

Вопросы к экзамену по «Управлению проектами»

I. История развития программной инженерии и 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 отчет 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 (маркетинговые 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 стартap, 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

многое 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 Unified 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 должен быть адаптивным в управлении и ориентирован на управление людьми.
- 4 важных пункта:
 - (1) Интерактивность.
 - (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 проведения 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.

Важно определить такие проблемы, решения которых никто не знает. Документ: концепция проект. После её создания издается приказ об открытии проекта. Определить приоритет проекта.

Оценки

1. Финансовая ценность. Окупаемость.

- Высокая – окупаемость до года, доход превышает их в 1.5 раза.
- Средняя – 1-3 года, 25%.
- Низкая – внутренние проекты.
- Шкала привлекательности (0-3).

2. Стратегическая ценность.

- Высокая – обеспечивает стратегическое преимущество на рынке (3 балла).
- Средняя – временное конкурентное преимущество (до года), выполнение обязательств для заказчиков.
- Низкая – поддержание репутации компании.

3. Уровень рисков.

- Низкий – когда цели и требования предельно понятны и документированы, определены четко масштаб и рамки проекта, доступны ресурсы нужной квалификации в полном объеме и когда не требуется новых технологий или платформ.
- Средний – есть размытость по требованиям, ресурсы доступны в основном, новая платформа.
- Высокий – плохо с требованиями, мало ясности.

Какая-то формула: $P = F + S + (4 - R)$.

// Белых предлагает $\frac{(F+S)}{R}$

11. Концепция проекта.

Концепция проекта является достаточным документом для открытия проекта. Является основой плана. Без неё нельзя открыть проект. Сначала одобряется куратором проекта, а потом – директором. По сути, это *дорожная карта*.

Состав:

1. Название. Может быть длинным, должно отображать ответы на вопросы (*).
2. Ценность.
3. Цели.
4. Задачи.
5. Ожидаемые результаты.
6. Техничко-экономическое обоснование (обоснование полезности).
7. Допущения и ограничения.
8. Ключевые участники (заинтересованные стороны).
9. Ресурсы проекта.
10. Сроки.
11. Критерии приемки.
12. Риски.

(*) Концепция отвечает на вопросы «кто?» и «зачем делает?».

Пример

Внутренний проект

- Ценность: разбалловка 1-1-3.
- Цель проекта – повышение эффективности отдела X.
- Задачи – снижение затрат и сроков на обработку заявок, повышение оперативности доступа к базе и пр.
- Ожидаемые результаты (функционал) – базовое по, документация, обучение юзеров, ТЭО – улучшение показателей, что было-что стало, например, ещё оптимизация персонала и пр.
- Допущения и ограничения – те технологические платформы, которые предусматриваются в проекте, методы, лицензии, сертификаты, нормативные, законодательные акты (например, акты по безопасности).
- Участники и заинтересованные люди – все кроме команды: заказчик проекта, спонсор проекта, потребитель; заинтересованные: куратор проекта, руководитель проекта.
- Ресурсы – материальные (столы, компы, книги и пр), нематериальные (софт, лицензии), бюджет, люди (трудоемкость – человекочасы, человекомесяцы, сколько отработает, сколько заплатим → бюджет).

// разработка занимает 25–35% времени проекта, длительность $= 2.5N^{1/3}$.

- Сроки – начало проекта, конец проекта, обосновать, указать вехи разработки.
- Критерии приёмки – требования рынка/заказчика, количественные оценки.
- Риски – какие риски видно на этапе разработки.

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. Трудоёмкость и сроки выполнения проекта.

Трудоёмкость и сроки выполнения проекта

Трудоёмкость – грубая оценка кураторами и техлидами времени на разработку. Проект открывается, если знаем как решать проблему. Если проблема не решается, то открывается предпроектный проект.

Для всего этого на этапе *концепции* оценивается *трудоёмкость*.

