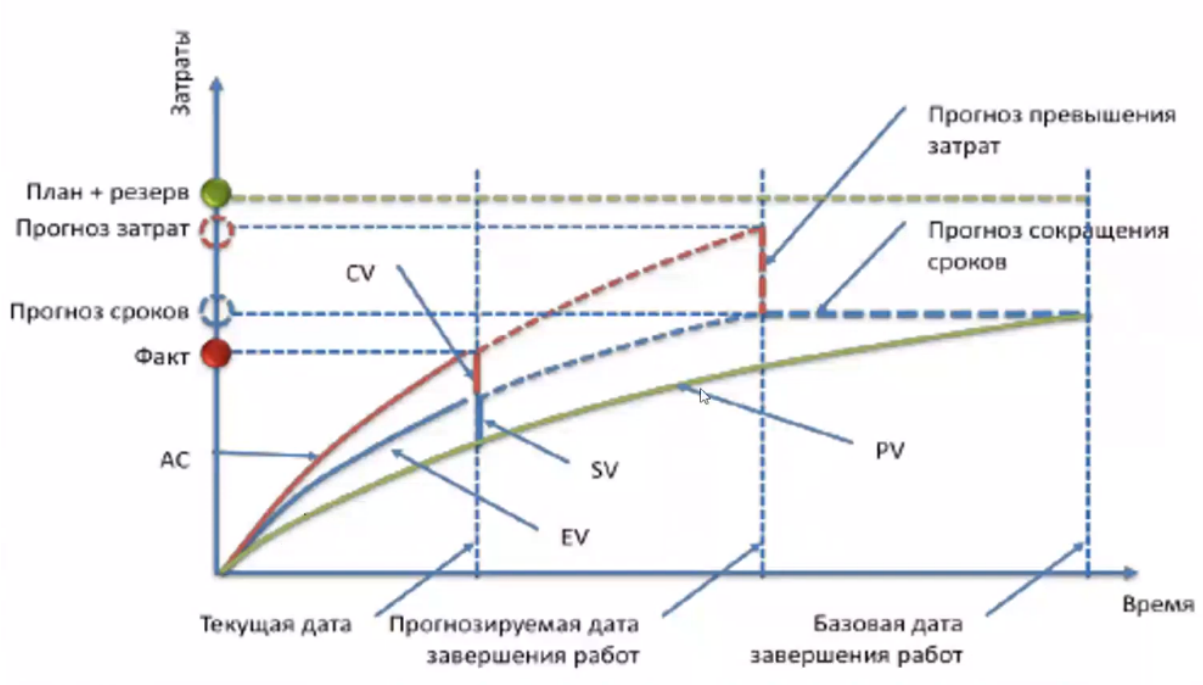
   Далее это можно переводить в денежное выражение

   > Например:
   > 15 требований за 40 чел/час по 1000 р/час
   > $PV = 15 * 40 * 1000 = 600000\text{р.}$
   > $EV = 20 * 40 * 1000 = 800000\text{р.}$
   > $SV = EV - PV = 200000\text{р.}$
   >
   > > $SV > 0$ - опережение
   > > $SV < 0$ - отставание

   Отклонения по затратам:
   $\text{ОЗ} = \text{ОО} - \text{ФЗ}$,
   где ОЗ - отклонения по затратам (Cost variance)
   ОО - освоенный объём
   ФЗ - фактические затраты (Actual costs)

   > Например:
   > $\text{АС} = 20 * (30 * 1000) + 10 * 2000 = 1000000\text{р.}$
   > $CV = EV - AC = -200000\text{р.}$

> $CV < 0$ - перерасход
>
> $CV > 0$ - экономия



Индексы сроков и затрат:

Индекс выполнения сроков (Schedule performance index):

ИВС $= $ ОО$/$ПО или $SPI = EV/PV$

> $SPI > 1$ - всё хорошо
>
> $SPI < 1$ - плохо

Индекс затрат:

Индекс выполнения стоимости (Cost Performance Index)

ИВС $= $ ОО$/$ФЗ или $CPI = EV/AC$

> $CPI > 1$ - всё хорошо
>
> $CPI < 1$ - плохо

2. *Показатель прогресса* - отношение реализованных требований к общему числу
3. *Показатель стабильности* - доля принятых изменений в плане проекта ко всему количеству требований. Чем больше, тем хуже.
4. Показатель производительности (KSLOC/чел*мес)
   KSLOC - Kilo Source Line of Code (релизного кода).

> Комментарии:

> 1. Кажется, что чем больше, тем вроде бы лучше, но от этого зависит и тестирование, а тут уже наоборот.
> 2. Функция не линейна, т.к. при росте общего объёма кода увеличивается время на интеграцию
> 3. Характерное значение KSLOC для проектов разных типов:
>   1. интранет = 800
>   2. бизнес-система = 600
>   3. интернет = 300
>   4. системное ПО = 150
>   5. системы РВ = 80
>   6. наукоёмкие системы = 50

5. Показатель качества - количество дефектов
   Например, $\text{количество\_выявленных\_дефектов}(\text{багов})/KSLOC$

   > Комментарий:
   >
   > Есть критичные дефекты, есть некритичные и можно учитывать только критичные

   Также можно измерять в

   1. Средних затратах (трудозатратах) на сопровождение
   2. Документированности кода (число строк с комментариями к общему числу строк)
   3. И так далее

Управленцу надо следить за этими показателями, но не прям гнаться (не жестко, пример с опозданием на 2-3 минуты, которое обсуждают полчаса)

Если намечается отклонение от плана, которое всегда случается, то вы их мониторите и есть допустимые, критичные и недопустимые отклонения. И на них нужно адекватно реагировать.

1. *Допустимые* - реакция - мониторинг (например, кто-то изучает новую технологию и на следующей неделе нагонит, но нужно “держать руку на пульсе”)
2. *Критичные* - нужно тщательно проанализировать (например, возросло число багов в каком-то функционале, в итоге может быть надо будет дать помощь)
3. *Недопустимые*

V. Завершение проекта.

18. Этапы внедрения программного продукта. Итоговая отчетность.

Этапы внедрения программного продукта

1. *Программа и методика испытаний* (*Целевая группа* — группа, на которой идёт тестирование, помимо альфа-версии на серверах исполнителя).
2. *Опытная эксплуатация* — бета-версия, обучение.
3. *Промышленная эксплуатация* — сопровождение и тех. поддержка.

Каждый этап заканчивается *актом сдачи и приёмки*.

Итоговая отчетность

Очень важно сохранить весь опыт, который был накоплен руководителем и командой в репозитории компании. Т.е. на основании дневника (Дневник РП — руководителя проекта), который рекомендуется вести руководителю, для документа *аналитический отчет*.

Пункты *аналитического отчета*:

1. Цель (достижение целей проекта).
2. Дополнительные полезные результаты.
3. Сроки (Как соблюдались все сроки).
4. Расходы.
5. Отклонения от целей (описать, обосновать).
6. Отклонения от требований (какие требования заказчика были проигнорированы и почему, обычно что-то неважное, какие-то капризы).
7. Уроки (чему удалось научиться в проекте).
8. Проблемы (которые возникли вследствие мб рисков, как они были решены).
9. Материальные ресурсы и программные компоненты для возможного использования в других проектах.
10. Предложения (по изменению процессов, стандартов в компании).

Служит хорошим основанием для внутренней презентации с отчетом по проекту.