Andersen

DevOps 2021:

Вводные данные.

Определение DevOps.

Жизненный цикл программных продуктов.

DevOps в цикле разработки.

Чего мы ждём от участников курса:

- Участия xD
- Ответственного отношения к заданиям
- Культурной коммуникации
- Вопросов к преподавателям
- Обратной связи по качеству курса/материалов

Andersen

Что мы предлагаем:

- Обзор ключевых компетенций DevOps-специалистов
- Коварные домашние задания
- Два месяца доступа к знаниям преподавателей-экспертов
- Social networking для начинающих специалистов
- Приглашения на работу в Andersen лучшим курсантам

Andersen

Определение DevOps:

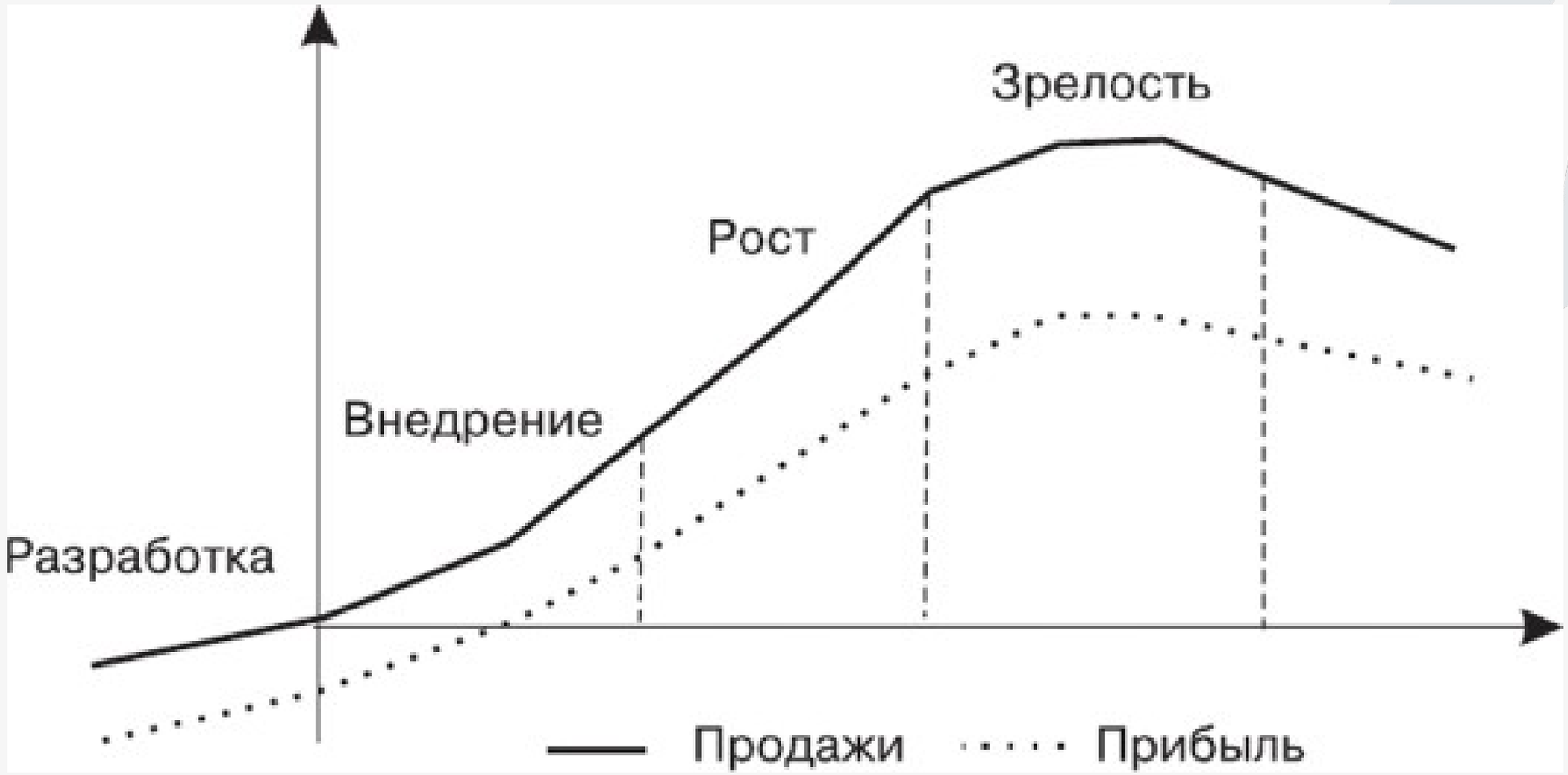
- Методология активного взаимодействия специалистов по разработке со специалистами по информационно-технологическому обслуживанию и взаимная интеграция их рабочих процессов друг в друга для обеспечения качества продукта.
- DevOps-инженер отвечает за любую автоматизацию задач, связанных с настройкой и развертыванием приложений. На его плечи ложится и мониторинг программного обеспечения. Для решения этих задач он применяет различные системы управления конфигурациями, решения виртуализации и облачные инструменты для балансировки ресурсов.

