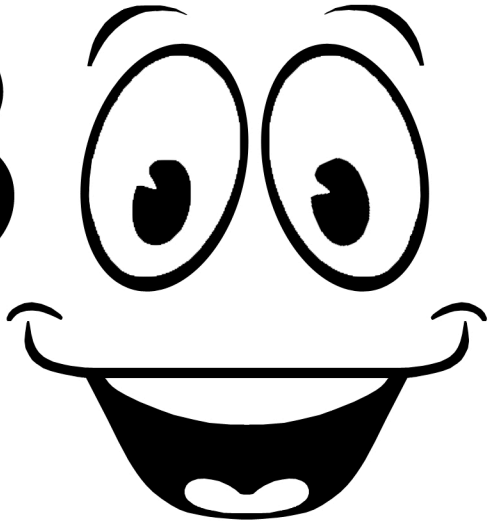
QA Light

HELP BOOK



СТРУКТУРА

- ВВЕДЕНИЕ
- QC
- ТЕСТИРОВАНИЕ ПО
- QA
- АРТЕФАКТЫ ТЕСТИРОВАНИЯ
- РАЗРАБОТКА ПРОГРАММНОГО ОЕСПЕЧЕНИЯ
- WATERFALL
- V-MODEL
- AGILE
- AGILE МАНИФЕСТ
- ЦЕННОСТИ AGILE
- SCRUM
- АРТЕФАКТЫ AGILE
- ТЕСТИРОВАНИЕ ПО
- КОМАНДА
- ТРЕБОВАНИЕ
- КРИТЕРИИ АНАЛИЗА ТРЕБОВАНИЙ
- МЕТОДЫ ТЕСТИРОВАНИЯ ТРЕБОВАНИЙ
- ВИДЫ ТЕСТОВ
- УРОВНИ ТЕСТИРОВАНИЯ
- ОШИБКИ НА УРОВНЯХ ТЕСТТИРОВАНИЯ
- ВИДЫ ТЕСТОВ
- ЦИКЛ ТЕСТИРОВАНИЯ
- РОЛИ В ПРОЦЕССЕ УПРАВЛЕНИЯ ДЕФЕКТОМ

СТРУКТУРА

- ЖИЗНЕНЫЙ ЦИКЛ ДЕФЕКТА
- ЖИЗНЕНЫЙ ЦИКЛ ДЕФЕКТА ПО РОЛЯМ
- ТИПЫ НЕИСПРАВНОСТЕЙ
- УЩЕРБ ОТ ОШИБКИ | ПРИОРИТЕТ ОШИБКИ
- ОПРЕДЕЛЕНИЕ ДЕФЕКТА
- ОТЧЕТ О ДЕФЕКТЕ
- ХОРОШИЙ ОТЧЕТ О ДЕФЕКТЕ
- ТЕСТ ПЛАН
- TEST CASE
- ТЕХНИКИ ТЕСТ ДИЗАЙНА
- ГРАНИЧНЫЕ УСЛОВИЯ | КЛАССЫ ЭКВИВАЛЕНТНОСТИ
- МЕТОД (ВСЕХ) ПАР
- ТЕСТОВОЕ ПОКРЫТИЕ
- МАТРИЦА ПОКРЫТИЯ
- ESTIMATE TIME
- СОБЕСЕДОВАНИЕ
- ПЕРВЫЙ ДЕНЬ
- КАЧЕСТВА ТЕСТИРОВЩИКА
- ИНСТРУМЕНТЫ ТЕСТИРОВЩИКА

ВВЕДЕНИЕ

Quality Control (контроль качества)	**Quality Assurance (обеспечение качества)**
это процесс проверки качества каждой сущности и всех факторов, влияющих на процесс разработки	это ряд мероприятий, направленных на то, что все процессы и требования к качеству разрабатываемого продукта выполняются в полной мере

Quality Assurance гарантирует, что процесс поставлен правильно и дает предсказуемый результат, в то время как Quality Control гарантирует, что продукт удовлетворяет указанному набору требований.

Тестирование - процесс исследования ПО с целью получения информации о качестве продукта

Тестирование ПО - процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом И который осуществляется специально подготовленными QC/QA инженерами.

АРТЕФАКТЫ ТЕСТИРОВАНИЯ

План тестирования (Test Plan)	- это документ описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения
Тест кейс (Test Case)	- это последовательность действий, по которой можно проверить соответствует ли тестируемая функция установленным требованиям.
Набор тест кейсов и тестов (Test set & Test suite)	- Это набор тест кейсов(тест кейсов) обьеденнёных в категорию по определённым признакам
Дефекты / Баг Репорты (Bug Reports / Defects)	- это документы, описывающие ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.
Матрица покрытия (traceability matrix)	Таблица, содержащая отношение требований к подготовленным тестовым сценариям (Test Cases)
Чеклист (Check-list)	Таблица создана для ускорения процесса тестирования. Имеющая расширенное краткое описание теста и место для отметки о его выполнения.

РАЗРАБОТКА ПРОГРАММНОГО ОЕСПЕЧЕНИЯ

ЖИЗНЕНЫЙ ЦИКЛ ПРОЭКТА

Инициация → Планирование → Выполнение → Контроль и мониторинг → Завершение

sprint

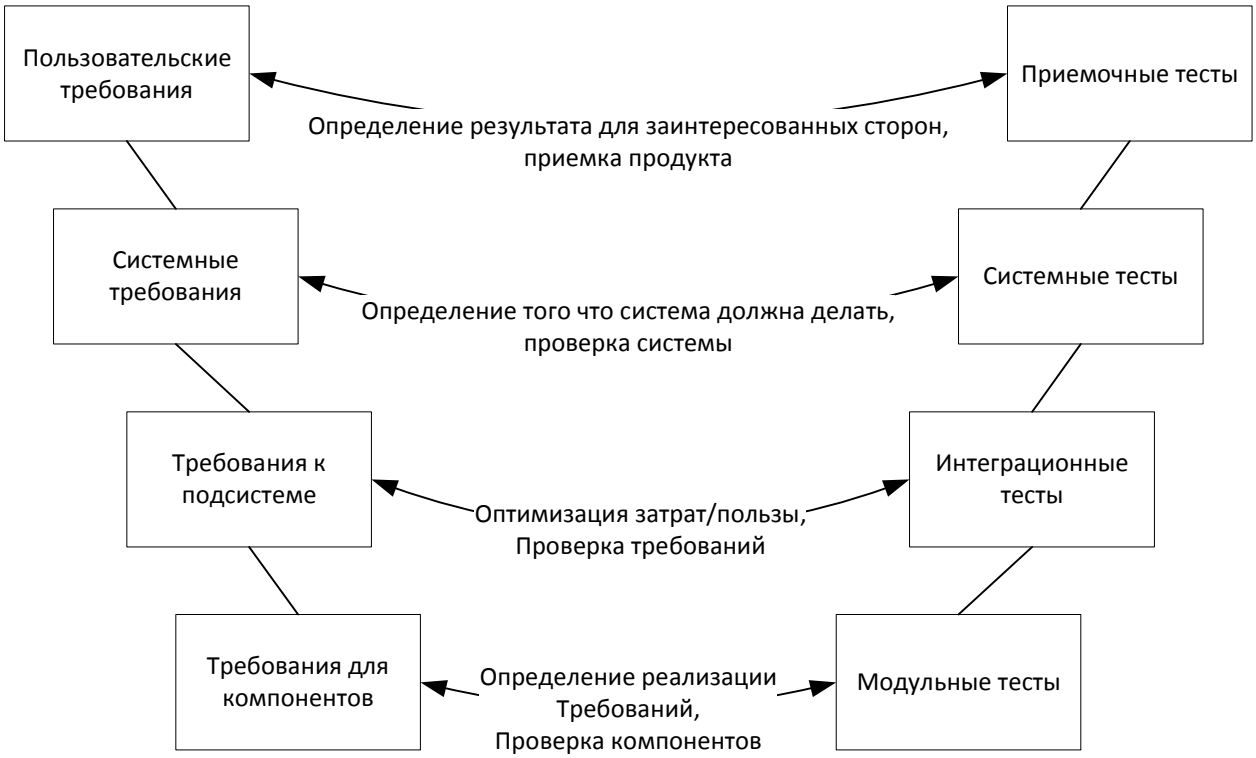
Стадии процесса

Тестирование

Анализ	Проэктрирование	Програмирование	Документирование	Сопровождение
+ Процесс сбора требований к ПО, + их систематизация, + документирование, + анализ, + выявление противоречий + разрешение конфликтов в процессе разработки ПО	+ Процесс создания ПО; + Определение внутренних свойств системы и детализация ее внешних свойств а основе требований к ПО.	+ Разработка комплекса алгоритмов; + Написание исходного кода; + Преобразование в машинный код (компиляция); + Тестирование и отладка;	Печатные руководства пользователя, диалоговая документация и справочный текст, описывающий функции и алгоритм использования ПО Виды: - Архитектурная/ Проектная; - Техническая; - Пользовательская; - Маркетинговая	+ Сбор и анализ информации от пользователей; + Создание отчетов об ошибках; + Требования по выпуску исправлений (hot-fixes, updates, service packs etc)
	Нотации – схематическое выражение характеристик: - Блок-схемы; - ER-диаграммы; - UML-диаграммы; - Макеты.			

Водопадная модель (Waterfall)

V-образная модель (V-model)



При внесении изменения необходимо выполнить следующую работу:
- Принять или отклонить изменение
- Согласовать изменение с заказчиком
- Организовать работы по переделке

AGILE

Scrum | Kanban | XP | TDD

Agile манифест

Личности и их взаимодействие важнее, чем процессы и средства	**Работающее ПО** важнее, чем исчерпывающая документация	**Сотрудничество с заказчиком** важнее, чем обязательства по контракту	**Реагирование на изменения** важнее, чем строгое следование плану

Ценности Agile

Гибкость и простота	Частые релизы	Самоорганизующаяся команда	Больше общения
Agile-процессы готовы к изменениям требований даже на поздних этапах разработки. Важна простота - искусство увеличения объема работ, которых удалось избежать.	Наивысший приоритет - удовлетворенность заказчика: - Ранние и периодические поставки ПО - ПО работающее и ценное для заказчика Продолжительность каждой итерации - от пары недель до пары месяцев. Предпочтение - коротким интервалам.	Над проектом работают мотивированные люди. Создаются все условия, поддержка и полное доверие. Самые лучшие архитектуры, требования и дизайны систем создаются самоорганизующимися командами. Команда сама организует оптимальный процесс.	Потенциальные пользователи системы и разработчики должны работать **вместе** на протяжении всего проекта. Самый действенный и эффективный способ обмена информацией как внутри команды разработчиков, так и с внешним миром - **непосредственное общение**.

