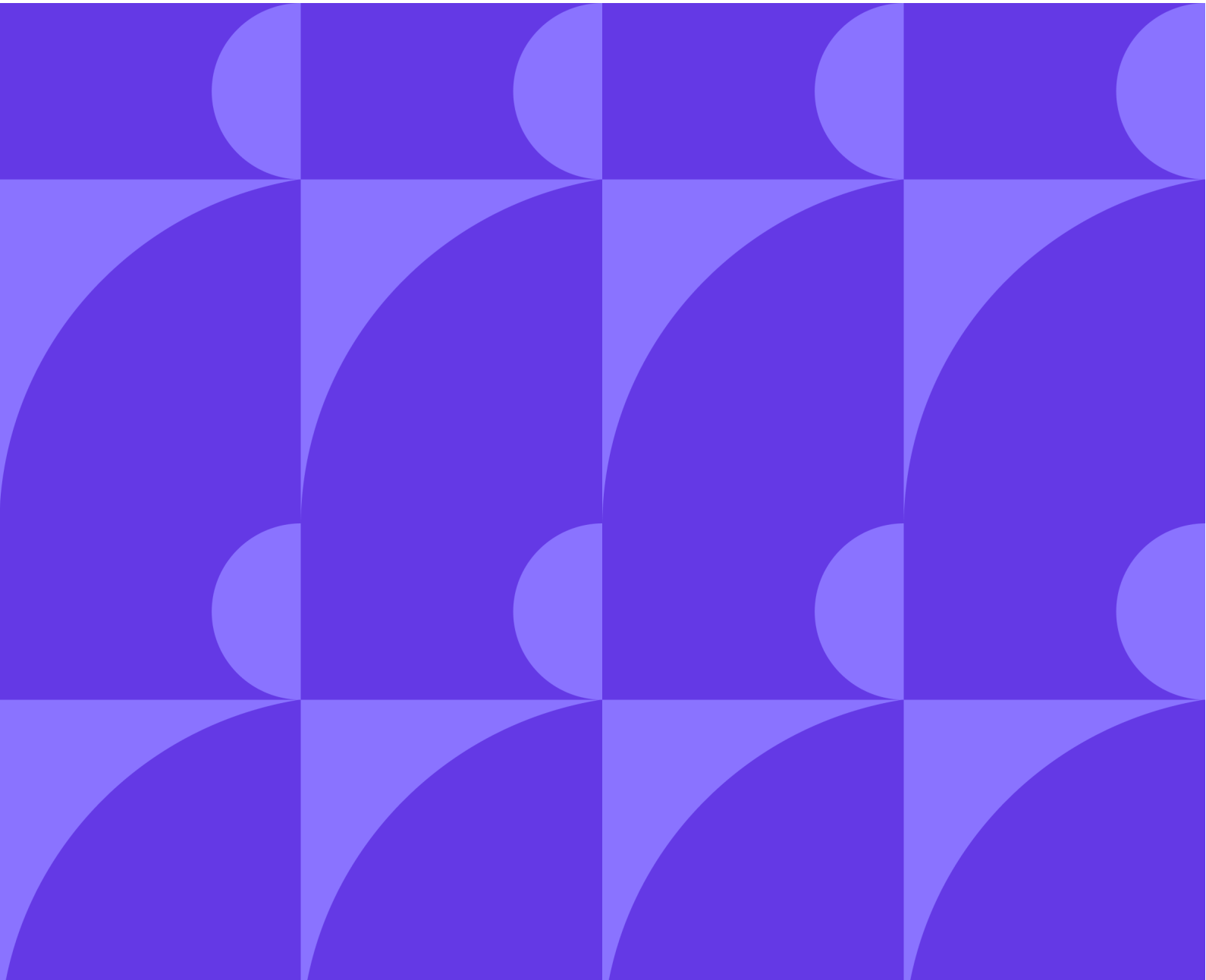
Гибкие методологии

Как выбрать гибкую методологию и не ошибиться



На этом уроке

1. Поговорим, на что в первую очередь необходимо обратить внимание при выборе методологии управления разработкой.
2. Обсудим, какие основные ошибки допускают новички.
3. Рассмотрим кейсы выбора гибких методологий разработки продукта.

Оглавление

Глоссарий

Заинтересованные стороны — это лица и организации, которые активно участвуют в проекте, интересы которых могут быть затронуты в ходе исполнения или в результате завершения проекта. Могут влиять на проект или на его результаты. Находятся на разных уровнях внутри организации и имеют разные уровни полномочий, а также могут быть внешними по отношению к исполняющей организации проекта. Примеры заинтересованных сторон — заказчики, спонсоры, исполняющая организация и общественность.

ПО — программное обеспечение.

ТЗ — техническое задание.

Теория урока

1. Почему это важно
2. Основные методологии управления разработкой
3. Чем руководствоваться при выборе методологии управления разработкой
4. Критерии выбора методологии управления разработкой продукта

Почему важно в самом начале выбрать подходящую методологию?

Методология управления разработкой подобна технологической дорожной карте или набору задач для IT-проекта. Она предоставляет проектной команде комплект инструкций, показывает набор необходимых процессов и фреймворков для успешной разработки IT-продукта.

Методология поможет команде быстро приступить к работе, стандартизировать результаты и ускорить принятие решений.

Часто проектные команды меняют методологию, чтобы создать подход, соответствующий потребностям клиента, проекту и своим навыкам. Встречаются случаи, когда команда берёт уже существующую методологию и адаптирует её под реалии своего проекта, а иногда создаёт симбиоз нескольких методологий, фреймворков и подходов.

Но для этого нужно сперва подобрать наиболее подходящую для проекта методологию.

