Эволюция vs Революция: как мы делаем mos.ru

Роман ИВЛИЕВ

О чём рассказ?

- О перехвате дел.
- Об инженерах и про инженеров.
- О командах и их организации.
- О процессах.
- О прозрачности.
- Об ошибках и способах их исключить.

Немного про проект mos.ru

- Самый главный портал города.
- Сотни функциональных заказчиков.
- Десятки (пока) интеграций с отраслевыми системами и сервисами.
- Разработка полного цикла.
- Лучший портал страны в своём сегменте, а Москва лучшая в мире среди цифровых городов (по рейтингу ООН).

Немного про проект mos.ru

- Огромный поток разных задач.
- Пиксель-пёрфект требования.
- Всё, что сделано попадает на поддержку навечно.
- За ошибки коллег отвечаем мы.

- Сложносочинённые сервисы с десятками исполнителей.
- Инжерерно мы достаточно технологичны: микросервисная архитектура, современные паттерны, умерено высокие нагрузки.

С чего всё началось?

История проекта - 2 года назад

- Разработка - аутсорсинг.
- DevOps - аутсорсинг.
- Всё, что только можно, кроме идей - аутсорсинг.
- 4 инженера, из них 3 в QA.

История проекта

- В моменте стало понятно, что:
  - Скорость реакции на изменения не очень устраивает.
  - КПД полезных задач не очень устраивает.
  - Удобство взаимодействия с исполнителем не очень устраивает.

История проекта

- В моменте стало понятно, что:
  - Скорость реакции на изменения не очень устраивает.
  - КПД полезных задач не очень устраивает.
    - Удобство взаимодействия с исполнителем не очень устраивает.
- Частая история для продуктов с долгим жизненным циклом.
- Рано или поздно к этому приходят все.

Как мы забирали проект in-house

- Что мы оставили “на заказ”:

☑ Непрофильную для наших инженеров разработку (например, Битрикс).

☑ Много однотипных операций, например, типовые страницы органов власти.

☑ То, где не хватало экспертизы здесь и сейчас - DevOps. И не было возможности получить её быстро.

Как мы забирали проект

- Что мы начали забирать практически сразу:
  - ☑ Всё новое.
  - ☑ То, что нужно менять быстро.
  - ☑ То, на что размытые требования.
  - ☑ То, на что сроки уже сорваны.

Что мы взяли на вооружение сразу

- В первую очередь мы:

☑ Абсолютно всех исполнителей пересадили на наш таск-трекер и баг-трекер.

☑ Привели в единообразие работу с хранилищем кода.

В итоге:

☑ Jira, Confluence, Git (gitlab и bitbucket).

Что мы не стали сразу стандартизировать:

- ☑ Методику сбора и описания требований.
- ☑ Среды разработки.
- ☑ CI/CD - сейчас у нас их две и это не сильно мешает жить.

Инженеры, какие они в Мос.ру сейчас?

Инженеры, какие они?

- Full-stack инженеров у нас нет. Мы сразу отказались от этой идеи потому, что:
  - Уровень сложности решаемых задач обычно не очень высокий.
  - На самом деле уклон всё-таки всегда есть.
  - Профессионалов реально очень мало, и они стоят кучу денег.

Инженеры, какие они?

- Специализированные.
  - Front-end и Back-end.
  - Обилие технологий и фреймворков диктует направленность и ограничивает возможность участия, иногда сложно искать.
  - Но за это - они профессионалы своего дела.
  - Правда часто зашорены на одной технологии, но мы это лечим:).
  - Как следствие, парни могут решать почти любую задачу одинаково качественно.

Команды, какие они в Мос.ру?

Команды, какие они?

- Команды разработки:
  - Смешанный состав: back+front.
  - Команды платформы: back only, front only.
- Три службы:
  - QA as a Service.
  - DevOps as a Service.
  - Operation as a Service.

Команды, какие они?

- Почему мы к этому пришли:
  - Смешанный состав - команда может решать любые задачи разработки и поддержки.
- Из плюсов:
  - Можем никого не ждать.
- Из минусов:
  - Органичная пропускная способность по специализации.
  - Не всегда работа идёт синхронно, например, в проекте больше back-end разработки.

Команды, какие они?

- Как мы боремся с минусами:
  - Следим и работаем с техническим долгом.
  - Свободное время идёт на помощь командам платформы.
  - Много работаем с менеджерами над планированием и приоритезацией.

Команды, какие они?

- Почему мы к этому пришли:
  - Команды платформы - полноценно развивать сервис и делать продукт не получилось ввиду сложности бизнес-модели.
- Из плюсов:
  - Автономность и узкая направленность.
- Из минусов:
  - Органичная пропускная способность.
  - Не всегда работа идёт синхронно, т.е. в проект нужен компонент, а его нет.