СОСТАВЛЯЮЩИЕ

Product backlog

User-story

Product owner

Sprint

Sprint-backlog: tasks

Sprint planning

Demo

Ретроспектива спринта

Abnormal Termination

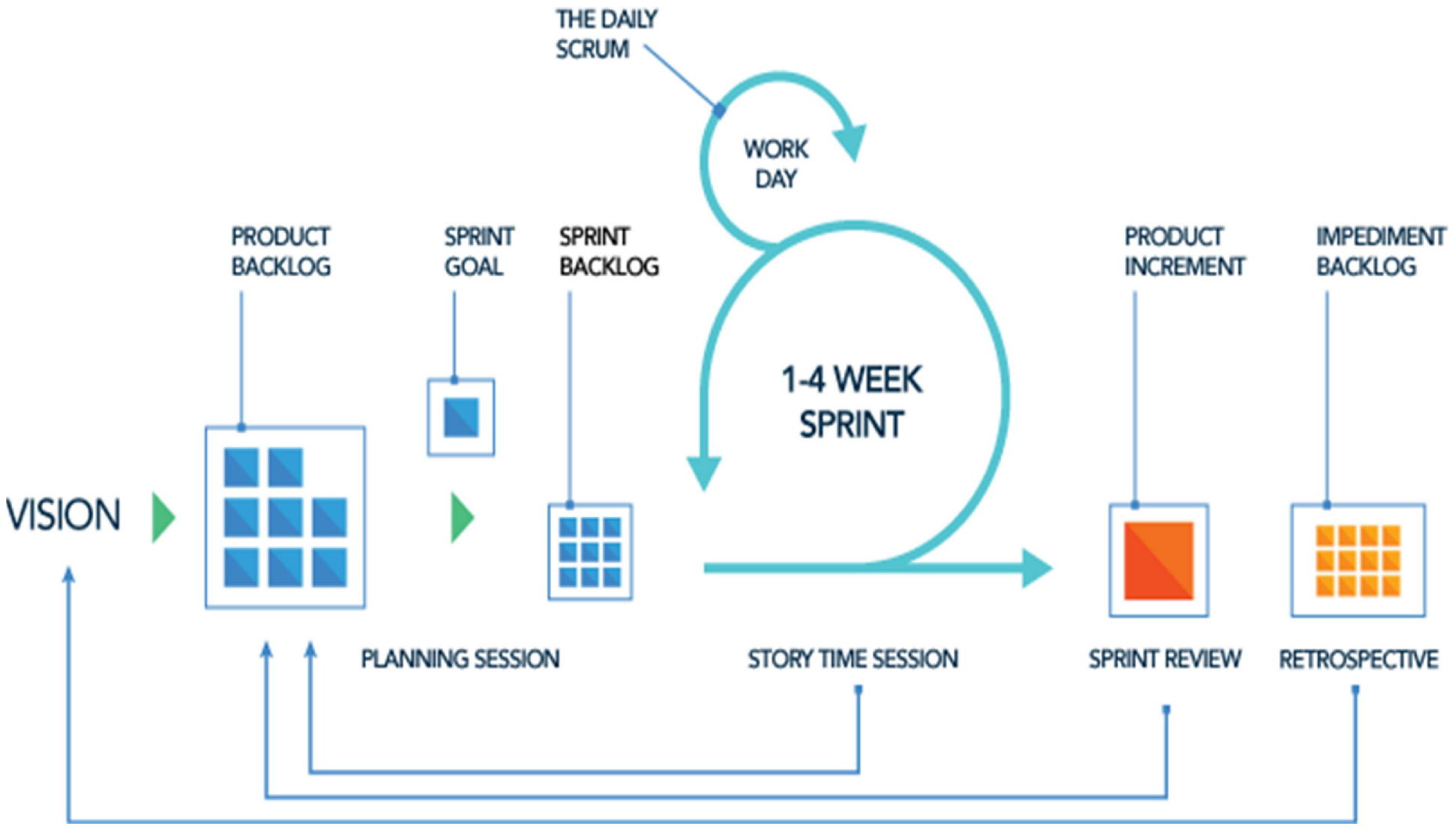
Daily scrum

Scrum master

Taskboard

Burndown chart

SCRUM

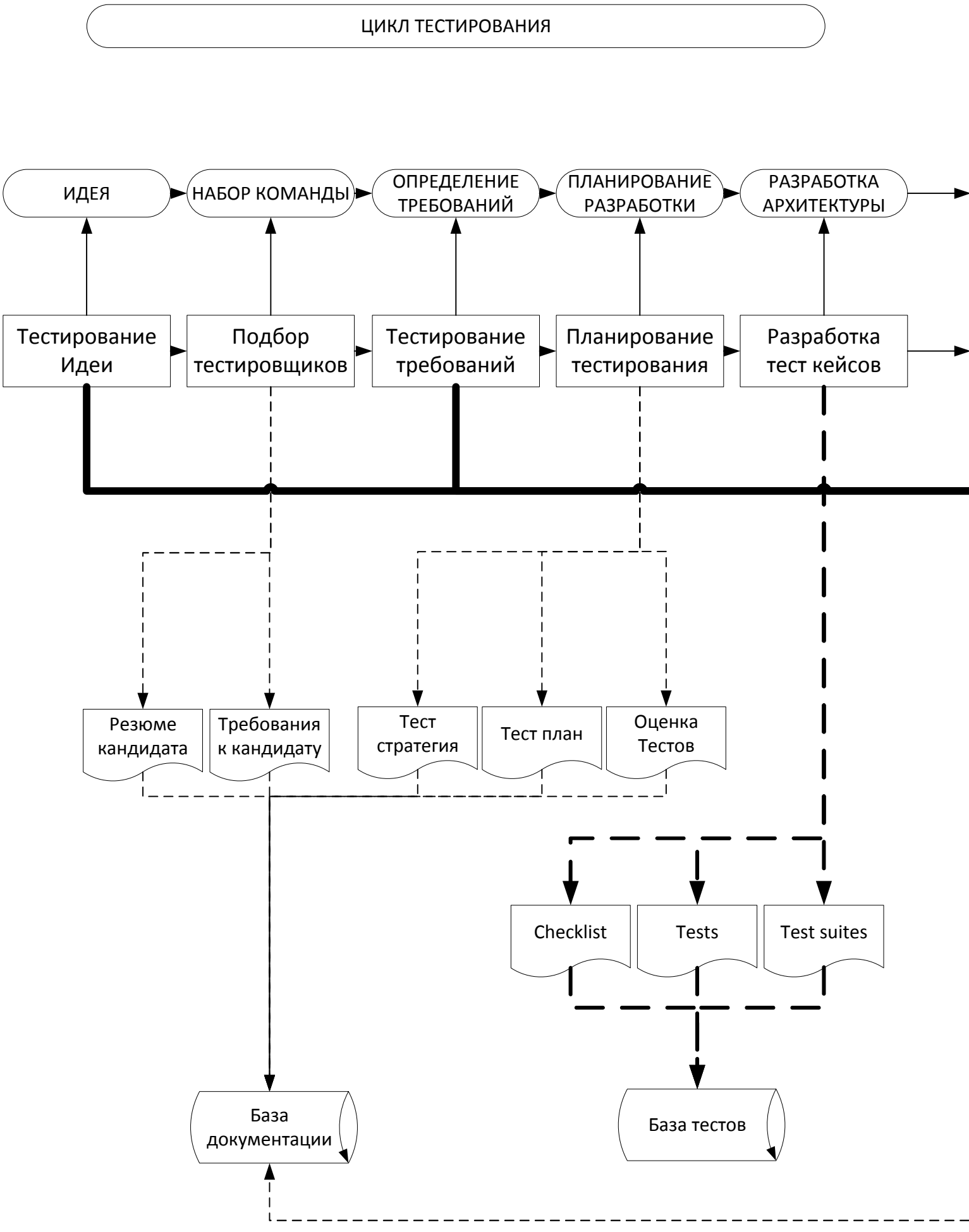

THE DAILY SCRUM
WORK DAY
PRODUCT BACKLOG
SPRINT GOAL
SPRINT BACKLOG
1-4 WEEK SPRINT
PRODUCT INCREMENT
IMPEDIMENT BACKLOG
VISION
PLANNING SESSION
STORY TIME SESSION
SPRINT REVIEW
RETROSPECTIVE

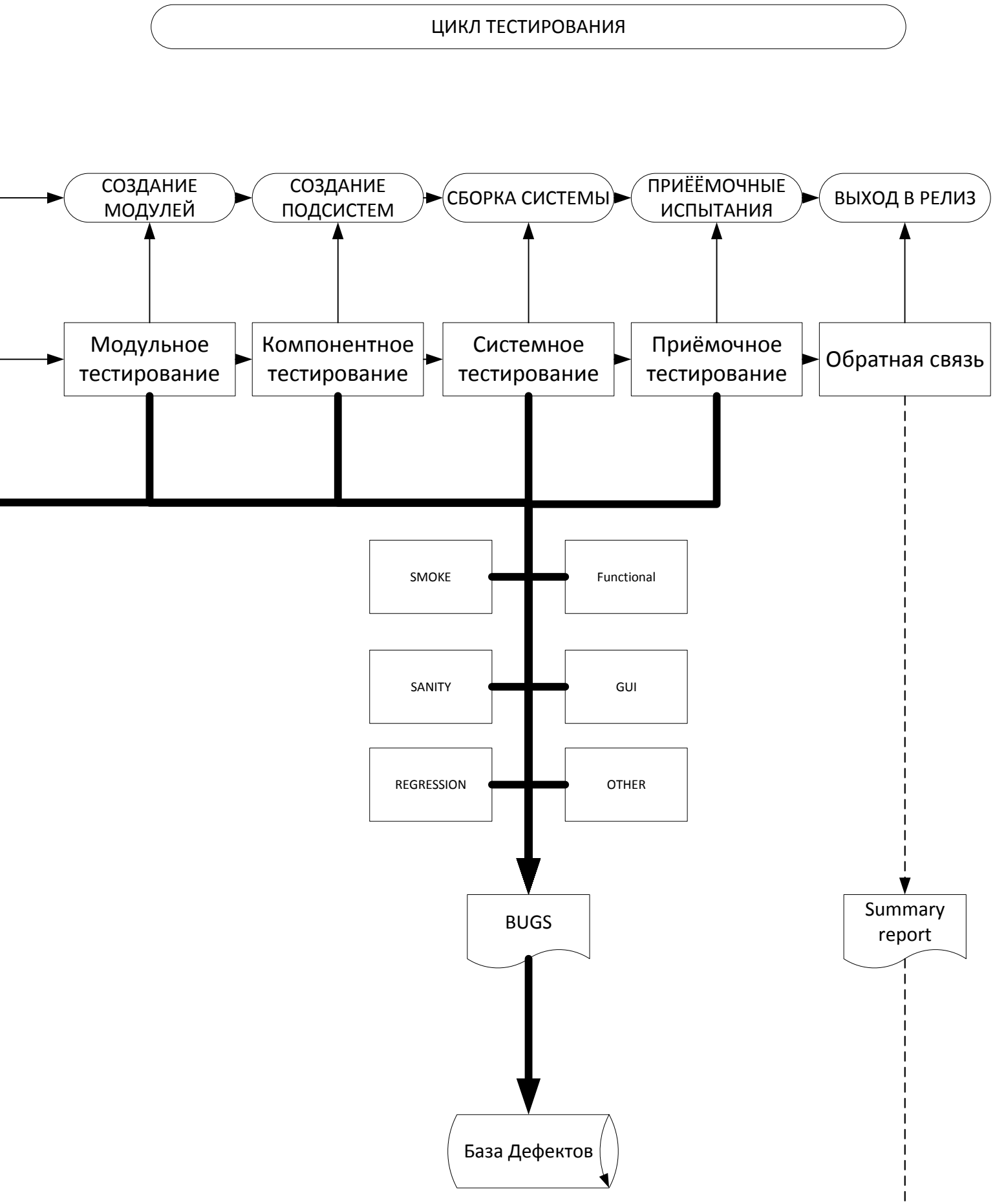
AGILE

Product backlog	Product backlog один на весь релиз Им владеет менеджер продукта (“**product owner**”) Он не статичен – записи можно добавлять, удалять, менять им приоритет Общедоступен, но поддерживается одним человеком
User-story	Содержит список функциональных единиц системы (“user stories”), запланированных на след релиз
Sprint	Фаза разработки состоит из нескольких итераций – спринтов. Обычно спринт длится 2-4 недели. Этапы: Планирование Разработка Демонстрация Ретроспектива
Sprint-backlog: tasks	Описывает задачи, запланированные командой на спринт Задачи – действия, необходимые для реализации запланированной на спринт функциональности В описание задачи входит ее оценка
Daily scrum	Проводится каждый день в фиксированное время Рекомендуется проводить стоя в течение 10-15 минут 1 минута на каждого 3 Вопроса: Что сделал; Что буду делать; Возникшие проблемы Если что-то нужно обсудить, назначается время после скрама
Sprint planning	Проводится в начале спринта Участвует вся команда User stories разбиваются на задачи и оцениваются членами команды В результате команда подписывается на ту функциональность, на которую хватает времени спринта

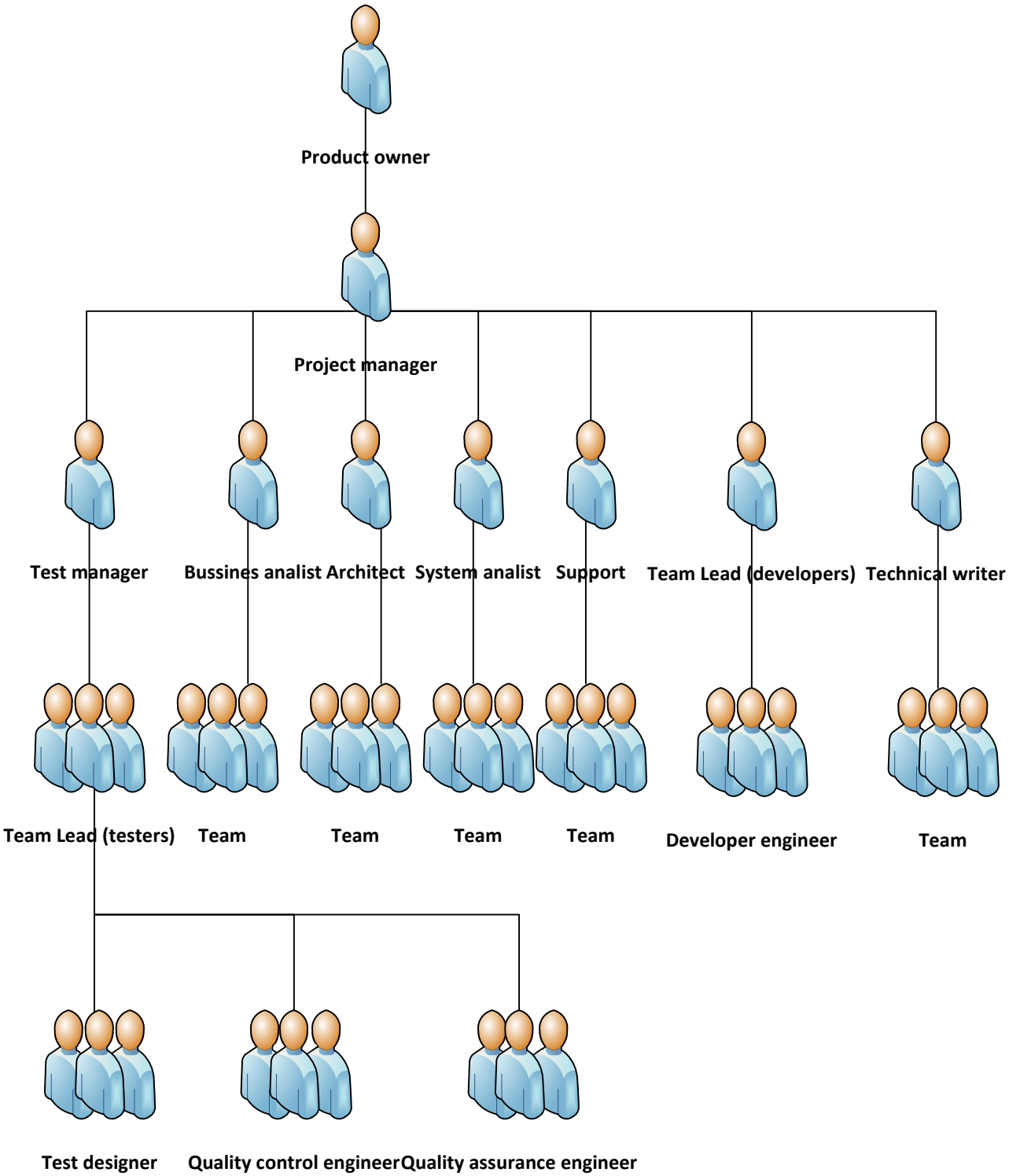
AGILE

Taskboard	
Burndown chart	Диаграмма сгорания задач показывающая количество сделанной и оставшейся работы Для спринта; Для проекта. Обновляется ежедневно для визуализации работы команды График должен быть общедоступен
Abnormal Termination	Остановка спринта раньше срока в исключительных ситуациях Команда понимает, что не может достичь цели проекта в срок Владелец проекта понимает, что исчезла необходимость в реализации цели спринта Обсуждение причины остановки Запуск нового спринта
Demo	В конце каждого спринта проводится ревью Это демонстрация реализованной функциональности В ней может участвовать любой человек, задействованный в проекте В идеале после каждой демонстрации можно отправлять продукт заказчику
Ретроспектива спринта	После каждого спринта (ревью) Участвуют все члены команды Цель - осознать: Что было хорошо? Что могло бы быть лучше Это обсуждение процесса, а не технических сложностей


