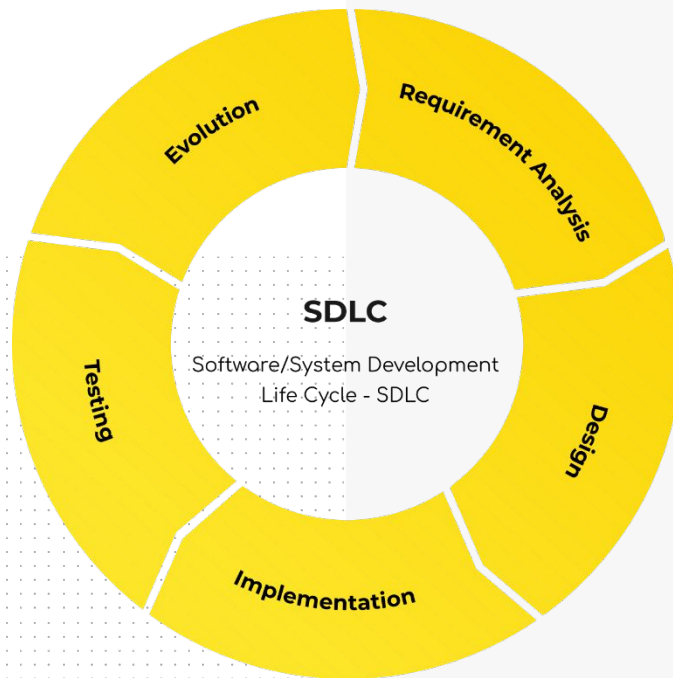


< Teach  
Me  
Skills />

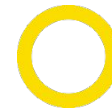
## Введение в тестирование программного обеспечения



# Жизненный цикл ПО (SDLC)



# Тестирование и качество



- Качество ПО - это совокупность характеристик ПО, относящихся к его способности удовлетворять установленные и предполагаемые потребности.



# QA vs Тестирование ПО



- Обеспечение качества (QA) - комплекс мероприятий направленный на обеспечение качества разрабатываемого продукта, на всех стадиях разработки.
- Тестирование ПО (Software Testing) - проверка соответствия между реальным и ожидаемым поведением программы, с помощью конечного набора тестов.



# Объекты тестирования

- Программы при их непосредственном запуске и исполнении (software)
- Код программ без запуска и исполнения (code)
- Прототип программного продукта (product prototype)
- Функциональные спецификации к продукту (functional specifications)
- Руководства по установке (Installation guide) и использованию (user manual)

# Этапы тестирования



- Анализ требований
- Планирование испытаний
- Тест-дизайн
- Запуск тестов
- Редактирование тестов
- Регрессионное тестирование
- Приемочные испытания
- Эксплуатация и сопровождение



# Анализ требований



1

Изучаем спецификации требований и функциональные требования к системе

2

Отвечаем на вопросы:

- a. что нам предстоит тестировать
- b. как много будет работы
- c. какие есть сложности
- d. всё ли необходимое у нас есть

# Планирование испытаний



1

Определяем объем испытаний и ресурсы

2

Пишем расписание того, когда будем выполнять намеченные действия



# Тест-дизайн



1

Определяем:

- i. Цель тестирования
- ii. Специфику входных данных (классы эквивалентности...)
- iii. Модули и подмодули приложения
- iv. Архитектуру тестов (детализация от крупного к деталям)

2

Пишем тесты

# Запуск тестов



1

Проверяем наши тесты в действии

2

Анализируем тестовые результаты

# Редактирование тестов



1

Пересматриваем и корректируем тесты

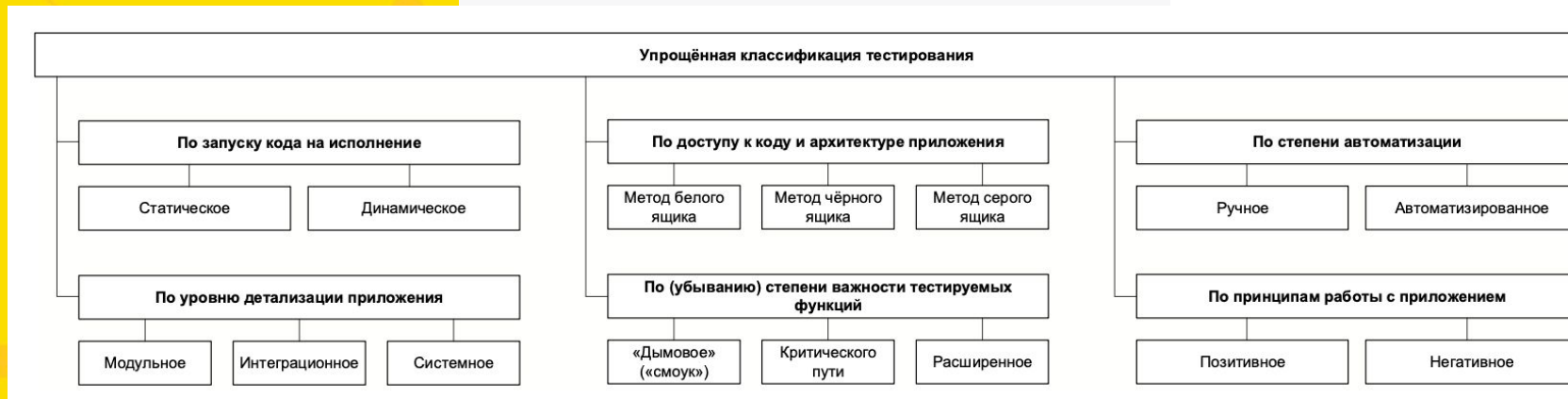
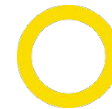
2