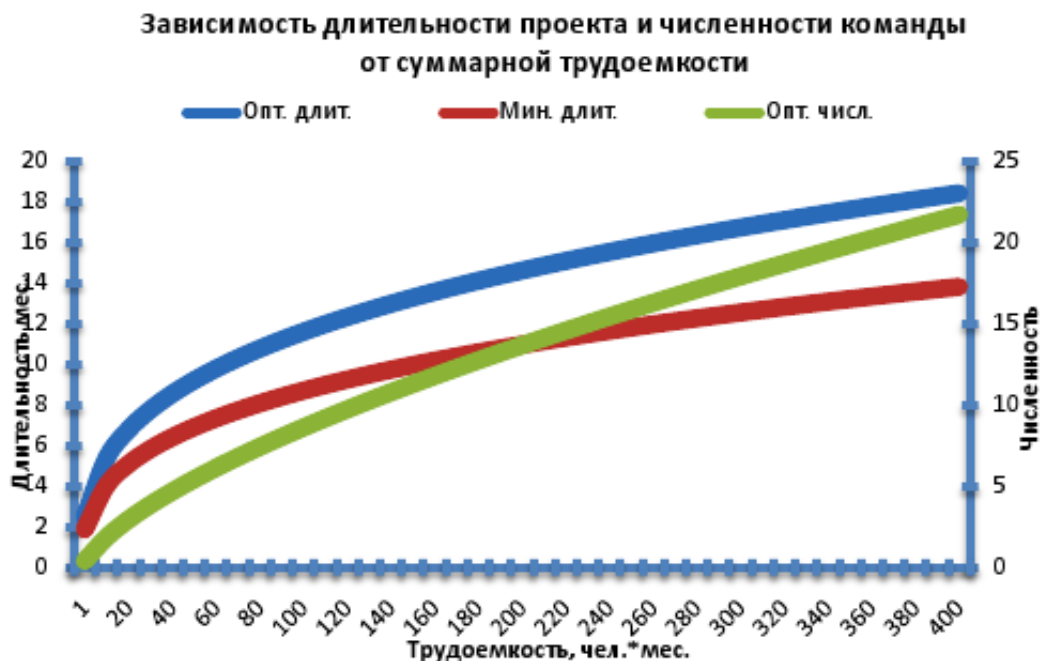
Нужно указывать даты начала и окончания проекта (его продолжительность) \Rightarrow трудоёмкость, указывать даты вех проекта.

Закон Бозма время выполнения проекта $T = 2.5\sqrt[3]{N}$, N – трудоемкость проекта, типичный проект: 50–100 человекомесяцев.

Сроки проектов программистов обычно составляют 0.5–1 год.

Для оценки трудоёмкости предлагается по каждому функционалу умножить на число π оценку программиста (человекомесяцев).

Следствием закона Бозма является кривая, дающая оптимальную численность проектной команды:



Стадии разработки

10%		15%		
Аналитика		Проектирование		
25-30%	25-20% (50% - разработка)	5%	10%	10%
Разработка	Тестирование	Документирование	Конфигурирование	Управление

IV. Реализация проекта.

15. Управление рисками проекта.

Риск – это неопределённое событие или условие, наступление которого отрицательно (или положительно) сказывается на развитии проекта и достижение его целей. Это проблема, которая ещё не реализовалась.

Проблема – это уже реализовавшийся риск.

Характеристики риска

1. Причина/источник – обстоятельство или явление, которое обуславливает риск
2. Симптомы – то, что указывает, что какое-то событие может произойти. Например, коронавирус – появление кашля, отсутствие обоняния, но это ещё не сама проблема.
3. Последствия (проблема) – проблемы, которые могут реализоваться в

результате риска.

4. Влияние – влияние риска на реализацию проекта.

Риск всегда связан с *вероятностью его наступления*.

Пример с генератором случайных чисел

Генератор от 1 до 9. 4, 5, 6 – выигрывает ИН, если остальные, то студент

Ставка 100р/1000р/10000р/100000р //генератор встроенный

Вторичные риски – преподаватель и студент, стоит ли играть с преподавателем (он может стать предвзятым на экзамене)

Риски можно разделить на две части на *известные* и *неизвестные* – те, которые можем предусмотреть и те, которые мы не можем.