ЦИКЛ ТЕСТИРОВАНИЯ
ИДЕЯ
НАБОР КОМАНДЫ
ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ
ПЛАНИРОВАНИЕ РАЗРАБОТКИ
РАЗРАБОТКА АРХИТЕКТУРЫ
Тестирование Идеи
Подбор тестировщиков
Тестирование требований
Планирование тестирования
Разработка тест кейсов
Резюме кандидата
Требования к кандидату
Тест стратегия
Тест план
Оценка Тестов
Checklist
Tests
Test suites
База документации
База тестов


ЦИКЛ ТЕСТИРОВАНИЯ
СОЗДАНИЕ МОДУЛЕЙ
СОЗДАНИЕ ПОДСИСТЕМ
СБОРКА СИСТЕМЫ
ПРИЁЁМОЧНЫЕ ИСПЫТАНИЯ
ВЫХОД В РЕЛИЗ
Модульное тестирование
Компонентное тестирование
Системное тестирование
Приёмочное тестирование
Обратная связь
SMOKE
Functional
SANITY
GUI
REGRESSION
OTHER
BUGS
Summary report
База Дефектов

КОМАНДА


Product owner
Project manager
Test manager
Bussines analist
Architect
System analist
Support
Team Lead (developers)
Technical writer
Team Lead (testers)
Team
Team
Team
Team
Developer engineer
Team
Test designer
Quality control engineer
Quality assurance engineer

ТРЕБОВАНИЕ

Требование -- это потребность или ожидание, которое:
- (а) установлено в явном виде, или
- (б) «наследуется» как обязательное из других систем требований (напр., государственных и ведомственных законоположений), или
- (в) подразумевается (является «обычным»)

ТИПЫ ТРЕБОВАНИЙ

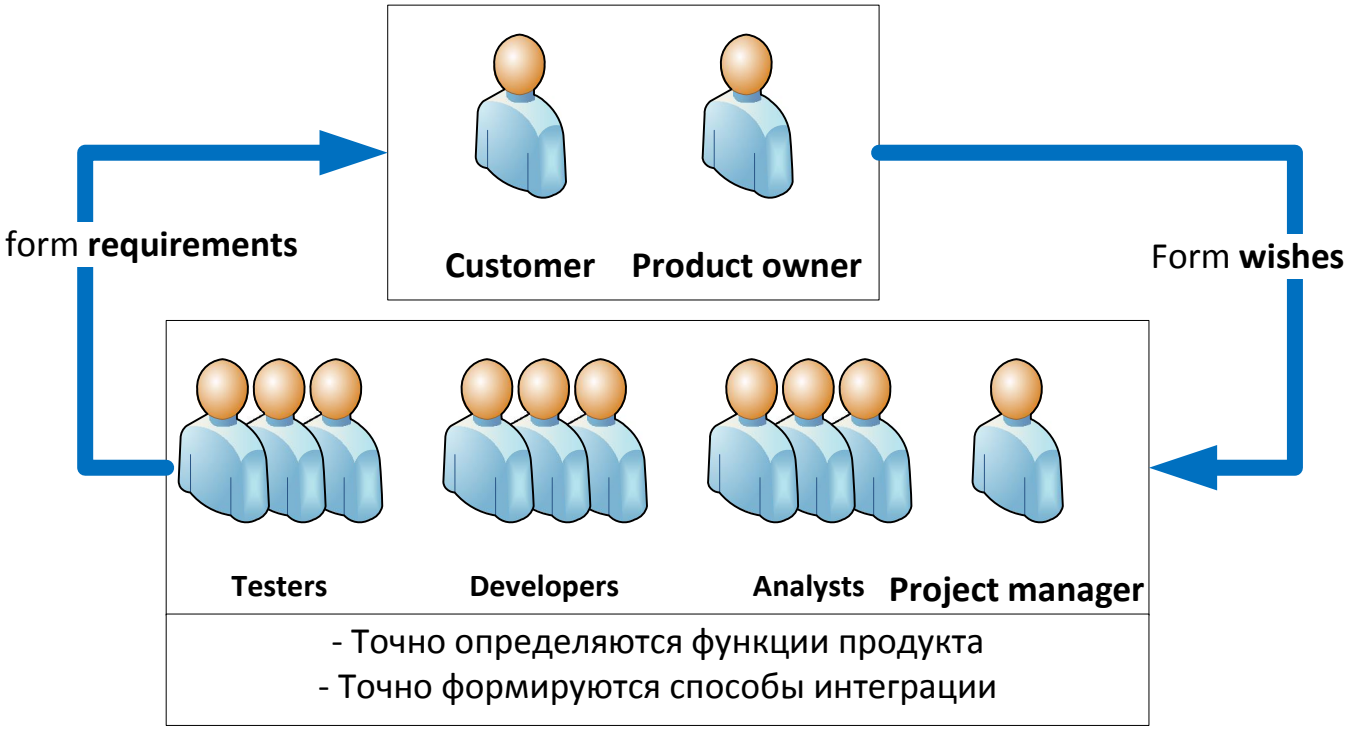
Бизнес требования
Цели и задачи продукта.

Нефункциональные требования
Описывают **как** должна работать система.

Функциональные требования
Описывают **что** должна делать система.

Предположения и ограничения
Специфика доменной области, которая накладывает ограничения на поведение или дизайн системы.

Требования пользователей
Потребности отдельных групп заказчиков/ пользователей.



КРИТЕРИИ АНАЛИЗА ТРЕБОВАНИЙ

Реализуемость

- Каждое из требований возможно реализовать ;
- Учитываются доступные ресурсы и время.

Корректность (Правильность)

- Каждое требование должно точно описывать то, что должно быть разработано.

Прослеживаемость

- Уникальный идентификатор;
- Система идентификации позволяет добавлять, удалять и разбивать требования без изменения идентификатора других требований

Однозначность

- Одинаковая интерпретация требования командой;
- Требование описано четко, просто, кратко;
- Все термины и аббревиатуры описаны и определены.

Полнота

- Все требования задокументированы;
- Каждое требование содержит всю информацию для проектирования, разработки и тестирования;
- Нет ссылок не несуществующие данные (таблицы, иллюстрации, документы);

Тестируемость

- Требование должно быть сформулировано так, чтобы можно было доказать соответствие системы предъявленному требованию;
- Требование не должно содержать неизмеримых или нетестируемых формулировок.

Непротиворечивость

- Требование не конфликтует с другими требованиями;
- Требование не конфликтует с под-требованиями;
- Требование не конфликтует с законами, стандартами.

МЕТОДЫ ТЕСТИРОВАНИЯ ТРЕБОВАНИЙ

ПРОВЕРКА ДОКУМЕНТАЦИИ	АНАЛИЗ ПОВЕДЕНИЯ СИСТЕМЫ	ПРОТОТИПИРОВАНИЕ
- Последовательный просмотр и проверка всех доступных требований;	- Формирование требований в формате «вход-выход», «событие-последствие», «условие-ответ». - Позволяет проверить на полноту, понятность, однозначность	- Создание модели будущей системы. - Позволяет проверить на полноту, корректность, реализуемость
Применяется - Заказчиками; - Аналитиками; - ПМами; - Тестировщиками.	**Применяется** - Тестировщиками (test cases); - Аналитиками (use cases).	**Применяется** - Архитекторами; - Аналитиками.
Плюсы - Простота использования; - Отсутствие специальных требований к проверяющему; - Покрывает много критериев качества; - Меньше затраты времени.	**Плюсы** - Хорошо проверяет требования; - Представляет требования в структурированном и понятном виде; - Результаты легко использовать для создание тест кейсов.	**Плюсы** - Пользователи получают возможность проверить решение; - Наглядное пособие для разработчиков и тестировщиков; - Проверка требования на реализуемость.
Минусы Качество проверки зависит от проверяющего; Вовлечение различных специалистов; Наличие документов с требованиями	**Минусы** - Требует большего количества времени; - Требует специальной подготовки.	**Минусы** - Требует значительного времени; - Специальная подготовка для создания прототипа.

ВИДЫ ТЕСТОВ

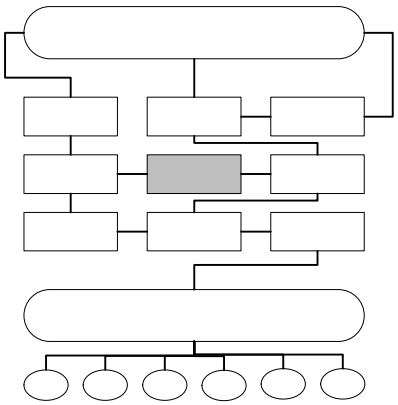
Уровни тестирования

Модульное тестирование	Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования
Интеграционное тестирование	Тестирование части системы, состоящей из двух и более модулей. Цель интеграционного тестирования - поиск дефектов, связанных с ошибками в реализации и интерпретации интерфейсного взаимодействия между модулями.
Подсистемное тестирование	Тестирование отдельных функциональных или архитектурных частей относительно соответствующего объема требований Возможно использование тест-драйверов.
Системное тестирование	Система в целом Пользовательский интерфейс - неверное использование ресурсов системы, - непредусмотренные комбинации данных пользовательского уровня, - несовместимость с окружением, - Непредусмотренные сценарии использования, - отсутствующая или неверная функциональность, - неудобство в применении и т.д.

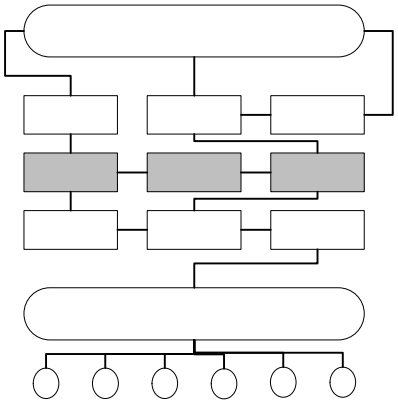
ВИДЫ ТЕСТОВ

Уровни тестирования

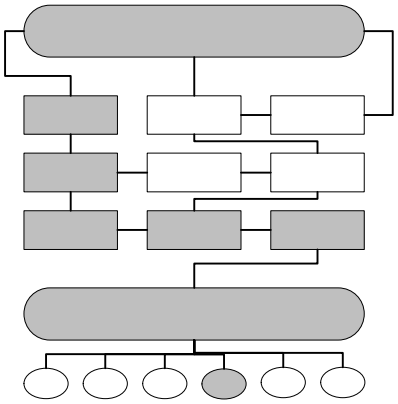
- Выполняется программистом
- Тестируется минимальный элемент кода
- Тестируетсяелементов не зависящих от других элементов
- Поиск ошибок внутри кода
- Поиск ошибок в работе алгоритмов



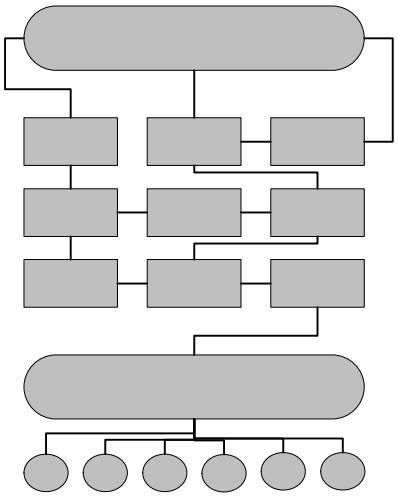
- Выполняется програмистом
- проверяет правильность входных и выходных данных в\из модуля
- Проверяется взаимодействие с другими модулями
- Использование заглушок



- Выполняется тестировщиком или программистом
- Проверка взаимодействия двух независимых подсистем.
- Использование тест-драйверов и подстановок
- Поиск ошибок на пути выполнения



- Выполняется тестировщиком
- Проверка правильного использования системы
- Проверка выполнения всех требований
- Проверка не предусмотренных шагов
- Поиск неверной функциональности
- Проверка удобства пользования
- Поиск не заявленных функций



ВИДЫ ТЕСТОВ

White-box (Glass-box)	**Black-box**
Объект известен вдоль и поперёк	Только готовый объект, ничего не известно о его внутренем состояние

Grey-box

Запуск продукта (исполняется ли програмный код)

Статическое тестирование	**Динамическое** тестирование

Способ реализации	
Ручное тестирование	**Автоматизирование** тестирование

Способ разработки тестов

Нисходящее | BIG BANG | Восходящее

Место в цикле

Smoke-testing	Sanity-testing	Regression-testing
- основного функционала	- наиболее проблемного функционала	-предыдущих тест-кейсов

Приёмо-сдаточные

испытания

Альфа (другая команда)

Бета (невовлеченные люди)

