GeekBrains

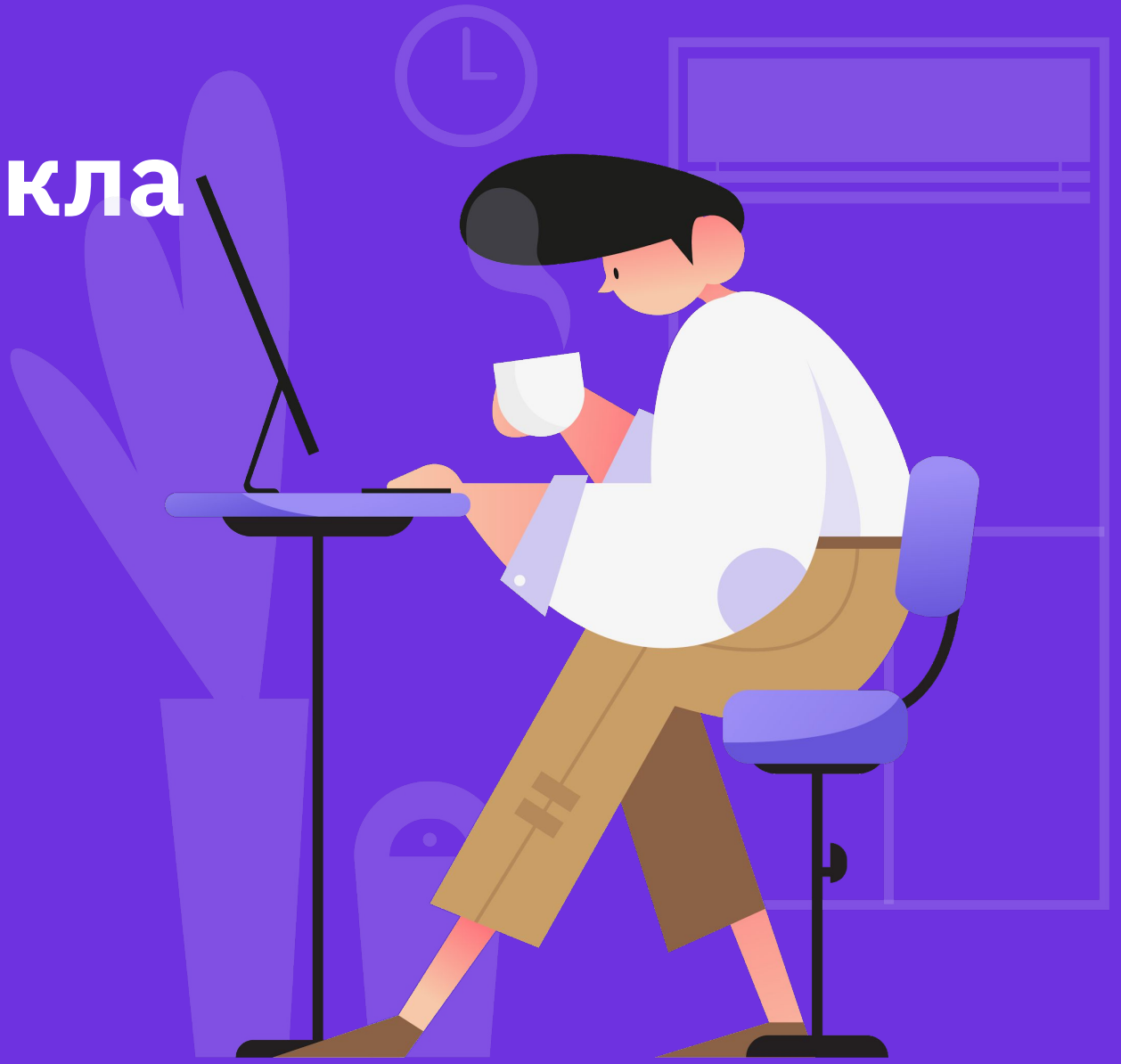
Жизненный цикл ПО

Основы ручного тестирования

Что будет на уроке

1. Модели разработки ПО.
2. Методологии разработки ПО.
3. SCRUM и Kanban.
4. Жизненный цикл тестирования.
5. Уровни независимости тестирования.
6. Правила коммуникации с командой разработки.

Модели жизненного цикла ПО



GeekBrains

Что такое модель жизненного цикла ПО?