Держим тесты в актуальном состоянии

3

Дополняем новыми тестами

# Виды тестирования



# Методы тестирования

Мы работаем с кодом системы?

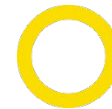
- НЕТ = метод «черного» ящика (black-box)
- Частично = метод «серого» ящика (grey-box)
- ДА = метод «белого» ящика (white-box)



# Регрессионное тестирование

Регрессионное тестирование — это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде. Регрессионными могут быть как функциональные, так и нефункциональные тесты.

# Регрессионное тестирование



- Проверяем всю систему в целом
- Получаем сведения о качестве ПО



# Приемочные испытания

Smoke Test – короткий цикл тестов, чтобы убедиться, что после сборки кода (нового или исправленного) приложение стартует и выполняет основные функции

Smoke Test – выполнение минимального набора тестов для выявления явных ошибок критичного функционала



# Поддержка и сопровождение



1

Тестирование багов, найденных в релизном продукте

2

Регрессионное тестирование билда после внесенных исправлений

# Уровни тестирования

Над чем проводим тесты? (модуль -> группа модулей -> система)

- Компонентное/модульное (component/unit testing)
- Интеграционное (integration testing)
- Системное (system testing)





## Типичные роли

- Руководитель проекта (Project Manager)
- Архитектор (Architect/Designer)
- Бизнес-аналитик (Business Analyst)
- Разработчик (Developer)
- Тестировщик (QA/Test Engineer)

# Waterfall

Переход от одной фазы к другой происходит только после полного и успешного завершения предыдущей



# Agile

Гибкая методология разработки (англ. гибкий) – серия подходов к разработке ПО, ориентированных на использование итеративной инкрементальной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля

