1. Что это? О\_о

Гибкая методология разработки (англ. Agile software development, agile-методы) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, DSDM, Scrum, FDD.

2. Зачем это нужно?

Задач всегда больше, чем ресурсов. Всем заказчикам нужно сделать задачи как можно раньше, все поднимают приоритеты своих задач. У разработчиков уходило слишком много времени на то, чтобы разобраться, какие из этих неотложных задач самые неотложные. Это очень тормозило развитие, и нужно было что-то предпринимать. Сделать так, чтобы каждый разработчик знал, какими именно задачами ему заниматься сегодня, а какие можно отложить на завтра, следующую неделю, месяц.

3. Какие проблемы решает?

Если же для убеждения вашего руководства вам мало этих основных причин, то среди набора возможных выгод, вы можете использовать следующие фразы:

* увеличение продуктивности
* улучшение качества
* наглядность ситуации в проекте
* уменьшение рисков
* упрощение процессов
* уменьшение стоимости проектов
* лучшая поддерживаемость проектов в дальнейшем
* улучшение морали команд(ы)
* налаживание инженерных дисциплин
* и, даже, организация работы распределенных команд

4. Описание и особенности.

Agile — семейство процессов разработки, а не единственный подход в разработке программного обеспечения, и определяется Agile Manifesto. Agile не включает практик, а определяет ценности и принципы, которыми руководствуются успешные команды.

Agile Manifesto разработан и принят 11-13 февраля 2001 года на лыжном курорте The Lodge at Snowbird в горах Юты. Agile Manifesto содержит 4 основные идеи и 12 принципов. Примечательно, что Agile Manifesto не содержит практических советов.

Основные идеи:

* люди и взаимодействие важнее процессов и инструментов;
* работающий продукт важнее исчерпывающей документации;
* сотрудничество с заказчиком важнее согласования условий контракта;
* готовность к изменениям важнее следования первоначальному плану.

Принципы, которые разъясняет Agile Manifesto:

* удовлетворение клиента за счёт ранней и бесперебойной поставки ценного программного обеспечения;
* приветствие изменений требований даже в конце разработки (это может повысить конкурентоспособность полученного продукта);
* частая поставка рабочего программного обеспечения (каждый месяц или неделю или ещё чаще);
* тесное, ежедневное общение заказчика с разработчиками на протяжении всего проекта;
* проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
* рекомендуемый метод передачи информации — личный разговор (лицом к лицу);
* работающее программное обеспечение — лучший измеритель прогресса;
* спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределённый срок;
* постоянное внимание улучшению технического мастерства и удобному дизайну;
* простота — искусство не делать лишней работы;
* лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
* постоянная адаптация к изменяющимся обстоятельствам.

5. Сравнительный анализ. Преимущества и недостатки.

Agile плохо совместим с окружающей проект средой (компанией заказчика)

Если компания давно уже как-то работает, менять сложившийся процесс бывает сложно а зачастую и невозможно (например, если по закону предприятие обязано объявить конкурс и зафиксировать требования и деньги, то...). Фиксированные требования и цена — явные признаки V-модели, и при работе, например, с российскими госкомпаниями у вас никакого чистого Agile не получится.

Просто потому, что если действовать, будто «готовность к изменениям важнее следования первоначальному плану» и «сотрудничество с заказчиком важнее согласования условий контракта», в конце проекта вам выкатят неустойку за недопоставленный функционал.

Иногда все не так жестко, и приходится бодаться не с внешними силами, а с какими-то внутренними ограничениями, типа требования следовать шаблонам при написании спецификаций (несовместимые с концепцией user story) или выдавать в определенном виде ежемесячный отчет о ходе проекта. При таких порядках декларация «работающий продукт важнее исчерпывающей документации» может вызвать недоумение у верхнего менеджмента.

С другой стороны, совсем вот брать и выкидывать старые правила, потому что они «не Agile», тоже не стоит. Есть всяческие полезные, хоть и занудные и бумагозатратные процедуры типа передачи кода в поддержку, или какой-нибудь внутренний или внешний аудит по безопасности продукта…

в мире есть много задач, в которых никакой Agile не будет работать.  
  
Самым ярким, конечно же, будет пример написания софта, управляющего космическими аппаратами — ну вот сложно раз две недели демонстрировать заказчикам посадку зонда на комету, получая в ответ замечания, что именно с точки зрения ученых-физиков стоило бы сделать по-другому.  
  
Еще один яркий пример — какой-нибудь телеком. Когда вы пишете прошивку для телефона, вам совершенно не нужны демо имплементации каждого следующего класса сообщений GSM протокола потенциальным пользователям нового телефона.  
  
Еще хороший пример неприменимости конкретно Scrum — всяческие команды поддержки, когда внезапно прилетает тикет от юзера с нулевым приоритетом, и все бросают всё и чинят. Никакой предсказуемости и ритмичности уже не будет.  
  
С другой стороны, даже если мы разрабатываем космические аппараты, мобильные телефоны или программируем бытовую технику, найдутся задачи, которые можно и нужно делать Agile, показывая промежуточные результаты потенциальным пользователям.  
  
Итак, мы начали делать в режиме Agile проект, который не совместим с идеей:

* Прекратить пытаться делать Agile и начать строить более подходящий процесс. При этом совершенно необязательно отказываться от каких-то элементов процесса, которые ассоциируются с Agile, но работают везде.
* V-модель с ее, на первый взгляд монструозной трассировкой требований до кода и от кода до тесткейса, работает как надо в случаях, для которых предназначена. Если что-то не подходит для творящего под iPhone стартапа, это не значит, что это плохо.
* Может иметь смысл почитать про концепции ITIL, им уже 20 лет, и многие работают именно в них. В некоторых случаях Service Desk можно скрестить с Kanban, и получить на самом деле работающий true Agile-процесс.
* Бывают еще всяческие индустриальные практики и собственные разработки больших компаний, которые могут подойти, особенно если вы решаете задачу, похожую на то, для чего они созданы.

6. Agile-дзен

На самом деле, каждый проект несчастен по-своему, но, если копнуть глубже, все можно свести к трем основным причинам:

* Это неправильная команда, и она делает неправильный Agile.
* Agile плохо совместим с окружающей проект средой (т. е. компанией).
* Этот проект по природе не совместим с Agile.

Если есть ощущение «неправильного Agile»:

* Начать с высокоуровневого целеполагания — убедиться, что и мы, и пользователи понимаем однозначно критерии успеха проекта (и нет, это не в срок поставить весь скоуп по оговоренной цене. Это скорее «пользователям очень надо, чтобы решилась вот эта проблема, для чего мы хотим сделать возможными вот эти use case»). Очень полезны также будут хотя бы базовые знания разработчиков в доменной области и их представление о пользователях.
* Разобрать с командой практики и технологии процесса. Показать, как ежедневное честное обновление статуса помогает в достижении глобальных целей проекта.
* Убедиться на всякий случай, что мы не забыли такие ключевые вещи, как acceptance criteria, early demos, ретроспективы и обратная связь.