Для реагирования на неизвестные риски формируется *управленческий резерв* (он входит в бюджет проекта, но распоряжаться им можно только при наступлении риска и с разрешения руководства).

Цель управления рисков – снизить вероятность наступления рисков и снизить тяжесть последствий в случае наступления.

Комментарии:

Отсутствие плана по управлению рисками значит, что вы выходите в море, а спасательных средств нет.

Чтобы правильно управлять рисками и планировать управление рисками, то нужно выделить время и ресурсы для этой проблемы. Определить общие основания для оценки рисков.

План управления рисками должен быть документом, а не просто списочком.



План управления рисками включает в себя:

1. Определение подходов и инструментов для управления рисками.

2. Распределение ролей и ответственности.
3. Выделение ресурсов и оценка стоимости мероприятий по преодолению рисков.
4. Определение сроков и частоты мероприятий по мониторингу рисков.
5. Сделать категории рисков (классифицировать их).



Рисунок 25. Пример иерархической структуры рисков проекта

- Организационные (в первую очередь) – люди, ресурсы (материальные / нематериальные), финансы, приоритеты (приоритезация/балансировка ресурсов между проектами).
 - Технические – требования, технология (новые технологии), качество (баги).
 - Внешние – заказчик (неприятные события), рыночные условия, плохая работа поставщиков/аутсорса.
6. Подходы для определения уровней вероятностей, шкалы воздействия и близости наступления рисков в проекте. Их можно условно определить количественно в денежном отображении.

Категории воздействия

1. умеренное – условно небольшие денежные потери (< \$10K),
2. критичное – существенный ущерб (\$10K–\$100K),
3. катастрофическое – например, закрытие компании (> \$100K).

Как идентифицировать риски?

- *Репозитории* с других проектов, где что-то накоплено (берется подобный проект и рассматривается то, как были оформлены риски),
- *Открытые источники* (начиная от научных работ, маркетинговых, форумы руководителей проектов и пр).

Классические методы

- *привлечь экспертов* (предоставить им как можно более полную информацию),
- *мозговой штурм* с проектной командой (выбросить идеи откуда могут прилететь риски),
- *метод Дельфи* (анонимный итерационный опрос экспертов, на каждой итерации опроса вы предоставляете всю информацию, в том числе и риски, которые на первой итерации были),
- *карточки Кроуфорда* (набор карточек, какой риск наиболее важный, в итоге получается ранжированный список рисков).

Список 11 наиболее распространённых рисков

1. Нереалистичные сроки.
2. Нереалистичный бюджет.
3. Дефицит специалистов.
4. Разрыв квалификации специалистов (математик может плохо знать физику т.е. есть условный разрыв между физиком и математиком).
5. Поток изменений (непрекращающийся).
6. Перфекционизм.
7. Реализация несоответствующей функциональности.
8. Производительность разработанной системы (если требуется, иногда точность важнее производительности).
9. Неправильный пользовательский интерфейс (похоже на 7-ой).
10. Недостатки в работе аутсорса.
11. Дефицит информации о внешних компонентах по отношению к продукту (куда он будет интегрироваться).

Наиболее важные источники рисков

1. Плохое календарное планирование
2. Текучка в проекте
3. Раздутые требования
4. Нарушение спецификаций
5. Низкая производительность разработчиков

Качественный анализ рисков – вероятность, тяжесть, близость наступления и, как следствие, определение ранга риска.

		Вероятность				
		1	2	3		
Воздействие	3	3	6	9	Ранг риска	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="width: 15px; height: 15px; background-color: #800000; margin-bottom: 5px;"></div> -высокий <div style="width: 15px; height: 15px; background-color: #FF8C00; margin-bottom: 5px;"></div> -средний <div style="width: 15px; height: 15px; background-color: #32CD32; margin-bottom: 5px;"></div> -низкий </div>
	2	2	4	6		
	1	1	2	3		

Управление рисками

1. *Уклонение от риска* – разработать план, чтобы уйти от риска.
2. *Передача риска* – переложение негативных последствий и ответственности на

третью сторону.