Команды, какие они?

- Как мы боремся с минусами:
  - Много времени уделяем работе с дизайнерами и продукт-менеджерами.
  - Ретроспективно изучаем прошлые проекты, выявляем общие компоненты. Мы храним историю всех ретроспектив и пересматриваем их.
  - Сохраняем и работаем с техническим долгом.
  - Стандарты и договорённости на разработку позволяет подключать “свободных” агентов.

Команды, какие они?

- Почему мы к этому пришли:
  - Три службы - наш ответ неравномерности нагрузки.
  - Из плюсов:
    - В рамках служб мы можем балансировать нагрузку более аккуратно.
    - Более быстро идёт накопление и обмен опытом.
- Из минусов:
  - Сложность планирования и администрирования.

Производственный цикл - цикл Деминга

Влияние производственного цикла на команды

- ☑ Всё, что мы делали на всех этапах жизни команды и продолжаем делать - подстраивались под общий производственный цикл продукта.

Влияние производственного цикла на команды

- ☑ Мы очень внимательно смотрели на то, как устроен цикл разработки наших продуктов.
- ☑ И устраивали команды подходящим образом.
- ☑ Сейчас у нас столько уникальных процессов в командах, сколько команд и продуктов, которые они делают.
- ☑ Но ядро у процессов единое!

Влияние производственного цикла на команды

- ☑ Мы очень внимательно смотрели на состав и квалификацию инженеров.
- ☑ И подбирали их подходящим образом.
- ☑ Сейчас у нас сбалансированная команда из лидов, синьоров, мидлов и джунов. Это позволяет заниматься внутренним обучением.

Влияние производственного цикла на команды

- ☑ Всё, что мы делали на всех этапах жизни команды - подстраивались под производственный цикл.
- ☑ Цикл производства для нас первичен.

Влияние производственного цикла на команды

- ☑ Вначале, когда нас было мало - мы не делали команды, мы собирали команды под проекты.
- ☑ Когда нас стало больше - мы сделали фиксированные проектные команды.
- ☑ Когда прошло время - мы привязали команды к продуктам.
- ☑ Когда выросла нагрузка мы запустили “Карусель”.

Форматы организации команд

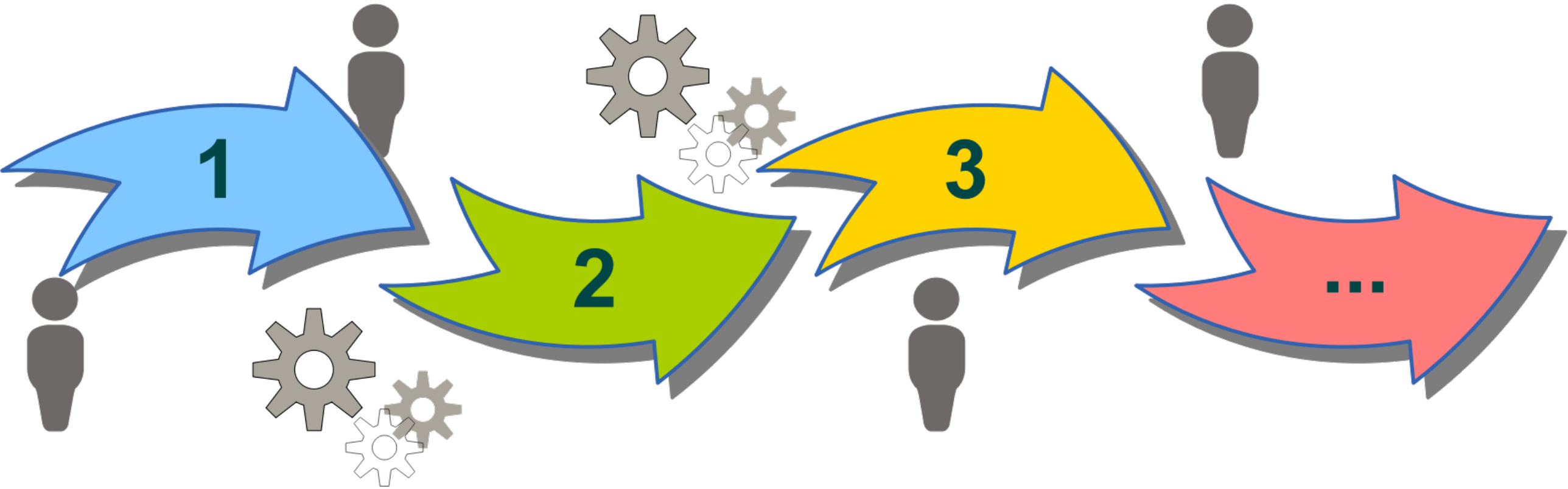
В чём суть карусели?