Валидация

Верификация

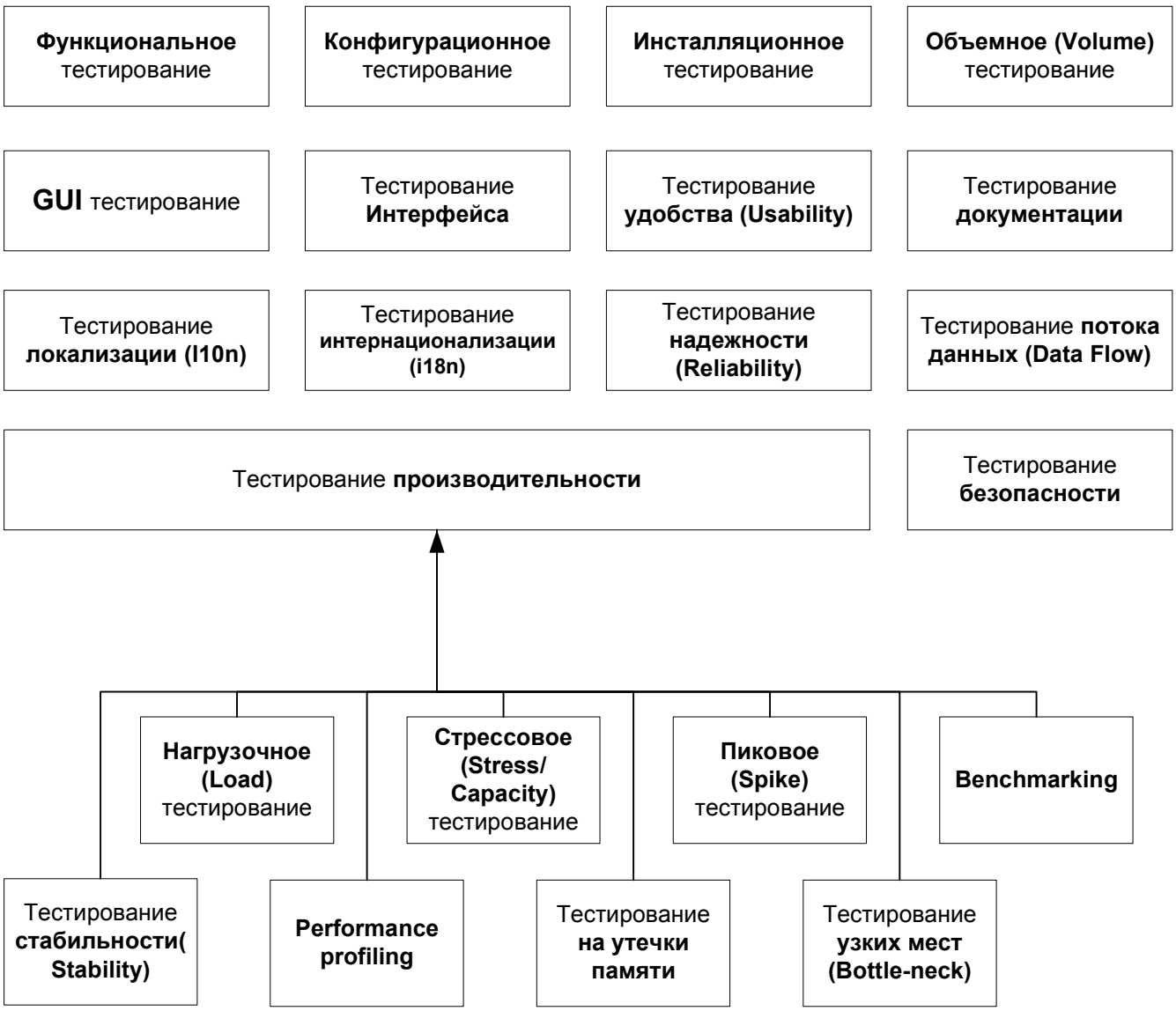
СОПРОВОЖДЕНИЕ

Тестирование системы в ходе эксплуатации для измерения параметров качества, в том числе производительности, использования системных ресурсов, а также для определения корректирующих действий

Тестирование **изменений**

Регрессионное тестирование

ВИДЫ ТЕСТОВ


Функциональное тестирование
Конфигурационное тестирование
Инсталляционное тестирование
Объемное (Volume) тестирование
GUI тестирование
Тестирование **Интерфейса**
Тестирование **удобства (Usability)**
Тестирование **документации**
Тестирование **локализации (l10n)**
Тестирование **интернационализации (i18n)**
Тестирование **надежности (Reliability)**
Тестирование **потока данных (Data Flow)**
Тестирование **производительности**
Тестирование **безопасности**
Нагрузочное (Load) тестирование
Стрессовое (Stress/ Capacity) тестирование
Пиковое (Spike) тестирование
Benchmarking
Тестирование **стабильности(Stability)**
Performance profiling
Тестирование **на утечки памяти**
Тестирование **узких мест (Bottle-neck)**

ВИДЫ ТЕСТОВ

1) **Функциональное тестирование** Функциональное тестирование проверяет функции и сервисы, которые программа должна выполнять на системном или пользовательском уровне, а так же что продукт НЕ ДОЛЖЕН выполнять. Игнорируются внутренние механизмы системы или компонента и фокусируются на откликах системы (outputs).
Тестирование вычисления прибыли в любой день между началом и концом депозита учитывая процентную ставку по депозиту.

2) **GUI Тестирование** Тестирует графические компоненты приложения и их соответствие требованиям ОС. Включает в себя проверку наличия, внешнего вида, поведения и функционирования каждого GUI элемента.
Убедиться, что кнопка "Login" располагается по центру экрана, имеет синий цвет (#0000CD), размер 20x15 пикселей.

3) **Тестирование Интерфейса** Проверяет взаимодействие с другими системами или окружением: входные данные, получаемые от этих систем и выходные данные отсылаемые в эти системы.
Так же включает проверку формата данных.
Проверить, что программа может импортировать данные из документа MS Word.

4) **Объемное (Volume) тестирование** Проверяет систему на большие объемы данных, с которыми она работает.
Тестируется, что программа может работать с определенным объемом входных, выходных и внутренних данных.
Проверить, что программа может импортировать текстовый документ размером до 1Гигабайта.

5) **Инсталляционное тестирование** Проверка установки программы включая любые опции или условия, которые могут повлиять на установку.
Так же проверяются опции инсталлятора: апгрейд, восстановление и починка, удаление.
Проверить, что программа установлена в папку, которая была указана в инсталляторе.

6) **Тестирование потока данных (Data Flow)**Тестирование пути потока данных между модулями системы.
Проверить, что данные корректно интерпретируются во время их передачи между компонентами системы.
Проверить создание пользователя в интернет-магазине. В эту проверку будет входить: Добавление пользователя в базу -> Отсылка активационного письма на указанный адрес -> Проверка активационного номера-> Активация пользователя -> Начисление скидки.

7) **Конфигурационное тестирование** Тестирование продукта на разных конфигурациях программного/ аппаратного (software/hardware) обеспечения.
Зачастую для этого требуется список самых часто используемых конфигураций.
Так же может включать в себя проверку переноса (portability) на другие конфигурации.
Убедиться, что система работает на Lion OS 10.7.3 и Safari 6.0

8) **Тестирование удобства (Usability)**Проверка удобства использования продукта для конечного пользователя.
Тестирование «человеческого фактора» – удобство, ясность, простота, легкость, интуитивная понятность, единый стиль, "user-friendly" и т.д.
Убедиться, что самые часто используемые функции программы находятся на самых видных местах (верхняя панель программы, первые пункты выпадающих меню).

9) **Тестирование безопасности** Проверка несанкционированного доступа к данным или проведения несанкционированных операций. Тестирование уровней доступа для разных пользователей и/или групп пользователей.
Так же включает проверку триальных, ограниченных и истекаемых (trial/limited/expirable) версий продукта.
Проверить, что доступ к админ странице сайта есть только у пользователей из группы «admins».

10) **Тестирование надежности (Reliability)**Сбор и сравнение статистических параметров: частота сбоев, время между сбоями, распределение сбоев по severity.
Тестирование механизмов резервного копирования и восстановления (backup and recovery).
Убедиться, MS Word делает резервную копию текущего файла с определенным интервалом и что данные можно восстановить из этого файла после сбоя.

11) **Тестирование интернационализации (i18n)**Проверка интернационализации продукта.
Тестирование возможности работы продукта на разных платформах (в смысле языка и культуры).
Убедиться, что продукт работает на китайской версии Windows.

12) **Тестирование локализации (l10n)**Проверка локализации продукта и его работы на определенной платформе (в смысле языка и культуры).
Проверка GUI элементов, справок, документации, установка и апгрейд на локализированном окружении.
Тестирование арабской версии продукта (вязь, написание справа налево, праздники в календаре и т.д).

13) **Тестирование документации** Проверка что вся документация описанная в требованиях есть в наличии и доступна пользователям. Документация адекватно описывает возможности и функции программы Ссылки и кросс-указатели(cross-references) присутствуют и работают.
Документация имеет определенный внешний вид, формат и правописание
Она опубликована правильным способом, доступна в определенном месте
Проверить, что Readme.txt и Release Notes.txt есть в пакете приложения.

ВИДЫ ТЕСТОВ

Тестирование производительности

Проверка производительности системы на соответствие требованиям.

1) Нагрузочное (Load) тестирование Тестирование программы в реальных условиях эксплуатации и под реальными нагрузками.
Разнообразные виды нагрузки приложения, но в рамках ожидаемых условий.
Вход в приложение разного количества пользователей но меньше максимально разрешенных 100.

2) Стрессовое (Stress/Capacity) тестирование Тестирование программы в условиях близких к максимальным и превышающих их. Позволяет убедиться, что производительность системы не падает и система работает при максимальной нагрузке. *Одновременный вход в приложение максимально разрешенного количества пользователей.*

3) Пиковое (Spike) тестирование Проверка нагрузкой, которая в несколько раз превышает допустимую. Позволяет проверить как система реагирует на чрезмерную нагрузку и в какой момент сломается.
Одновременный логин на сайт тысячи пользователей. Проверить как сайт (сервер) будет работать после такой нагрузки.

4) BenchmarkingТестирование производительности в сравнении с похожими продуктами и/или предыдущими версиями. Позволяет обнаружить ухудшение производительности новых версий.
Сравнение производительности Windows XP и Windows Vista.

5) Тестирование стабильности(Stability) Позволяет определить доступность программы на ожидаемом временном отрезке.
Позволяет проверить что определенные характеристики продукта (время отклика, время обработки запроса) остаются в допустимых пределах на длительном периоде времени.
Время логина в программу должно занимать не больше 30 секунд на протяжении 6 месяцев после рестарта.

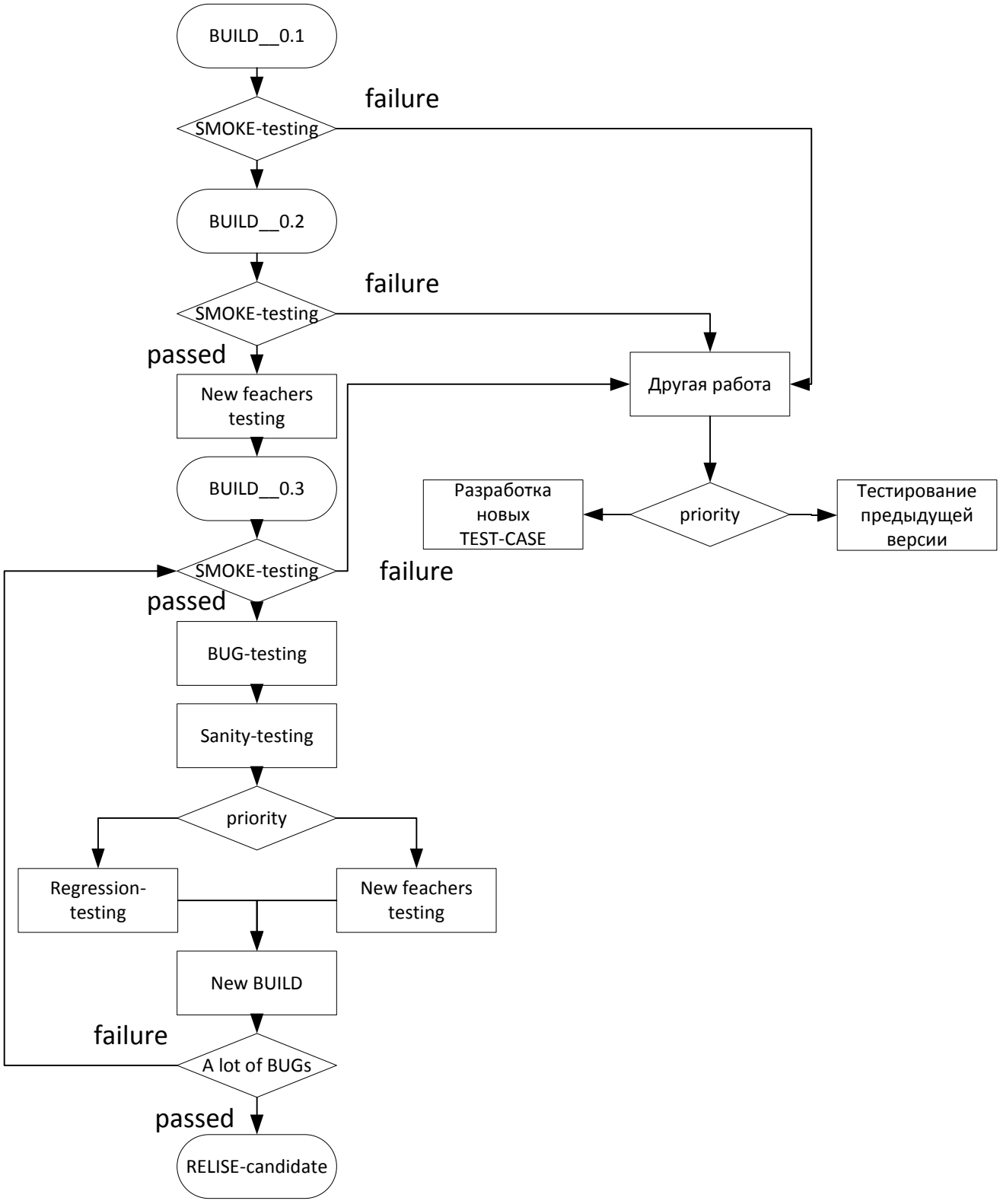
6) Performance profiling Прямое слежение за характеристиками производительности:
Время выполнения определенной операции, Использование CPU, Использование памяти.
Приложение должно использовать не больше 20% CPU.