Основные методологии управления разработкой

Рассмотрим основные методологии для управления IT-проектами. Некоторые из них более популярны в практическом применении, некоторые редко встречаются в чистом виде — их совмещают с другими методологиями управления разработкой.

Каскадная модель (Waterfall)

Первая методология управления IT-проектами. Очень последовательна, включает набор автономных этапов:

1. анализ требований;
2. проектирование;
3. разработка;
4. тестирование;
5. техническая поддержка.

⚠ **Внимание!**
Перейти к следующему этапу можно только после завершения предыдущего. Внесение исправлений в этапы практически запрещено.

В методологии Waterfall большое внимание уделяется подробной документации — это необходимо для перехода на следующий этап реализации проекта. При применении методологии важно максимально чётко представлять конечный результат.

Waterfall — простая в управлении методология. Её особенности — чёткая структура и конкретные сроки: как общего выполнения IT-проекта, так и каждого этапа.

V-образная модель

V-образная модель была придумана как улучшение каскадной модели для IT-проектов, нуждающихся в тестировании компонентов продукта по мере разработки.

Модель учитывает специфику проектов для государственных органов:

- фиксированные требования;
- фиксированная стоимость;
- фиксированное время.

Отличие от каскадной методологии в том, что этап анализа и проектирования связан с этапом тестирования.

Разберём на примере: представим, что мы разрабатываем сайт для государственного органа — Налоговой инспекции.

Во время анализа требований одновременно изучаем подходы к тестированию, во время проектирования архитектуры системы разрабатываем высокоуровневые планы и сценарии тестирования, во время проектирования компонентов системы изучаем способы тестирования компонентов и их взаимодействия, создаём сценарии тестирования, пишем утилиты, помогающие в тестировании, инструкции, скрипты и так далее.

Всё это помогает лучше понять требования и спроектировать систему. При этом мы можем сразу протестировать то, что разрабатываем: разработали систему сбора налогов с физических лиц — протестировали; увидели ошибки — исправили их и пошли дальше разрабатывать систему сбора налогов с юридических лиц. И так пока не завершим весь проект.

Почему выбираем такую методологию?

- изначально понимаем, что в итоге хочет заказчик: уже есть нормативные акты, регулирующие сбор налогов, осталось только его воплотить в цифровом автоматизированном продукте;
- заказчик — государственный орган, у него есть определённые нормативные акты, по которым он должен формировать бюджет на разработку и планировать сроки реализации проекта — как в целом, так и отдельно по этапам;
- если выберем каскадную модель, не сможем протестировать отдельные этапы всей системы сбора налогов — это приведёт к тому, что в самом конце проекта можем получить плохо интегрированную систему с кучей ошибок в коде.

Однако здесь, как и в каскадной модели, нежелательно, чтобы требования менялись во время разработки.

Итеративная модель

Итеративная разработка — это процесс создания ПО небольшими этапами, в ходе которых ведётся анализ промежуточных результатов, выдвигаются новые требования и корректируются предыдущие этапы работы. Итеративная модель способствовала зарождению Agile-философии.

Жизненный цикл проекта при итерационной разработке разбит на последовательность итераций. Каждая из них — это проект в миниатюре, включающий в себя все процессы разработки ПО:

- сбор и анализ требований;
- составление спецификаций;
- реализацию;
- тестирование;
- запуск.

Как правило, цель каждой итерации — получить версию ПО, которая включает в себя как новые возможности, так и функциональность предыдущих итераций. А результат финальной итерации содержит весь нужный функционал продукта.

Обычно бюджет и сроки реализации финальной версии изначально не устанавливаются, так как не определяется общий объём задач, а требования формируются по ходу работы.

В чистом виде итеративная модель почти не встречается на проектах. Разработчики понимают, что нужно двигаться итерациями, постепенно наращивать функционал продукта, но чтобы сделать это на практике, нужно выбрать одну из существующих Agile-методологий: Scrum, Kanban и другие.

Спиральная методология

Спиральная методология — подход к разработке ПО, комбинирующий модель Waterfall и итерационной модели. Её особенность — фокус на рисках, влияющих на организацию жизненного цикла.

Каждый виток спирали соответствует созданию фрагмента или версии ПО. На витке уточняются цели и характеристики IT-проекта, определяется его качество и планируются работы следующего витка. Детали проекта углубляются и последовательно конкретизируются, а в результате выбирают обоснованный вариант, который доводят до реализации.

Каждый виток разбит на четыре сектора:

1. определение целей;
2. оценка и разрешение рисков;
3. разработка и тестирование;
4. планирование следующей итерации.

Разберём на примере:

Недавно мы работали над созданием автоматизированной системы обработки багажа в аэропорту. Учитывая современную нестабильную ситуацию, есть риски, что IT-система может не пригодиться. Возможно, упадёт пассажиропоток и нечего будет обрабатывать. К рискам прибавляется заказчик в лице аэропорта с государственной долей в капитале, а значит необходимо в соответствии с внутренними нормативными актами чётко спланировать бюджет и сроки реализации проекта.

Таким образом:

- применить гибкую методологию нельзя из-за того, что нужно сразу определить бюджет и сроки реализации проекта;

- если применить каскадную модель, сильно возрастёт риск вложить огромный бюджет в проект, который может не окупиться в будущем и не пригодиться из-за сильного снижения пассажиропотока.

Принимаем решение разделить проект на несколько витков спирали и действовать постепенно:

1. Сначала сделали систему сбора багажа, применив на этом витке каскадную модель. Определили бюджет и сроки, чётко прописали ТЗ, разработали систему.
2. Начали думать над следующим витком — расширение этой системы на несколько терминалов.
3. Для инициации нового витка проанализировали риски падения пассажиропотока, поняли, что всё в норме, начали разработку.

