Технологии проектирования ИС и ИТ

ФИО преподавателя: Смирнов Михаил Вячеславович

e-mail: smirnov.mirea@gmail.com

Лекция 2

Методологии разработки ПО ИС и ИТ

Содержание:

- Каноническое проектирование
- Стадии разработки программного средства.
- Модели канонического проектирования ПО.
- Водопадная модель, стадии водопадной модели разработки ПО.
- Модель с верификацией, инкрементная модель разработки ПО.
- Итеративная модель, спиральная модель разработки ПО.
- Гибкая модель разработки ПО, ликбез по моделям в целом.
- Разнообразие методологий разработки ПО, DevOps.

Вводные определения лекции

- Организация *канонического проектирования ИС* подразумевает полное завершение некоторого типа работ перед переходом к следующему этапу, на котором выполняется другой тип работ.

Стадии канонического проекта разработки ИС и ИТ. Формирование требований.

- обследование объекта и обоснование необходимости создания ПО;
- формирование требований пользователей к ИС;
- оформление отчета о проведенном обследовании.

Стадии канонического проекта разработки ИС и ИТ. Образец потенциальных требований участников проекта.

- Требования директора по персоналу:
  a. «Управление вакансиями на сайте сделать по аналогии с сайтом SuperJob».
  b. «Создать внутрикорпоративный портал и базу знаний».
  c. «Создать личные странички сотрудников».
  d. «Создать возможность проведения вебинаров для сотрудников через сайт».

- Требования директора по рекламе, маркетолога:
  a. «Создать удобное управление баннерной системой – как для показа своих баннеров, так и чужих».
  b. «Сделать возможность выгрузки данных о товарах и услугах».
  c. «Создать функционал проведения опросов через сайт».
  d. «Создать возможность комментирования и обсуждения товара на сайте».
  e. «Организовать кросспостинг материала в социальные сети».
  f. «Создать рейтинги товаров».

- Требования администратора сайта:
  a. «Создать возможность одновременной работы нескольких человек в административном разделе».
  b. «Создать функционал автоматического архивирования сайта».
  c. «Создать возможность разграничения прав доступа для редакторов сайта».

Стадии канонического проекта разработки ИС и ИТ. Разработка концепции.

- изучение объекта автоматизации, проведение необходимых научно-исследовательских работ;
- разработка вариантов концепции ИС и ИТ, удовлетворяющих требованиям пользователей;
- оформление отчета и утверждение концепции.

Стадии канонического проекта разработки ИС и ИТ. Ключевые положения концепции. Бизнес-требования.

В ходе сбора и анализа требований формулируются цели, для достижения которых предназначена система, и задачи, которые она должна решать для достижения поставленных целей.

Как правило, целями является решение определённых проблем, например:

- сократить штат сотрудников, задействованных в автоматизируемых процессах, до N человек;
- увеличить прибыль на N рублей в год;
- сократить длительность расчёта R до N минут;
- увеличить точность рассчитываемых результатов до N;
- обеспечить выполнение изменившихся норм законодательства <перечисление разделов законодательных актов, подлежащих учёту>.

Стадии канонического проекта разработки ИС и ИТ. Ключевые положения концепции. Цели и задачи.

Цели должны быть конкретными и проверяемыми. Достижение поставленных целей является критерием успешности реализации и внедрения нового IT-актива и, по сути, определяет его ценность.

Задачи, которые нужно решить для достижения поставленных целей, могут затрагивать разные аспекты, например:

- реализовать клиентское приложение для ОС Linux в связи с увеличением количества потенциальных пользователей, использующих эту платформу;
- перейти на версию СУБД V.V для увеличения производительности БД;
- реализовать функцию X (или бизнес-процесс P);
- модифицировать функцию F для учёта изменившегося законодательства;

Стадии канонического проекта разработки ИС и ИТ. Ключевые положения концепции. Функции.

Возможности системы обычно разделяются на функциональные (обеспечивающие реализацию бизнес-требований) и нефункциональные (служебные, обеспечивающие реализацию нефункциональных требований). Перечень функциональных возможностей напрямую получается детализацией задач, стоящих перед системой. Нефункциональные возможности обеспечивают такие показатели качества системы, как сопровождаемость и мобильность.

Обычно возможности системы классифицируются по важности (критичности) (см. например "метод MoSCoW"):

- обязательные – должны быть реализованы в системе обязательно;
- критические – подлежат реализации, за исключением случаев, когда реализация не возможна по объективным причинам;
- важные – влияют на качество системы, но не критичны для её функционирования;
- перспективные – возможности, повышающие способность системы к развитию в перспективе.

Стадии канонического проекта разработки ИС и ИТ. Эскизный проект.

- разработка предварительных проектных решений по системе и ее частям;
- разработка эскизной документации на ИС и ее части.

Стадии канонического проекта разработки ИС и ИТ. Технический проект.

- разработка проектных решений по системе и ее частям;
- разработка документации на ИС и ее части;
- разработка и оформление документации на поставку комплектующих изделий;
- разработка заданий на проектирование в смежных частях проекта.

Стадии канонического проекта разработки ИС и ИТ. Рабочая документация.

- разработка рабочей документации на ИС и ее части;
- разработка и адаптация программ.

Стадии канонического проекта разработки ИС и ИТ. Ввод в действие.

- подготовка объекта автоматизации;
- подготовка персонала;
- комплектация ИС поставляемыми изделиями (программными и техническими средствами, программно-техническими комплексами, информационными изделиями);
- строительно-монтажные работы;
- пусконаладочные работы;
- проведение предварительных испытаний;
- проведение опытной эксплуатации;
- проведение приемочных испытаний.

Стадии сопровождения проекта ИС и ИТ.

- выполнение работ в соответствии с гарантийными обязательствами;
- послегарантийное обслуживание.

Методологии разработки ПО и ИТ.

Основные методологии разработки ИС и ИТ

- Waterfall model (водопадная модель);
- V-model (модель с верификацией);
- Incremental model (инкрементная модель);
- Iterative model (итеративная модель);
- Spiral model (спиральная модель);
- Agile Model (гибкая модель).

Принципы водопадной модели разработки

- Документы и инструкции - это важно, всё должно быть зафиксировано.
- Следующий этап работы не начинается, пока не закончится предыдущий.
- Пропускать этапы нельзя.
- Если требования к продукту изменились после согласования - переписываем ТЗ.
- Нельзя возвращаться на предыдущий этап, чтобы что-то изменить.
- Нет итераций, есть один общий процесс создания продукта.
- Выявлять и исправлять ошибки - только на этапе тестирования.
- Клиент не участвует в создании продукта после постановки ТЗ.

