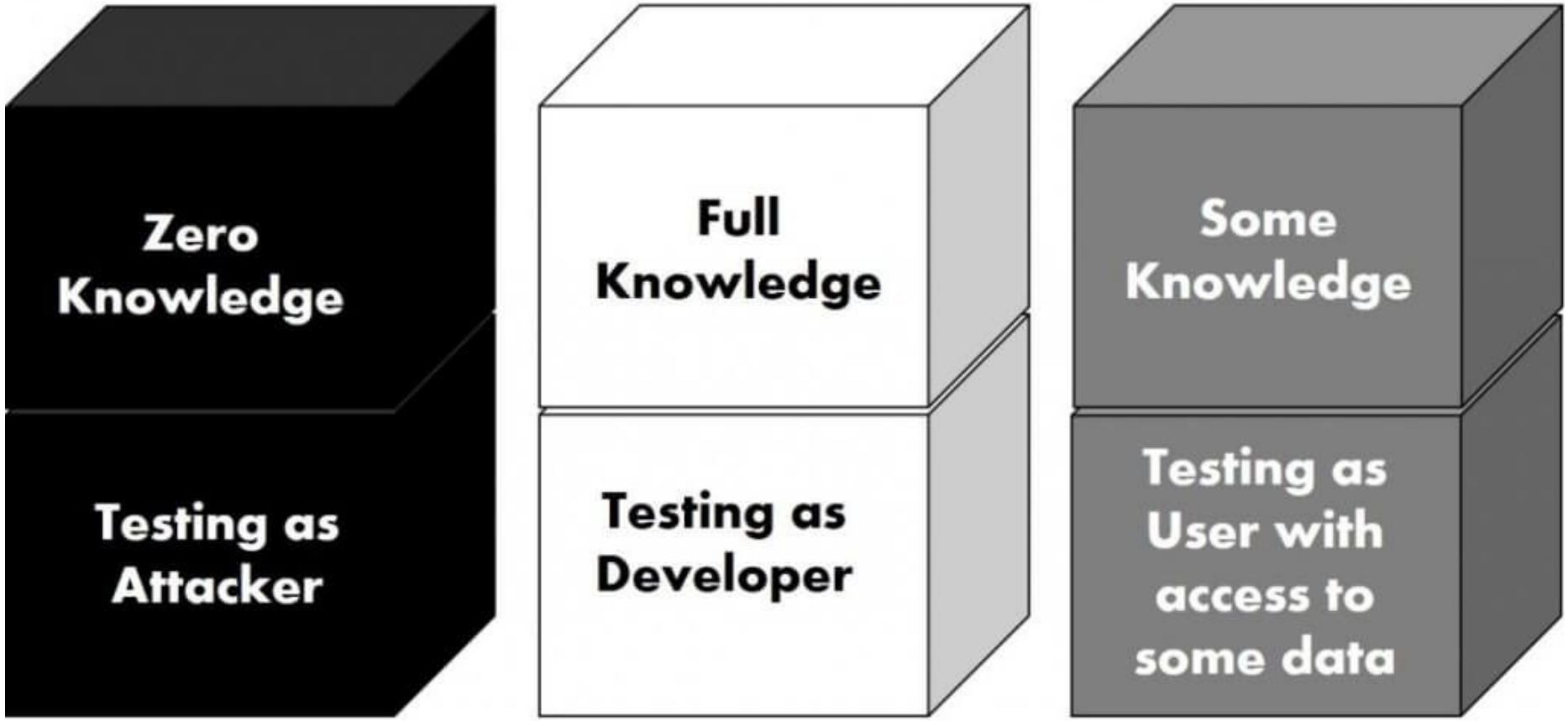
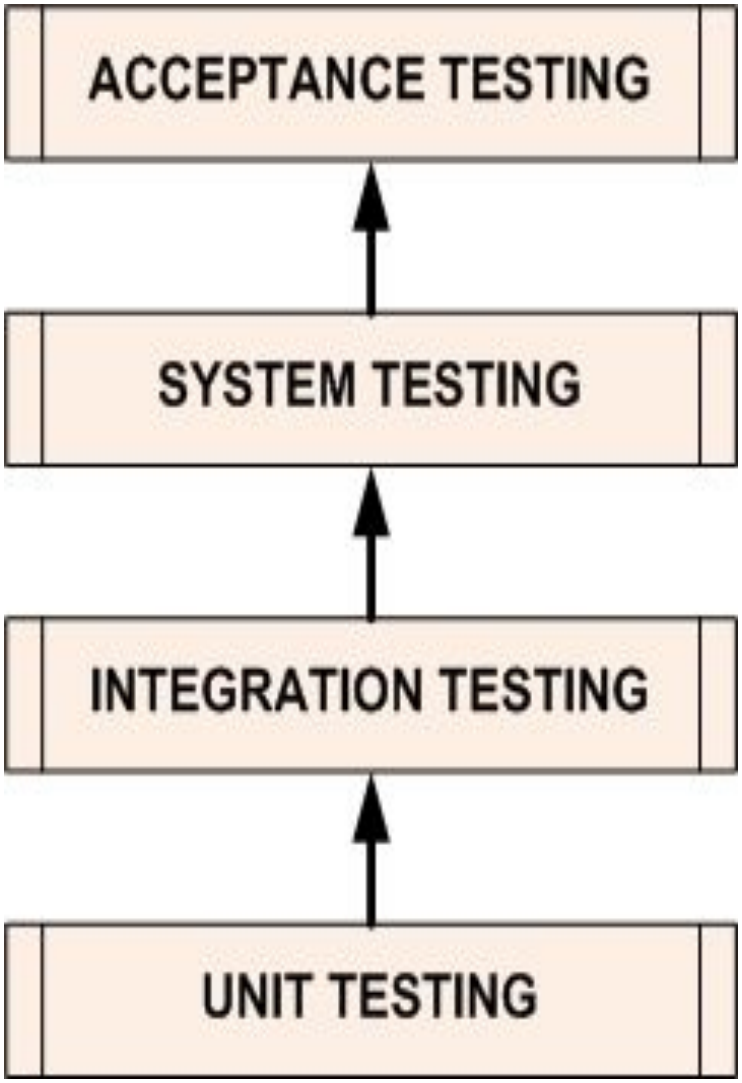
Классификации видов тестирования

- По доступу к коду (по знанию системы)
- По степени изолированности компонентов
- По степени автоматизации
- По степени подготовленности к тестированию
- По признаку +/- сценариев (по требованиям)
- По запуску кода
- По объекту тестирования

По доступу к коду (по знанию системы)		
Black box	**White box**	**Grey box**
Есть доступ к ПО только через интерфейсы, которые будут предоставлены заказчику.	Есть доступ к исходному коду и при выполнении тестов, мы как правило работаем с кодом. Пример: Unit testing.	Есть частичный доступ к исходному коду или БД, и при выполнении тестов мы можем более детальные и весомые проверки.


Zero Knowledge
Testing as Attacker
Full Knowledge
Testing as Developer
Some Knowledge
Testing as User with access to some data

По степени изолированности компонентов		
Unit testing	**Integration testing**	**System testing**
тестируются по отдельности небольшие блоки системы, максимально отделенные от других элементов и, в то же время, пригодные для тестирования.	тестируются объединенные компоненты системы, зачастую это некоторый блок взаимодействующих между собой элементов.	Тестируется система целиком на соответствие всем функциональным и нефункциональным требованиям.