И так пока не закончим весь наш большой IT-проект.

Agile-философия

Agile-философия предполагает набор гибких методологий, фреймворков и подходов по управлению разработкой ПО. Появилась как ответ на провал методологии Waterfall для управления сложными IT-проектами.

В подходе и идеологии Agile — это противоположность Waterfall. Она ориентирована на быстрый и гибкий подход к разработке. В ней нет большого сбора требований, этапы итеративны, допускаются постепенные изменения в ответ на меняющиеся требования рынка и заказчиков. Из-за особенностей подхода у IT-проекта нет чёткого времени окончания и невозможно сразу определить общую стоимость.

Agile-философия включает набор гибких методологий, фреймворков и подходов — Scrum, Kanban, Lean и другие, менее популярные.

Scrum

Scrum — это не полнофункциональная методология управления IT-проектами, а фреймворк, который описывает подход к гибкому управлению с акцентом на проектные команды, короткие спринты и ежедневные встречи.

У Scrum есть особые методы и тактики для управления IT-проектами. Например, фреймворк делает акцент на структуру команды: часто в ней нет руководителя, ожидается, что команда будет самоорганизующейся и самоуправляемой.

Методология идеально подходит для высококвалифицированных и опытных команд.

Kanban

Kanban — ещё одна разновидность Agile-методологий. Подходит для быстрой разработки ПО, требует обмен информацией в режиме реального времени и полную прозрачность работы. Главный инструмент для достижения этих целей — Kanban-доска со столбцами:

1. to do (что нужно сделать);
2. in progress (в процессе);
3. in review (на проверке);
4. done (сделано).

Столбца «in review» может и не быть — всё зависит от специфики проекта.

⚠ **Внимание!**
В зависимости от проекта количество столбцов может меняться.

Визуализация на доске позволяет членам команды отслеживать состояние каждой части работы. При этом нет разделения на конкретные роли — команда направлена на быстрое завершение задач, поэтому одни сотрудники могут помогать другим, даже если это не их зона ответственности.

Совмещение Waterfall и Agile

Часто команда приходит к тому, что нужно объединить Waterfall и Agile. Получается гибридная методология или **структурированный Agile**, совмещающий лучшие черты обоих подходов.

При совмещении получается гибкий, но структурированный подход с фокусом на сборе и анализе требований на начальном этапе (Waterfall) и последующими быстрыми итерациями и внесением изменений (Agile).

Чем руководствоваться при выборе методологии управления разработкой?

Критерий «Заинтересованные стороны»

При выборе методологии управления нужно оценить требования заинтересованных сторон (заказчик, исполнитель и другие ключевые участники проекта).

Некоторые методологии требуют, чтобы заинтересованные стороны регулярно привлекались на каждом этапе IT-проекта.

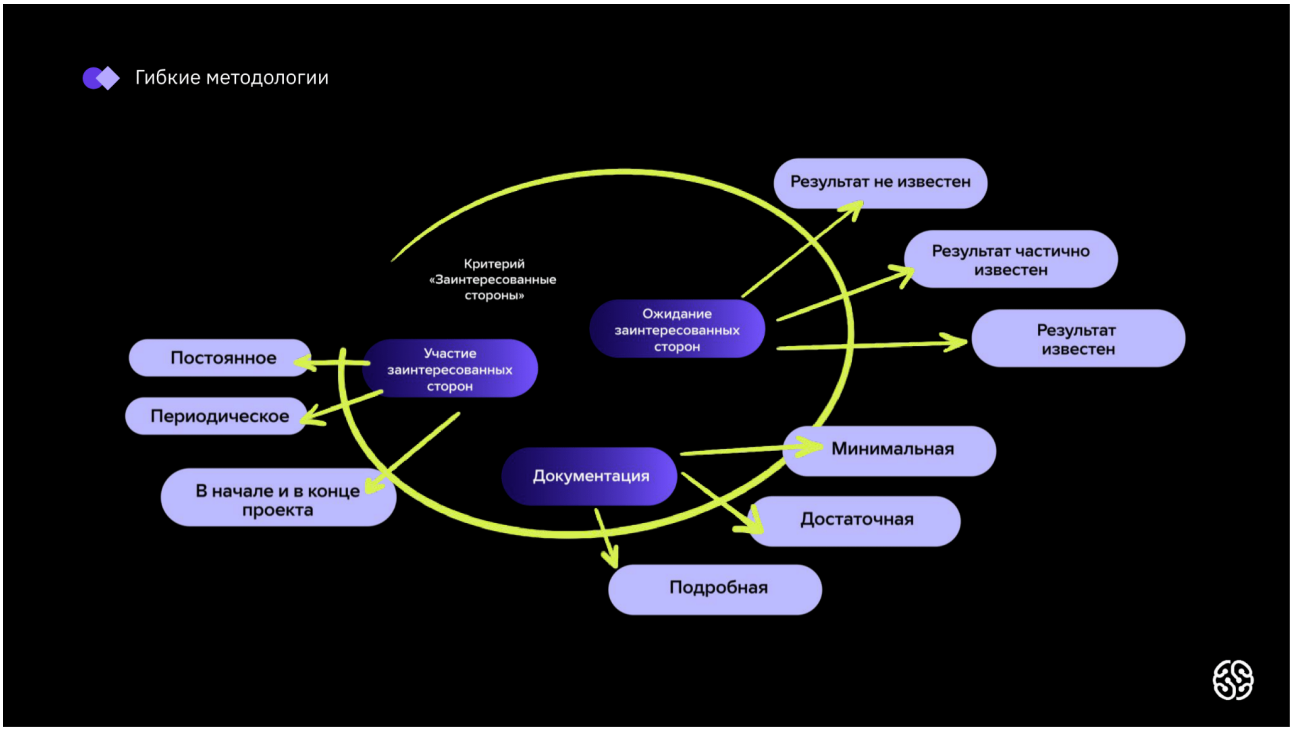
Например, в Agile-методологиях важно, чтобы заинтересованные лица были регулярно доступны для обратной связи. Если они заняты, стоит выбрать методологию, где их участие снижено. Например, гибридную методологию — Waterfall + Agile.

