**16. Особенности продвижения информационно-технологического проекта.**

1) Стоит начать с проработки стратегии продвижения проекта. Что обязательно стоит проработать:

* Сегментирование целевой аудитории;
* Описание болей ЦА и то, как их решает ваш продукт;
* Как измениться жизнь клиента после покупки;
* Cильные и слабые стороны конкурентов;
* Просчитать юнит-экономику.

2) Проработать четкую и понятную информацию о продукте.

Чтобы продукт хотелось купить, пользователь должен как минимум понять, какую проблему проект закрывает, что входит в стоимость, есть ли поддержка и т.д.

3) Проработать информацию о своей компании.

* Должна быть проработана юридическая информация компании;
* Интересное описание вашей компании и команды;
* Ваши достижения и причины вам доверять.

4) Проработка кейсов и историй успеха. Крайне важно в сфере ПО и IT уметь рассказать на конкретных примерах — кому и как помог данный продукт.

5) Возможность попробовать бесплатно. Потенциальному клиенту важно скачать демо и пощупать своими руками продукт. Если возможности нет, то сделать уклон на онлайн-демонстрацию продукта.

**17. Контроль исполнения информационно-технологического проекта.**

Контроль выполнения проекта — это сравнение плановых сроков и бюджетов с фактическими на отдельных этапах работы над проектом.

Для контроля исполнения IT проекта для начала надо разделить задачи соответственно ролям участником команды. Далее Руководитель или назначенное им ответственное лицо должен разделить работу над всем проектом на задачи вплоть до самого низкого уровня и установить для каждой задачи сроки возможного их выполнения. В последствии руководителю или ответственному по контролю остается контактировать с командой для соблюдения установленного календарного плана.

**18. Правила разработки WBS.**

1. Каждый элемент WBS должен обеспечивать достижение ощутимого результата.

2. Каждый элемент WBS должен являться агрегатом всех подчиненных элементов, перечисленных непосредственно под ним.

3. Результаты должны логически декомпозироваться до уровня, на котором можно определить, как они будут достигаться (проектирование, поставки, заключение договоров, производство). Декомпозиция результатов, начиная от верхнего уровня WBS (проекта) до нижнего уровня должно быть логически связано.

4. Результаты пакетов работ должны быть уникальными и отличаться от результатов других пакетов работ того же уровня. Они должны декомпозироваться до уровня детализации, обеспечивающей успешное планирование, координацию и контроль работ, связанных с достижением поставленных целей.

5. Процесс разработки WBS должен представлять собой гибкий механизм, позволяющий корректировать WBS, особенно когда объем работ по проекту может изменяться. Однако, для успешного управления проектом, необходимо тщательно обеспечить процесс контроля изменений для документирования и управления изменениями содержания проекта. При изменении содержания проекта WBS должна быть откорректирована.

6. Каждый элемент WBS(пакет работ), представляющий собой объем работ подрядчика или других внешних организаций, должен быть согласован непосредственно с соответствующими элементами WBS подрядчика.

7. Все результаты в явном виде должны быть включены в WBS.

8. Для всех важных событий, связанных с отчетностью (например, Штабы, ежемесячные отчеты, отчеты о проведении испытаний и т.д.) должны быть включены и определены соответствующие пакеты работ.

9. Все пакеты работ должны быть совместимы с организационной структурой и структурой затрат.

10. Результаты должны быть четко определены так, чтобы исключить дублирование объемов работ внутри элементов WBS, в целом по организации или отдельными ответственными за выполнение работ.

11. Результаты должны иметь размер, достаточный для эффективного управления, но не настолько малый, чтобы сделать затраты на контроль чрезмерными.

**19. Виды группировки задач в WBS.**

По стадиям жизненного цикла проекта (например, отдельно описываются результаты фаз планирования, анализа, разработки, приемки и проч.) – это самый простой и популярный подход, особенно если проект идет по утвержденному процессу и всем понятно, что должно быть на выходе какой фазы.

По высокоуровневым результатам проекта (проект разбивается на ключевые результаты, например, готовая система, обученные пользователи, разработанная нормативная документация, согласованное использование системы с государственными органами и проч.)