Acceptance testing

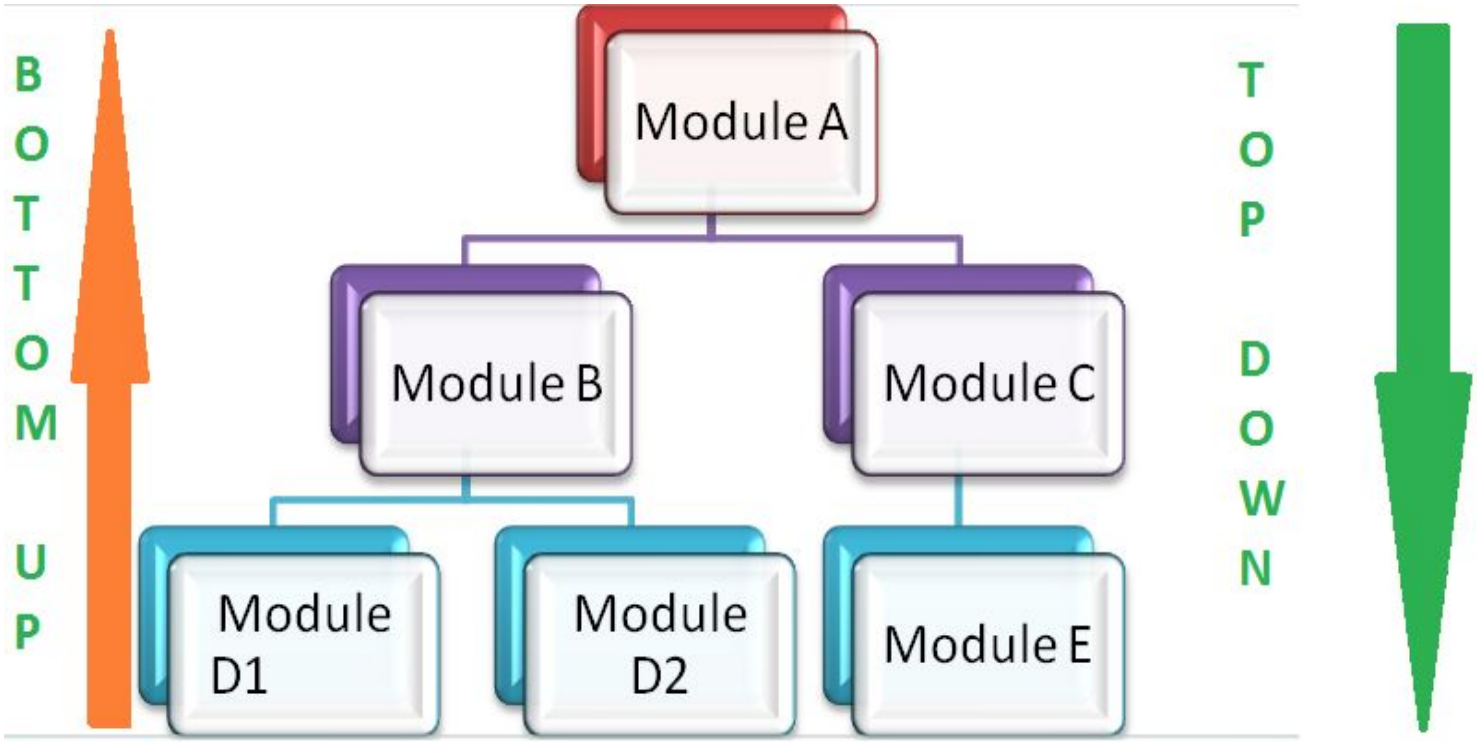
это тестирование готового продукта конечными пользователями на реальном окружении, в котором будет функционировать тестируемое приложение. Приемочные тесты разрабатываются пользователями, обычно, в виде сценариев.

Подходы к интеграционному тестированию:

Снизу вверх (Bottom Up Integration)
Все низкоуровневые модули, функции собираются воедино и затем тестируются. После чего собирается следующий уровень модулей для проведения интеграционного тестирования.

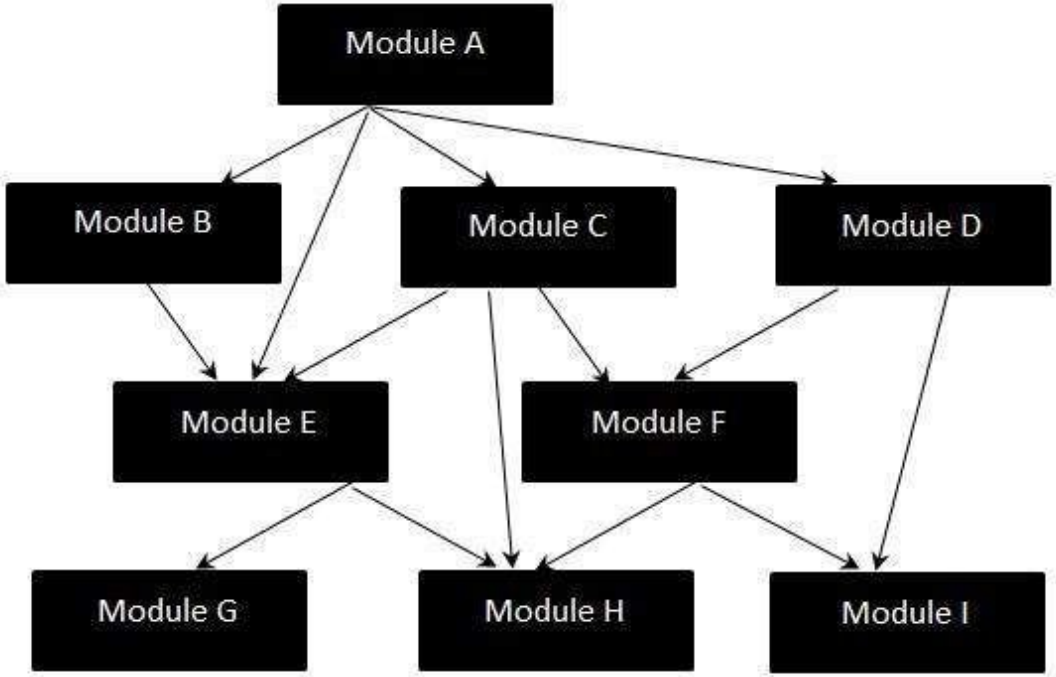
Сверху вниз (Top Down Integration)
Вначале тестируются все высокоуровневые модули, и постепенно один за другим добавляются низкоуровневые. Все модули более низкого уровня симулируются заглушками, и по мере готовности они заменяются реальными активными компонентами.

Высокоуровневые модули



Низкоуровневые модули

Большой взрыв ("Big Bang" Integration)
Все или практически все разработанные модули собираются вместе в виде законченной системы или ее основной части, и затем проводится интеграционное тестирование.



По степени автоматизации:		
Manual	Auto	Semi-auto

По признаку позитивности сценариев (по требованиям):	
Positive	Negative

По степени подготовленности к тестированию:	
By documentation	Ad-hoc (exploratory)

По запуску кода:	
Static	Dynamic

При статическом тестировании программный код не выполняется — анализ программы происходит на основе исходного кода, который вычитывается вручную, либо анализируется специальными инструментами. Пример: code-review.

Также к статическому тестированию относят тестирование требований, спецификаций, документации.

Остальное - это динамическое тестирование.