Маленький, но  
жизнеспособный

## HOW TO BUILD A MINIMUM VIABLE PRODUCT

### NOT LIKE THIS



1



2



3



4

### LIKE THIS



1



2



3

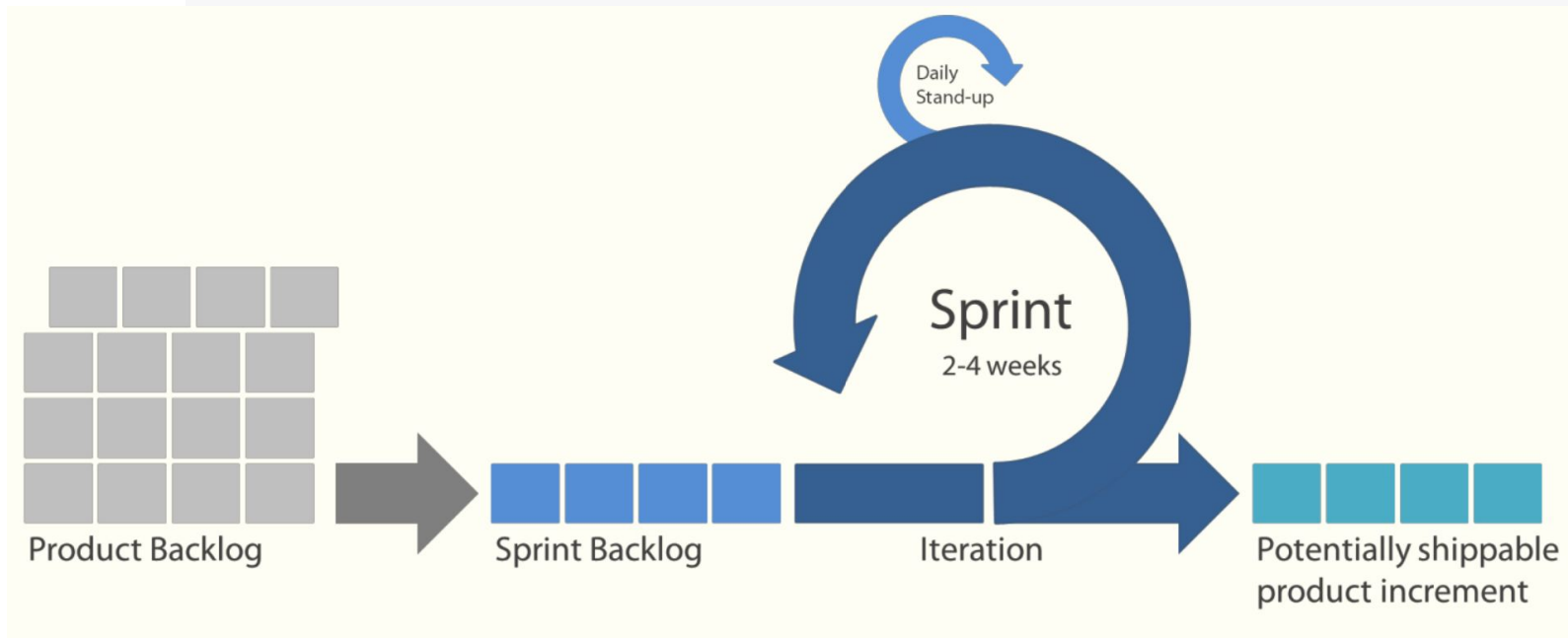


4



5

# Scrum. Process



# Agile. Основные Определения

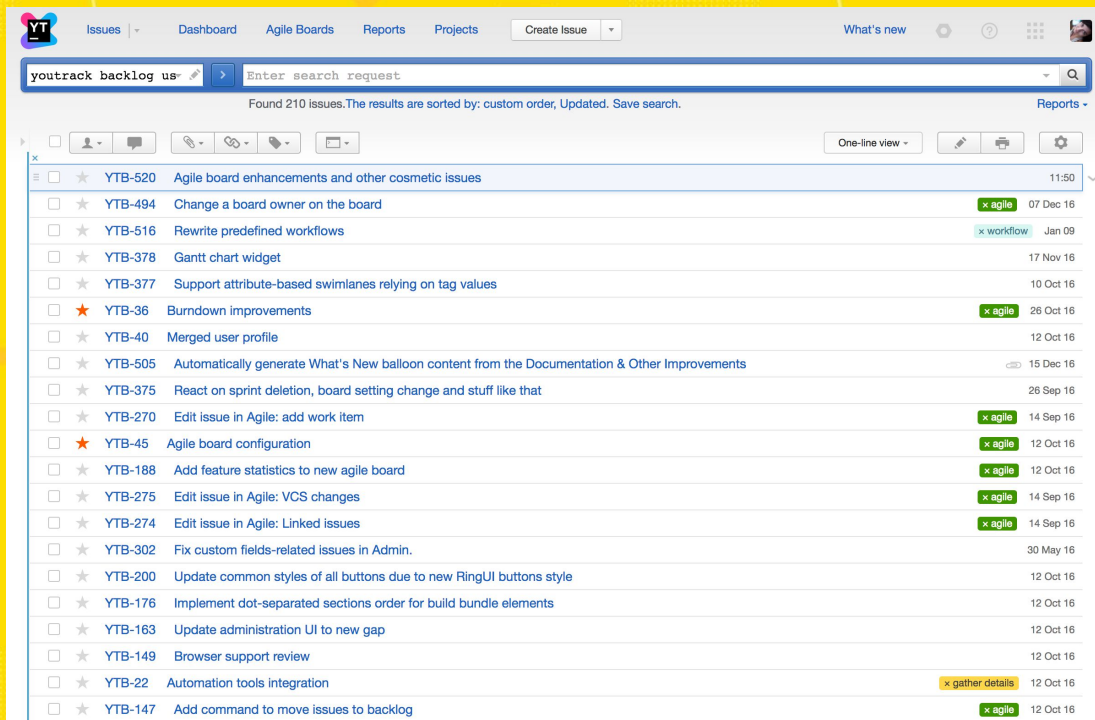
**Sprint** - это циклы выполнения задач. Их длительность может быть от одной до четырех недель. За это время команда должна создать потенциально готовый к работе продукт, чтобы пользователи могли его протестировать и поделиться впечатлениями. Спринт состоит из нескольких этапов: планирование, работа, презентация результата и анализ. Команда сама оценивает свои силы на один спринт и формирует столько задач, сколько сможет выполнить





# Agile. Основные Определения

**Product Backlog** - это приоритизированный набор пользовательских историй. В процессе этот список может обновляться всеми членами команды



# Agile. Основные Определения

**Sprint Backlog** - это список задач и объем работы, которую нужно сделать за один спринт. К примеру, за одну или две недели.