Модель жизненного цикла ПО — это описание, как проходит процесс разработки.

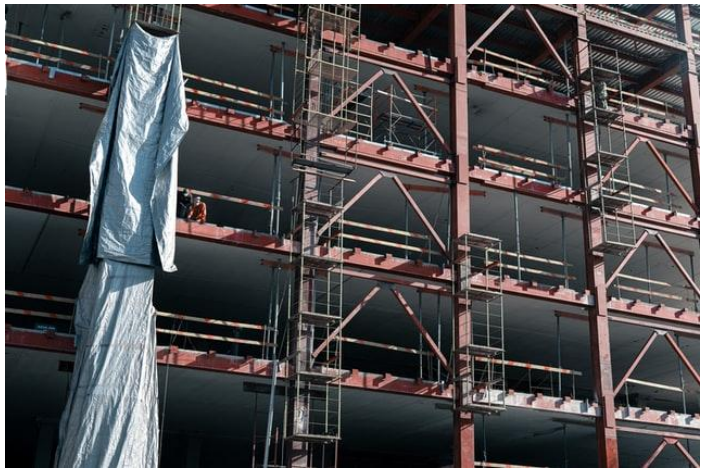
Модель — это, прежде всего, этапы разработки и их последовательность.

Каскадная модель (водопад, waterfall)

Этапы жизненного цикла выполняются последовательно.

Невозможно начать следующий этап, если предыдущий не завершился.

Пример из жизни: строительство дома



Каскадная модель

Преимущества

- стабильные требования перед началом разработки;
- у каждой стадии проверяемый результат;
- в каждый момент команда выполняет один вид работы;
- легко контролировать.

Недостатки

- позднее получение результата;
- позднее обнаружение конструктивных ошибок;
- сложность возврата к предыдущим этапам;
- отсутствие обратной связи от пользователей;
- неравномерная загрузка участников проекта.

Каскадная модель — сферы использования

- крупные проекты со стабильными требованиями — космическая отрасль, медицинское ПО;
- недорогие, несложные, средние проекты;
- есть позитивный опыт разработки аналогичных систем;
- создание новой версии ранее разработанного продукта, когда вносимые изменения определены и управляемы;
- перенос уже существующего продукта на новую платформу.

V-модель

Усовершенствованная водопадная модель.

Тестирование появляется в начале, и каждому этапу модели соответствует свой этап тестирования.

V-модель

Преимущества

- простота в использовании;
- планирование и проектирование тестирования на ранних этапах разработки;
- верификация и аттестация всех внешних и внутренних полученных данных, а не только конечного продукта.