7) Тестирование на утечки памяти Тестирование нацелено на определение ситуаций когда программа не высвобождает память. Это может привести к постепенному заполнению ОП и существенному ухудшению производительности.
Выполнение определенной операции большое количество раз. Если данная операция имеет проблемы с выделением памяти, то при замере использования ОП эта проблема будет выявлена.

7) Тестирование узких мест (Bottle-neck)Нахождение критических мест в потоке данных, которые замедляют работу системы . Bottle-neck (бутылочное горлышко) – модуль и компонент системы, увеличив производительность которого мы добьемся увеличения производительности программы.
Модуль записи в БД использует неоправданно сложный запрос, который делает полное сканирование базы.

8) Тестирование «условия гонок» (Race conditions)Тестирование систем, которые позволяют одновременный доступ к ресурсам.
«Условия гонок» - аномальное поведение программы при одновременном возникновении событий и/или одновременном доступе к одному ресурсу.
Попытка одновременного редактирования одного файла на Sharepoint.

ЦИКЛ ТЕСТИРОВАНИЯ



Роли в процессе управления дефектом

Customer

Product owner

Project manager

+ Первичная настройка репозитория (роли, код проекта, и т.п)
+ Назначение исполнителей для поиска дефектов, отслеживание выполнения, переназначение
+ Контроль за отложенными дефектами
+ Предварительный анализ дефекта (отклонение, планирование работ по фазам, дупликация) – вместе с разработчиками
+ Определение процесса в рамках которого родился дефект – с участием архитектора, аналитика, ТМ

Bussines analist

+ Анализ дефектов
+ Поиск и определение возможных причин дефектов на уровне требований
+ Исправление дефектов в требованиях

Developer engineer

+ Исправление дефектов и реализация запросов на изменение
+ Передача сборки на тестирование

Technical writer

+ Назначение исполнителей для поиска дефектов в документации, отслеживание выполнения, переназначение
+ Контроль за отложенными дефектами документации
+ Предварительный анализ дефектов документации (отклонение, планирование работ по фазам, дубликаты)
+ Определение процесса в рамках которого родился дефект

Test manager

+ Настройка репозитория
  - Система уведомления
  - Настройка динамических списков
  - Ведение номеров сборок
  - Ведение списка сценариев тестирования (test cases) и log folders
  - Связь с системой поддержки требований
  - Дополнительные запросы, отчеты, правила, поля в проекте
+ Регистрация дефектов, полученных от заказчика, от членов проектной команды, не уполномоченных на регистрацию

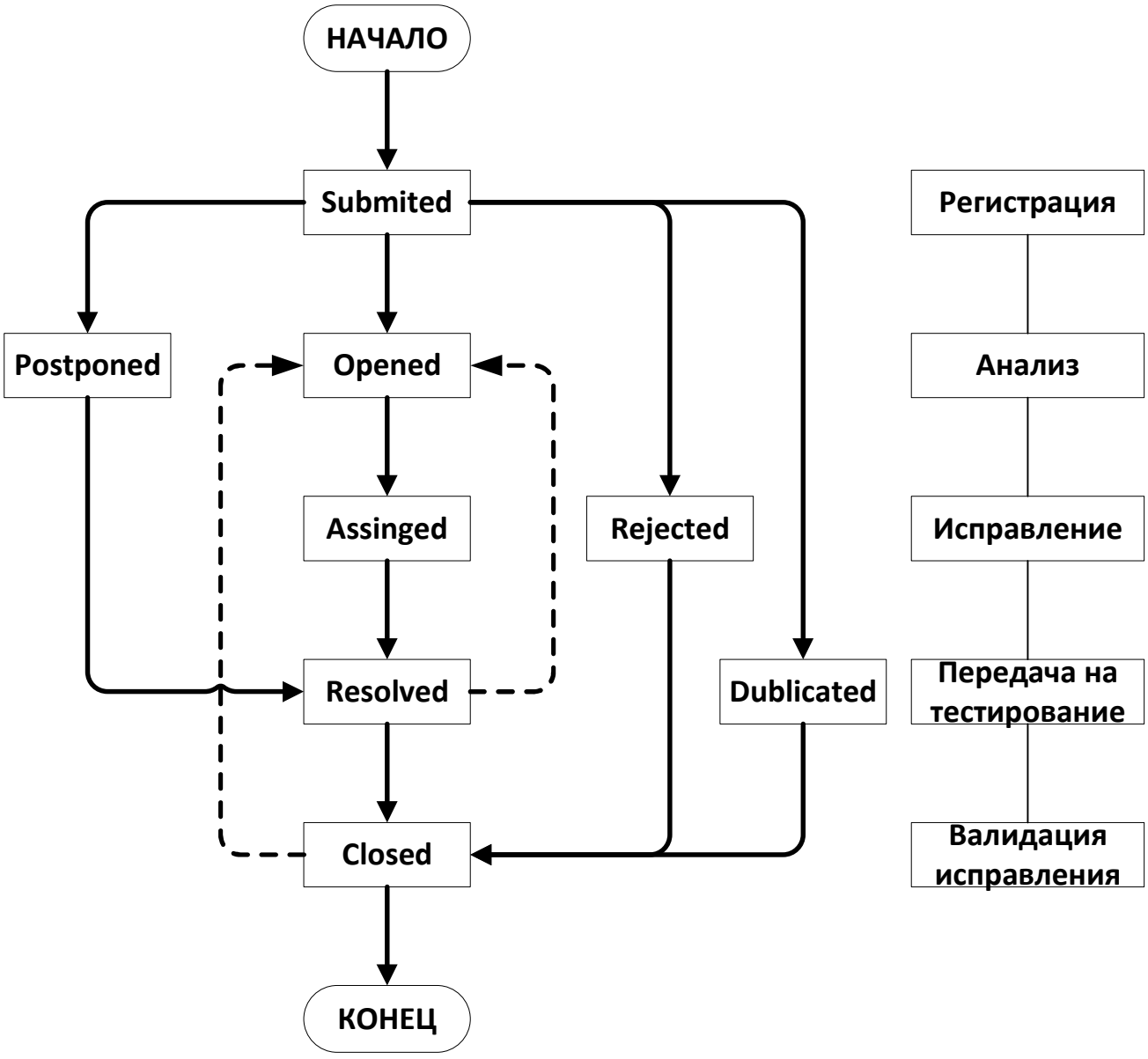
+ Контроль за соблюдением правил регистрации и обработки дефектов
+ Анализ списков дефектов, диаграмм распределения и трендов для планирования работ и отчетности
+ Контроль за дефектами, обнаруженными вне раунда тестирования -> доработка планов тестирования
+ Обработка отклоненных дефектов

Контроль за неподтвержденными дубликатами

Quality control engineer

+ Регистрация дефектов, обнаруженных в раунде тестирования
+ Контроль реализации запросов на изменения и исправления дефектов, включая дубликаты
+ Обработка отклоненных дефектов по запросу ТМ

ЖИЗНЕНЫЙ ЦИКЛ ДЕФЕКТА



ЖИЗНЕНЫЙ ЦИКЛ ДЕФЕКТА ПО РОЛЯМ



Quality control engineer | Project manager | Developer engineer

ТИПЫ НЕИСПРАВНОСТЕЙ

Error
(ошибка)
несоответствие между вычисляемым, наблюдаемым, или измеренным значением или состоянием и ожидаемым теоретически правильным значением или состоянием.

Defect
(дефект)
проблема внешнего вида программного продукта, поведения, несоответствующего спецификации, отсутствие ожидаемого функционала.

Fault
(неисправность)
неверный шаг, процесс или данные, которые влекут за собой непреднамеренные или непредвиденные действия в программе.

Failure
(отказ, сбой)
неспособность системы или компонента системы выполнять требуемые функции при заданных требованиях к производительности.

Bug
ошибка в программе, которая заставляет программу выполнять непреднамеренные или непредвиденные действия.

Ущерб

Блокирующие (Blocker)
– Ошибки, из-за которых дальнейшая работа с системой становится невозможной.

Важные (Major)
— Из-за таких ошибок система, в целом, работает, но что-то работает не так

Обычные (Normal)
— Как правило, к этой категории баги относят очень редко. В качестве примера, могу написать что-то вроде не работает кнопка «Запомнить меня» на сайте

Малозначимые (Minor)
— к таким, как правило, относятся небольшие баги, типа опечаток, «плавания» вёрстки в IE6 на определенной странице в админке и т.п. Редко исправляются по одному, собираются в несколько десятков/сотен/тысяч в зависимости от продукта и фиксятся «пачкой»

ПРИОРИТЕТ

FIX IN RELEASE
— Пофиксить в новой версии продукта. Как правило, относится к багам, обнаруженным в процессе тестирования нового функционала

MUST FIX
— Пофиксить как можно быстрее. Как правило включает в себя блокирующие баги, которые должны быть исправлены в специальном сервис паке, до выхода новой версии

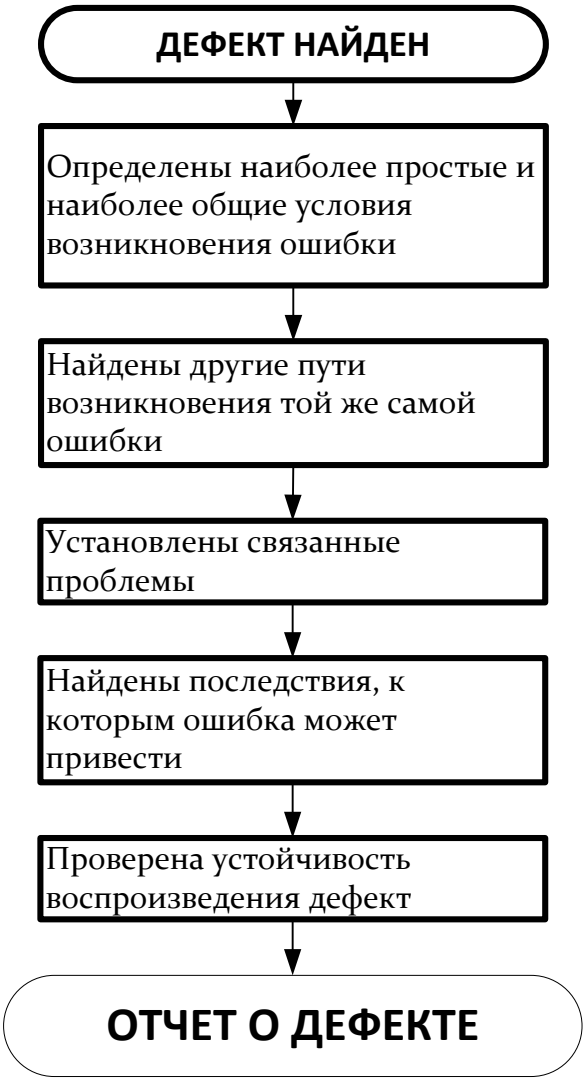
FIX IF TIME
— «Пофиксить, если есть врёмя» — к этой категории как правило относятся минорные баги

NEVER FIX
— «Не фиксить никогда». Например, какая та фича будет удалена из следующей версии продукта, либо найдена в продукте, который уже более не поддерживается, или его поддержка прекратится в ближайшее время

ОПРЕДЕЛЕНИЕ ДЕФЕКТА

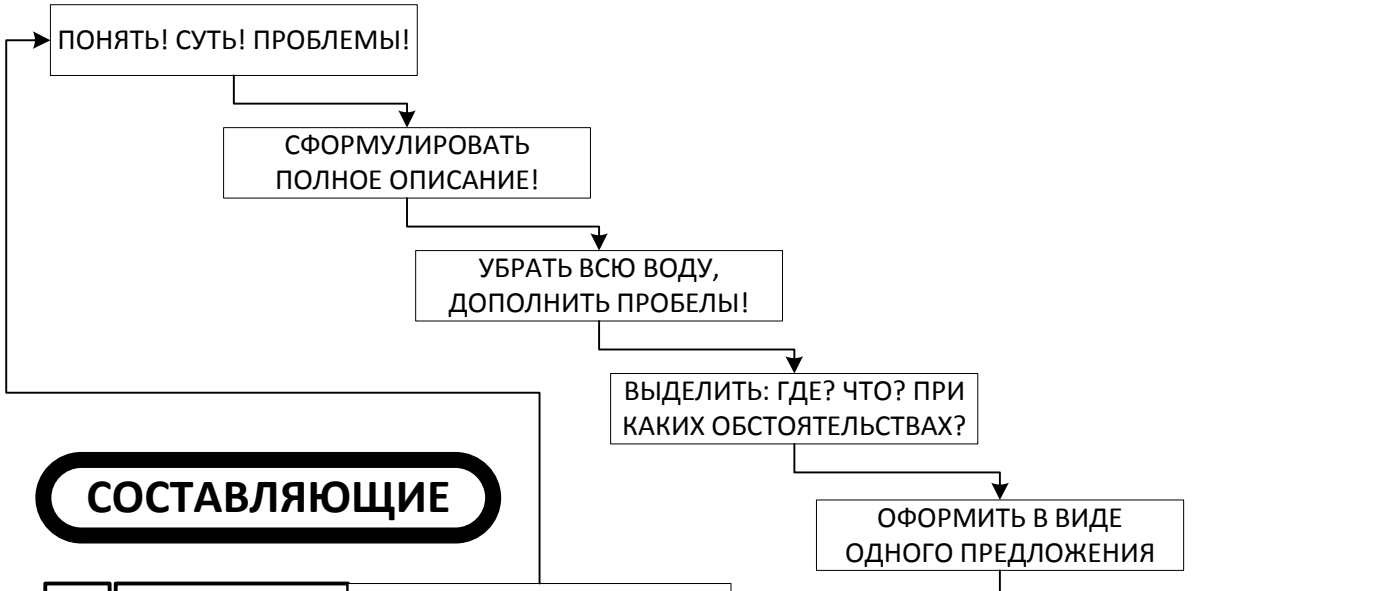
Дефект есть несоответствие между фактическими и требуемыми характеристиками объекта тестирования