По организационной структуре (например, вы, заказчик, подрядчик(и) и проч.) – этот вариант удобен, когда вам надо жестко разграничить ответственность за результаты работ.

По срокам (например, по кварталам) – если для проекта критична привязка к срокам.

По техническим областям (производство, маркетинг, закупки и проч.)

По источникам финансирования (какая часть результатов за какие средства достигается или за бюджет какого года, например).

**20. Основные виды бригад разработчиков и характеристика ролей в них.**

В обычной бригаде старший программист (лидер бригады) непосредственно руководит работой младших программистов. Недостатки такой организации непосредственно связаны со спецификой разработки ПС: программисты разрабатывают сильно связанные части программной подсистемы, сам процесс разработки состоит из многих этапов, каждый из которых требует особенных способностей от программиста, ошибки отдельного программиста могут препятствовать работе других программистов. Успех работы такой бригады достигается в том случае, когда ее руководитель является компетентным программистом, способным предъявлять к членам бригады разумные требования и умеющим поощрять хорошую работу.

В неформальной демократической бригаде поручаемая ей работа обсуждается совместно всеми ее членами, а задания между ее членами распределяются согласованно в зависимости от способностей и опыта этих членов. Один из членов этой бригады является лидером (руководителем) бригады, но он также выполняет и некоторые задания, распределяемые между членами бригады. Неформальные демократические бригады могут весьма успешно справляться с порученной им работой, если большинство членов бригады являются опытными и компетентными специалистами. Если же неформальная демократическая бригада состоит, в основном, из неопытных и некомпетентных членов, в деятельности бригады могут возникать большие трудности. Без наличия в бригаде хотя бы одного квалифицированного и авторитетного члена, способного координировать и направлять работу членов бригады, эти трудности могут привести к неудаче проекта.

В бригаде ведущего программиста за разработку порученной программной подсистемы несет полную ответственность один человек, называемый ведущим программистом и являющийся лидером бригады: он сам, конструирует эту подсистему, составляет и отлаживает необходимые программы, пишет документацию к подсистеме. Ведущий программист выбирается из числа опытных и одаренных программистов. Все остальные члены такой бригады, в основном, создают условия для наиболее продуктивной работы ведущего программиста. Дублер ведущего программиста также является квалифицированным и опытным программистом, способным выполнить любую работу ведущего программиста, но сам он эту работу не делает. Главная его обязанность – быть в курсе всего, что делает ведущий программист. Он выступает в роли оппонента ведущего программиста при обсуждении его идей и предложений, но решения по всем обсуждаемым вопросам принимает единолично ведущий программист. Администратор базы данных разработки отвечает за сопровождение всей документации (включая версии программ), возникающей в процессе разработки программной подсистемы, и снабжает членов бригады информацией о текущем состоянии разработки. Эта работа выполняется с помощью соответствующей инструментальной компьютерной поддержки. В зависимости от объема и характера порученной работы в бригаду могут быть включены дополнительные члены, такие как

* распорядитель бригады, выполняющий административные функции;
* технический редактор, осуществляющий доработку и техническое редактирование документов, написанных ведущим программистом;
* инструментальщик, отвечающий за подбор и функционирование программных средств, поддерживающих разработку программной подсистемы;
* тестовик, готовящий подходящий набор тестов для отладки разрабатываемой программной подсистемы;
* один или несколько младших программистов, осуществляющих кодирование отдельных программных компонент по спецификациям, разработанным ведущим программистом.

**21. Управление рисками.**

Управление рисками в проекте — это процесс поиска, оценки и предотвращения потенциальных проблем. Этот процесс регулярный, превентивных действий на старте проекта недостаточно.

**Временные риски.** Это вероятность того, что на выполнение задач в проекте уйдёт больше времени, чем запланировано. Помните о сроках, потому что время — это ресурсы. Если команда тратит много времени на задачи, растёт и фонд оплаты труда. Кроме того, стейкхолдеры проекта могут разочароваться из-за задержек.