По объекту тестирования:	
Functional testing	Non-functional testing
Smoke testing Sanity testing New feature testing Regression testing Alpha testing Beta testing	Performance testing Load testing Stress testing Stability testing Volume testing UI testing Usability testing Compatibility Security Localization

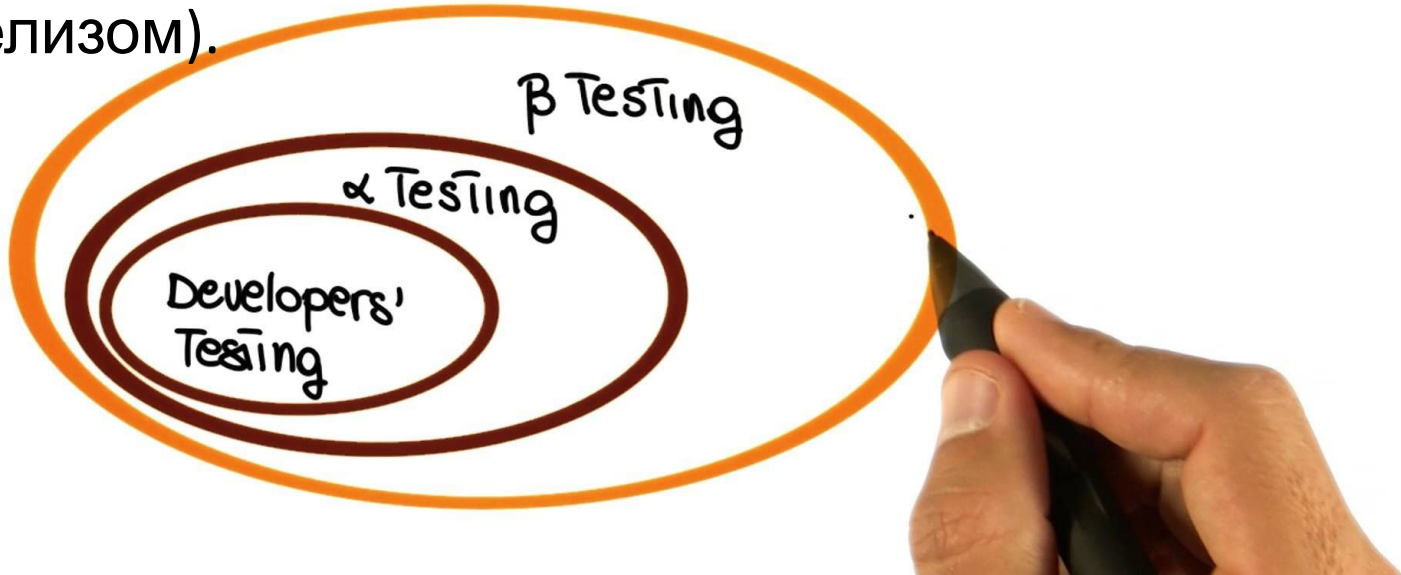
Smoke тестирование - это минимальный набор написанных тест-кейсов, определяющий, что билд готов к передаче в тестирование. Цель для команды тестирования – не нахождение дефектов, а убедиться, что вся функциональность работает стабильно и готова к тестированию. Занимает от 15 минут до 2х часов. Если не работают элементарные вещи, то билд отдают на доработку. Можно использовать средства автоматизации.

Sanity (bug-fix) testing заключается в том, чтобы проверить только исправленные дефекты, изменения из баг-трекинговой системы. Сосредоточен на узкой части функциональности.

New feature testing – это по сути модульное тестирование нового функционала, когда мы проверяем что новый функционал работает в соответствии с заявленными требованиями.

Альфа тестирование - это тестирование, обычно проводимое на ранней стадии разработки продукта и включающее имитацию реального использования продукта штатными разработчиками либо его реальное использование потенциальными клиентами.

Бета-тестировании - интенсивное использование почти готовой версии продукта с целью выявления максимального числа ошибок в его работе для их последующего устранения перед окончательным выходом (Релизом).



Regression testing – повторное тестирование после внесение изменений в программное обеспечение или в его окружение (в новой версии приложения), чтобы убедиться, в том, что функции, которые работали в предыдущей версии системы, по-прежнему работают так, как ожидалось.

99 маленьких багов в коде,
99 маленьких багов в коде,
Один нашли, пофиксили.

127 маленьких багов в коде

Тестирование производительности – тестирование поведение системы при различных нагрузках и при различных сценариях использования.

Основные виды тестирования производительности:

- Stress testing
- Load testing
- Stability testing
- Volume testing

Стрессовое тестирование (Stress testing) – проверка системы при пиковых нагрузках, ограниченных ресурсах и восстановление после возвращению к нормальному состоянию.

Нагрузочное тестирование (Load testing) - проверка систем на различных уровнях нагрузки. Определяем, при какой максимальной нагрузке (максимальном количестве пользователей) система способна функционировать в соответствии с требованиями к производительности.

Тестирование стабильности (Stability testing) - оценка работоспособности системы при длительной нагрузке. Главная задача - выявить утечки памяти или другие проблемы, которые не позволяют системе стабильно работать.

Объемное тестирование (Volume testing) - тестирование проводится с увеличением не нагрузки и времени работы, а количества используемых данных, которые хранятся и используются в приложении.

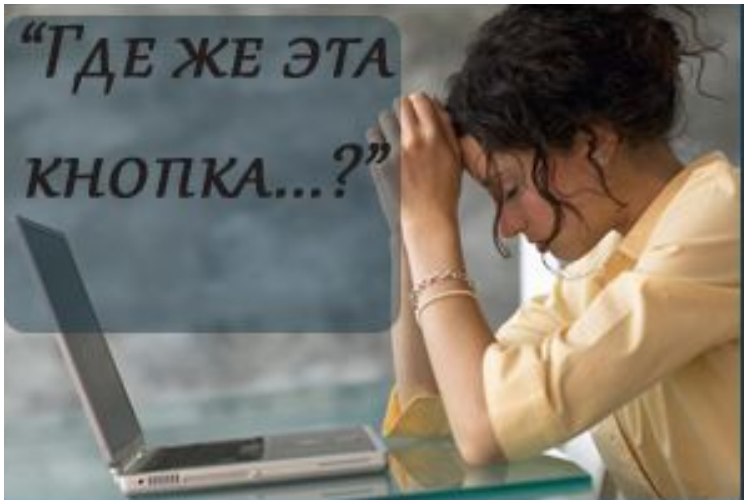
При тестировании производительности нас интересует:

- изменение времени выполнения операций в зависимости от интенсивности операций (где интенсивность операций = кол-во пользователей * кол-во операций * единицу времени).
- определение границы приемлемой производительности (где приемлемая производительность - это либо четко прописанное в ТЗ среднее время отклика системы, либо такая скорость работы, когда уже с приложением нормально работать невозможно).
- определение количества пользователей, которые могут одновременно работать с приложением.