Жизненный цикл программных продуктов:

- анализ требований,
- проектирование,
- кодирование (программирование),
- тестирование и отладка,
- эксплуатация и сопровождение.

Andersen

Жизненный цикл программных продуктов:



Модели разработки:

- Waterfall
- Incremental
- Iterative
- Spiral
- AgileSpiral

Каскадная модель (Waterfall):

Преимущества Waterfall:

- Разработку просто контролировать. Заказчик всегда знает, чем сейчас заняты программисты, может управлять сроками и стоимостью.
- Стоимость проекта определяется на начальном этапе. Все шаги запланированы уже на этапе согласования договора, ПО пишется непрерывно «от и до».
- Не нужно нанимать тестировщиков с серьёзной технической подготовкой. Тестировщики смогут опираться на подробную техническую документацию.

Недостатки Waterfall:

- Тестирование начинается на последних этапах разработки. Если в требованиях к продукту была допущена ошибка, то исправить её будет стоить дорого. Тестировщики обнаружат её, когда разработчик уже написал код, а технические писатели — документацию.
- Заказчик видит готовый продукт в конце разработки и только тогда может дать обратную связь. Велика вероятность, что результат его не устроит.
- Разработчики пишут много технической документации, что задерживает работы. Чем обширнее документация у проекта, тем больше изменений нужно вносить и дольше их согласовывать.

Инкрементная модель (Incremental):

Инкрементная модель подходит для проектов, в которых точное техзадание прописано уже на старте, а продукт должен быстро выйти на рынок.

Преимущества инкрементной модели:

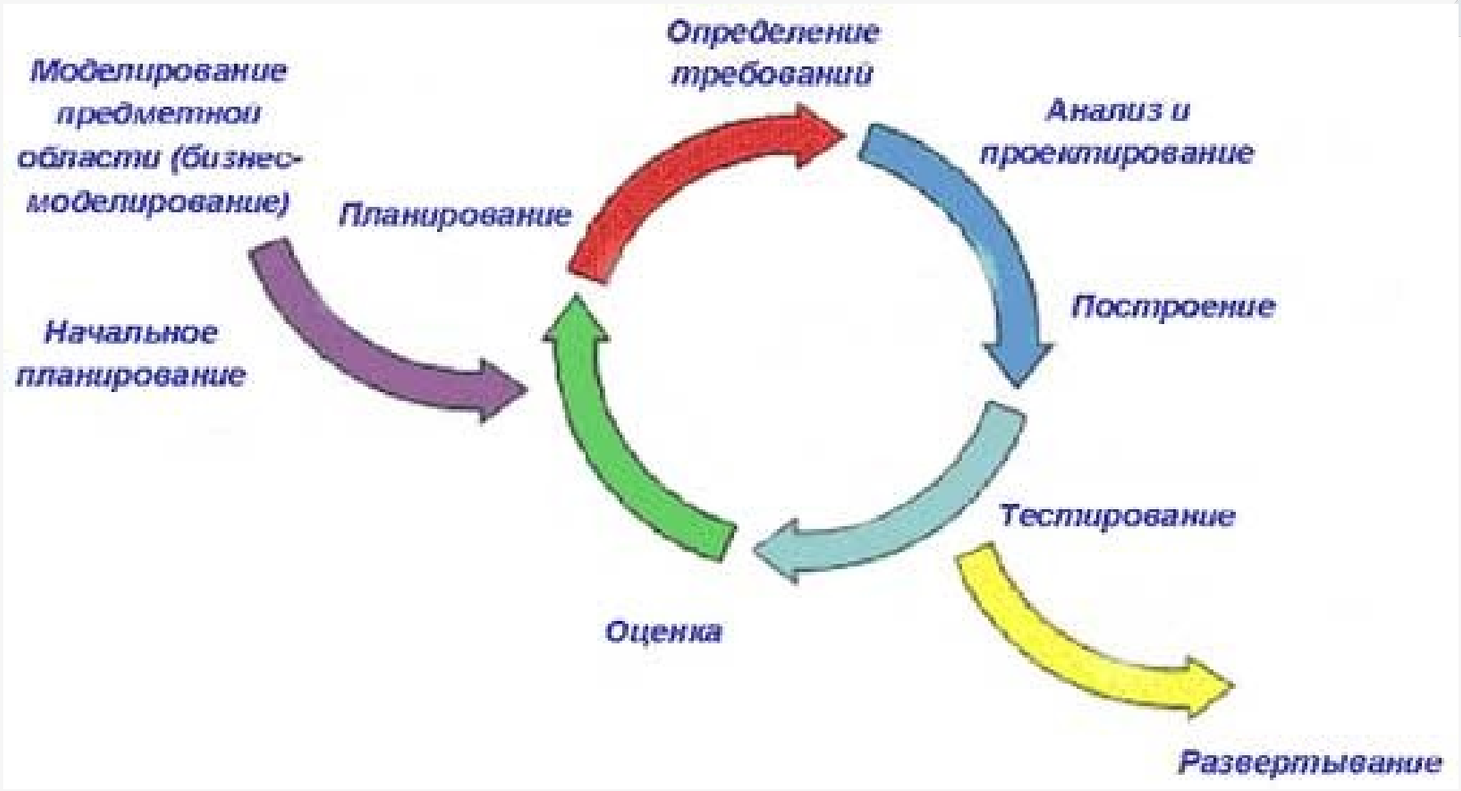
- Не нужно вкладывать много денег на начальном этапе. Заказчик оплачивает создание основных функций, получает продукт, «выкатывает» его на рынок — и по итогам обратной связи решает, продолжать ли разработку.
- Можно быстро получить фидбэк от пользователей и оперативно обновить техническое задание. Так снижается риск создать продукт, который никому не нужен.
- Ошибка обходится дешевле. Если при разработке архитектуры была допущена ошибка, то исправить её будет стоить не так дорого, как в каскадной модели.

Недостатки инкрементной модели:

- Каждая команда программистов разрабатывает свою функциональность и может реализовать интерфейс продукта по-своему. Чтобы этого не произошло, важно на этапе обсуждения техзадания объяснить, каким он будет, чтобы у всех участников проекта сложилось единое понимание.
- Разработчики будут оттягивать доработку основной функциональности и «пилить мелочёвку». Чтобы этого не случилось, менеджер проекта должен контролировать, чем занимается каждая команда.

Итеративная модель (Iterative):

Это модель, при которой заказчик не обязан понимать, какой продукт хочет получить в итоге, и может не прописывать сразу подробное техзадание.



Преимущества итеративной модели:

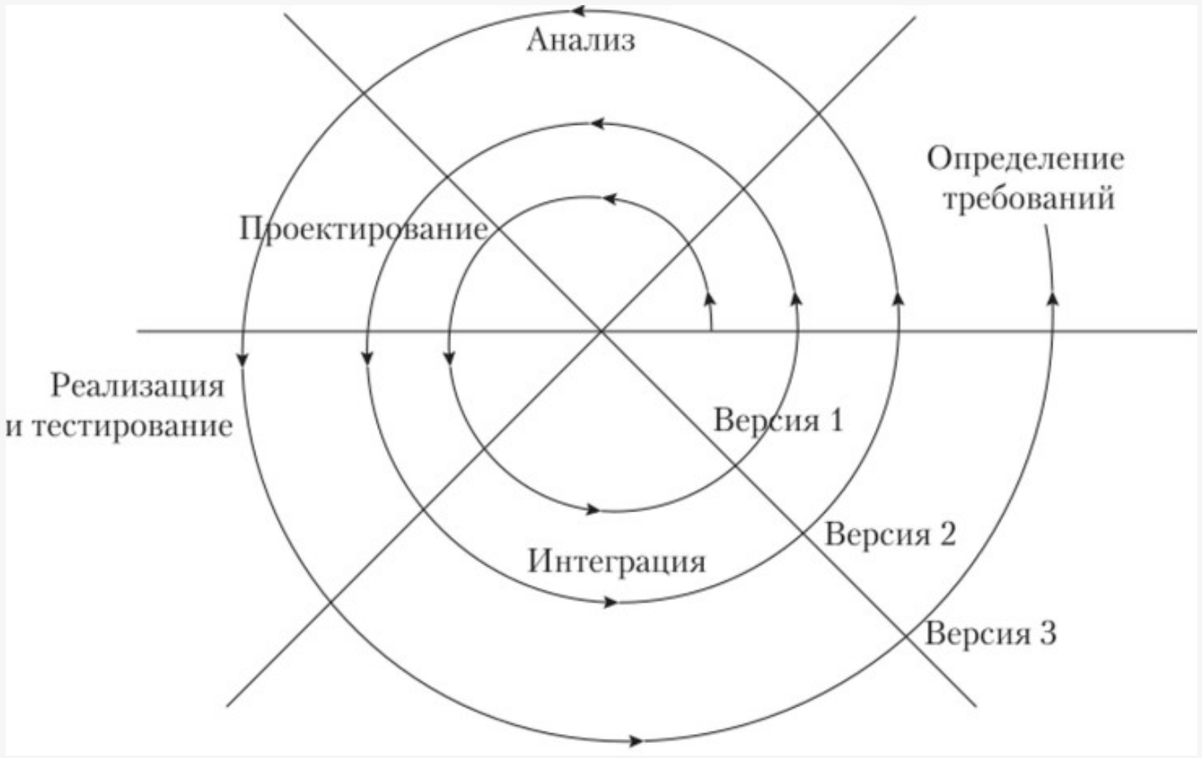
- Быстрый выпуск минимального продукта даёт возможность оперативно получать обратную связь от заказчика и пользователей. А значит, фокусироваться на наиболее важных функциях ПО и улучшать их в соответствии с требованиями рынка и пожеланиями клиента.
- Постоянное тестирование пользователями позволяет быстро обнаруживать и устранять ошибки.

Недостатки итеративной модели:

- Использование на начальном этапе баз данных или серверов — первые сложно масштабировать, а вторые не выдерживают нагрузку. Возможно, придётся переписывать большую часть приложения.
- Отсутствие фиксированного бюджета и сроков. Заказчик не знает, как выглядит конечная цель и когда закончится разработка.

Спиральная модель (Spiral):

Спиральная модель похожа на инкрементную, но здесь гораздо больше времени уделяется оценке рисков.



Преимущества спиральной модели:

- Большое внимание уделяется проработке рисков.

Недостатки спиральной модели:

- Есть риск застрять на начальном этапе — бесконечно совершенствовать первую версию продукта и не продвинуться к следующим.
- Разработка длится долго и стоит дорого.