Если известно, что заказчик точно представляет, каким должен получиться конечный продукт, и у него уже есть чётко сформулированное ТЗ, прописанное до малейших деталей, подходит структурированный подход — Waterfall.

Если у заказчика нет чёткого представления, а масштаб проекта меняется, лучше выбрать более гибкую методологию.

Также важно сразу обсудить с заказчиком формат предоставления результатов — нужно ли документировать каждый этап или достаточно предоставить частично готовый продукт?

Ниже на рисунке — показатели критерия (выделены жёлтым) и их возможные значения:



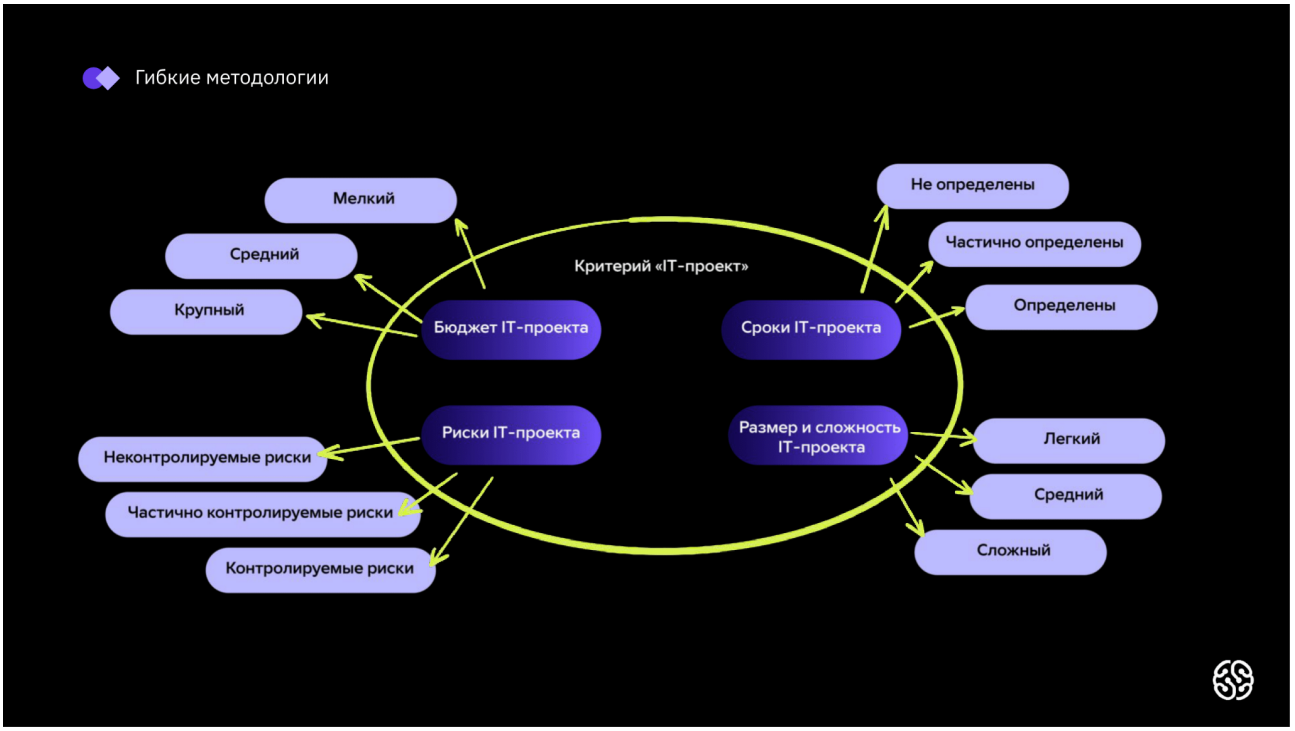
Критерий «IT-проект»

У любого IT-проекта есть сроки и бюджет. Сроки могут быть плавающими и окончательными, от бюджета будет зависеть состав команды разработки.

Эти критерии могут напрямую повлиять на выбор методологии управления IT-проектом. Например, если объём работы большой, а сложность высокая, нужна большая и разнообразная проектная команда и гибкая методология разработки.

При подборе методологии управления важно оценить существующие и потенциальные риски. В основном они зависят от сложности проекта, бюджета и ожидаемого результата. Если заказчик чётко видит результат, то риски минимальны, можно использовать Waterfall-методологию. Если ожидаемый результат расплывчат, стоит рассматривать гибкие методологии.

К критерию «IT-проект» относятся показатели:



Критерий «Организация»

То, как компания организована, её культура и прошлые IT-проекты будут влиять на выбор методологии управления. Например, некоторые методологии подходят только крупным организациям с установленной иерархией, где хорошо налажены процессы.

Понятие «культура» эквивалентно совокупности понятий «традиции», «привычки» и «общепринятые ценности». Культура обусловлена историей компании, маркетинговой стратегией, структурой собственности. На неё влияют месторасположение компании (страна, город) и род её деятельности.