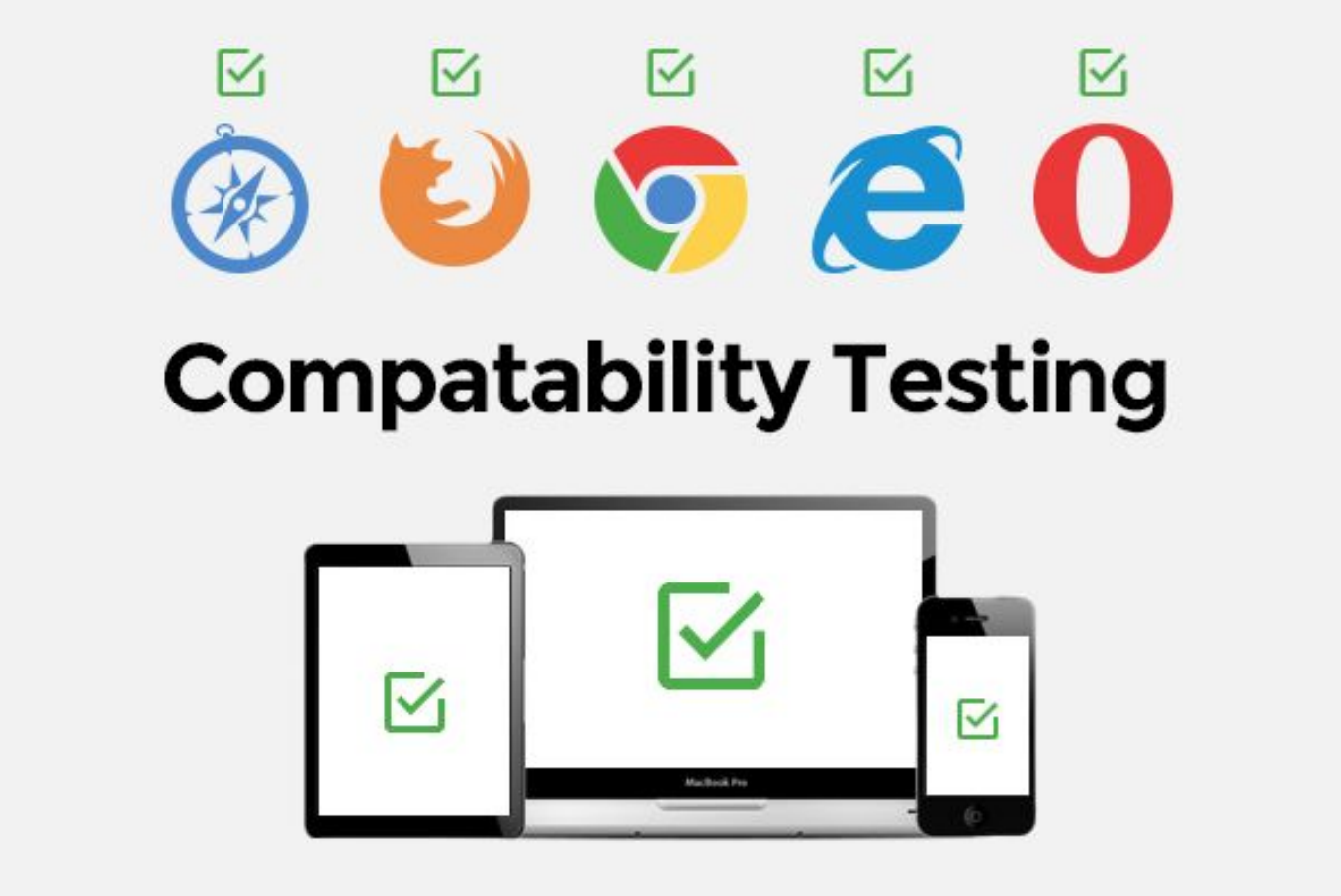
Тестирование интерфейса пользователя (UI testing) - тестирование графического интерфейса пользователя для того, чтобы убедиться, что он соответствует принятым стандартам и их требованиям.

Тестирование удобства использования (Usability testing) - тестирование, определяющее, насколько продукт отвечает требованиям той аудитории, для которой он пишется.

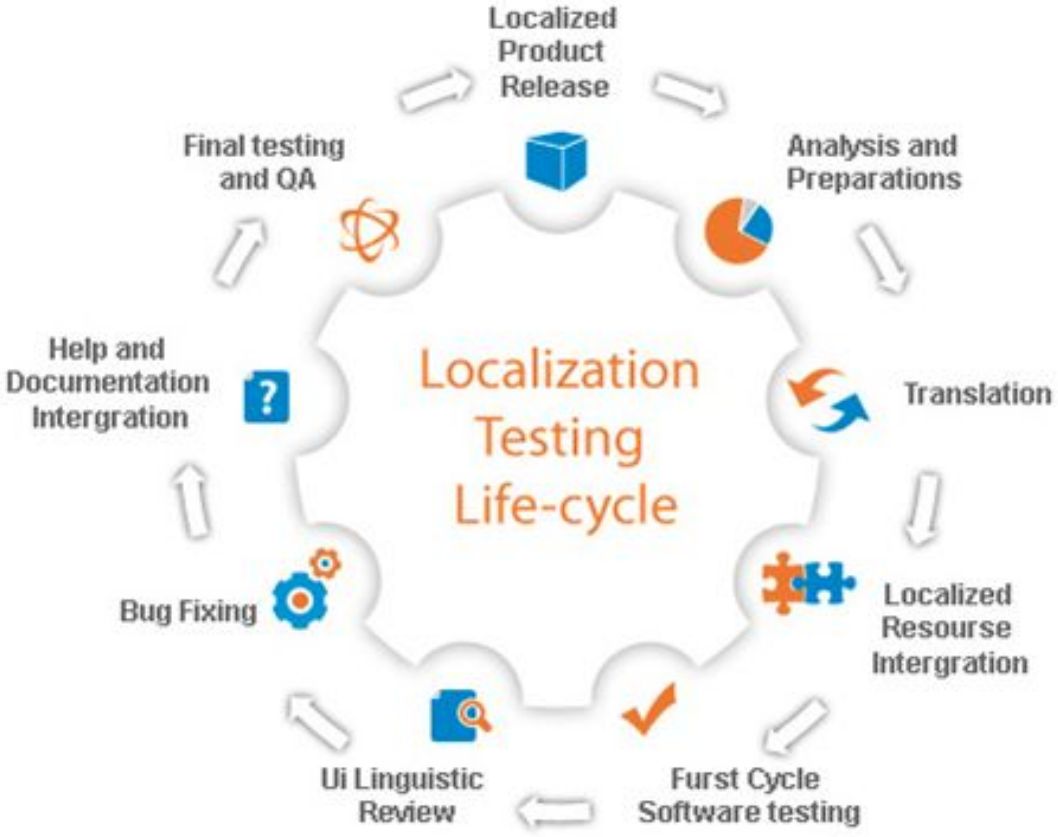
Обычно результатом выполнения UI и Usability тестов является список рекомендаций и предложений по улучшению.



Тестирование совместимости (compatibility testing) - проверить, что приложение совместимо с определенными конфигурациями оборудования, операционными системами, базами данных, брау



Localization testing - проверяет, правильно ли локализован продукт. То есть, переведен на другой язык и корректно работает с учетом национальных особенностей страны или региона.



2.3 Методологии разработки ПО

Модель жизненного цикла программного обеспечения - структура, содержащая процессы действия и задачи, которые осуществляются в ходе разработки, использования и сопровождения программного продукта.

- Водопад или каскадная модель
- Водоворот или каскадная с промежуточным контролем
- V модель - разработка через тестирование
- Спиральная модель
- Итеративная модель
- Семейство Agile: Scrum, XP, Kanban

"Водопад" или каскадная модель

"Водопад" или каскадная модель

Модель предусматривает последовательное выполнение всех этапов проекта в строго фиксированном порядке. Переход на следующий этап означает полное завершение работ на предыдущем этапе. Требования, определенные на стадии формирования требований, строго документируются в виде технического задания и фиксируются на все время разработки проекта. Каждая стадия завершается выпуском полного комплекта документации, достаточной для того, чтобы разработка могла быть продолжена другой командой разработчиков.

"Водоворот" или каскадная модель с промежуточным контролем - в этой модели предусмотрен промежуточный контроль за счет обратных связей.

V модель - разработка через тестирование которая предполагает регулярное тестирование продукта во время разработки.