Этапы водопадной модели разработки

- **Аналитика**
  Команда собирает требования к будущему продукту. Потом пишет подробное техническое задание, планирует график работ и возможные риски. Переходит к следующему этапу, только когда все требования прописаны и есть план. А в плане — инструкции, что и когда делать.

- **Проектирование**
  Команда создаёт прототип и готовит дизайн-макеты. Когда это готово, подключаются разработчики.

- **Разработка**
  На этом этапе пишут код продукта согласно плану, макетам и требованиям. Ни шагу в сторону, всё четко по ТЗ.

- **Тестирование**
  Код готов, начинается тестирование. Тут могут появиться проблемы. Например, команда обнаружит серьёзные ошибки в коде и потратит много времени, чтобы их исправить. Это главный минус каскадной модели разработки.

- **Эксплуатация и поддержка**
  Проект передают заказчику и следят заранее определённое время, чтобы всё работало.

Краткое описание модели

1. Последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей.

2. В модели Waterfall легко управлять проектом. Благодаря её жесткости, разработка проходит быстро, стоимость и срок заранее определены.

3. Водопадная модель будет давать отличный результат только в проектах с четко и заранее определенными требованиями и способами их реализации. Нет возможности сделать шаг назад, тестирование начинается только после того, как разработка завершена или почти завершена.

4. Стоимость внесения изменений высока, так как для ее инициализации приходится ждать завершения всего проекта.

Когда использовать водопадную модель?
- требования известны, понятны и зафиксированы. Противоречивых требований не имеется.
- нет проблем с доступностью программистов нужной квалификации.
- в относительно небольших проектах.

V-модель разработки

V-модель, пояснения по этапам тестирования

Модульное тестирование (юнит-тестирование) - процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы, наборы из одного или более программных модулей вместе с соответствующими управляющими данными, процедурами использования и обработки. Примеры утилит тестирования: QTest, QSignalSpy.

Интеграционное тестирование (I&T) - Интеграционное тестирование в качестве входных данных использует модули, над которыми было проведено модульное тестирование, группирует их в более крупные множества, выполняет тесты, определённые в плане тестирования для этих множеств, и представляет их в качестве выходных данных и входных для последующего системного тестирования. Пример ПО для тестирования: VectorCAST/C++

V-модель, пояснения по этапам тестирования

Функциональное тестирование - это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает. Пример ПО тестирования: Ranorex.

Приемо-сдаточное тестирование (UAT) - это тип тестирования, выполняемый конечным пользователем или клиентом для проверки / принятия системы программного обеспечения перед перемещением приложения в производственную среду. UAT выполняется на заключительном этапе тестирования после выполнения функциональных, интеграционных и системных испытаний.

Краткое описание модели

1. Особенностью модели можно считать то, что она направлена на тщательную проверку и тестирование продукта, находящегося уже на первоначальных стадиях проектирования.

2. Стадия тестирования проводится одновременно с соответствующей стадией разработки, например, во время кодирования пишутся модульные тесты.

Когда использовать V-модель?

- если требуется тщательное тестирование продукта;

- для малых и средних проектов, где требования четко определены и фиксированы;

- в условиях доступности инженеров необходимой квалификации, особенно тестировщиков.

Инкрементная модель разработки

Инкрементная модель разработки (социальная сеть)

Заказчик решил, что хочет запустить соцсеть, и написал подробное техническое задание. Программисты предложили реализовать основные функции - страницу с личной информацией и чат. А затем протестировать на пользователях, «взлетит или нет».

Команда разработки показывает продукт заказчику и выпускает его на рынок. Если и заказчику, и пользователям социальная сеть нравится, работа над ней продолжается, но уже по частям.

Программисты параллельно создают функциональность для загрузки фотографий, обмена документами, прослушивания музыки и других действий, согласованных с заказчиком. Инкремент за инкрементом они совершенствуют продукт, приближаясь к описанному в техническом задании.

Краткое описание модели

1. В инкрементной модели полные требования к системе делятся на различные сборки.

2. Имеют место несколько циклов разработки, и вместе они составляют жизненный цикл «мульти-водопад».

3. Цикл разделен на более мелкие легко создаваемые модули.

4. Каждый модуль проходит через фазы определения требований, проектирования, кодирования, внедрения и тестирования.

5. Процедура разработки по инкрементной модели предполагает выпуск на первом большом этапе продукта в базовой функциональности, а затем уже последовательное добавление новых функций, так называемых «инкрементов».

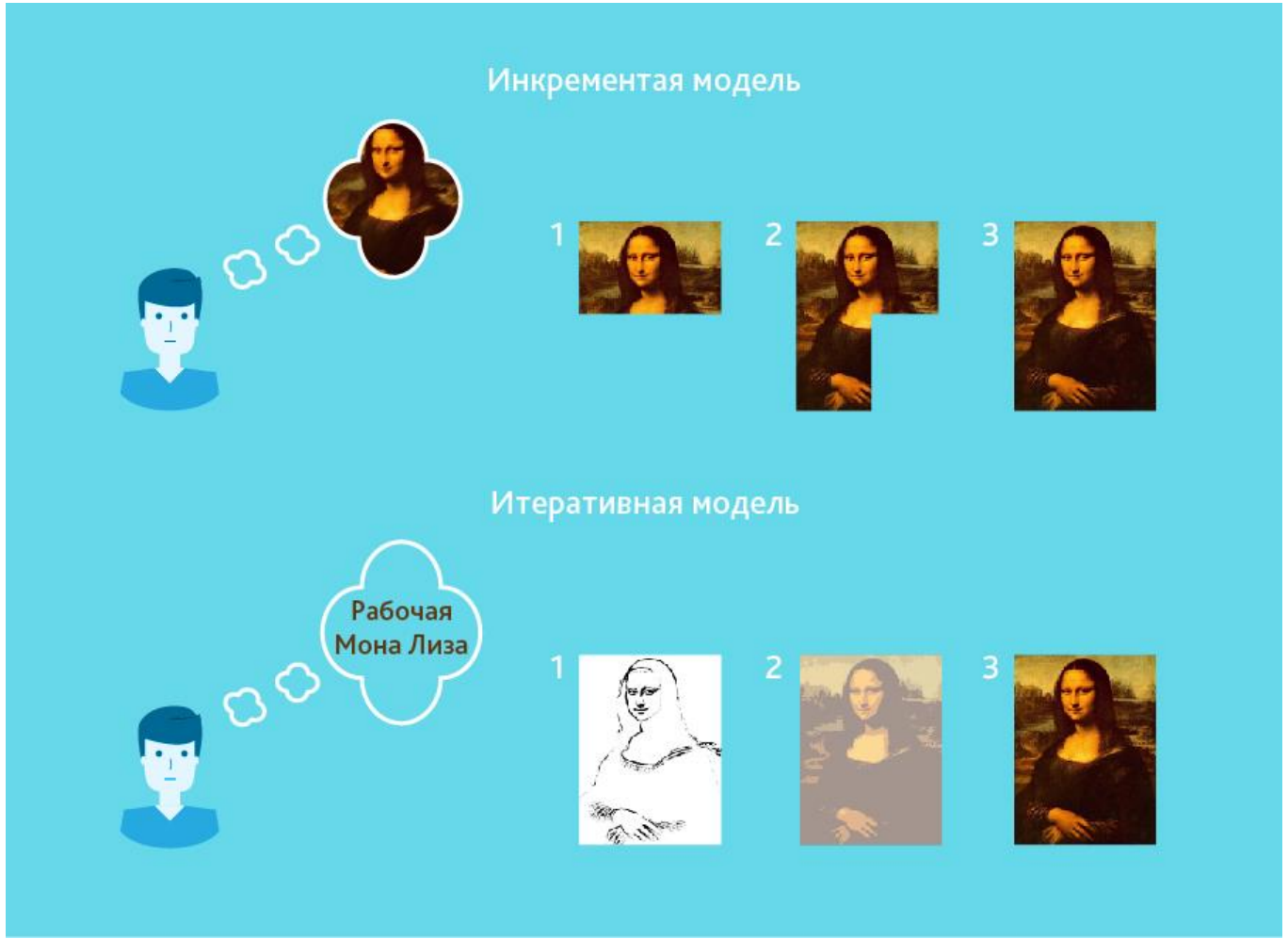
Когда использовать инкрементную модель?