Недостатки

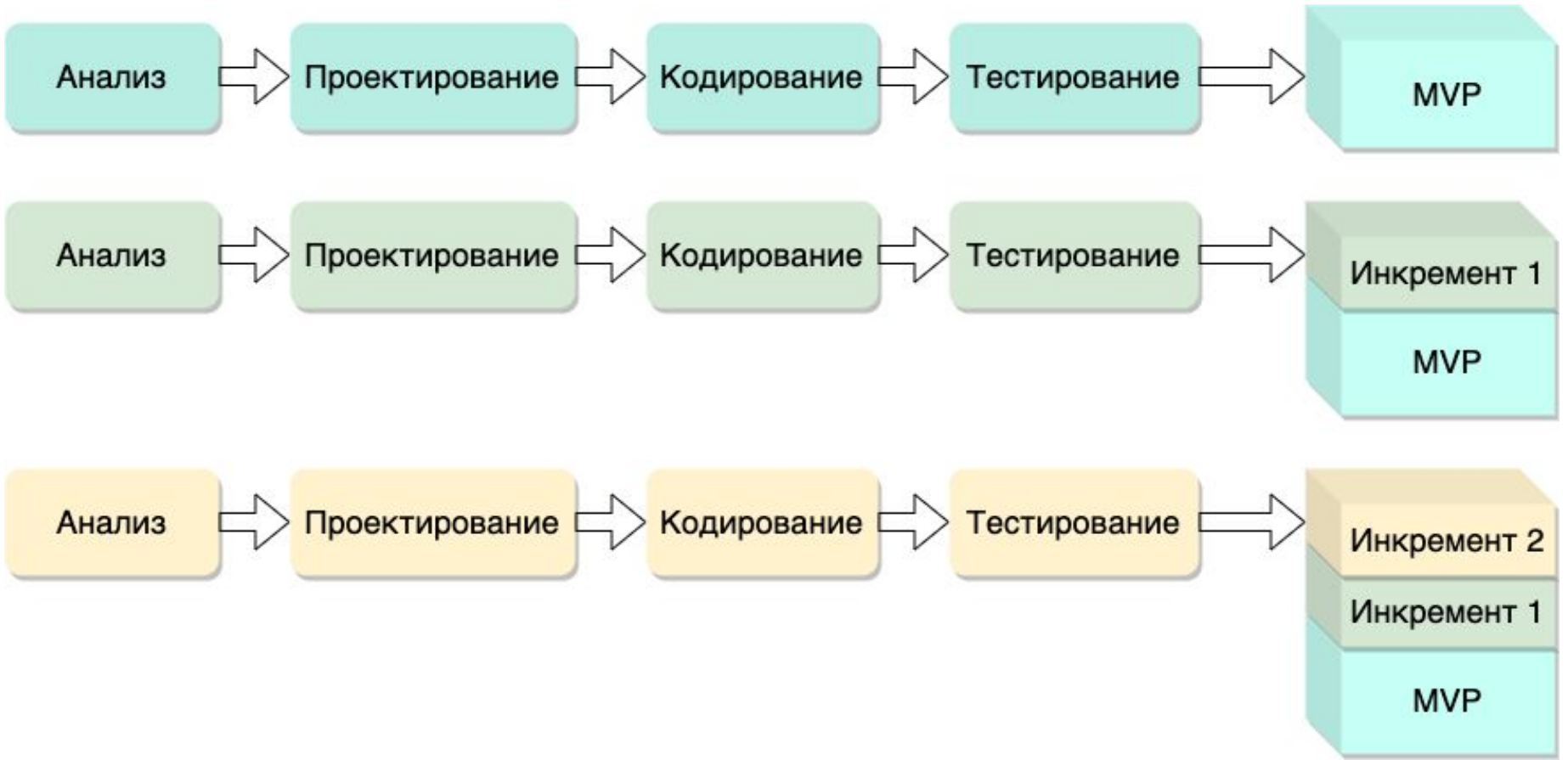
- сложно вносить изменения;
- нет действий, направленных на анализ рисков;
- другие недостатки «водопада».

V-модель: сферы использования

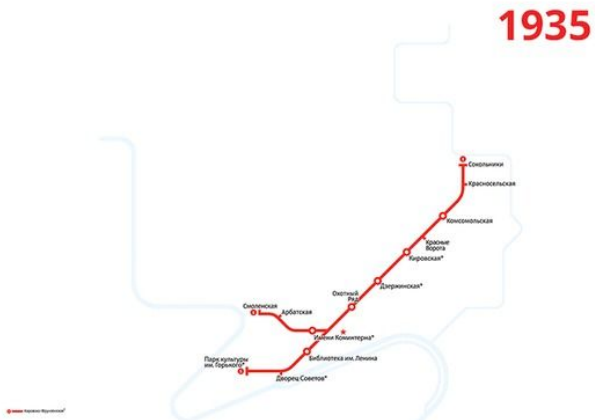
- управление разработкой ПО в немецкой администрации;
- стандарт для немецких правительственных и оборонных проектов;
- системы, в которых требуется высокая надёжность, например, прикладные программы для наблюдения за пациентами в клиниках;
- встроенное ПО для устройств управления аварийными подушками безопасности в автомобилях.

Инкрементная модель

Инкремент — это постоянно увеличивающаяся величина.



Пример из жизни: метро



Инкрементная модель

Преимущества

- позволяет уменьшить затраты на первоначальных этапах разработки;
- ускоряет процесс создания функционирующей системы;
- позволяет получать отзывы заказчика в процессе разработки;
- снижает риск неудачи при изменении требований.

Недостатки

- в начале разработки не всегда определяется функциональность ПО;
- сложность приоритизации инкрементов;
- может возникать тенденция к откладыванию сложных решений на будущее.

Инкрементная модель: сферы использования

Проекты, в которых требования сформулированы заранее, и требуется быстрая поставка на рынок.