3. *Снижение риска* – снизить вероятность и последствия (лучше снижать вероятность).

Если не срабатывает ни один из способов рисков, то нужно его принять. Те неизвестные риски, которые не предусмотрели, обрабатываются за счёт резерва.

Российская действительность, причины провала проектов

1. Отсутствуют или часто меняются требования заказчиков.
2. Отсутствие взаимодействия с заказчиком.
3. Отсутствие необходимых ресурсов и опыта.
4. Дефицит специалистов.
5. Неполнота (некачественное) планирование.
6. Ошибки в оценке трудоёмкости и сроков.

Что часто упускается в требованиях

- Функциональные требования (установка, настройка, миграция данных, интерфейс с внешними системами, документация).
- Общесистемная (производительность, надёжность, кроссплатформенность, масштабируемость, открытость, эргономичность, безопасность).

Как взаимодействовать с заказчиком

Должен быть налажен процесс принятия изменений в требованиях и ответственный за это.

Если серьёзное требование меняется, то должна быть переоценка стоимости проекта, можно требовать компенсацию затрат в случае изменений, закладывание резервов с помощью завышенной трудоёмкости.

Должно быть постоянное взаимодействие с заказчиком (многие считают, что это стресс)

Обязанности помимо основных (сопровождение действующих продуктов, участие в совещаниях, отпуска, больничные, обучение).

80% функционала – основная часть, 20% – вторичная (необязательная).
Реализовывать сначала легкие (вторичные) требования – это ошибка.

16. Управление командой проекта.

Несколько важных вещей

1. Нельзя управлять людьми как вещами. Тот, кто так делает – плохой руководитель.
2. Сложнее управлять интеллектуальными людьми, т.к. они в большей степени требуют индивидуального подхода.

3. Эффективные команды работают с признанным, в команде, лидером. Лидера нельзя назначить, лидером становятся.
4. Неуправляемый человек никогда не станет управленцем (не просто повиновение, а с пониманием).
5. Эмоциональный коэффициент (или харизма). Если начальник харизматичный, то за ним пойдут не смотря на его интеллект (баланс может достигать 80% в сторону эмоционального интеллекта и 20% в сторону интеллекта).
 1. Начальник должен быть компетентным и харизматичным.
 2. Руководитель должен быть признанным, то есть совокупный потенциал начальника должен быть выше, чем у подчинённого.
 3. Должно быть доверие к действиям лидера (лидер должен быть честным, порядочным). Репрессивный аппарат неэффективен, особенно в наукоемких областях – здесь должна быть свобода творчества.

12 качеств и компетенций, которыми должен обладать лидер

1. Видение целей и путей их достижения (стратегическое и тактическое мышление)
2. Глубокий анализ проблем
3. Нацеленность на успех
4. Способность понимать состояние членов команды
5. Искренность и открытость в общении
6. Навыки в разрешении конфликтов
7. Умение создавать творческую атмосферу, положительный микроклимат
8. Терпимость и умение принимать право людей на собственное мнение и на ошибку (не систематическую)
9. Умение мотивировать (мотивировать профессиональное поведение)
10. Умение правильно распоряжаться людскими ресурсами (у лидера должно быть стремление выявлять и реализовывать индивидуальные способности каждого и способствовать их профессиональному росту)
11. Обладать “пробивной способностей”. Надо уметь бороться и отстаивать.
12. Ответственность (самое важное качество)

Комментарий:

Технический лидер далеко не обязательно удовлетворяет всем этим свойствам (даже скорее не удовлетворяет).

Стратегии управления

1. Директивное – на уровне распоряжений начальник доводит до коллектива и контролирует их исполнение
2. Объяснение – объяснение своих решений
3. Участие – привлечение членов команды для принятия решений
4. Делегирование – делегирует полномочия и старается не мешать, пока не требуется

Управленец должен быть адаптивным, должен менять стратегии.