Особенности V модели:

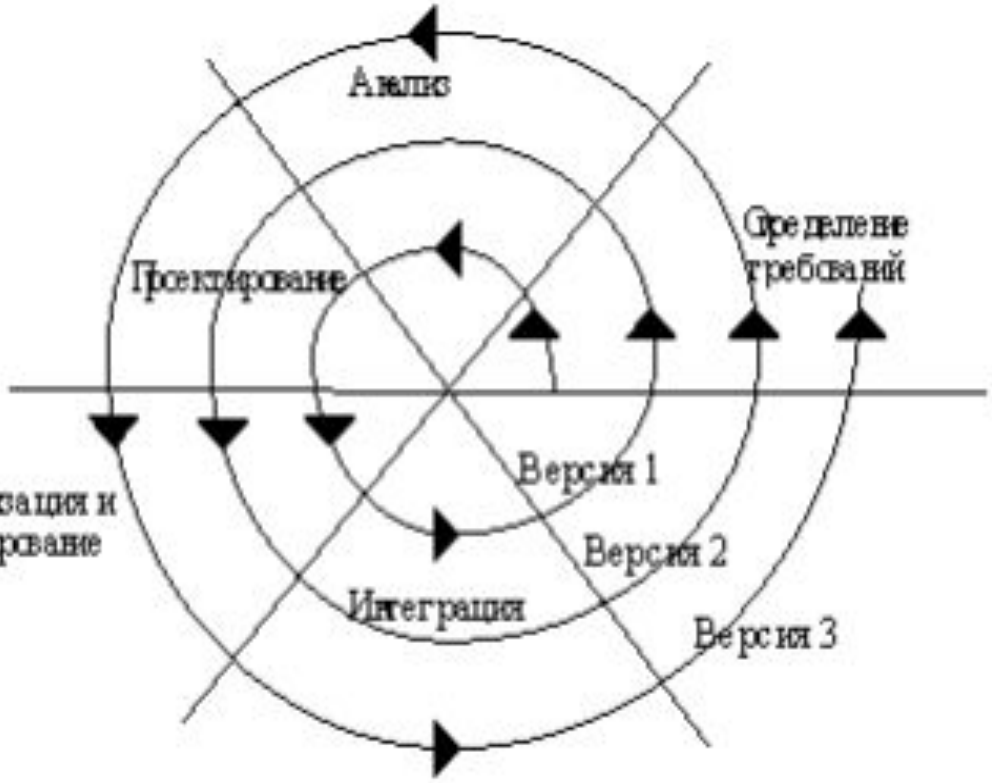
• детализация проекта возрастает при движении слева направо, одновременно с течением времени, и ни то, ни другое не может повернуть вспять

• приемо-сдаточные испытания основываются, прежде всего, на требованиях, системное тестирование — на требованиях и архитектуре, комплексное тестирование — на требованиях, архитектуре и интерфейсах, а компонентное тестирование — на требованиях, архитектуре, интерфейсах и алгоритмах

Спиральная модель

Общая идея спирального процесса заключается в том, чтобы на каждой итерации строить очередную версию программы, используя в качестве основы ее предыдущую версию.

Включает в себя постадийное проектирование и прототипирование.

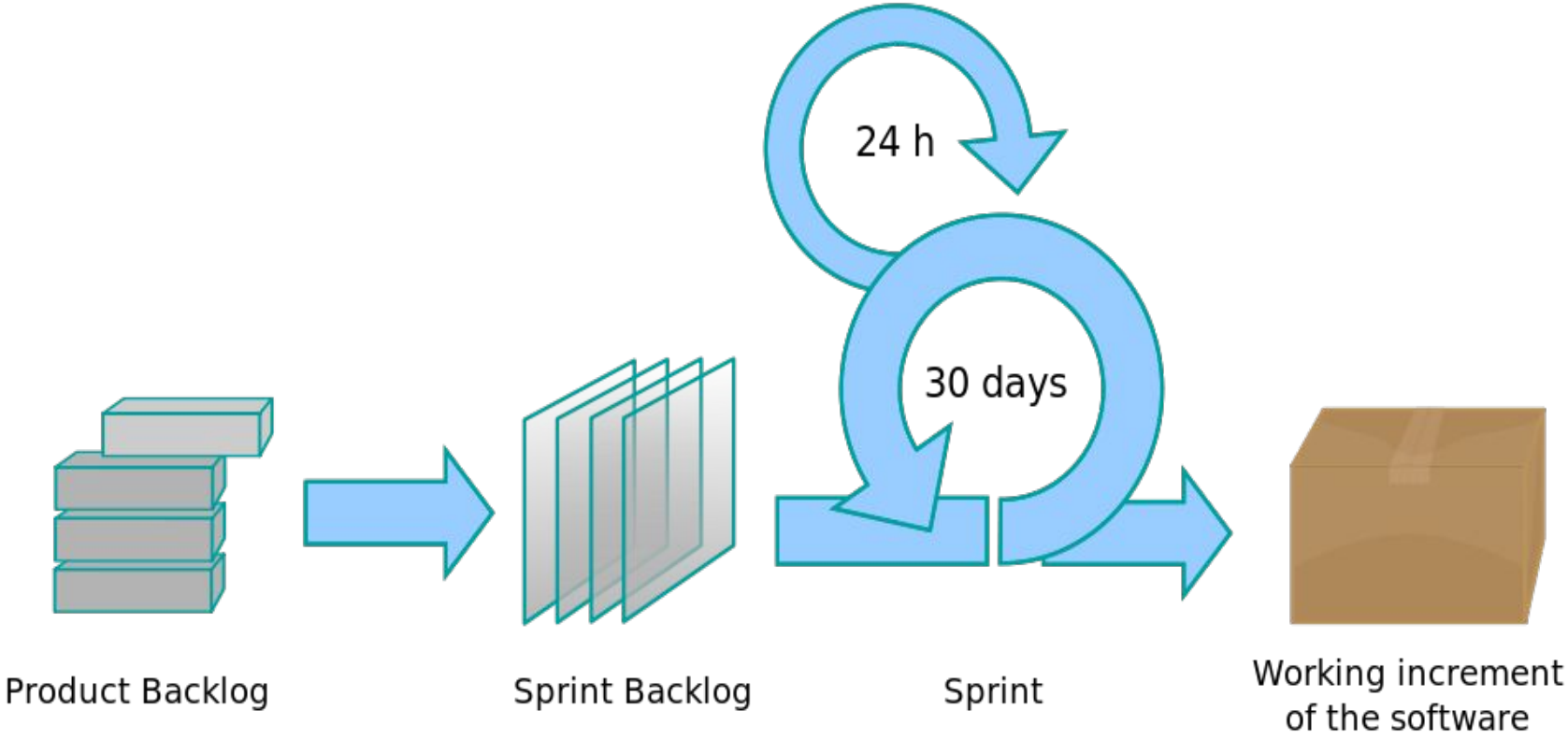


Итеративная разработка:
Итеративный подход — это выполнение работ параллельно с непрерывным анализом полученных результатов и корректировкой предыдущих этапов работы. Проект при этом подходе в каждой фазе развития проходит повторяющийся цикл PDCA (Планирование – Реализация – Проверка - Оценка).