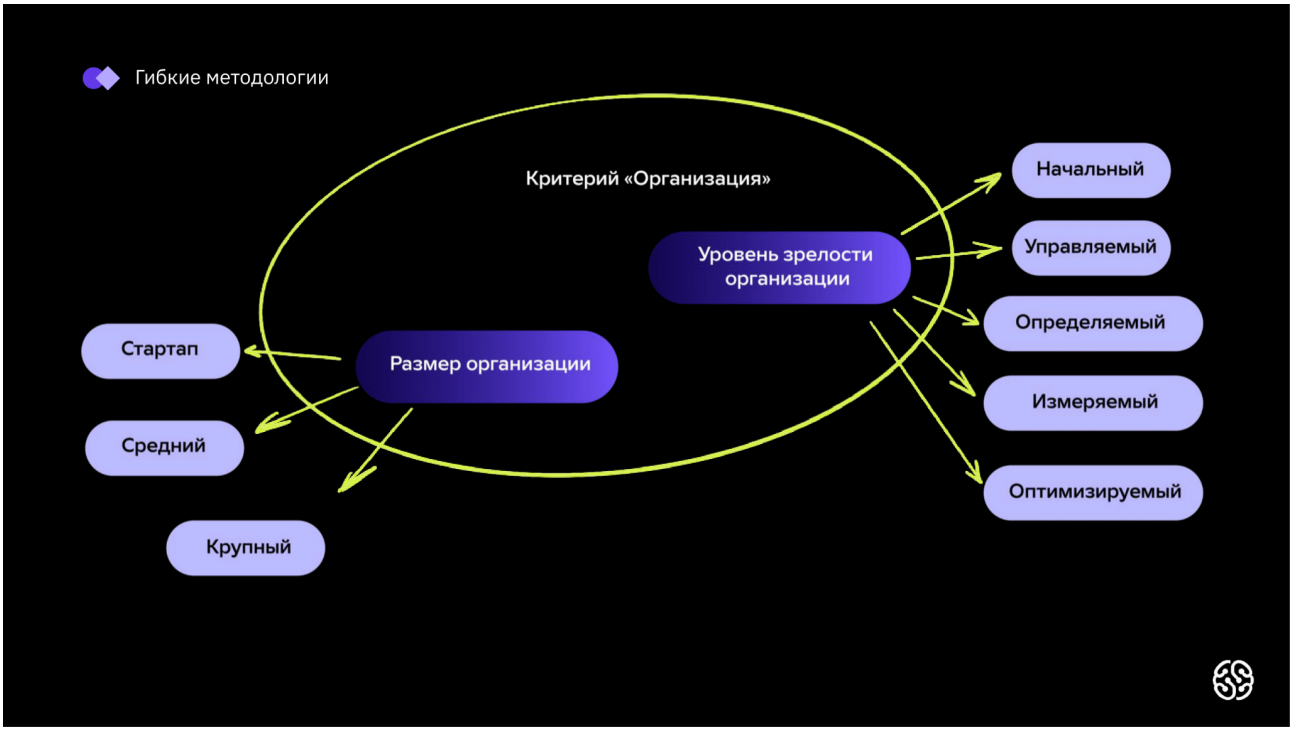
У любой организации есть своя культура. Она же является основным фактором, обеспечивающим повторяемость процессов, их устойчивость к изменению/увеличению требований, сокращению сроков.

Сотрудники компании должны понимать, почему процесс разработки построен именно так, и к каким последствиям может привести нарушение процесса. Только процесс,

построенный на основе собственного опыта и требования заказчика, будет повторяемым, и только для такого процесса существует возможность улучшения.

Всё это определяет уровень зрелости организации.

К критерию «Организация» относятся показатели:



Критерий «Проектная команда»

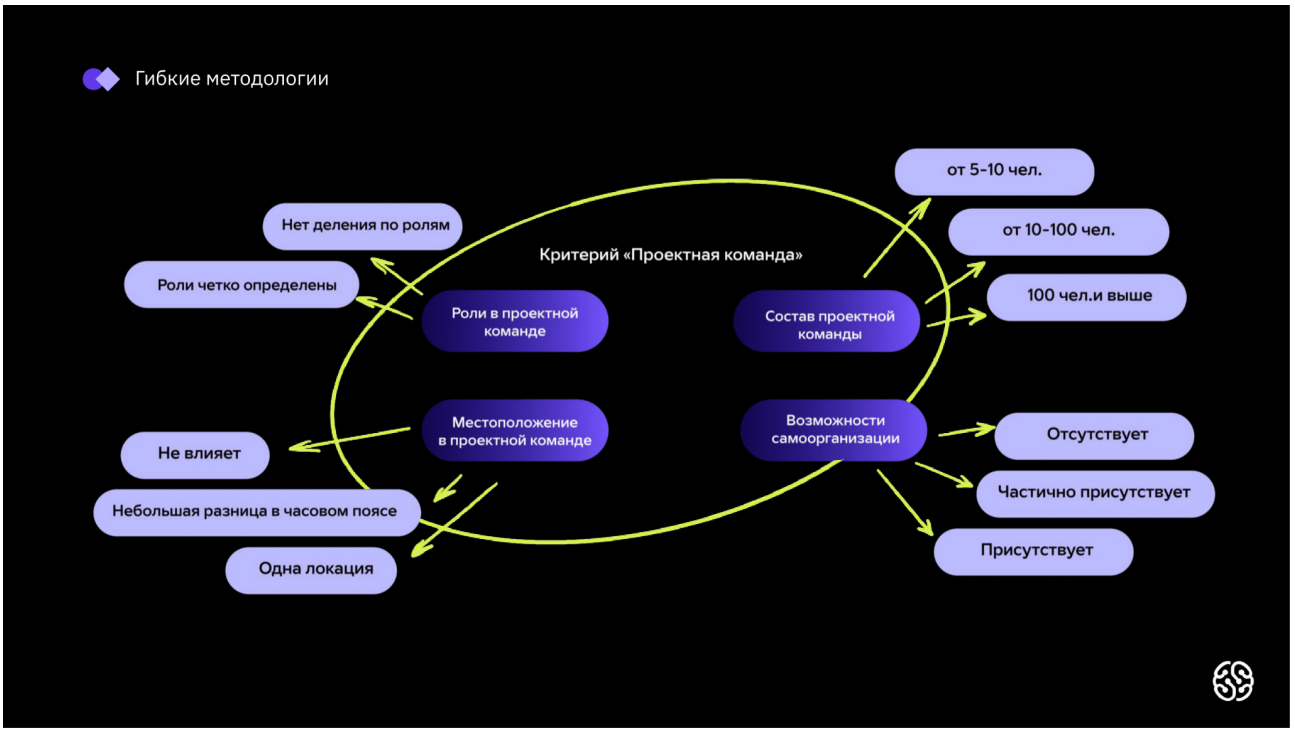
Методология управления IT-проектом — это, по сути, план проекта, который показывает проектной команде, что и когда создавать. Но для этого проектная команда должна быть в состоянии прочитать и понять план.