- когда основные требования к системе четко определены и понятны. В то же время некоторые детали могут дорабатываться с течением времени;

- требуется ранний вывод продукта на рынок;

- есть несколько рисковых feature или целей.

Итеративная модель разработки



Итеративная модель разработки (мессенджер)

Заказчик решил, что хочет создать мессенджер. Разработчики сделали приложение, в котором можно добавить друга и запустить чат на двоих.

Мессенджер «выкатили» в магазин приложений, пользователи начали его скачивать и активно использовать. Заказчик понял, что продукт пользуется популярностью, и решил его доработать.

Программисты добавили в мессенджер возможность просмотра видео, загрузки фотографий, записи аудиосообщений. Они постепенно улучшают функциональность приложения, адаптируют его к требованиям рынка.

Краткое описание модели

1. Модель не требует для начала полной спецификации требований.

2. Создание начинается с реализации части функционала, становящейся базой для определения дальнейших требований. Этот процесс повторяется.

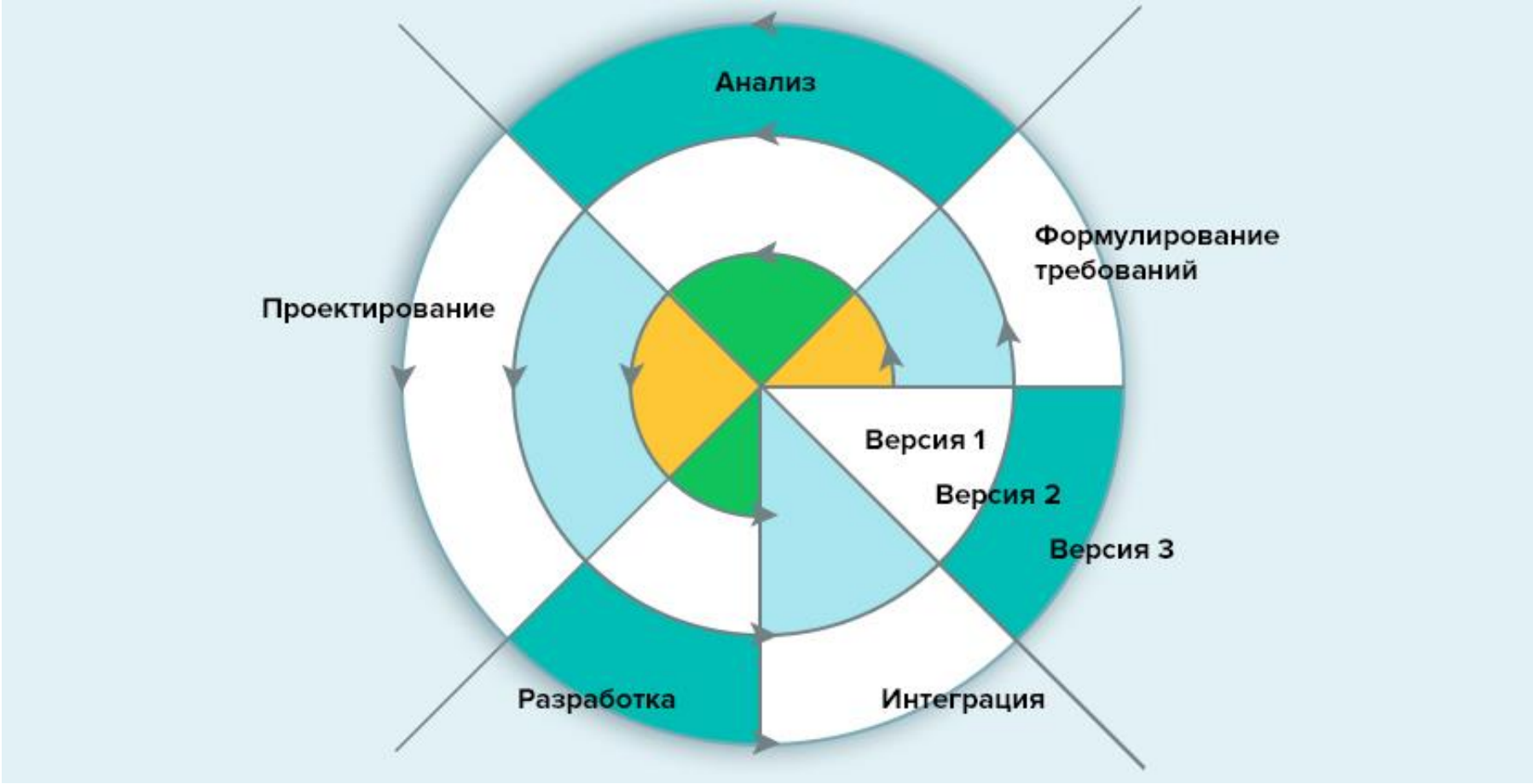
Когда оптимально использовать итеративную модель?