Итерационная модель

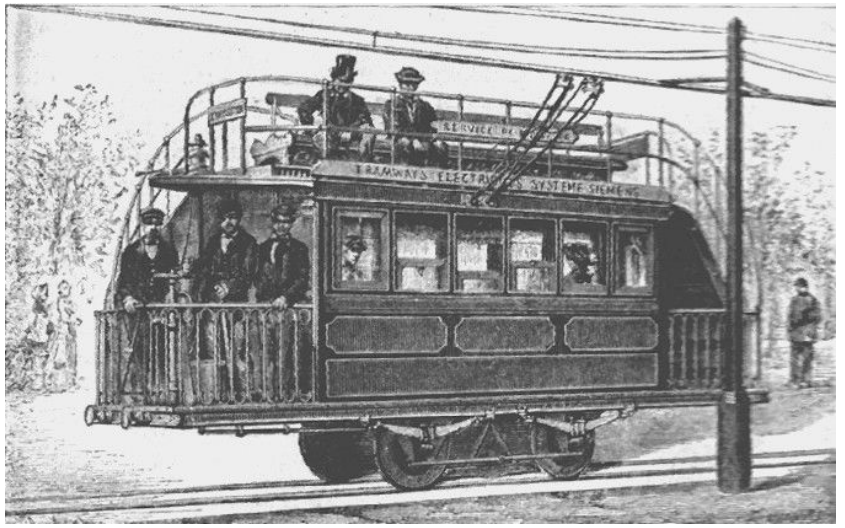
Разработка идёт циклами — итерациями.

Итерация — временной интервал:

- в течение которого проходят все фазы разработки, например, анализ, проектирование, разработка, тестирование;
- который многократно повторяется.

Пример из жизни: трамвай

1828 год — лошадь вместо двигателя, 1 вагон



1873 год — канатная тяга, паровой двигатель вместо лошади

2020 год — приводится в движение электричеством

Итерационная модель

Преимущества

- быстрый выпуск минимального продукта;
- не требуется полной спецификации требований заранее;
- эффективная обратная связь с заказчиком;
- раннее обнаружение ошибок в требованиях;
- равномерная загрузка участников проекта.

Недостатки

- могут возникнуть проблемы с архитектурой;
- отсутствие фиксированного бюджета и сроков;
- нет конечной цели, может привести к затягиванию проекта.

Итерационная модель: сферы использования

- большие проекты с неопределёнными требованиями;
- проекты по разработке ПО, которое носит инновационный характер и основано на бизнес-гипотезах, требующих проверки.

Инкрементно-итерационная модель

- с точки зрения жизненного цикла, модель считается **итерационной**, так как подразумевает многократное повторение одних и тех же стадий;
- с точки зрения развития продукта, приращения его полезных функций, модель представляется **инкрементальной**.

Спиральная модель

Частный случай итерационно-инкрементальной модели, где акцент делается на управлении рисками, в особенности влияющими на организацию процесса разработки проекта и контрольные точки.

- проработка целей, альтернатив и ограничений;
- анализ рисков и прототипирование;
- разработка продукта (промежуточной версии);
- планирование следующего цикла.

Спиральная модель

Преимущества

- качественный анализ рисков;
- активное участие заказчиков, особенно на стадии планирования;
- усовершенствование процесса на каждом витке.

Недостатки

- высокая стоимость разработки;
- сложно определить момент перехода на следующий виток спирали;
- сложная структура, трудная для понимания;
- спираль может продолжаться бесконечно.

Спиральная модель: сферы использования

- проекты с высоким риском;
- исследовательские проекты;
- проекты, в которых изначально неизвестно, возможна ли их реализация в текущих условиях.

Методологии разработки

Определение

Методология разработки — это набор методов по управлению разработкой ПО, набор практических правил и техник разработки. Или по-другому, методология — это детализированный набор правил, практик и принципов, применяемый при реализации той или иной модели.

Фреймворк процессов — это методология, содержащая большое число правил. Необязательно использовать их все, можно выбрать только те, что нужны сейчас, и построить на их основе процесс разработки.

Гибкие методологии

Agile — это философия разработки, на которую опираются гибкие методологии.

Agile объединяет гибкие методологии разработки.

1. SCRUM.
2. Kanban.
3. Lean.
4. прочее.

Гибкие методологии и Agile Manifesto

Гибкие методологии разработки — обобщающий термин для нескольких подходов и практик, основанных на ценностях манифеста гибкой разработки программного обеспечения и 12 принципах, лежащих в его основе.