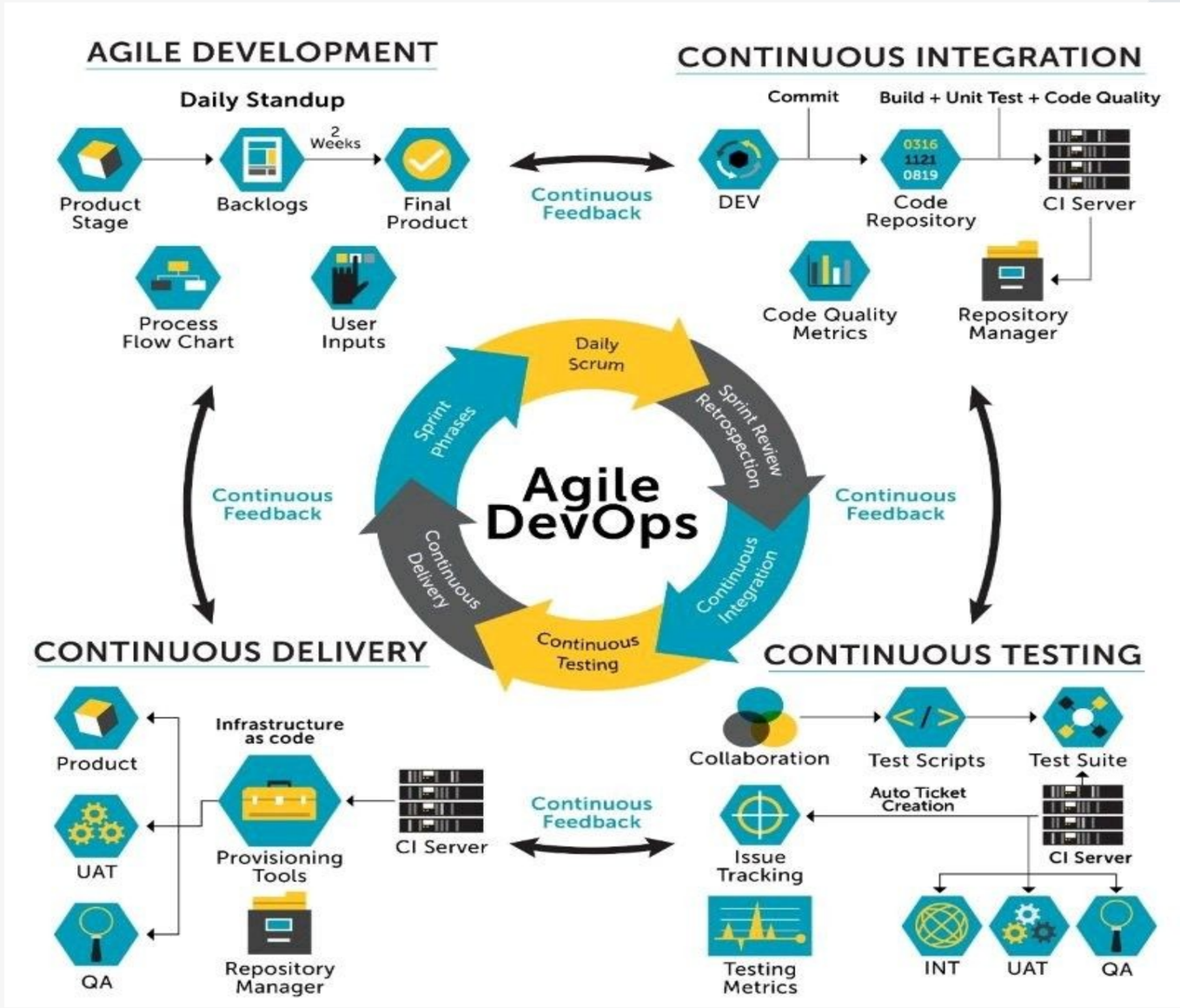
Подход к разработке Agile:

Преимущества Agile:

- Agile-управление обладает максимальной гибкостью.
- Проверка качества осуществляется по окончании каждого спринта.
- Быстрое реагирование на изменения.
- Позволяет поддерживать максимально тесную связь между. заказчиком и проектной командой.

Недостатки спиральной модели:

- Сроки сдачи готового проекта могут постоянно переноситься.
- Разработка длится долго и стоит дорого.
- Потребность в мотивированных и высококвалифицированных специалистах.


AGILE DEVELOPMENT
Daily Standup
Product Stage
Backlogs
2 Weeks
Final Product
Process Flow Chart
User Inputs
Continuous Feedback
CONTINUOUS INTEGRATION
Commit
Build + Unit Test + Code Quality
DEV
0316
1121
0819
Code Repository
CI Server
Code Quality Metrics
Repository Manager
Daily Scrum
Sprint Review Retrospection
Continuous Integration
Continuous Testing
Continuous Delivery
Sprint Phrases
Agile DevOps
Continuous Feedback
Continuous Feedback
CONTINUOUS DELIVERY
Product
UAT
QA
Infrastructure as code
Provisioning Tools
Repository Manager
CI Server
Continuous Feedback
CONTINUOUS TESTING
Collaboration
Test Scripts
Test Suite
Auto Ticket Creation
CI Server
Issue Tracking
Testing Metrics
INT
UAT
QA

Andersen

Домашнее задание:

- Завести репозиторий на github.com
  - один репозиторий
  - директория под каждое домашнее задание
  - README.md в каждой директории!
- README.md driven development:
  - создать Readme.md с markdown разметкой
  - написать коротко о себе
  - указать 3 пункта, почему вам интересен курс
  - указать 3 пункта, чего вы сами ожидаете от курса
  - commit-messages & readmes in English :-)
- Устроим блог из репозитория, ежедневное задание: TIL
  *TIL == Today I’ve Learned