Если проектная команда не знакома с выбранной методологией, будет сложно добиться результатов. Сперва придётся посвятить время изучению методологии (которой могут противостоять некоторые члены команды), что приведет к задержкам в реализации проекта.

Чтобы оценить знания методологии в команде, сперва нужно провести анализ её состава — определить количество участников, выявить их слабые и сильные стороны.

Например, ****если команда высоко мотивирована и дисциплинирована, ей подойдёт SCRUM. Если человеческие ресурсы ограничены, — Kanban, в котором нет чётких ролей.

К критерию «Проектная команда» относятся показатели:



Критерий «Продукт»

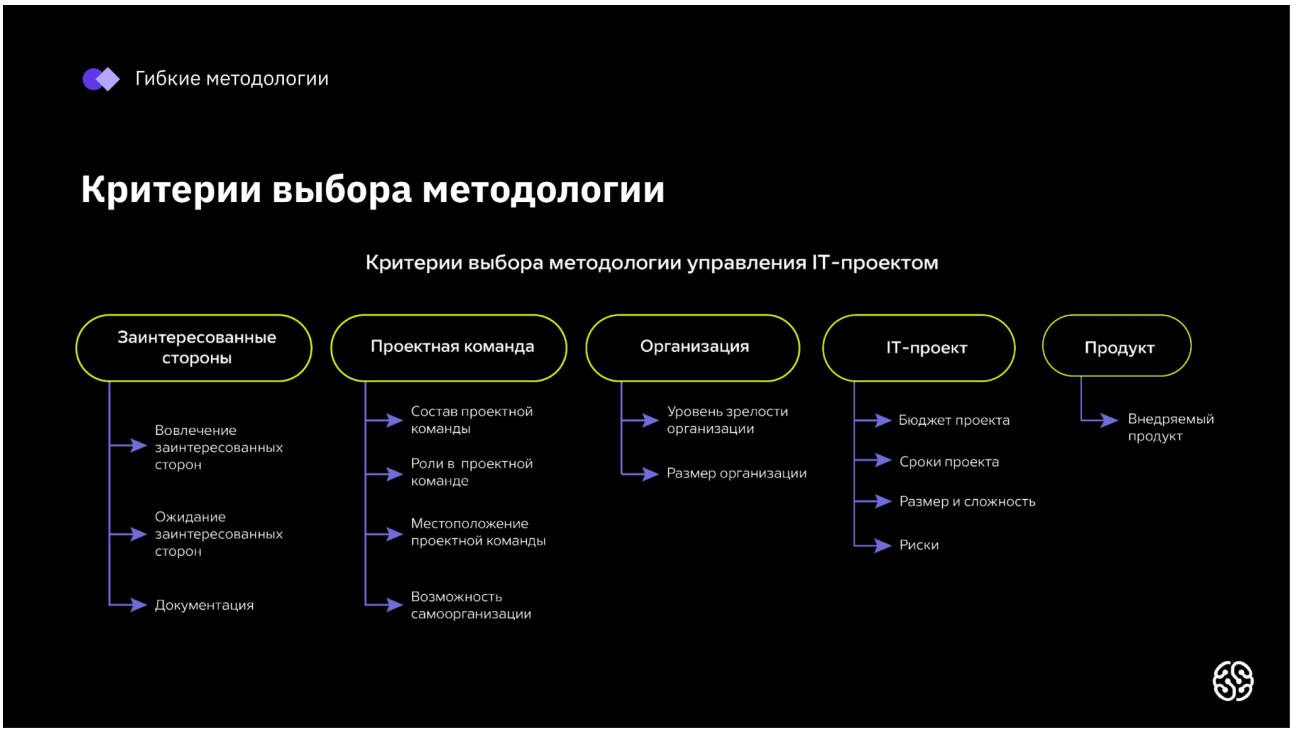
Компания может предоставлять собственную разработку или быть вендором. Нужно проанализировать, будет ли это разработка с нуля под нужды заказчика или внедрение готового решения / доработка готовой функциональности.

Для критерия «Продукт» можно выделить показатель — внедряемый продукт и определить его значения:

- разработка с нуля;
- внедрение готового решения;
- доработка готового решения.