Agile-манифест — основные идеи, на основании которых выстраиваются процессы в гибких методологиях разработки:

- люди и их взаимодействие важнее процессов и инструментов;
- готовый продукт важнее документации по нему;
- сотрудничество с заказчиком важнее жёстких контрактных ограничений;
- реакция на изменения важнее следования плану.

Принципы гибкой разработки

Наивысший приоритет — удовлетворение потребностей заказчика благодаря регулярной и ранней поставке нового инструментария. Чем раньше заказчик увидит новую функциональность, тем быстрее команда проекта получит обратную связь и скорректирует свою работу.

Изменение требований приветствуется даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

Работающий продукт следует выпускать как можно чаще с периодичностью от двух недель до нескольких месяцев.

Особенности Agile-команд

- при планировании работ учитываются интересы всех членов команды;
- работа над завершением задач более приоритетного уровня идёт в первую очередь;
- признание проблем;
- члены команды помогают друг другу.

Agile-команды

Преимущества

- быстрая поставка;
- взаимодействие с заказчиком;
- возможность получения быстрой обратной связи от конечных пользователей;
- адаптируемость к изменению внешних условий;
- в процессе разработки есть возможность корректировать требования.

Недостатки

- сложно применять в непростых проектах;
- опасность возникновения неучтённых рисков;
- часто недостаточно документации.

SCRUM

SCRUM — набор принципов, ценностей, политик, ритуалов, артефактов, на которых строится процесс SCRUM-разработки. Процесс его разработки позволяет в жёстко фиксированные и небольшие по времени итерации, называемые спринтами, предоставлять конечному пользователю работающий продукт с новыми бизнес-возможностями, для которых определён наибольший приоритет.

Роли в SCRUM-команде

1. Скрам-мастер (SCRUM Master).
2. Владелец продукта (Product Owner).
3. Команда разработки (Development Team):
    - разработчики;
    - тестировщики;
    - аналитики;
    - и другие специалисты.

Особенности SCRUM

1. Разработка идёт итерациями — спринтами. В конце итерации — новая версия продукта.
2. Все активности жёстко регламентированы. Они называются «церемониями».
3. Основной артефакт — SCRUM-доска.

SCRUM-церемонии

Планирование — определение набора задач, который планируется реализовать в процессе спринта (бэклог спринта).

Ежедневный скрам-митинг — встреча всей команды для обсуждения текущей ситуации на проекте. Основная цель — оценка прогресса команды и оперативное выявление возникающих проблем.

После спринта проводится демонстрация полученных результатов — **демо**. На демонстрацию приглашаются все заинтересованные лица и владелец продукта.

Отдельно для команды проекта в SCRUM предусматривается проведение **ретроспективы** по итогам спринта и демо. На встрече подводятся итоги спринта, обсуждаются возникшие проблемы или успехи.

Kanban

Kanban — японский термин, который начали использовать в 60-х годах XX века в компании Toyota.

В основе — конвейерный метод производства и различные скорости выполнения отдельных технологических операций на производстве.

Kanban

Допустим, есть автомобильный завод и конвейер по сборке автомобилей. Левые и правые зеркала производятся на разных станциях. По каким-то причинам на «левой» станции произвели 10 зеркал, а на «правой» — 20.

В это время на конвейере сборки ожидает 15 автомобилей. Отсутствие 5 левых зеркал тормозит процесс сборки.