Примеры:

1. Назначается новый сотрудник-начальник, принятый в команду (признания нет, доверия нет, но, допустим, есть информация, что он компетентный), лучше всего директивная стратегия.
2. Если компетенций нет – лучше стратегия участия.
3. Стратегия объяснения – когда вы уже были в коллективе, стали начальником и ваши компетенции как начальника ещё не признаны
4. Многолетняя связь с командой (есть кому доверять) – делегирования

Эффективный командный игрок (как исполнитель, так и руководитель)

Должен:

1. Четкое осознание личных и общих целей
2. Активная позиция и ответственность
3. Профессиональный рост
4. Уверенность в себе и коллегах
5. Получение удовольствия от своей работы и стремление, чтобы удовольствие получали другие
6. Оптимизм (любая новая проблема воспринимается как испытание)

То, чего не должно быть:

1. Непорядочность (лживость, отсутствие справедливости)
2. Эгоцентризм, неуважение и невнимание к коллегам
3. Завышенная самооценка (недооценка вклада других и переоценка своего)
4. Безответственность
5. Паразитизм

Для эффективной работы члена команды необходимо

1. Понимание целей и задач
2. Умение выполнить эти задачи
3. Возможность выполнить эту работу
4. Желание сделать работу (главное)

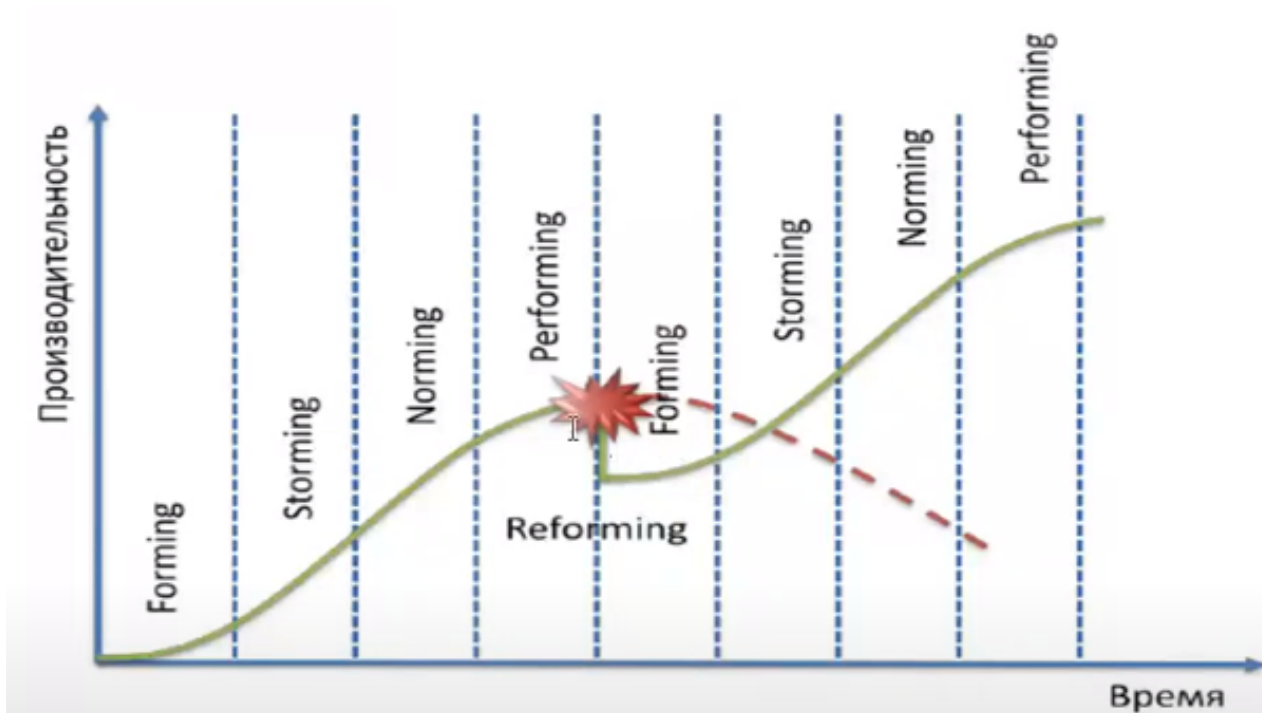
Что должен делать руководитель

1. Направить (сам руководитель должен объяснить цели и задачи, если их не понимают)
2. Обучить (курсы, наставничество)
3. Убирать препятствия
4. Мотивация (важно давать возможности)