Критерии выбора методологии: схема

Обобщённо все критерии и их показатели, влияющие на выбор методологии для управления разработкой, можно представить так:



Критерий	Показатель	Waterfall	Agile	Гибридная методология	Scrum	Kanban	Спиральная методология
ИТ-проект	***Бюджет проекта***	Крупный	Мелкий/ Средний	Крупный/ Средний	Мелкий/ Средний	Мелкий/ Средний	Крупный/ Средний
	Сроки проекта	Определены	Не определены/ Частично определены	Определены	Частично определены	Не определены	Не определены
	Размер и сложность	Легкий/ Средний	Легкий-Сложный	Легкий-Сложный	Легкий-Сложный	Легкий	Легкий-Сложный
	Риски	Частично контролируемые риски/ Контролируемые риски	Неконтролируемые риски/ Частично контролируемые риски	Частично контролируемые риски/ Контролируемые риски	Частично контролируемые риски/ Контролируемые риски	Неконтролируемые риски/ Частично контролируемые риски/ Контролируемые риски	Неконтролируемые риски/ Частично контролируемые риски/ Контролируемые риски
Проектная команда	***Состав проектной команды***	10-100 чел./ 100 чел. и выше	5-10 чел.	10-100 чел./ 100 чел. и выше	5-10 чел.	5-10 чел.	10-100 чел./ 100 чел. и выше
	Роли в проектной команде	Роли четко определены	Роли четко определены	Роли четко определены	Роли четко определены	Нет деления по ролям	Роли четко определены
	Местоположение проектной команды	Не влияет	Небольшая разница в часовом поясе	Не влияет	Одна локация	Одна локация	Не влияет
	Возможность самоорганизации	Отсутствует	Частично присутствует	Отсутствует	Присутствует	Присутствует	Отсутствует
Организация	***Уровень зрелости организации***	Управляемый/ Измеряемый/ Оптимизируемый	Управляемый/ Измеряемый/ Оптимизируемый	Управляемый/ Измеряемый/ Оптимизируемый	Управляемый/ Измеряемый/ Оптимизируемый	Отсутствующий/ Начальный/ Управляемый	Управляемый/ Измеряемый/ Оптимизируемый
	Размер организации	Средний/ Крупный	Стартап/ Средний	Средний/ Крупный	Стартап/ Средний	Стартап/ Средний	Средний/ Крупный
Заинтересованные стороны	***Вовлечение заинтересованных сторон***	В начале и в конце проекта	Постоянное	Периодическое	Постоянное	Постоянное	Постоянное
	Ожидание результата	Результат известен	Конечный результат не известен	Конечный	Конечный результат не известен	Конечный результат	Конечный результат не известен

	заинтересованных сторон			результат частично известен		частично известен	
	Документация	Подробная	Минимальная	Достаточная	Минимальная	Минимальная	Достаточная
Продукт	***Внедряемый продукт***	Внедрение готового решения/ Доработка готового решения	Разработка с нуля/ внедрение готового решения/ Доработка готового решения	Внедрение готового решения/ Доработка готового решения	Разработка с нуля/внедрение готового решения/ Доработка готового решения	Разработка с нуля/ Доработка готового решения	Разработка с нуля/ внедрение готового решения/ Доработка готового решения

аблица поможет выбрать методологию управления IT-проектом, которая будет соответствовать его целям, возможностям команды и требованиям заинтересованных сторон. Всё это повысит процент успеха в реализации проектов компании.

Пересечение диапазонов по показателям даёт чёткое представление того, какая методология управления больше подходит проекту, и позволяет достичь качественного результата — то есть максимально попасть в ожидания заказчика.

Как применить Lean?

Lean (бережливое производство) — это концепция, которая помогает максимально избежать потерь. Сама по себе она не работает — её нужно встроить в уже применяемую методологию управления разработкой.

Даже традиционную Waterfall можно усовершенствовать, внедрив в неё Lean-концепции. То есть, если вы видите, что где-то возникают потери времени, ресурсов, информации, важных данных, встраивайте в эти процессы Lean-концепцию:

1. Нарисуйте VSM (карту потоков создания ценности / Value stream mapping);
2. Проанализируйте процессы;
3. Выработайте подход по исключению ненужных потерь;
4. Внедряйте новые подходы.

Кейсы выбора методологии управления разработкой

Кейс №1

Проект — автоматизация всей финансовой системы одной крупной нефтегазовой компании. В рамках поставленных задач вместе с командой нужно создать полностью автоматизированную систему управления всеми финансовыми потоками предприятия.

Проект достаточно крупный, бюджет большой, заказчик — государственная компания с определёнными внутренними правилами планирования финансирования таких проектов.

Во-первых, нужно разделить большой проект на маленькие итерации, в рамках которых будем наращивать функционал. Это непросто, но у нас уже есть чётко сформированное ТЗ от заказчика, где детально прописан весь функционал системы, и даже указано, что всё должно быть реализовано на базе программы SAP. Это серьёзно облегчает задачу — мы сразу можем увидеть, что хочет заказчик, сколько под эти задачи нужно людей, сколько это будет стоить.

Во-вторых, возникает вопрос: есть ли риски? Может ли требуемый сейчас функционал стать ненужным, пока идёт работа над проектом? После общения с заказчиком понимаем, что таких рисков нет. Следовательно, нам не придётся оценивать риски старта каждой следующей итерации, а значит сразу уходит предложение о выборе спиральной модели.

Желание заказчика сразу увидеть бюджет проекта и понять сроки его создания загоняет нас в жёсткие рамки планирования. Уйти от каскадной модели не получится — мы должны максимально точно всё спланировать. Планируем весь проект, детально изучаем ТЗ, собираем требования заказчика, вносим вместе с ним корректировки на начальном этапе в уже имеющееся ТЗ. Начинаем считать потребность в ресурсах (человеческих, технических, материальных, временных).