- общие требования к конечной системе в целом понятны;

- проект большой или очень большой;

- основная задача должна быть определена, но детали реализации могут эволюционировать с течением времени.

Спиральная модель разработки



Спиральная модель разработки (умный дом)

Заказчик решил, что хочет сделать такую систему, и заказал программистам реализовать управление чайником с телефона. Они начали действовать по модели «водопад»: выслушали идею, провели анализ предложений на рынке, обсудили с заказчиком архитектуру системы, решили, как будут её реализовывать, разработали, протестировали и «выкатили» конечный продукт.

Заказчик оценил результат и риски: насколько нужна пользователям следующая версия продукта — уже с управлением телевизором. Рассчитал сроки, бюджет и заказал разработку. Программисты действовали по водопадной модели и представили заказчику более сложный продукт, разработанный на базе первого.

Заказчик подумал, что пора создать функциональность для управления холодильником с телефона. Но, анализируя риски, понял, что в холодильник сложно встроить Wi-Fi-модуль, да и производители не заинтересованы в сотрудничестве по этому вопросу. Следовательно, риски превышают потенциальную выгоду. На основе полученных данных заказчик решил прекратить разработку и совершенствовать имеющуюся функциональность, чтобы со временем понять, как развивать систему «Умный дом».

Краткое описание модели

В спиральной модели жизненный путь разрабатываемого продукта изображается в виде спирали, которая, начавшись на этапе планирования, раскручивается с прохождением каждого следующего шага. Таким образом, на выходе из очередного витка получаем готовый протестированный прототип, который дополняет существующую сборку. Прототип, удовлетворяющий всем требованиям, готов к выпуску.

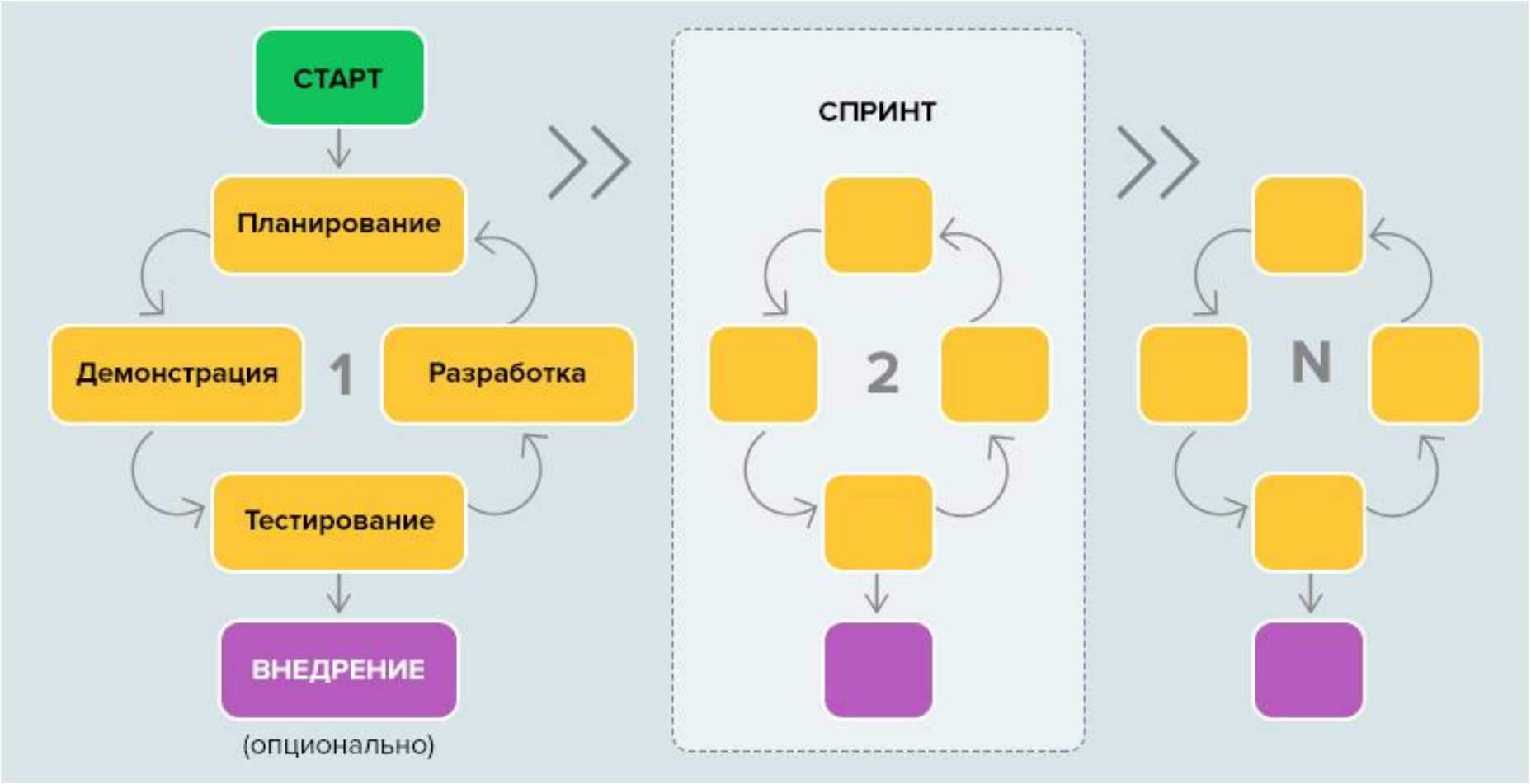
Плюсы:

- управлению рисками уделяется особое внимание;
- дополнительные функции могут быть добавлены на поздних этапах;
- есть возможность гибкого проектирования.