Дефект есть несоответствие фактического поведения системы разумным ожиданиям пользователя



Наблюдаемый дефект - это расхождение, *неувязка* между ожидаемым и полученным результатом, неверное поведение программного продукта.	**Внутренний дефект**, или **изъян** – это причина отказа, то, что привело к отказу; как правило, дефект в коде программы, но и: дефект в дизайне.
Отказ это симптом = внешнее проявление внутреннего изъяна, наблюдаемое при некоторых условиях Тестеры (и пользователи) наталкиваются на отказы/сбои, иначе: наблюдают симптомы	Разработчики находят и исправляют внутренние дефекты (ставят диагноз и проводят «лечение») Изъян кода может не приводить к отказу, т.е. может быть не замечен тестировщиком Один изъян может приводить к нескольким отказам в разных частях системы

ОТЧЕТ О ДЕФЕКТЕ



СОСТАВЛЯЮЩИЕ

ОБЯЗАТЕЛЬНО		
	SUMMARY	Где? Что? При каких обстоятельствах; Предельно кратко, но достаточно;
	STEPS TO REPRODUCE	Список шагов что привели в дефекту
	ACTUAL RESULT	Полученный результат
	EXPECTED RESULT	Ожидаемый результат

- Атомарно
- Последовательно
- Содержать ссылки на дополнение

Pre-conditions	Условия до начала тестирования					
Post-conditions	Приведения программы в начальное состояние					
description	Подробное описание дефекта					
Severity	Серьезность – степень влияния на систему	Bloker	Critical	Major	Normal	Minor
Priority	Приоритет – как скоро нужно исправить дефект	Fix in release	Must fix	Fix if time	Never fix	

ХОРОШИЙ ОТЧЕТ О ДЕФЕКТЕ

ПРОСТОТА	ПОЛНОТА	ОБЪЕКТИВНОСТЬ	НЕЙТРАЛЬНОСТЬ	НИЧЕГО ЛИШНЕГО
+ Простые грамматические структуры + однозначные выражения + короткие и ясные фразы + СПИСОК (идеальный вариант)	+ Только необходимую информацию для понимания дефекта - без лишних уточнений	+ предоставить объективные данные для принятия решений + не указывать на виновника	+ суть излагать ясным и деловым языком - без эмоций - без юмора и сарказма	Только то что необходимо

СНИМКИ ЭКРАНА	КРАТКОЕ ОПИСАНИЕ	ПОДРОБНОЕ ОПИСАНИЕ
+ Доказательство работы тестировщика и програмиста. + Снимок лишь части экрана - проблемной зоны (*лучше в формате JPEG или PNG*) + Снимок нужно прикрепить к отчёту о дефекте. + Название адекватно – названию дефекта	+ Краткое (в одну строку) описание проблемы. + Используеться ПМ-ом при анализе списка неисправленных дефектов. + Позволит выделить и подробнее рассмотреть только значимые дефекты. Оно должно включать : - Краткое, но достаточно точное описание, позволяющее понять суть дефекта. - Краткое указание на область и условия проявления дефекта (насколько проявление дефекта зависит от условий выполнения программы) - Указание на серьезность дефекта (помогающее представить последствия его наличия в продукте)	Подробное описание сути проблемы. - Используеться ПМ-ом и разработчиком при углубленном изучении сути дефекта. - Должно быть самодостаточным, концентрирующим в одном месте максимум информации о дефекте - Перечислите все переменные окружения (config и т.п.). - Если ошибка трудновоспроизводима (требует специальных условий - специфических входных данных, предварительных действий, напрямую не связанных с этой ошибкой т.п.), то четко опишите эти условия. - Правильное подробное описание не вызывает вопросов.

ВОСПРОИЗВОДИМОСТЬ		
+ **Всегда описывайте воспроизводимость дефекта**. - Никогда не говорите «да», пока не поймете, как добиться повторения этой ошибки. (Научитесь воспроизводить ошибку до составления отчета.) - Если после неоднократных попыток воспроизвести ошибку не удается, то напишите «нет» и объясните, какие действия вы предпринимали для ее воспроизведения.	+ **Всегда описывайте воспроизводимость дефекта**. - Если дефект проявляется нерегулярно и вы все еще не понимаете, почему, то напишите «иногда» и дайте поясните. - Не всегда возможно воспроизвести ошибку. Например: дефект, описанный пользователем, проявляющийся в специфической ситуации, которую трудно воспроизвести. **Обязательно**! Тестировщик должен воспроизвести ошибку в присутствии разработчика, если разработчик говорит, что не может ее повторить.	**Как воспроизвести дефект.** + Опишите шаги для воспроизведения этого дефекта. - Начните описание с известного места (например, с запуска программы или открытия входного документа) и - Затем опишите каждый шаг до проявления ошибки. - НУМЕРУЙТЕ ШАГИ. - Отделяйте шаги один от другого. + Опишите ожидаемое и фактическое поведение. - Разница между ними и будет сущностью дефекта.)

TEST PLAN

Project manager — осуществляет общий контроль над созданием тест плана

Test manager — создает тест план, определяет стратегию тестирования продукта

Тест План – документ, описывающий стратегию и подход к планированию и управлению процессом тестирования проекта (продукта)
Он гарантирует, что процесс тестирования будет проходить тщательно и организованно, а так же даст возможность тест команде определить качество продукта.

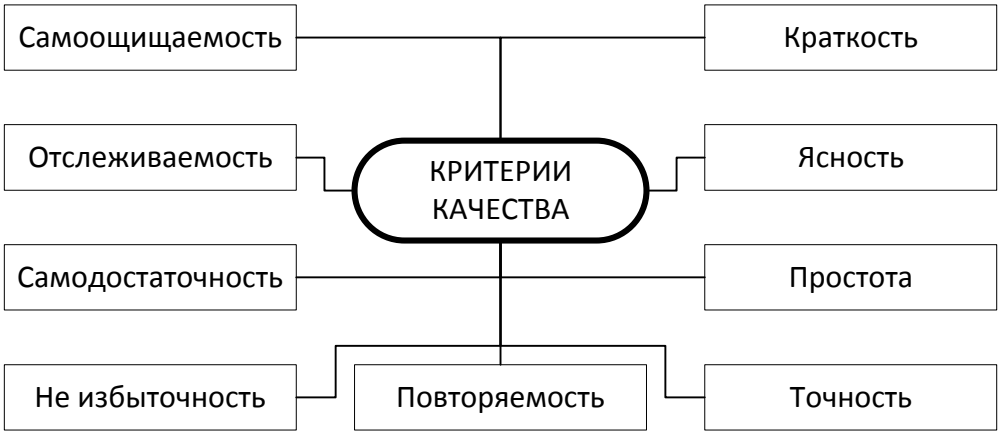
ОПРЕДЕЛЯЕТ

- Цели тестирование
- Методологии
- Инструменты
- Ожидаемые результаты
- Распределение ответственности
- Требуемые ресурсы

СОДЕРЖИТ

- **Идентификатор** — Уникальный номер для идентификации плана тестирования
- **Введение** — Цель – описывает причины написания данного тест плана;
  Терминология – список специфических терминов, сокращений, аббревиатур;
  Идентификатор проект – уникальный идентификатор тестируемого продукта
- **Тестируемые / не тестируемые функции** — Тестируемые функции – описание частей и модулей продукта, которые должны быть протестированы в данной версии продукта;
  Не тестируемые функции - описание частей и модулей продукта, которые не входят в задачи тестирования на данный релиз;
  Указываются причины по которым модули продукта не будут протестированы
- **Стратегии и подходы тестирования** — Определяет общую стратегию тестирования данного проекта;
  Определяет подход к тестированию каждой из функций системы или групп функций (подсистем);
  Перечисляются уровни, виды тестов которые буду применяться в данном релизе.
  Методы тест-дизайна
- **Критерий успешности / не успешности** — Определяет критерии прохождения/отказа для каждой группы тест кейсов;
  Определяет критерии входа в тестирование;
  Определяет критерии выхода из тестирования;
- **Критерий приостановки тестирования** — Определяет требования к остановке тестирования;
  На уровне отдельного тест кейса или тест типа;
  На уровне отдельного модуля или функции;
  На уровне всей системы.
- **Инструменты тестирования** — Перечень всех инструментов, которые будут использоваться в процессе тестирования;
  Система отслеживания ошибок;
  Система документооборота;
  Система хранения тест кейсов и тест наборов;
  Инструмент автоматизации;
  Инструмент отчетности.
- **Предоставляемая документация** — Список документов, предоставляемых перед, во время и после окончания тестирования;
  Тест план;
  Тестовые случаи;
  Тестовые наборы;
  Сценарии использования;
  Отчеты об ошибках;
  Отчет о тестировании
- **Тестовые стенды** — Определяет платформы для тестирования;
  Software;
  Hardware;
  Configurations.
- **Обязанности и назначения** — Распределение ролей в команде;
  Определение групп тестирования;
  Наем сотрудников;
  Обучение сотрудников.
- **Поэтапный график тестирования (Roadmap)** — Временные рамки для всего процесса тестирования;
  Временные рамки для каждого из этапов тестирования.
- **Управление рисками** — Определение рисков;
  Категоризация рисков;
  Меры предупреждения рисков;
  Методы компенсации рисков

TEST CASE



Header		
Identifier	Уникальный идентивикатор для каждого теста	
Owner	Составитель или исполнитель	
Date	дата написания или исполнения	
Requirement	Необходимые условия	
Configuration	Конфигурация на которой производились испытания	
Details		
Purpose	Цель	
Reason	Причина проведения	
Complexity	Сложность проведения	
Estimated time	Предположительное необходимое время	
Notes	Заметки (разное)	
Dependencies	Зависимости от разных условий	
Scenario		
Initialization	Шаги перед началом работы	
Step	**Action**	**Expected Result**
№ шага	действие	ожидаемый результат от действия
Finalization	шаги восстановления системы в исходное состояние	

TEST DESING

Набор входных данных, условий выполнения и ожидаемых результатов, разработанных с целью проверки соответствия тестируемого продукта выдвигаемым к нему требованиям.

Анализ Граничных Значений (Boundary Value Analysis - BVA)	**Эквивалентное Разделение (Equivalence Partitioning - EP)**
Метод проверки переменных программы на их границах	Метод сокращения числа тестов путем выбора одного теста из эквивалентного набора

Предугадывание ошибки (Error Guessing - EG)	**Причина / Следствие (Cause/Effect - CE)**	**Исчерпывающее тестирование (Exhaustive Testing - ET)**
Проверка важных проблем, которые могут возникнуть	Проверка продукта по наиболее частым и важным сценариям использования – use cases	Проверить все возможные комбинации входных значений, и в принципе, это должно найти все проблемы

Задачи тест дизайна

Исходные данные

A					B					C
1	2	3	4	5	6	7	8	9	10	11

Граничные условия

Граничные условия

+ Граничные значения – это значения, на которых программа меняет свое поведение;
+ Граничные значения – это границы классов эквивалентности;
+ Формула проверки граничных значений: {BV-1, BV, BV+1}.

	граничные условия		пригранничные условия					
A	1	5	**0**	1	**2**	**4**	5	**6**
B	6	10	**5**	6		10		**11**
C	11		**10**	11	**12**			

Классы эквивалентности

A	1	2	3	4	5
B	6	7	8	9	10
C	11				

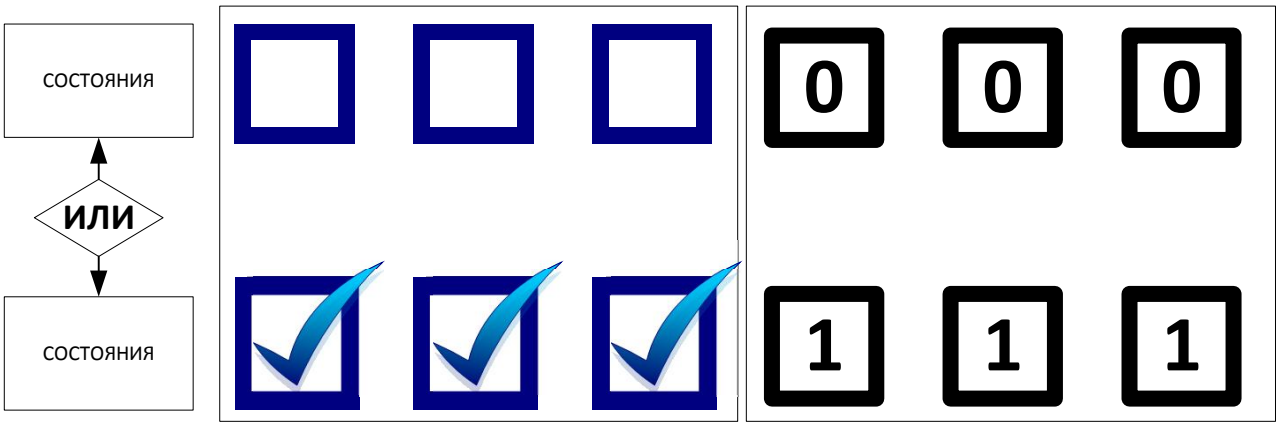
Классы эквивалентности

+ Класс эквивалентности — это класс, в рамках которого все тесты являются эквивалентными;
+ Эквивалентные тесты — это тесты, которые приводят к одному и тому же результату.

