02
Требования к интернет-системам

«ПРОЕКТИРОВАНИЕ ИНТЕРНЕТ-СИСТЕМ»

ПАВЕЛ КОЧУРКО, К.Т.Н., ДОЦЕНТ
КАФЕДРА ИИТ
EPLANE.COM

EPLANE
The Leading Aerospace Marketplace

БрГТУ

Developers are most motivated to improve code that affects operations when they **feel the pain of operations**, too. Operations must understand the development process if they are to be able to constructively collaborate.

—T.A.Limoncelli et al., *The Practice of Cloud System Administration: DevOps and SRE Practices for Web Services*

Поговорим о целях

Цель архитектуры программного обеспечения – уменьшить человеческие трудозатраты на создание и сопровождение системы

— Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения

Бизнес-цели

Идеальная система полностью решает цели и задачи бизнеса.

Бизнес цели: планируемы, измеримы, автоматизируемы

- Например:
  Продажа x товаров на сайте в месяц
  Предоставление сервиса 99,99% времени
  Обработка x миллионов заказов в месяц, с ростом 10%
  Вывод новых фич два раза в неделю
  Исправление мажорных багов за 24 часа

Предсказуемость и устойчивость целей бизнеса и проектной команды

PLANE
The Leading Aerospace Marketplace

БрГТУ

Поведение v Архитектура

- Бизнес-цели максимально быстро с ущербом для качества
  Мы сможем навести порядок потом, нам бы только выйти на рынок!
  - надо пилить новые фичи;
  - нарастает технический долг;
  - беспорядок возрастает;
  - вместо исправления беспорядок переносится с места на место.
- Грязный код может помочь быстро выйти на рынок, но в действительности он затормозит движение в долгосрочной перспективе

Поведение v Архитектура

- **Поведение** – удовлетворение бизнес-требованиям по функциональности
- **Архитектура** – удовлетворение требованиям качества (сопровождаемость, изменяемость, надежность, и т.д.)

– так что всё-таки важнее?

Правильный ответ:

- Если функциональные требования поменялись, а изменения вносить невозможно – правильно работавшая ранее программа становится бесполезна
- Если программа работает неправильно, но легко поддается изменениям, то её можно сделать правильной

Архитектура.

Что лучше для бизнеса?

Заинтересованные стороны:

- Топ-менеджмент – ???
- Маркетинг – за скорость в ущерб качеству
- Продажи – за скорость в ущерб качеству
- Разработчики – за правильную архитектуру и код
- Эксплуатация – за правильную архитектуру и код

Идеальная архитектура

- Очевидный запланированный путь развития с ростом популярности и трафика
  реакция на изменяющиеся требования
- Устойчивость к сбоям, избыточность и эластичность
  сбой не сюрприз, а естественная часть физики ИТ
- Максимальная независимость подсистем
  масштабирование, апгрейд, замена
- Электронное описание инфраструктуры, Infrastructure as a Code
  автоматическое воссоздание окружений

Традиционный процесс релиза

Разработчик размещает готовый код в репозиторий
Тестовые инженеры прогоняют код через тесты
Если тесты пройдены, выпускающий инженер строит пакеты для доставки ПО. Большинство файлов из репозитория, но некоторые – из сторонних источников, генерируемы и т.д.
Создается тестовое окружение
Тестовые инженеры тестируют взаимодействие подсистем
Если тесты пройдены – код в продакшн
Системные администраторы обновляют код систем, следя за сбоями
При наличии сбоев – откат до предыдущей версии

Ошибки на любом этапе - возврат на предыдущий этап

Много изменений разрабатывается и тестируется одновременно

Пакетные мега-релизы несколько раз в год

Чем больше одновременных изменений – тем **больше рис**

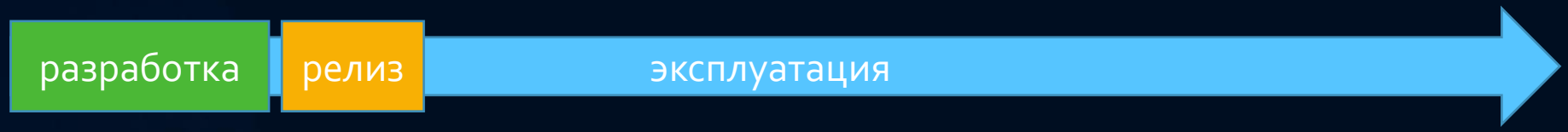
Релиз в идеальной системе

- Сборка, тестирование, выпуск, доставка автоматизированы
- Вместо мега-релизов – микро-релизы, хоть 100 раз в день

Как только разработчик размещает код в репозитории, система запускает серию автоматических тестов основного функционала
Если тесты пройдены, автоматически собираются пакеты
Запускается создание тестового окружения без человеческого вмешательства (IaaC)
Запускается серия автотестов на тестовом окружении
При успешном прохождении пакеты выкатываются с продакшн
Тестовые окружения строятся так же, как и продакшн, минимум различий

- При обнаружении ошибки на любом из этапов – остановка других релизов до исправления ошибки
- За счет минимального объема изменений – исправление ошибок быстрое, движение вперед не задерживается

Идеальная эксплуатация



- Измеримость всех показателей работы
  удовлетворение бизнес-требованиям
- Вместо ручного контроля – автоматическая реакция
  масштабирование вверх и вниз
- Обнаружение проблем до того, как они станут сбоями
  своевременное сообщение дежурным инженерам по эксплуатации
- Возможность включения/отключения отдельных частей, фич
  в т.ч. выкатка фич на ограниченное число пользователей
- Постоянно дополняемые инструкции по возможным сбоям и путям устранения
  вовлечение разработчиков в исправление эксплуатационных ошибок

Эксплуатационные требования
(иногда – нефункциональные требования)

- Автоматизация конфигурирования
- Корректный рестарт сервиса при перезагрузке или включении
- Очистка очереди перед выключением
- Обновление ПО без отключения сервиса
- Резервное копирование и восстановление, импорт данных
- Избыточность и реплицирование БД
- Переключатели отдельных фич
- Постепенная деградация при перегрузках
- Контроль доступа, мониторинг, аудит, средства отладки, сбор исключений, документирование эксплуатации,...

Дизайн для эксплуатации

Фичи, важные для эксплуатации, не берутся «ниоткуда», их необходимо реализовывать, а прежде – спроектировать.

- Встраивание в продукт с самого начала
  Это круто. Так не бывает (почти).
- Запрос разработки фич по мере идентификации их необходимости
  Часто. Необходимо определение приоритетов запросов.
- Самостоятельное написание командой эксплуатации
  Слишком часто. Разработчики могут не принять код. Плохой прецедент.
- Использование продуктов сторонних вендоров
  Необходимость адаптации процесса под продукт, а не наоборот

FPLANE
The Leading Aerospace Marketplace

БрГТУ

Продолжение следует

ЕСТЬ ВОПРОСЫ?