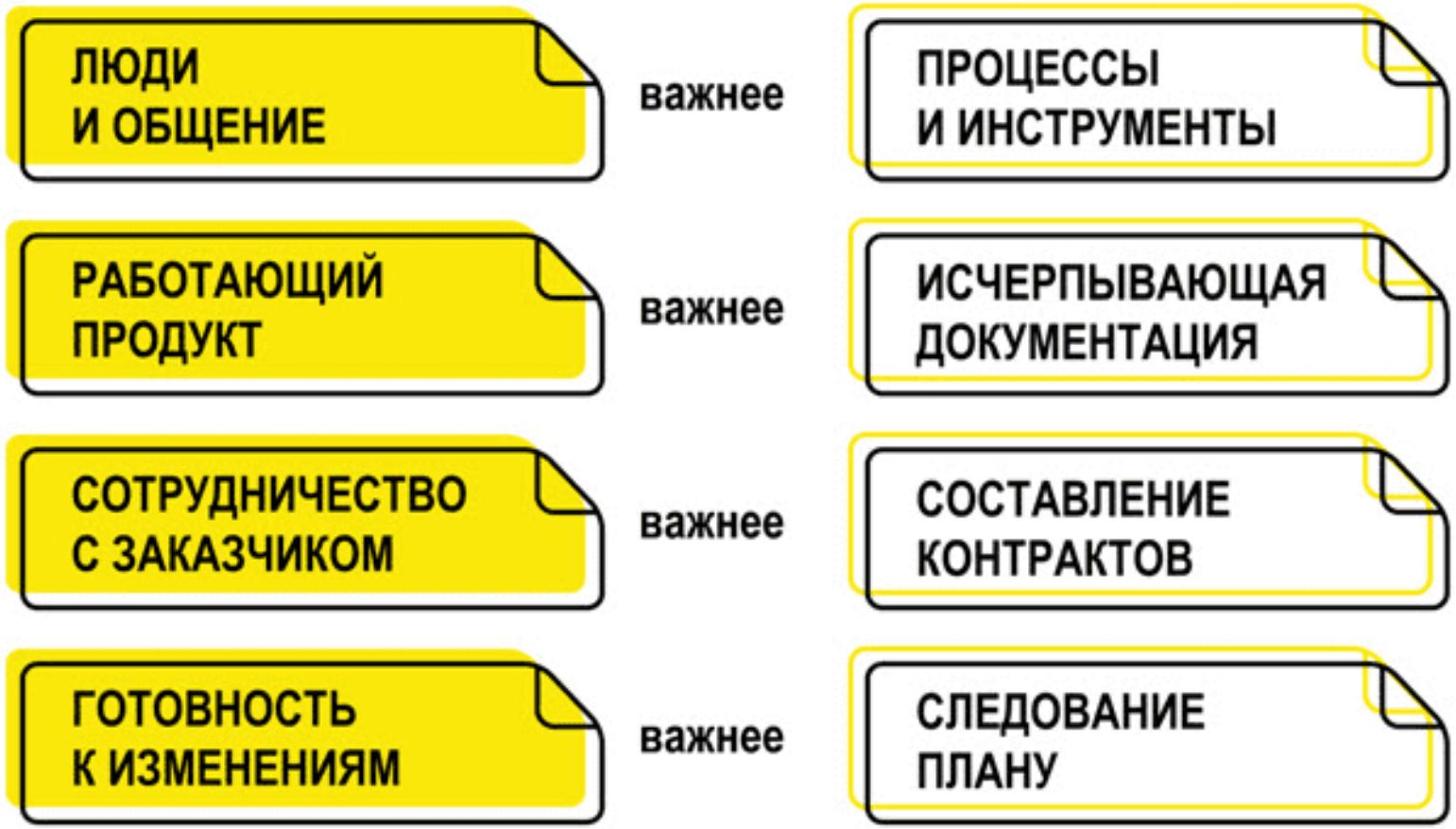
Минусы:

- оценка рисков на каждом этапе является довольно затратной;
- постоянные отзывы и реакция заказчика может провоцировать все новые и новые итерации, которые могут приводить к временному затягиванию разработки продукта;
- более применима для больших проектов.

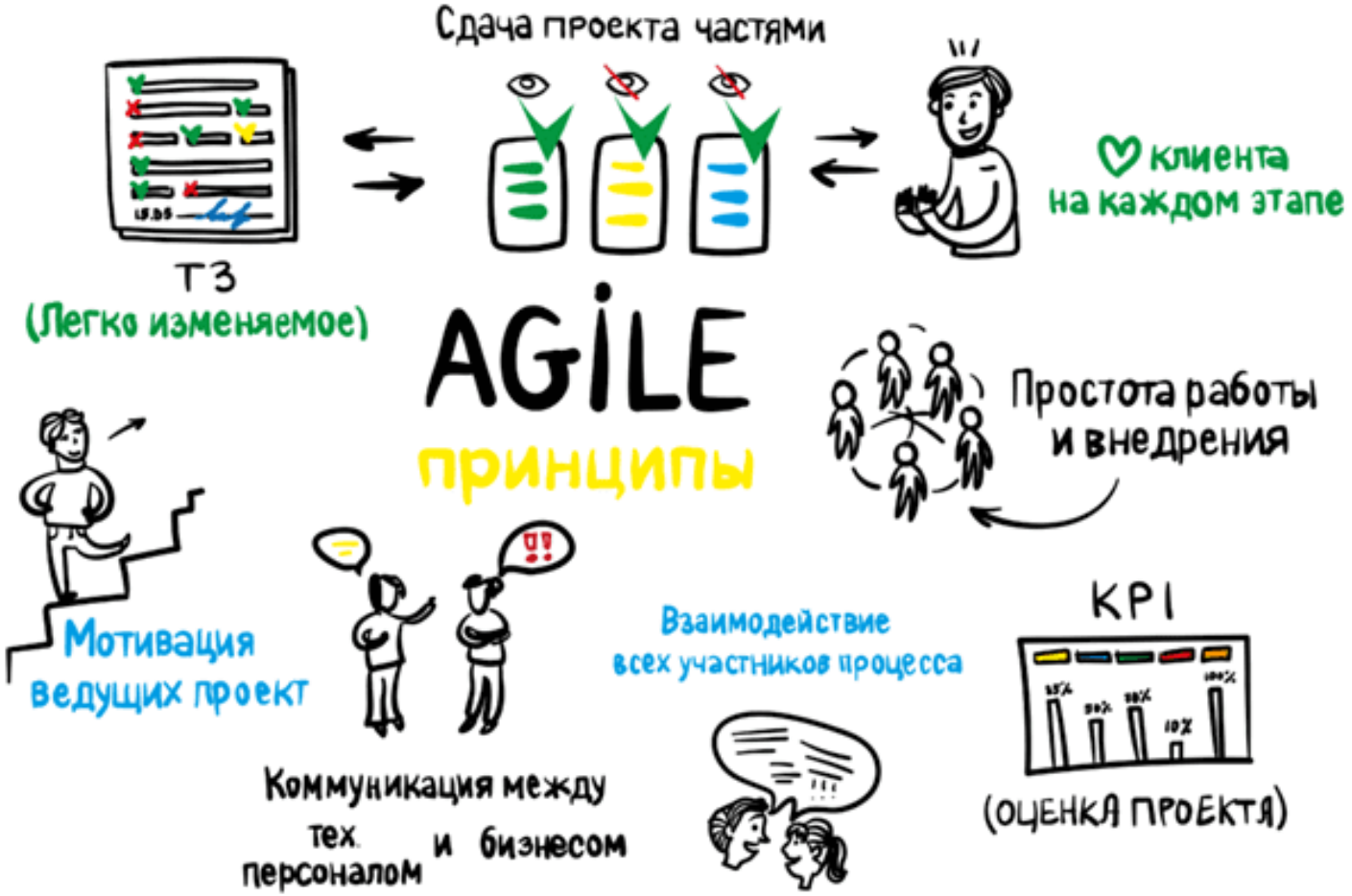
Модель гибкой разработки (Agile)