После всех подсчётов понимаем, что сможем это сделать за 100 млн. долларов в срок 2 года с командой 120 человек — применяем каскадную модель, всё детализируем и планируем, прописываем все технические особенности проекта сразу.

На следующем этапе мы понимаем, что идти только по каскадной модели глупо — IT-продукт огромный, нужно делить его на этапы (итерации), при этом оставляя возможность тестирования функционала каждого компонента продукта. Безусловно, нужна V-образная модель.

Затем, разделив наш крупный проект на малые этапы, чётко описав набор требований, бюджет и сроки каждого этапа, наладим работу команды разработчиков. У нас 120 человек, разделённых на несколько групп, каждая из которых занимается своими задачами. Чтобы организовать их работу и упорядочить поток незавершённых задач, принимаем решение выстроить их работу по Kanban ****— ****нарисовать Kanban-доску, записать весь перечень задач, ограничить количество задач, одновременно находящихся в работе, и задать ритм работы с помощью регулярных собраний. При этом мы взяли не все 7 каденций Kanban, а только нужные нам для разработки данного продукта:

- Kanban-митинги (каждый день по утрам на 15 минут) — уточняем и обсуждаем текущие моменты выполнения задач;
- Собрание по пополнению очереди (каждую неделю на полчаса) — обсуждаем, какие возьмём задачи в разработку, и кого на них назначим исполнителем;

- Собрание планирования поставки (по мере готовности компонента к интеграции в системе заказчика) — обсуждаем, когда будет выпуск компонентов нашего большого IT-продукта;
- Ревью сервиса поставки (раз в две недели на 30 минут) — обсуждаем с заказчиком, насколько он доволен скоростью разработки, возникли ли новые требования к продукту по мере того, как заказчик постепенно начинает работать с уже готовыми модулями разработки.

Конечно, интересно было бы применить Scrum, но тогда потребуется вовлекать владельца продукт со стороны заказчика, а заказчик не хочет погружаться в проблемы разработки и поставил условия, что будет смотреть только готовый компонент разработки — детали ему не интересны и не должны занимать его время.

Таким образом, мы выстроили разработку нашего продукта сразу по двум методологиям:

1. V-образная модель;
2. Kanban-метод.

Кейс №2

Проект — разработка модуля автоматизации системы учёта бензина и газа на автозаправочных станциях одной из крупнейших компаний России.

Компания крупная, с определённым процентом государственной доли в капитале, а значит есть нормативные акты, регулирующие процедуру закупки услуги IT-разработчиков. Как и в предыдущем кейсе, заказчик выставил требования по первоначальному определению бюджета и сроков реализации проекта на старте, а значит снова придётся использовать каскадную методологию для управления проектами.

Но, собрав требования заказчика, мы узнаём, что проект планируется реализовать в другой стране, и заказчик принимает решение провести закупку услуг через своё дочернее предприятия в форме собственности ООО (общество с ограниченной ответственностью). В этом предприятии уже нет нормативных актов, требующих жёсткого определения бюджета и сроков на старте. Можем выдохнуть и немного уйти от идеи использовать каскадную модель.

Также заказчик решил выставить условия: сначала нужно разработать модуль автоматизированной системы учёта бензина и спустя год сделать такой же модуль для учёта газа на автозаправочных станциях. При этом заказчик пояснил, что рисков отсутствия потребности в новом модуле нет. Спиральная модель отметается.

В ходе обсуждения выясняем, что есть масса пожеланий от заказчика, но нет чётко сформированного ТЗ. Заказчик видит, что нужно сделать, но довольно плохо представляет, как эти модули должны работать и взаимодействовать с уже существующими IT-системами на нефте- и газоперерабатывающих заводах.

Учитывая все обстоятельства, принимаем решение применить гибкий подход. Также, учитывая большую вовлечённость заказчика и его желание участвовать в проекте, выбираем фреймворк Scrum. Собираем команду и определяем роли: Product Owner, Scrum-мастер, команда разработчиков.

Следующий шаг — описать бэклог продукта. Этим занимается Product Owner. Когда написан бэклог, понимаем, что получилось большое количество задач, делим проект на спринты и начинаем разработку.

В процессе разработки видим, что сильно переоценили возможности команды разработчиков, понимаем, что нужно расширить штат или сократить одновременно выполняемую работу. Для расширения штата бюджета у заказчика нет. Значит нужно понять, где мы допускаем ошибки.

Чтобы изучить проблему детально, применяем Lean-концепцию и рисуем карту потока создания ценности. Видим, что несколько разработчиков работают над разными задачами, что сильно ухудшает качество выдаваемого кода — нужно срочно что-то делать. У нас есть ещё одна особенность — команда довольно активная и все участники готовы друг другу помогать. Это значит, что мы может применить подходы Kanban-метода, то есть теперь мы будем не просто рисовать доску с задачами и нужным количеством колонок, но ещё и ограничивать количество задач, находящихся одновременно в работе.

Применяя Kanban-метод, мы устраним те потери при переключении специалистов на разные задачи, которые мы установили с помощью Lean-концепции, и продолжим разработку по фреймворку Scrum, применив подходы Kanban-метода.

Используемые источники

1. Эндрю Стеллман, Дженнифер Грин «Постигая Agile. Ценности, принципы, методологии»
2. Стивен Деннинг «Эпоха Agile»
3. Торосян Е. К., Тюлькина А. С. «Критерии выбора методологии управления ИТ-проектами», Петербургский экономический журнал, 2020
4. В чём секрет популярности и эффективности методологии Agile
5. Как у российских компаний обстоят дела с внедрением Agile: масштабное исследование 2020 года
6. Как выбрать наилучшую методологию управления проектами
7. Блок-схема выбора оптимальной методологии разработки ПО