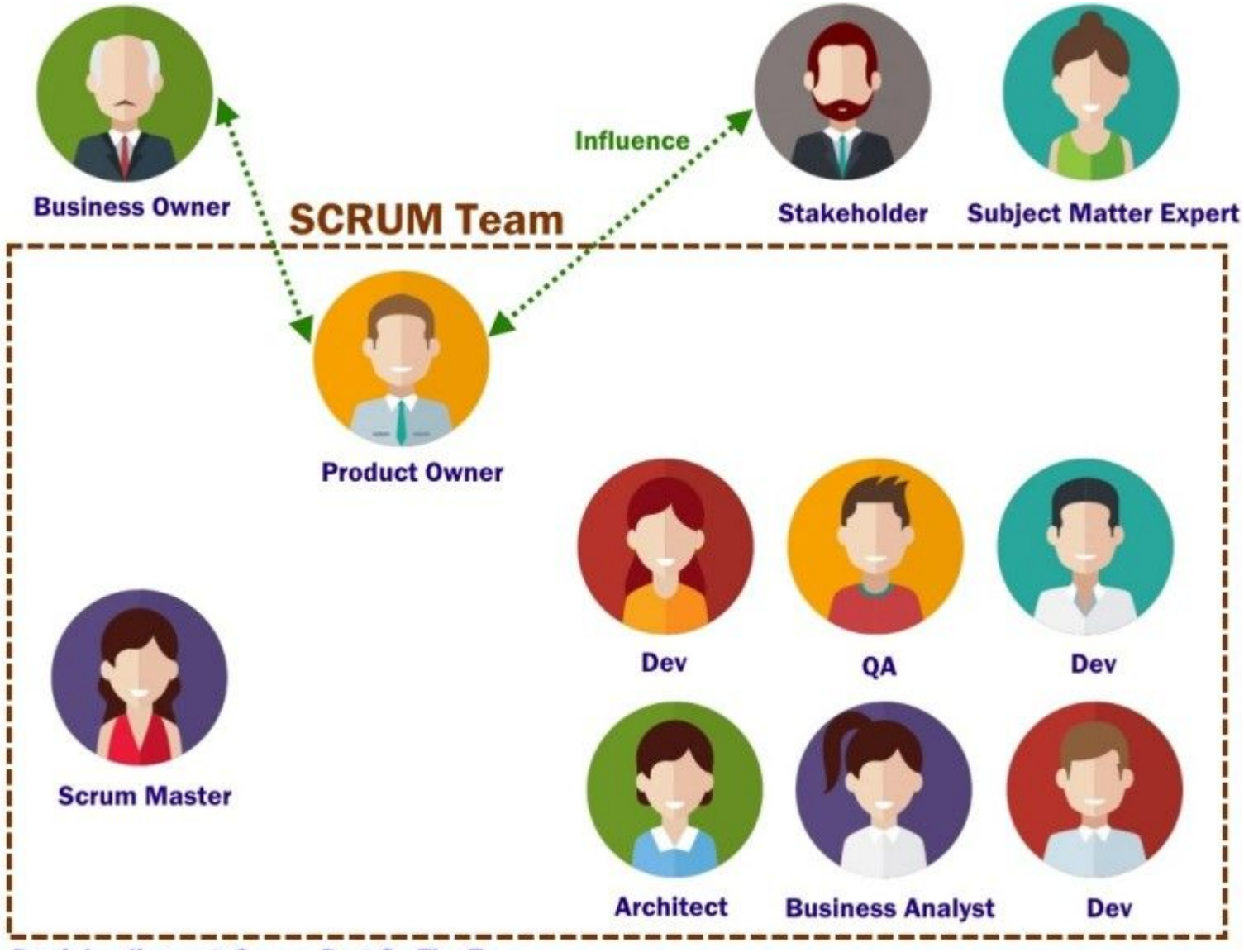
Agile – семейство гибких методологий разработки.

- Люди и взаимодействие важнее процессов и инструментов
- Работающий продукт важнее исчерпывающей документации
- Сотрудничество с заказчиком важнее согласования условий контракта
- Готовность к изменениям важнее следования первоначальному плану

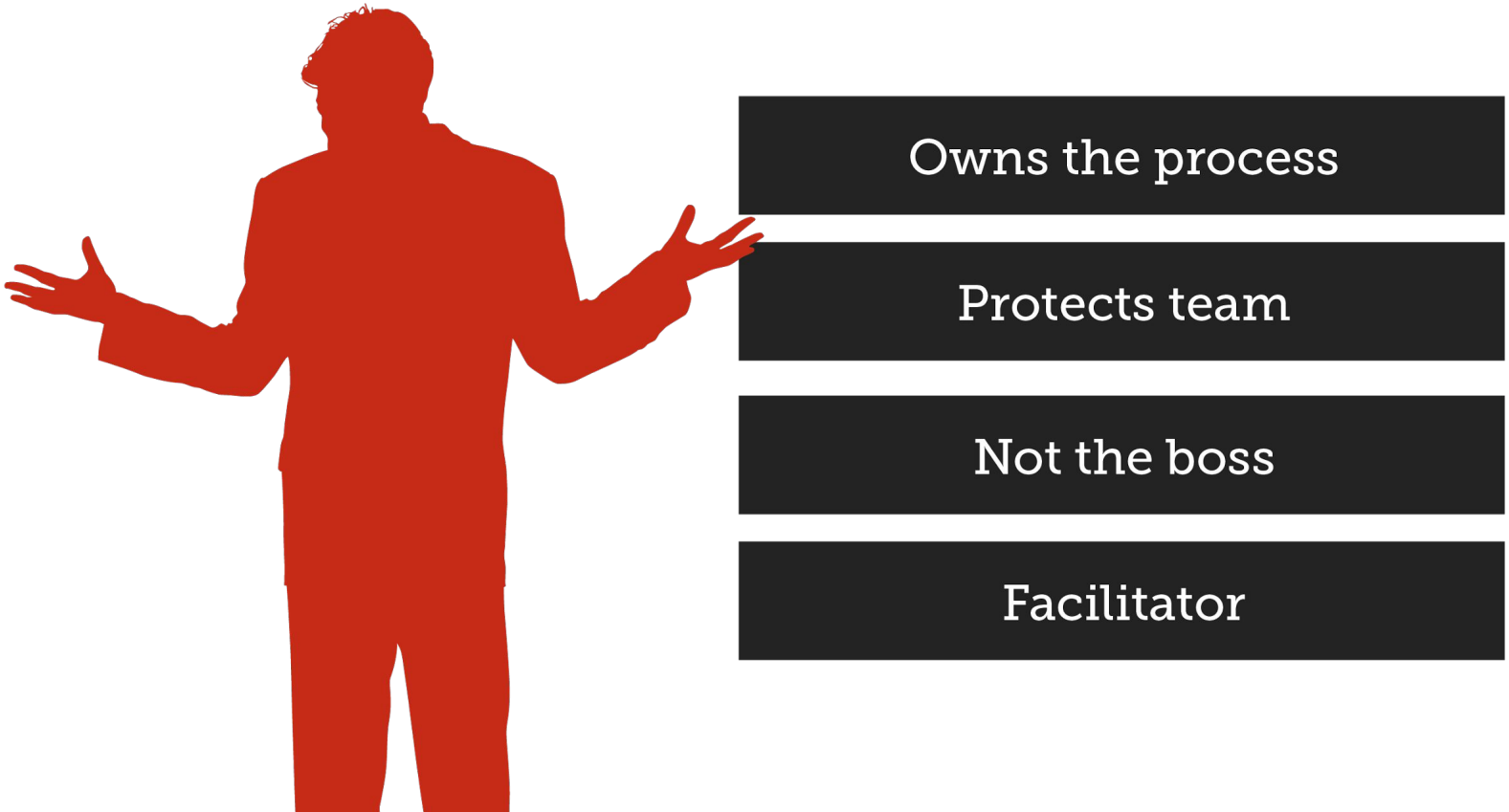
Scrum - одна из самых популярных методологий гибкой разработки. Одна из причин ее популярности - простота.



В Scrum всего три роли: Scrum Master, Product Owner, Team



Скрам Мастер (СМ) - отвечает за успех Scrum в проекте. По сути, СМ является интерфейсом (посредником) между менеджментом и командой. В Agile команда самоорганизующаяся и самоуправляемая.



Product Owner - это человек, отвечающий за разработку продукта. Как правило, это product manager для продуктовой разработки, менеджер проекта для внутренней разработки и представитель заказчика для заказной разработки. Единая точка принятия окончательных решений для команды в проекте.

- Voice of the customer
- Owns value
- Gathers feedback
- Makes decisions

Обязанности команды (7 +/- 2):

- ✓ Отвечают за оценку элементов бэклога
- ✓ Принимают решение по дизайну и имплементации
- ✓ Разрабатывают софт и предоставляют его заказчику
- ✓ Отслеживают собственный прогресс
- ✓ Отвечают за результат перед Product Owner

Commits to the work

Swarm on high value tasks

Has skills to deliver

Aims to be cross-functional

Особенности Scrum

- **Product Backlog** - приоритезированный список бизнес-требований и технических требований к системе. Данный документ постоянно обновляется - в него включаются новые требования, удаляются ненужные, пересматриваются приоритеты. За Product Backlog отвечает Product Owner.