Метод (всех) ПАР

ОПЦИИ

ТОЖЕ САМОЕ, Но другой вид



ВЕРОЯТНОСТЬ ОШИБКИ связи **ДВУХ** элементов намного **ВЫШЕ**, чем ошибки трех элементов

Тестируются **ПАРЫ** элементов

ТЕСТИРУЕМ **КАЖДЫЙ** С **КАЖДЫМ** по **ОТДЕЛЬНОСТИ**

НА ОСТАЛЬНЫЕ НЕ ОБРАЩАЕМ ВНИМАНИЯ

СОВМЕЩАЕМ. В необязательные элементы добавляем ЭЛЕМЕНТ для СОКРАЩЕНИЯ количества тестов

РЕЗУЛЬТАТ Намного **меньше** тест кейсов

ВСЕ ВОЗМОЖНЫЕ ТЕСТ-КЕЙСЫ

0	0	0
1	0	0
0	1	0
0	0	1
0	1	1
1	0	1
1	1	0
1	1	1

МЕТОД ВСЕХ ПАР

ручной

программыный

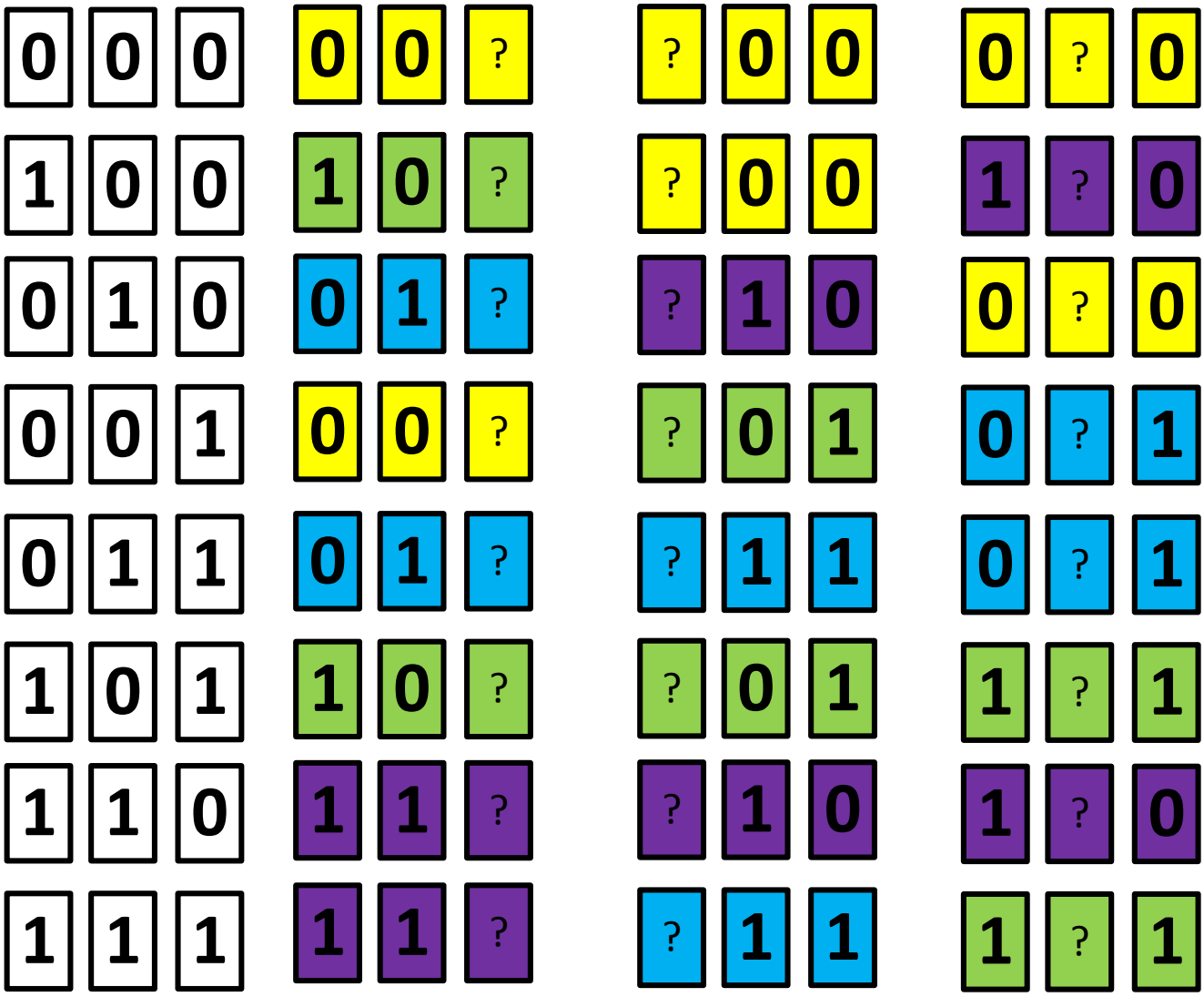
РЕЗУЛЬТАТ

0	0	0
1	1	0
1	0	1
0	1	1

Метод (всех) ПАР

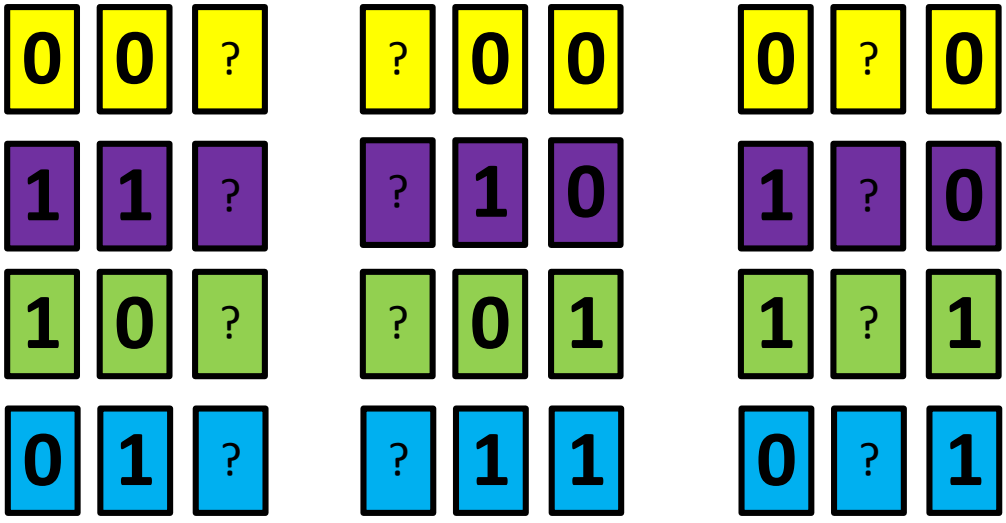
ПЕРЕБОР всех возможных вариантов

ПЕРЕБОР всех ПАР по вариантам



Убираем дубликаты

СОВМЕЩАЕМ

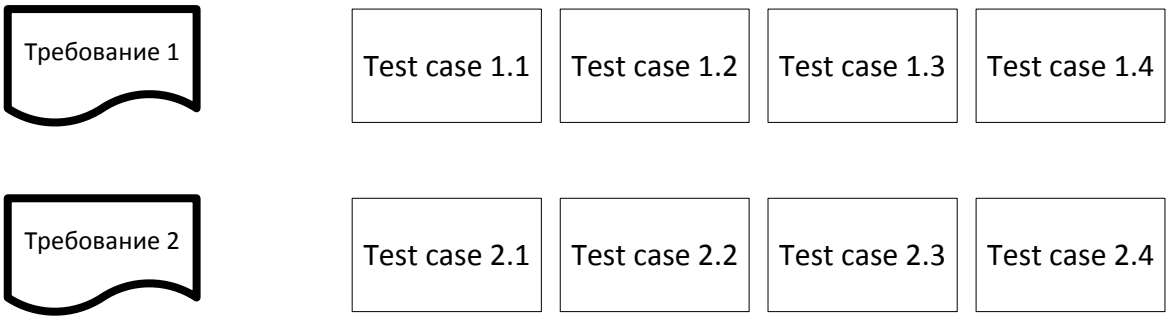


Тестовое покрытие

Тестовое покрытия – одна из важнейших метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

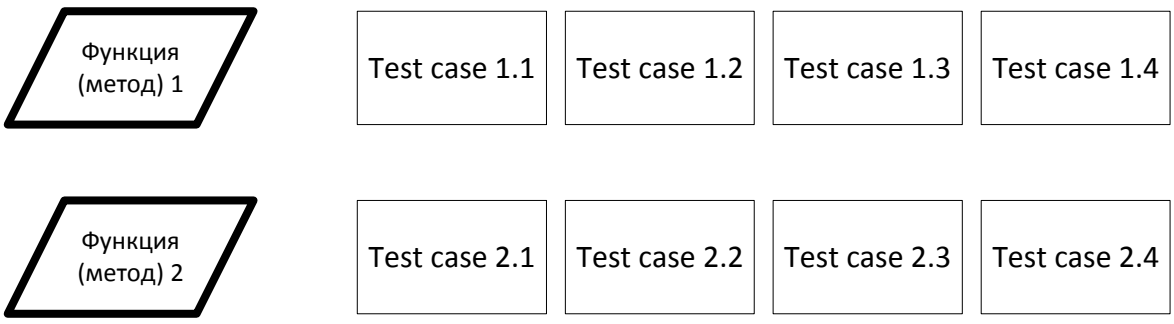
Покрытие требований (Requirements Coverage)

оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки (traceability matrix)



Покрытие кода (Code Coverage)

оценка покрытия исполняемого кода тестами, путем отслеживания непроверенных в процессе тестирования частей программного обеспечения.

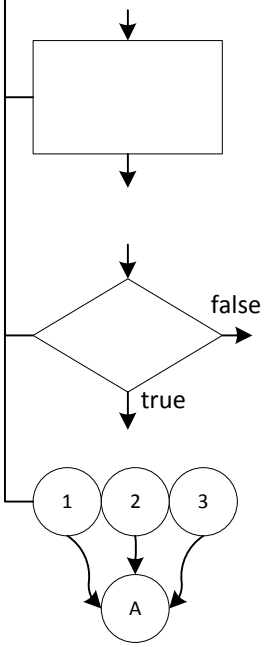


Тестовое покрытие

Покрытие на базе анализа потока управления

Оценка покрытия основанная на определении путей выполнения кода программного модуля и создания выполняемых тест кейсов для покрытия этих путей.

Фундаментом для тестирования потоков управления является построение графов потоков управления (Control Flow Graph)



	блок процесса - одна точка входа и одна точка выхода	Test case 1.1	Test case 1.2	Test case 1.3
false / true	точка альтернативы - одна точка входа, две и более точки выхода	Test case TRUE	Test case FALSE	Test case ALTERNATIVE
1 2 3 / A	точка соединения - две и более точек входа, одна точка выхода	Test case 1.A	Test case 2.A	Test case 3.A

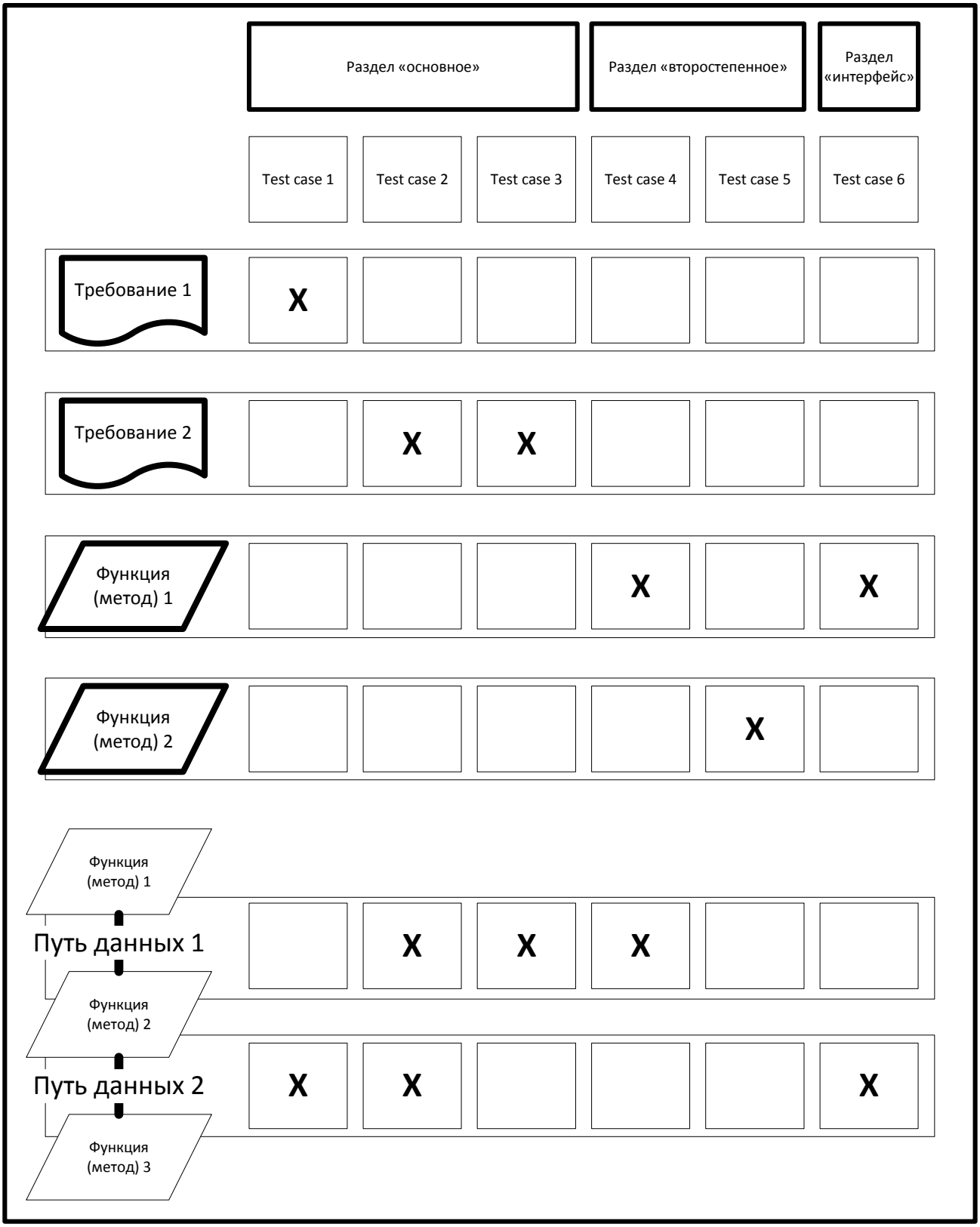
Уровни тестового покрытия

Уровень	Название	Краткое описание
Уровень 0	--	“Тестируй все что протестируешь, пользователи протестируют остальное” На английском языке это звучит намного элегантнее: *“Test whatever you test, users will test the rest”*
Уровень 1	Покрытие операторов	Каждый оператор должен быть выполнен как минимум один раз.
Уровень 2	Покрытие альтернатив [2] / Покрытие ветвей	Каждый узел с ветвлением (альтернатива) выполнен как минимум один раз.
Уровень 3	Покрытие условий	Каждое условие, имеющее TRUE и FALSE на выходе, выполнено как минимум один раз.
Уровень 4	Покрытие условий альтернатив	Тестовые случаи создаются для каждого условия и альтернативы
Уровень 5	Покрытие множественных условий	Достигается покрытие альтернатив, условий и условий альтернатив (Уровни 2, 3 и 4)
Уровень 6	“Покрытие бесконечного числа путей”	Если, в случае зацикливания, количество путей становится бесконечным, допускается существенное их сокращение, ограничивая количество циклов выполнения, для уменьшения числа тестовых случаев.
Уровень 7	Покрытие путей	Все пути должны быть проверены