- В зависимости от нагрузки мы:
  - Переводим инженеров из команды в команду.
  - Жертвуем командами платформы.
  - Разделяем задачу на две и более команды.
- Бывает так, что инженер работает на нескольких проектах одновременно.

В чём суть карусели?

- Почему у нас это работает:
  - Мы изначально шли к этому.
  - В рамках гильдий (так мы называем бэков и фронтов) стандартизировали основную деятельность.
  - Унифицировали стек, фреймворки, библиотеки.
  - Очень много работаем с тимлидами. Они - основа всей системы разработки.

Процессы и методологии



Когда процесс может навредить

- Если исходить из навязывания процессов по схеме - мне так удобно/я так привык.
- Если процесс строится для того, чтобы был процесс.
- Если продуктовый процесс создания продукта отличается от технологического процесса создания продукта.

Когда методология может навредить

- Всё про процессы с предыдущего слайда уместно и здесь.
- Если без строгого следования одной методологии нет движения вперед.

Что мы делаем с процессами и методологиями

- ☑ Мы с самого начала не стали внедрять никаких строгих процессов (ключевое слово - “строгих”).
- ☑ Гордо заявили, что у нас Скрам (правда не весь). Надо было с чего-то начинать.
- ☑ Начали накапливать опыт и постоянно проводить ретроспективы и адаптировать процесс.
- ☑ У нас нет описанного процесса на 100 листов, но это не означает, что у нас хаос.

Что мы делаем с процессами и методологиями

- В итоге:

☑ Скрам не прижился, потому что у нас проект с поддержкой без выделенной команды на это. Его место занял “kanban-mos.ru-edition".

☑ В условиях роста команды и появления новых задач - это позволило гибко и быстро подстраиваться под производственный цикл.

☑ Основные процессы практически утряслись только сейчас (1,5 года с начала процессной кампании!!)

Когда мы подключаем технарей и зачем?

Когда мы подключаем технарей и зачем?

- Представим, что будет,
  - Если забыть про то, что архитектура продукта не может быть изменена мгновенно?
  - Если забыть, что даже самые крутые инженеры всё-таки люди, и они могут ошибаться?
  - Если забыть, что один маленький “бантик” может породить кучу проблем?

Когда мы подключаем технарей и зачем?

- Не ранее, чем готова концепция или видение продукта (но и здесь будет полезно).
- На этапе построения дорожной карты проекта.
- На этапе планирования изменений или подготовки следующих версий.
- По сути, консультации полезны на любом этапе, вплоть до вывода в широкие массы.
- У нас есть kick-off мероприятия и post-mortem, и везде есть технари.