Обычно backlog состоит из User Stories следующего формата:

- As a <role>, I want <goal/desire> so that <benefit>
- As a <role>, I want <goal/desire>
- In order to <benefit> as a <role>, I want <goal/desire>

User Stories

Stories

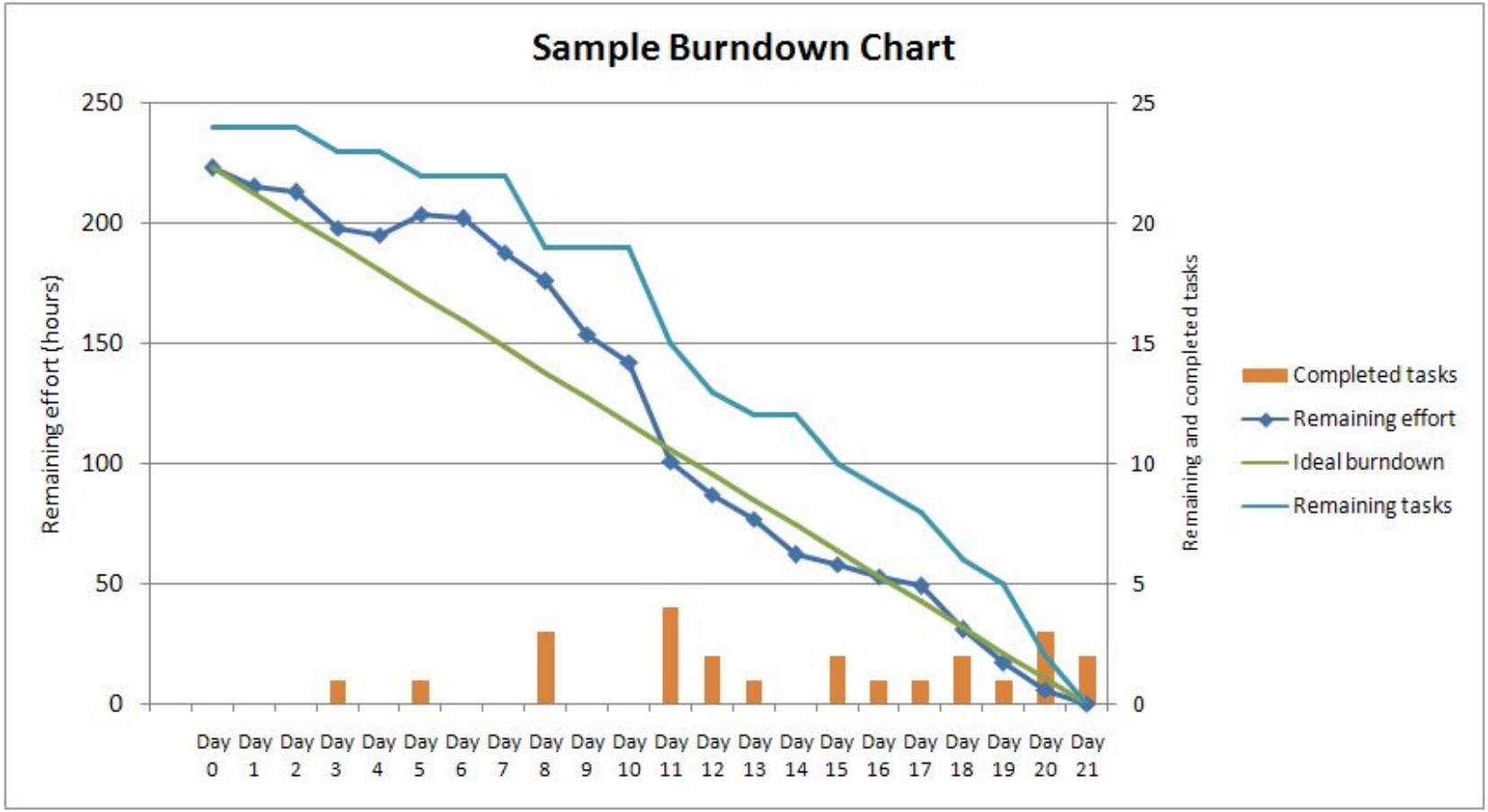
- As a thirsty person I want water to quench my thirst
- As a fashionable person I want umbrellas to make my drink look good
- As a thirsty person I would like lemon for added refreshment
- As a thirsty person I want a glass to hold the water in
- As a fashionable person I would like a straw to make me look cooler

Особенности Scrum

- **Sprint Backlog** - содержит функциональность, выбранную Product Owner на итерацию из Product Backlog. В Scrum итерация называется **Sprint** длительностью 2-4 недели. Результатом Sprint является готовый продукт (build), который можно передавать (deliver) заказчику (по крайней мере, система должна быть готова к показу заказчику). В течение спринта делаются все работы по сбору требований, дизайну, кодированию и тестированию продукта. Планирование спринта происходит в начале новой итерации, где выбираются задачи, обязательства по выполнению которых за спринт принимает на себя команда. При этом никто не может менять список задач утвержденный на Sprint.

Особенности Scrum

- **Burn-down diagram** - диаграмма, показывающая количество сделанной и оставшейся работы.



Особенности Scrum

- Daily Scrum meeting – ежедневное совещание, которое, длится не более 15 минут. В течение совещания каждый член команды отвечает на 3 вопроса:

- Что сделано с момента предыдущего совещания?
- Что будет сделано до следующего совещания?
- Какие проблемы мешают достижению целей спринта?

Особенности Scrum

Retrospective meeting - проводится после завершения спринта. Члены команды высказывают своё мнение о прошедшем спринте. Отвечают на два основных вопроса:

– Что было сделано хорошо в прошедшем спринте?

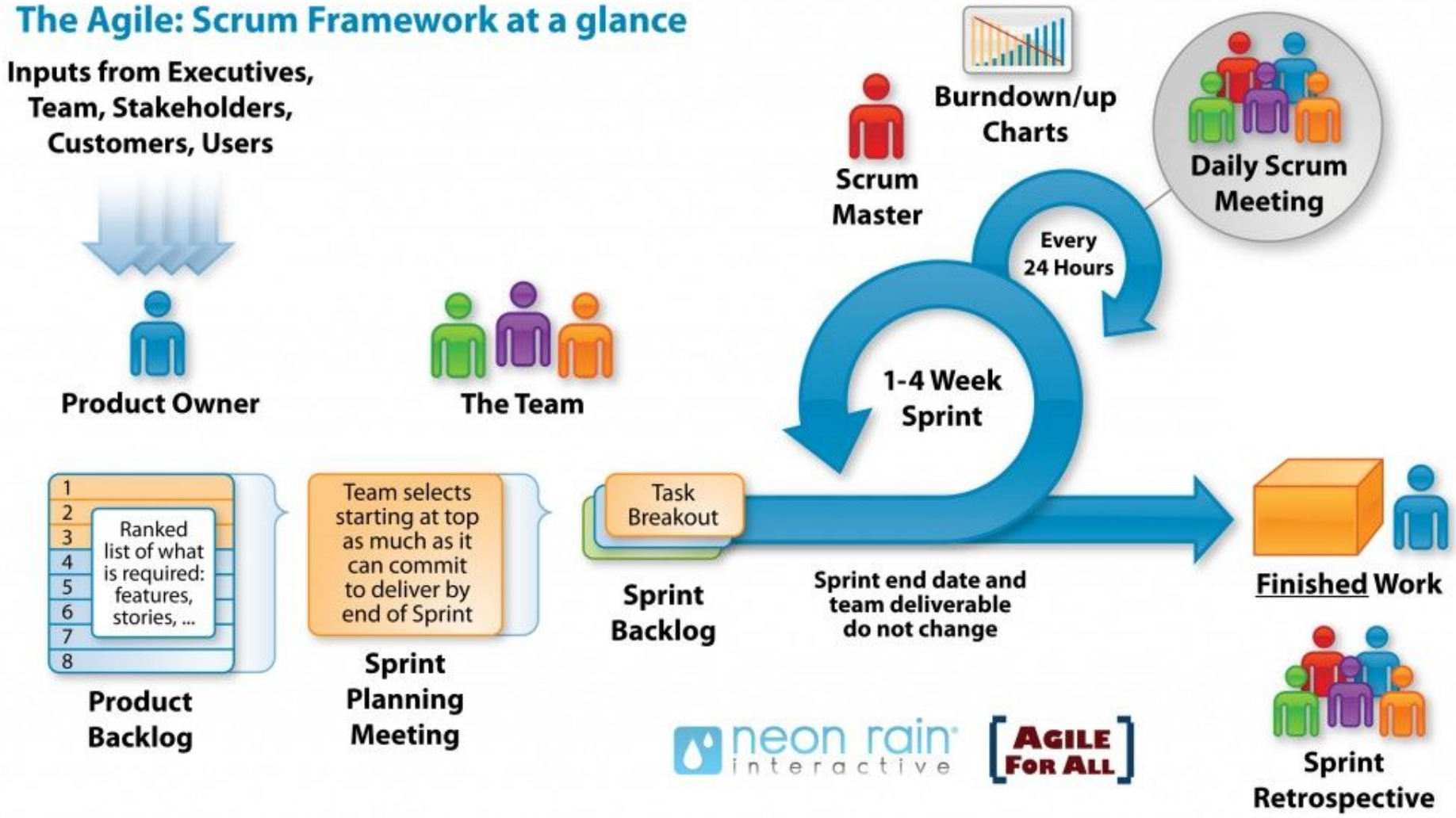
– Что надо улучшить в следующем?