The screenshot displays the JIRA Software interface for a project named "Teams in Space". The main view is the "Backlog", which shows a list of issues organized into sprints. The left sidebar contains navigation options like "Backlog", "Agile board", "Releases", "Reports", "All issues", "Components", and "Add-ons". Below these are "PROJECT SHORTCUTS" including "Mars Team HipChat Room", "Space Station Dev Roadmap", "Teams in Space Org Chart", "Orbital Spotify Playlist", and "Hyperspeed Bitbucket Repo".

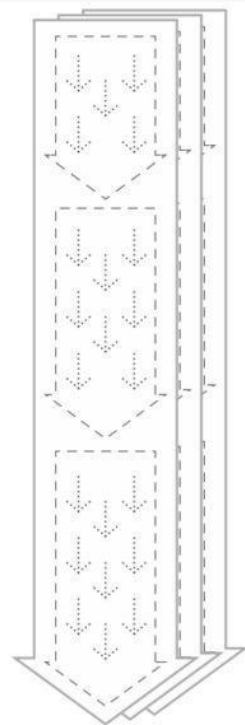
The main content area shows the "Backlog" with two sprints:

- Sprint 1**: 14 issues. Issues include:
  - TIS-25: Engage Jupiter Express for outer solar system travel (SeeSpaceEZ Plus)
  - TIS-37: When requesting user details the service should return prior trip info (Large Team Support)
  - TIS-9: After 100,000 requests the SeeSpaceEZ server dies (Local Mars Office)
  - TIS-7: 500 Error when requesting a reservation (Large Team Support)
  - TIS-10: Bad JSON data coming back from hotel API (Space Travel Partners)
  - TIS-18: Enable Speedy SpaceCraft as the preferred individual transit provider (Large Team Support)
- Sprint 2**: 6 issues. Issues include:
  - TIS-25: Engage Jupiter Express for outer solar system travel (Local Mars Office)
  - TIS-37: When requesting user details the service should return prior trip info (Space Travel Partners)
  - TIS-9: After 100,000 requests the SeeSpaceEZ server dies (Space Travel Partners)
  - TIS-7: 500 Error when requesting a reservation (Local Mars Office)

At the bottom, there is a "Backlog" section with 49 issues, showing the same set of issues as the sprints above.

# Agile. Основные Определения

**User Story** - способ  
описания требований к  
разрабатываемой  
системе,  
сформулированных как  
одно или более  
предложений на  
повседневном или  
деловом языке  
пользователя



"As a [persona],  
I want to [do something]  
so that I can [realize a reward]"

**Who is this user?**

**What makes them tick?**

**Who's an example of such a person?**

**Why do they want to do this?**

**What's the benefit/reward?**

**How will we know of it's  
working?**



# Agile. Основные Определения

**Grooming** – это тип встреч в Скраме, основная активность, которого делается на протяжении спринта для подготовки бэклога к следующему спринт-планированию.

Грумминг стоит проводить:

- активно в начале проекта или релиза
- затем регулярно (или по необходимости) в течение всего проекта, выделяя на эту активность время в каждом спринте



# Agile. Основные Определения

**Sprint Planning** - во время данного митинга Product Owner описывает наиболее приоритетные задачи команды. Команда в это время задаёт достаточное количество вопросов, чтобы более точно оценить и распределить задачи, которые будут решаться во время спринта

**Daily stand-up** (Daily Scrum, Kanban Meeting или просто дэйли) — регулярная короткая встреча Agile-команды, которая призвана синхронизировать всех участников, обеспечивать прозрачность рабочего процесса и таким образом поддерживать эффективность и производительность команды. Обычно задают 3 вопроса:

- Что сделал?
- Что буду делать?
- Какие блокирующие проблемы существуют?

# Agile. Основные Определения

**Sprint Demo** - митинг, основной целью которого является короткая демонстрация командой для Product Owner того как работает разработанная функциональность

**Retrospective** - митинг, на котором команда производит обзор того, что удалось сделать успешно во время спринта, какие проблемы возникали и как не допускать аналогичных ошибок/проблем в будущем



# Требования

Требования (requirements) - описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи. Требования должны быть независимы от внутренней архитектуры приложения, т.е. должны описывать то, что необходимо заказчику без деталей реализации (принцип «what, not how»). Правда, бывают исключения в виде ограничений, например, на операционную систему.

# Практическое задание

Изучить <https://sharelane.com> и написать к нему требования, например:

- Поле Zip Code должно состоять из 5 цифр
- Поля Last Name, Password и Confirm Password должны быть обязательными