Технический долг

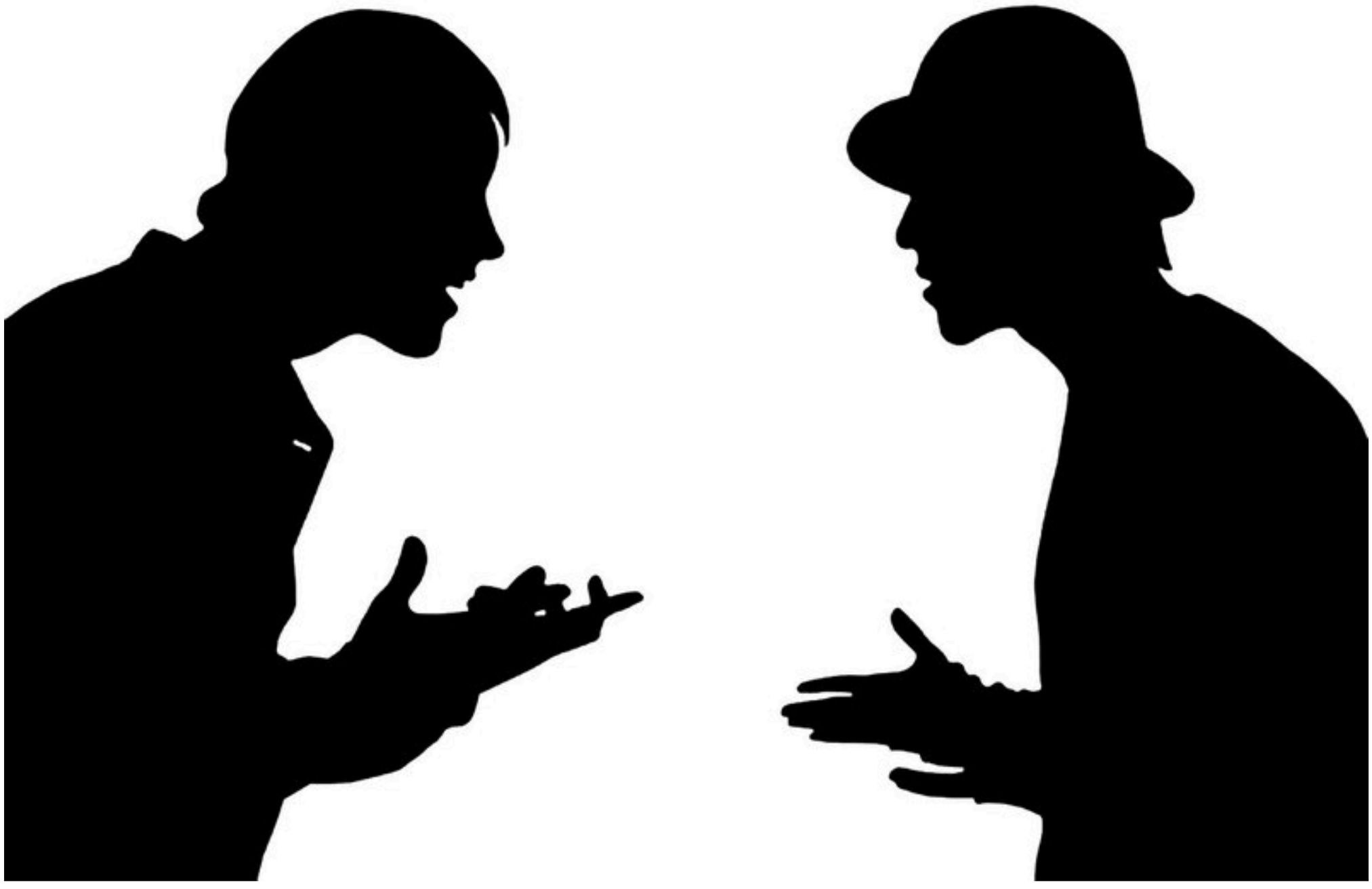
Технический долг

- Это - кредит!!! Придётся платить проценты.
- Время не на вашей стороне.
- Технологии и бизнес не на вашей стороне.
- “Технический дефолт” в худшем случае:
  - Быстрая разработка или внесение изменений будут невозможны!
  - Устранение технического долга будет невозможно!

Технический долг - наш подход:

- ☑ 1 “костыль” - 1 задача в бэклог.
- ☑ Долг - полноправный участник планирования.
- ☑ Долг - не уступает по приоритету (не весь, но некоторый).
- ☑ Парко-хозяйственный день - день, когда все смотрят на свои задачи в бэклоге.

Обо что мы переломали копья с менеджментом



Основные паттерны управления

- **Нет стандарта на разработку систем:**
  - “Сделай так, как у них”.
  - “Но они же как то сделали?”.
  - “Почему у них это работает?”
  - “Только у нас всё через }|{ОПУ?”

Основные паттерны управления

- **Не всё нужно здесь и сейчас:**
  - Перфекционизм - опасная штука в инженерии.
  - Стремление покорить мир - опасная штука в продуктовом менеджменте.
  - Сложная система всегда получается из простой системы.
  - Сразу сложная система не заработает.

Основные паттерны управления

- **Иногда самое простое решение - самое верное:**
  - Стоит привыкнуть к термину “костыль”.
  - “Костыль”:
  - Это не всегда плохо.
  - Не всегда технологично.
  - Но дешево и быстро.
  - Но теперь есть долг, и главное не забыть про него.

Основные паттерны управления

- **Использование опыта других не всегда оправдано:**
- Помните, да?
  - “Сделай так, как у них”.
  - “Но они же как то сделали?”.
  - “Почему у них это работает?”
  - “Только у нас всё через }|{опу?”

Основные паттерны управления

- А на самом деле:
  - Разные задачи, пусть и похожие.
  - Разная инфраструктура, пусть и похожая.
  - Разная квалификация, вот тут без пусть.
  - Разные возможности компании.

☑ Разработчики - лентяи, каких поискать, они при первой же возможности тырят чужие решения.

☑ Но не всегда решения, например, ~~Badoo~~ подойдут вашему продукту.

☑ Мы экспериментируем лидами и командами.

Основные паттерны управления

- **Нуре - мы его опасаемся, но не боимся:**
  - Это интересно и полезно.
  - Это любят инженеры.
  - Обязательно надо следить за ним.
- НО
  - Он может внезапно прирасти к системе.
  - Он дольше в производстве.
  - Он может перестать быть хайпом:)

КОНТАКТЫ

roman@ivliev.info

roman.ivliyev

http://ivliev.info