Стадии формирования и зрелости команды

1. Формирование команды
2. Разногласие и конфликты
3. Становление команды

4. Стадия отдачи



Красное место на рисунке – добавление новых членов

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:

$$ОГ = ОО - ПО \quad \text{или} \quad SV = EV - PV,$$

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:

$$OЗ = OО - ФЗ,$$

где OЗ – отклонения по затратам (Cost variance)

OО – освоенный объём

ФЗ – фактические затраты (Actual costs)

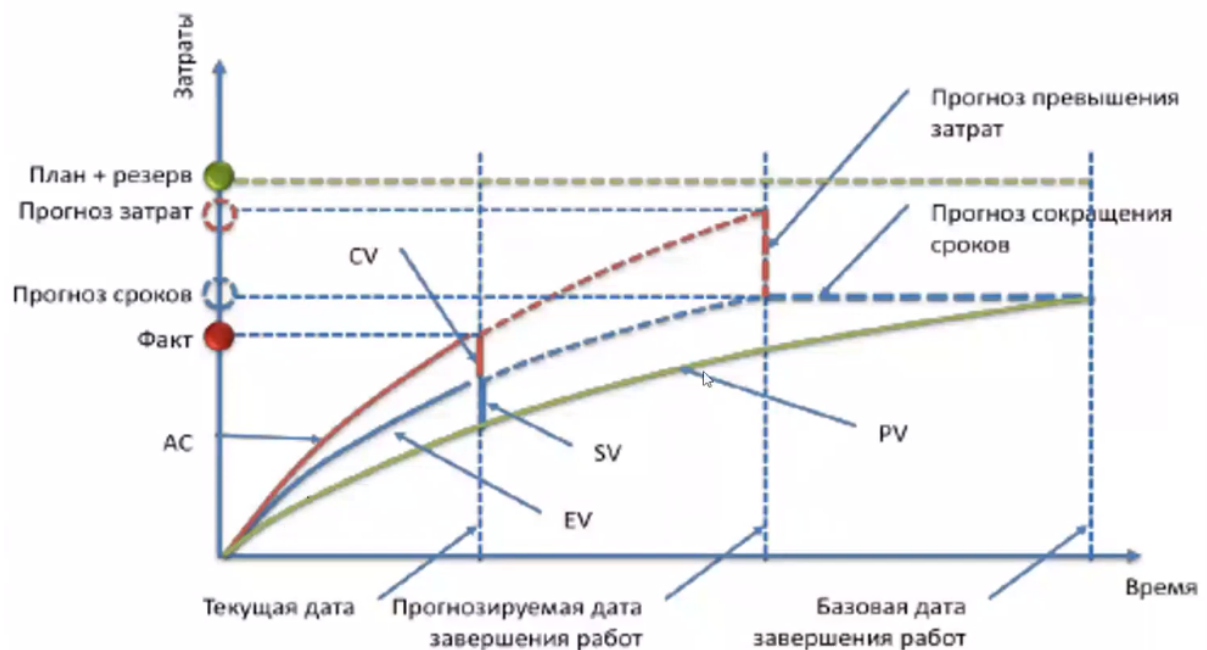
Например:

$$AC = 20 * (30 * 1000) + 10 * 2000 = 1000000\text{р.}$$

$$CV = EV - AC = -200000\text{р.}$$

$CV < 0$ - перерасход

$CV > 0$ - экономия



Индексы сроков и затрат:

Индекс выполнения сроков (Schedule performance index):

$$ИВС = OО/ПО \text{ или } SPI = EV/PV$$

$SPI > 1$ - всё хорошо

$SPI < 1$ - плохо

Индекс затрат:

Индекс выполнения стоимости (Cost Performance Index)

$ИВС = \text{ОО}/\text{ФЗ}$ 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, $\frac{\text{количество_выявленных_дефектов(багов)}}{KSLOC}$

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.