**Бюджетные риски.** Из-за плохого планирования стоимость проекта может оказаться больше, чем заложено в бюджете. Обычно бюджет закладывают перед запуском проекта, тогда же планируют траты по статьям. Если команда не уложится в план, потребуются дополнительные средства, и если их не будет, проект остановится.

**Риски изменения объёмов работы.** Они могут появиться, если исполнители не поняли требований заказчика или он сам внёс в проект изменения. Это может привести к пересмотру бюджета, сроков и списка задач.

**Внешние риски.** Это потенциальные события, которые находятся за пределами компании и которые компания не может контролировать. Например, на проект могут повлиять новые законы.

**Единая точка отказа.** Так называют единственное событие, которое может остановить всю работу над проектом. Ни один член команды не сможет дальше выполнять свои задачи, пока проблема не решится. Например, для интернет-магазина единой точкой отказа может стать отключение электричества в офисе. Если доступ к инструментам, таким как CRM, был только из офиса, вся команда не сможет выполнять задачи.

В результате команда не выполнит ни одной задачи. Зная об этой точке отказа, можно принять меры: создать резервные копии сервисов и информации в облаке.

**Есть четыре основные стратегии, как реагировать на риски.**

**Избегать.** Иногда вы можете избежать риска полностью. Например, если вы сомневаетесь в надёжности подрядчика, который часто не соблюдает сроки, вы можете перестать работать с ним.

**Принять.** Этот способ подойдёт для рисков с низкой или средней вероятностью и без тяжёлых последствий для проекта. Нужно принять, что такой риск существует, и отслеживать его всё время до окончания проекта.

Представим, что поставщик неожиданно заявил, что у него нет нужных вам компонентов, однако он пополнит запасы в ближайшее время. Возможно, это скажется на сроках проекта.

Можно начать работу с другим поставщиком, но такой риск лучше принять. Это имеет смысл, если задержки не критичны для проекта. Если не искать нового поставщика и смириться с риском, это избавит команду от лишней работы.

**Уменьшить или контролировать.** Для смягчения риска используйте дерево решений. Это блок-схема, которая показывает, какие решения существуют для каждой проблемы. Например, если компания работает с исполнителем, который срывает сроки, ему можно постоянно напоминать о задаче: отправлять имейлы каждый день или звонить.

**Передать риски.** Если команда понимает, что не может снизить риски для какой-то группы задач, она может передать их специализированным компаниям. Иногда это помогает сэкономить время и деньги.

**План по управлению рисками** — это документ, который описывает возможные риски и способы их снизить. Если у вас есть такой план, все члены команды и заказчики будут в курсе, какие проблемы могут возникнуть во время реализации проекта. Документ нужно постоянно дополнять, так как новые риски могут появиться на любом этапе проекта. Например можно создать таблицу, в которой вы распишете все возможные риски, оцените их и добавьте возможные решения для каждого.

**22. Управление требованиями.**

**Управление требованиями** — процесс, включающий идентификацию, выявление, документацию, анализ, отслеживание, приоритизацию требований, достижение соглашений по требованиям и затем управление изменениями и уведомление заинтересованных лиц. Управление требованиями — непрерывный процесс на протяжении всего жизненного цикла продукта.  
  
**Требование** — это любое условие, которому должна соответствовать разрабатываемая система или программное средство. Требованием может быть возможность, которой система должна обладать и ограничение, которому система должна удовлетворять.

Основная цель управления требованиями — обеспечить четкое, четкое и безошибочное выполнение требований для инженерной группы, чтобы они могли убедиться в обнаружении ошибок в системе и потенциально снизить стоимость проекта, а также риск.

Ключевые проблемы управления требованиями:

1. Конфликт интересов (будь то конфликт в коллективе или может быть даже между заказчиком и исполнителем)
2. Непринятие ответственности
3. Субъективность оценки требований
4. Нарушение сроков разработки
5. Сложность сбора и консолидации (под сложностью сбора требований понимается собирательный процесс. Сюда входит и удалённость владельцев информации, нежелательное или непонимание важности описания бизнес-процессов