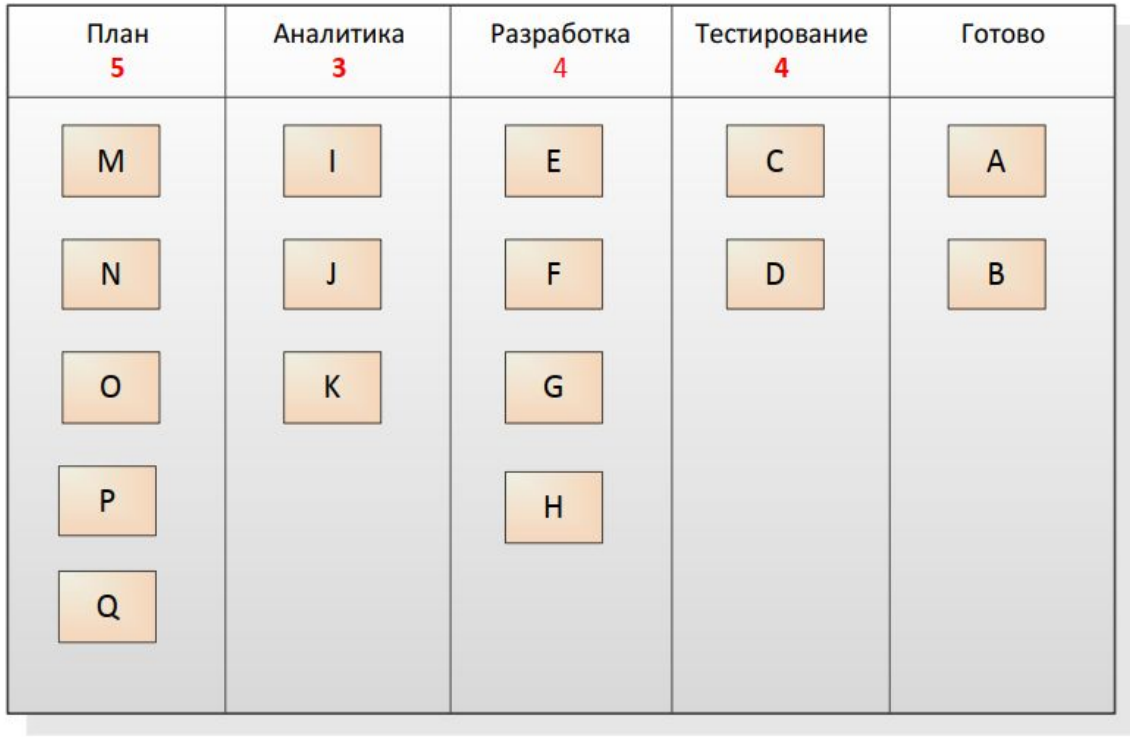
Kanban

1. Производственная станция имеет план производства деталей (backlog). План отсортирован по приоритету и может меняться в любое время. К примеру, станция, производящая слишком много левых зеркал, должна иметь возможность переключиться на правые как можно скорее.

2. Количество задач, выполняемых на станции одновременно, но ограничено. То есть производить не более заданного количества зеркал одновременно. Это ограничение требуется для управления скоростью производства на станции, а также скоростью реагирования на изменения плана.

Kanban

Основной инструмент в Kanban — доска. Главная её особенность — ограничение задач, которые могут быть одновременно в разработке или тестировании. Поэтому важно следить, чтобы в процессе работы не возникало «бутылочных горлышек» и большое число задач не скапливалось на одном этапе разработки.



Scrum VS Kanban

1. Гибкие технологии, придерживаются ценностей и принципов Agile Manifesto.
2. Ограничивают Work in progress.
3. Сосредотачиваются на выпуске программного обеспечения на раннем этапе.
4. Основываются на самоорганизующихся командах.
5. План выпуска постоянно оптимизируется на основе эмпирических данных.