Идеи гибкой разработки (Agile)



Принципы гибкой разработки (Agile)



Краткое описание модели

1. В «гибкой» методологии разработки после каждой итерации заказчик может наблюдать результат и понимать, удовлетворяет он его или нет.

2. При этом сложно оценить трудозатраты и стоимость, требуемые на разработку.

3. Методология подходит для больших или нацеленных на длительный жизненный цикл проектов, постоянно адаптируемых к условиям рынка. Соответственно, в процессе реализации требования изменяются.

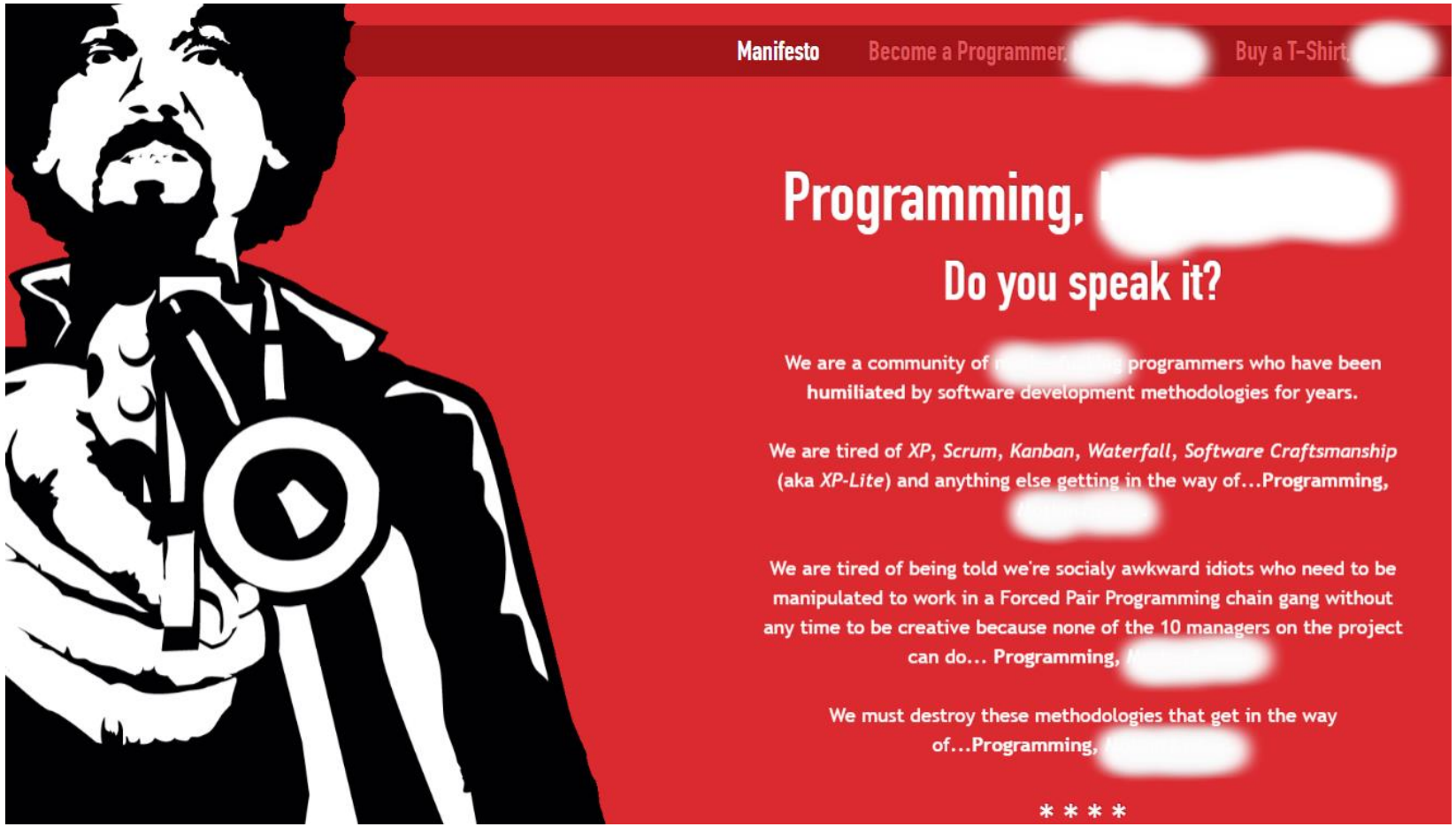
Когда использовать Agile?

- когда потребности пользователей постоянно меняются в динамическом бизнесе;