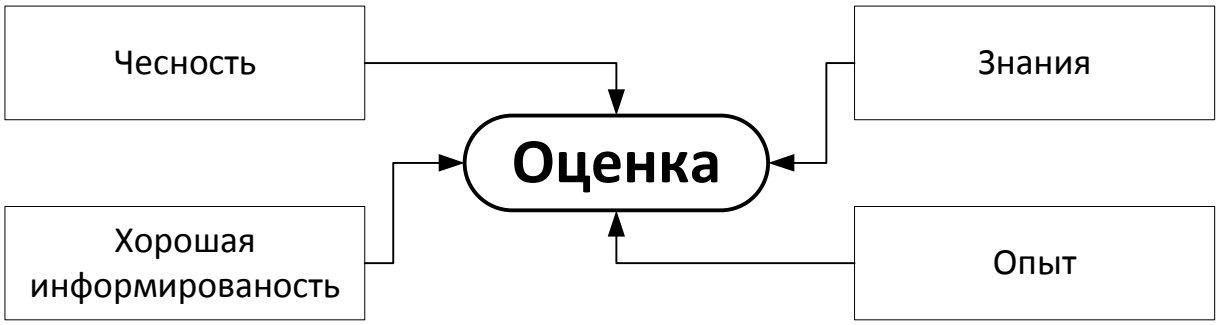
Матрица покрытия

Таблица, содержащая отношение требований к подготовленным тестовым сценариям (Test Cases)

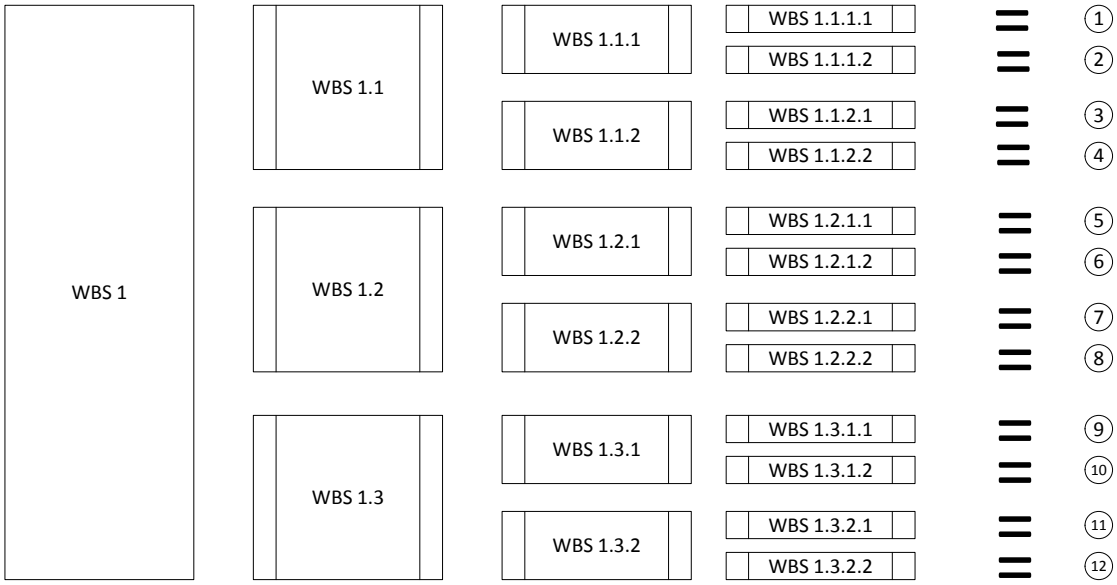
В заголовках колонок – требования, в заголовках строк – тестовые сценарии. На пересечении – отметка покрытия



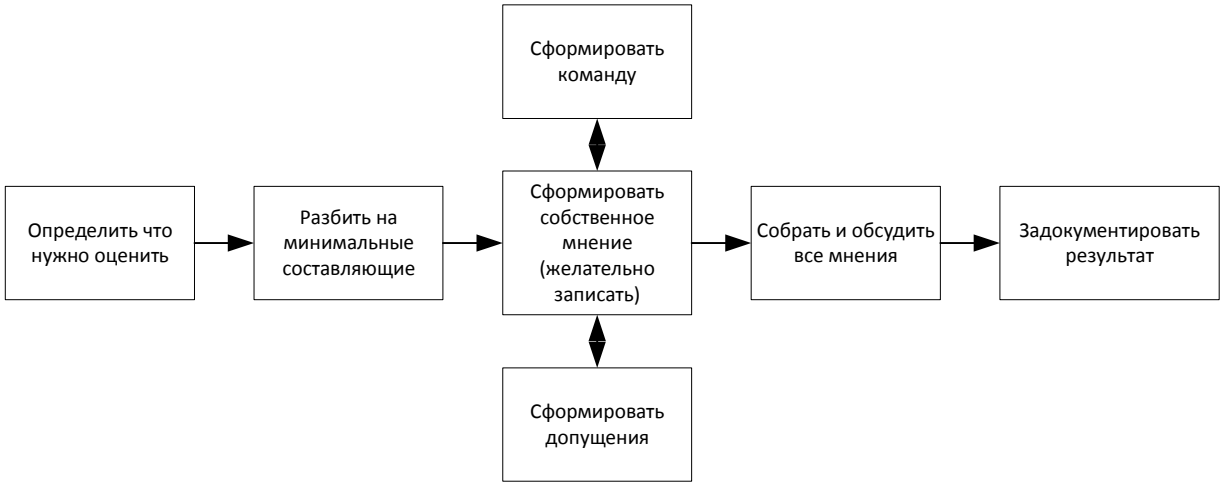
ESTIMATE TIME



WORK BREAKDOWN STRUCTURE



АЛГОРИТМ ОЦЕНКИ ВРЕМЕНИ



СОБЕСЕДОВАНИЕ

Сможет	Будет	Подойдет
Сможет ли человек решать поставленные задачи? (Опыт)	Будет ли он это делать? (Потенциал)	Подойдет ли он в данную команду, ритм работу проект? (Химия)

Задачки на логическое мышление | **Задачки на сообразительность** | **Задачки на внимательность**

- Заставляют кандидата думать, анализировать, строить предположения, генерировать идеи

Тестирование реальных предметов
- Предметы знакомы
- Принципы не сильно отличаются

Тестирование абстрактных программ
- Реальные задачи
- Плацдарм для новых вопросов

Тестирование реальных программ
- Более глубокий уровень понимания
- Охват многих областей

Игра «ДА/НЕТки» | **Заказкик-исполнитель** | **Ситуации из практики**

- Умение кандидата анализировать полученные ответы и строить новые

Ожидания

- Понимание всей картины
  - Понимание продукта
    - Расстановка приоритетов
- Как правильно совместить все части
  - Как должно работать с большей системой
    - Интеграция с другими компонентами
- Организация
  - Системный подход – составления плана
    - Разбитие на подпроцесы
- Практичность
  - Как правильно написать воспроизводимый тест-кейс

Тестирование предмета

- Кто будет использовать и зачем
  - Целевая аудитория
- Варианты использования
  - Обсудить и уточнить
- Каковы границы использования
  - Что? Где? Кем? Каким образом? Может исполльзоватся
- Условия отказа (Стресс-тест)
  - Каков отказ
    -Блокирующий
    -Информирующий
    -Фатальный
- Как осуществить тестирование
  - Автоматизация
    Рочное

СОБЕСЕДОВАНИЕ

Поиск и устранение неисправностей

Информация о проэкте

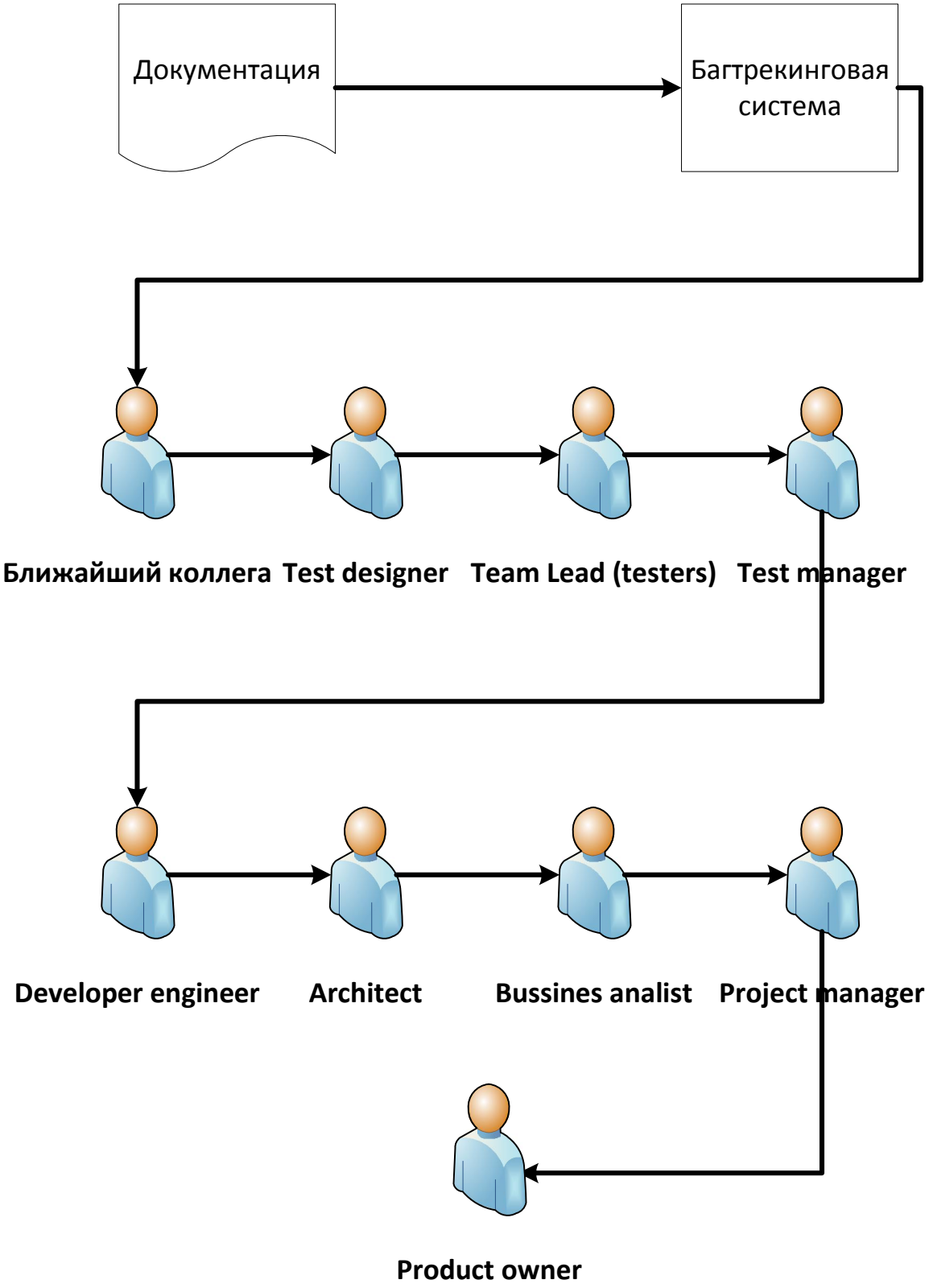
Процессы	Персонал	График работы	Оборудывание	Инструменты
Информационное обеспечение	Бюджет	Логистика	Домен	Критерии качества

Первый день

1	Знакомство с командой
2	Узнать Ментора (куратора)
3	Обсудить с ментором задачи на близжайшее время
4	Изучение продукта – ищем всю возможную информацию о продукте и изучаем её
5	Изучение рабочей зоны
6	Отчет

Первый день

ПОИСК ИНФОРМАЦИИ О ПРОДУКТЕ



КАЧЕСТВА ТЕСТИРОВЩИКА

Воспитаность - обладание тактом и дипломатичностью	**Грамотность** - Умение устно и письменно излагать проблему
Понимание - Способность представить себя на месте другого	**Наблюдательность и внимательность** - умение замечать неточности и противоречия
Логическое мышление	**Креативность, инициативность, азарт**
Алогичное мышление – умение мыслить глубже, шире и не так как того требует логика	**Критическое мышление** – умение во всем находить недостатки
Ясновиденье - умение предвидеть где находятся ошибки	**Многозадачность** - Умение одновременно выполнять несколько задач
Таймменеджмент - Умение распределять свое время и рабоать по расписанию	Склонность **эксперементировать** -

ИНСТРУМЕНТЫ ТЕСТИРОВЩИКА

ОСОБОЕ ВНИМАНИЕ

ЛОГИЧЕСКИЕ УТВЕРЖДЕНИЯ

Запрещенные к использованию

ERROR GUESSING