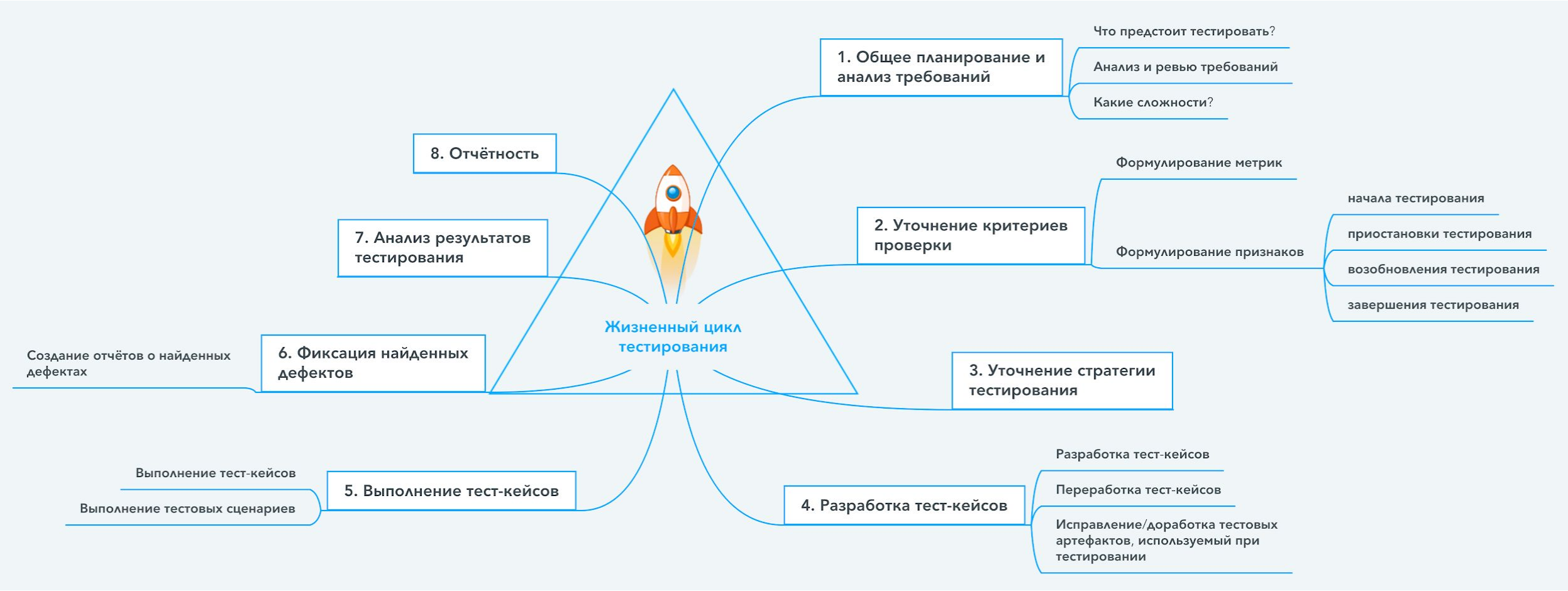
Scrum VS Kanban

	Scrum	**Kanban**
Роли	Есть, строго разграничены	Нет
Продолжительность итераций	Длина спринта неизменна	Фиксированной длины итераций нет
Ограничение WIP	В рамках спринта	В рамках «рабочей станции»
Состав команды	Кросс-функциональная	Разрешаются специальные команды
Work items	Помещаются в спринт	Нет ограничений на размер
Внесение изменений	После старта спринта изменять задачи нельзя	Изменения возможны, если позволяет лимит WIP
Доска	Новый спринт — новая доска	Одна доска на всё время работы
Церемонии	Обязательно: daily scrum meeting, планирование, груминг, ретро и т. д.	Нет
Цель	Закончить спринт	Выполнить задачу

Жизненный цикл тестирования

GeekBrains

Жизненный цикл тестирования



Тестирование на основе моделей

Тестовые модели — это абстрактные наглядные схемы, описывающие состояние, взаимодействия и связи системных компонентов.

Тестовые модели представляют собой схемы, таблицы, диаграммы переходов состояний и интеллект-карты.

Тестовые модели создаются на этапе проектирования системы и демонстрируют влияние одной части ПО на другую.

Тестирование на основе данных

Тестирование на основе данных (Data Driven testing, DDT) — это инфраструктура автоматизации тестирования, которая хранит тестовые данные в виде таблицы.

Входные значения считываются из файлов данных и сохраняются в переменную тестовых сценариев.

DDT объединяет положительные и отрицательные сценарии в один тест. Входные данные хранятся в файлах Excel, XML, CSV, базах данных.

Тестирование на основе риска

Тестирование на основе рисков (Risk Based Testing, RBT) — это тип тестирования, основанный на вероятности риска.

Риск — это возникновение неопределённого события, которое положительно или отрицательно влияет на измеримые критерии успеха проекта.

Чек-лист для тестирования на основе рисков

1. Важные функциональные возможности.
2. Видимая пользователю функциональность.
3. Функциональность, оказывающая наибольшее влияние на безопасность.
4. Финансово важные функциональные возможности.
5. Сложные области исходного кода и кодов, подверженных ошибкам.
6. Функции, добавленные в последнюю минуту.
7. Факторы подобных проектов, вызвавшие проблемы.
8. Факторы или проблемы аналогичных проектов, увеличившие эксплуатационные расходы.
9. Плохие требования.

Спасибо!

Каждый день вы становитесь лучше :)