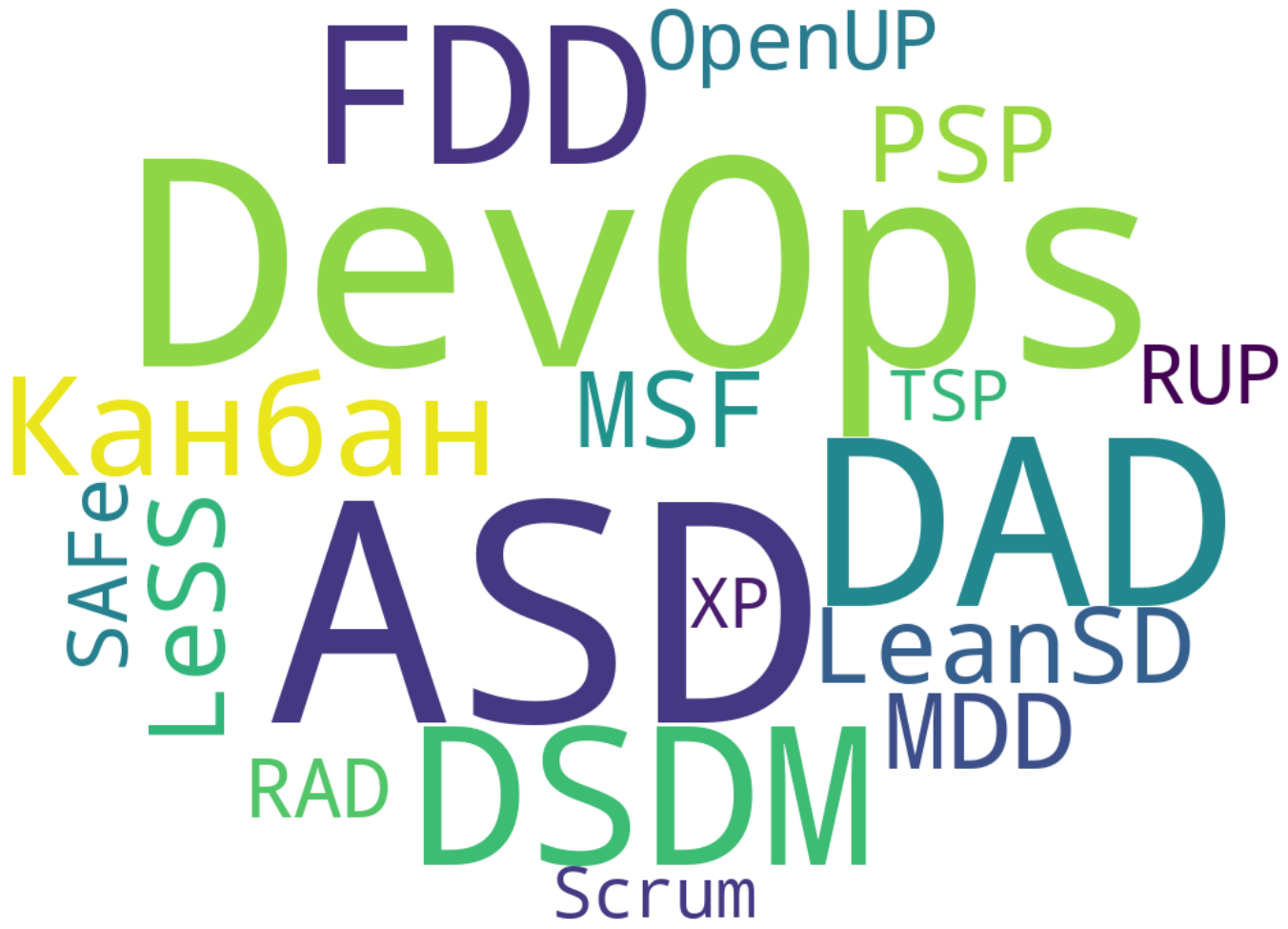
- изменения на Agile реализуются за меньшую цену из-за частых инкрементов;

- в отличие от модели водопада, в гибкой модели для старта проекта достаточно лишь небольшого планирования.

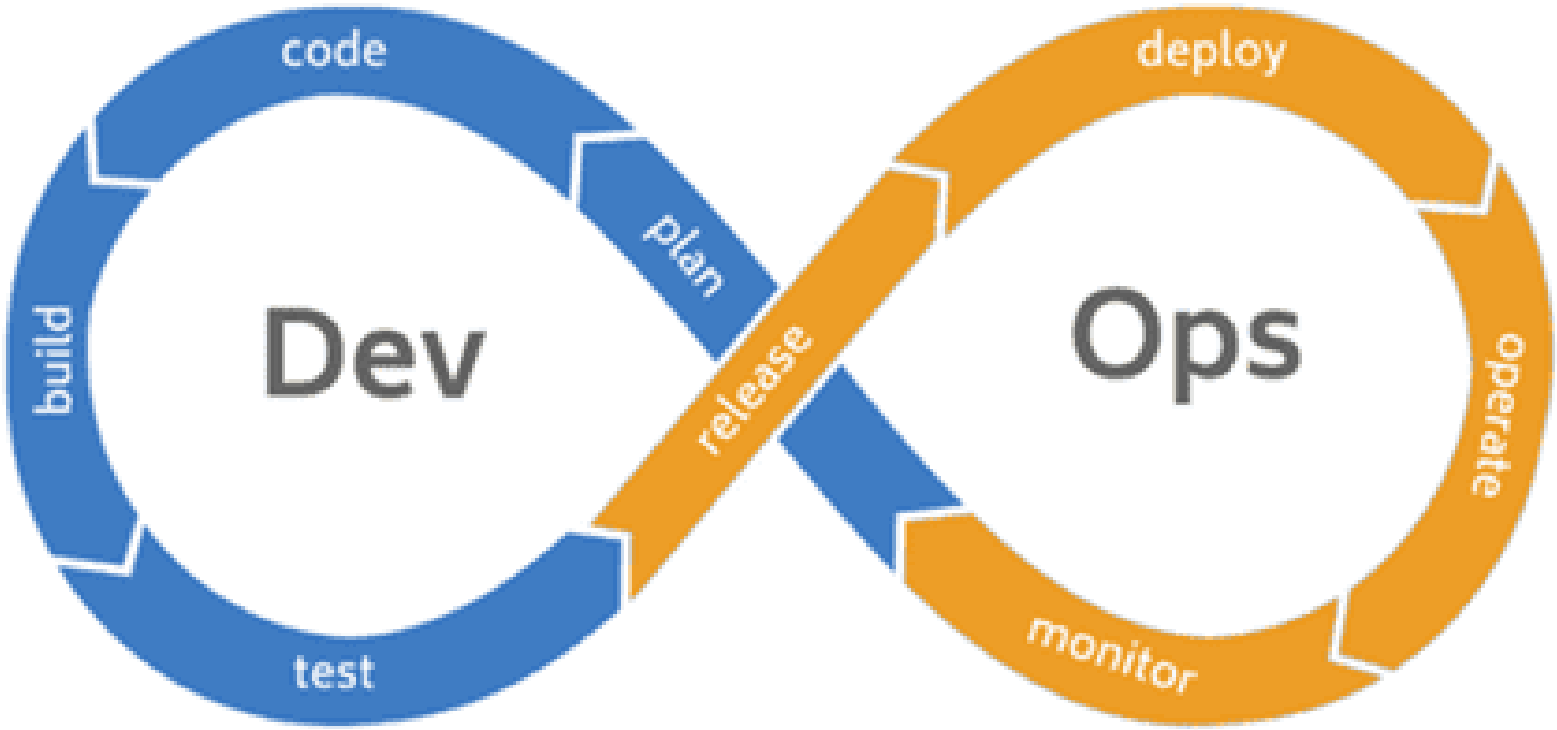
Немного про “цикличность” развития моделей (:



Множество методологий разработки ПО



Методология DevOps



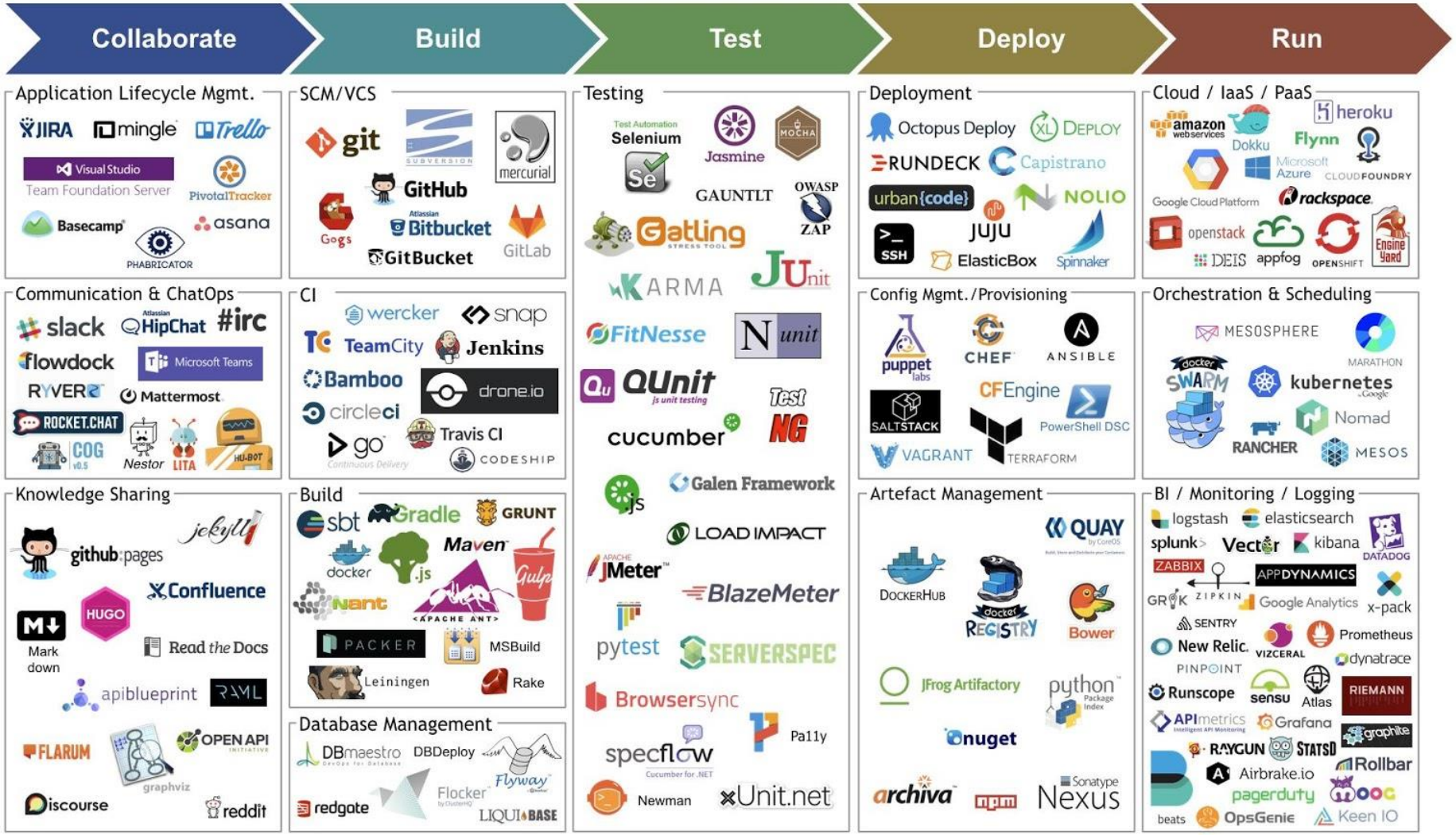
Методология DevOps

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. Основана на идее тесной взаимозависимости создания продукта и эксплуатации программного обеспечения, которая прививается команде как культура создания продукта.

Методология фокусируется на стандартизации окружений разработки с целью быстрого переноса программного обеспечения через стадии, способствуя быстрому выпуску версий.

Задача DevOps-инженеров — сделать процесс разработки и поставки программного обеспечения согласованным с эксплуатацией объединив их в единую команду, что позволяет организовать процессы, которые далее можно автоматизировать с помощью инструментов.

Инструментарий DevOps



Статьи для самостоятельного изучения

- Материалы лекции 2 на сайте http://msuniversity.ru
- По учебнику Технологии разработки ПО: Глава 1 Организация процесса разработки, стр. 22-35, http://msuniversity.ru/d/13
- Книга Стандартизация разработки программных средств, Стандарты комплекса ГОСТ 34, стр. 80-85. http://msuniversity.ru/d/13
- Книга (монография) Управление жизненным циклом информационных систем, стр. 16-30. http://msuniversity.ru/d/13
- Habr. Технология сбора требований в процесс проектирования сайта. https://habr.com/ru/post/142019/
- LJ_exp. Концепция информационной системы. https://m-i-kuznetsov.livejournal.com/158887.html
- Habr. Разработка Технического задания по ГОСТ 34 легко и просто. https://habr.com/ru/post/432852/
- Prj-exr.ru. Пояснительная записка к эскизному проекту. https://www.prj-exp.ru/patterns/pattern_draft_project.php

Статьи для самостоятельного изучения

- Habr. Организация процессов производства ИС. Внедрение ИС. https://habr.com/ru/post/351198/
- Prj-exr.ru. Акт приемки в опытную эксплуатацию. https://www.prj-exp.ru/patterns/pattern_act_of_trial_operation.php
- ecm-journal.ru. Организация обучения работе в информационной системе в крупной компании. https://ecm-journal.ru/card.aspx?ContentID=6959510
- edsd.ru (частично рекламный буклет). Гарантия на программное обеспечение или техническая поддержка. https://www.edsd.ru/ru/o_kompanii/novosti/21_nov_2012-garantiya-i-tehnicheskaya-podderjka-programmnogo-obespecheniya

Спасибо за внимание!