В процессе митинга решают вопросы и фиксируют удачные решения. Совещание ограничено одним-тремя часами.

Особенности Scrum

Planning Poker / Scrum poker — техника оценки, используемая для оценки сложности предстоящей работы или относительного объёма решаемых задач при разработке программного обеспечения.

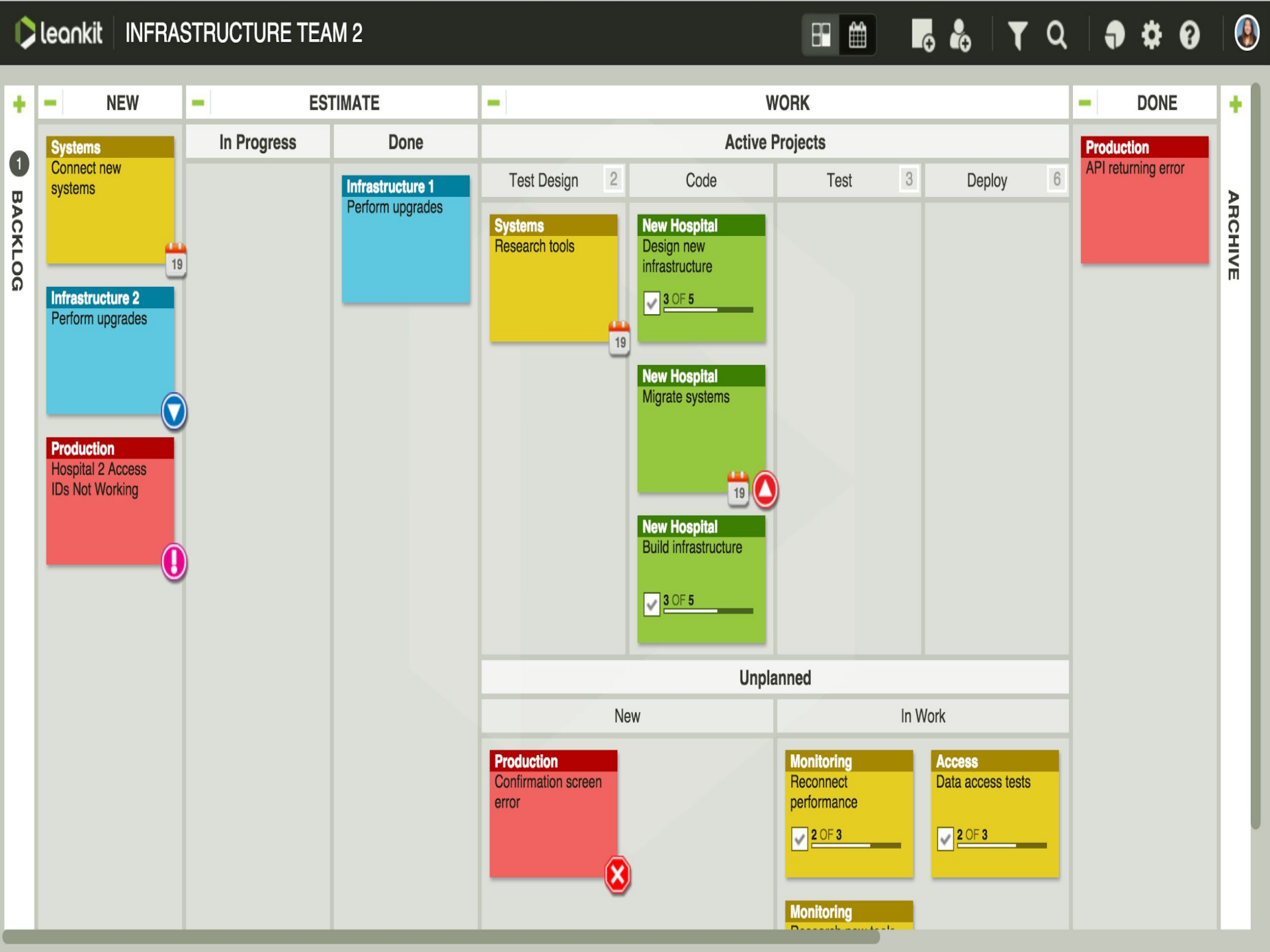
The Agile: Scrum Framework at a glance



Kanban - одна из разновидностей управления разработкой программного обеспечения. Перспективный вариант для аутсорсинговых компаний и фрилансеров, работающих с большим количеством заказов.

Особенности Kanban:

- Визуализация разработки
- Отметки о положении задач в разработке
- Ограниченное количество работ на каждом этапе
- Измерение времени цикла
- Оптимизация процесса
- Видно состояние проекта


leankit
INFRASTRUCTURE TEAM 2
BACKLOG
1
NEW
ESTIMATE
WORK
DONE
ARCHIVE
Systems
Connect new systems
19
Infrastructure 2
Perform upgrades
Production
Hospital 2 Access IDs Not Working
In Progress
Done
Infrastructure 1
Perform upgrades
Active Projects
Test Design 2
Code
Test 3
Deploy 6
Systems
Research tools
19
New Hospital
Design new infrastructure
3 OF 5
New Hospital
Migrate systems
19
New Hospital
Build infrastructure
3 OF 5
Unplanned
New
In Work
Production
Confirmation screen error
Monitoring
Reconnect performance
2 OF 3
Access
Data access tests
2 OF 3
Monitoring
Production
API returning error

Scrum	Kanban
Итерации ограниченные по времени	50 / 50
Фиксированный объем задач на итерацию	50 / 50
Кросс-функциональные команды	50 / 50
Мелкие задачи в спринте	50 / 50
Burn-down diagram	50 / 50
Оценки задач обязательны	50 / 50
Нельзя добавлять задачи в спринт	Можно добавлять
3 обязательные роли	Нет предписанных ролей
Приоритезированный backlog	50 / 50

1 Андрей Горбачук - капля воды
2 Анатолий Кононович - книжная полка
3 Мария Матасова - комната
4 Алена Некрасова - комп. стол
5 Наталья Петренко - лампочка
6 Егор Савелов - маркер
7 Константин Слюсар - м/п окно
8 Дима Солоид - мусорное ведро
9 Эрнест Степанян - облако (в небе)
10 Наталья Федорова - кусочек сахара
11 Владислав Юрима - утюг
12 Дмитрий Козачко - электрочайник
13 Юлия Козачко - купюра номиналом 100$
14 Евгений Рябченко - табурет