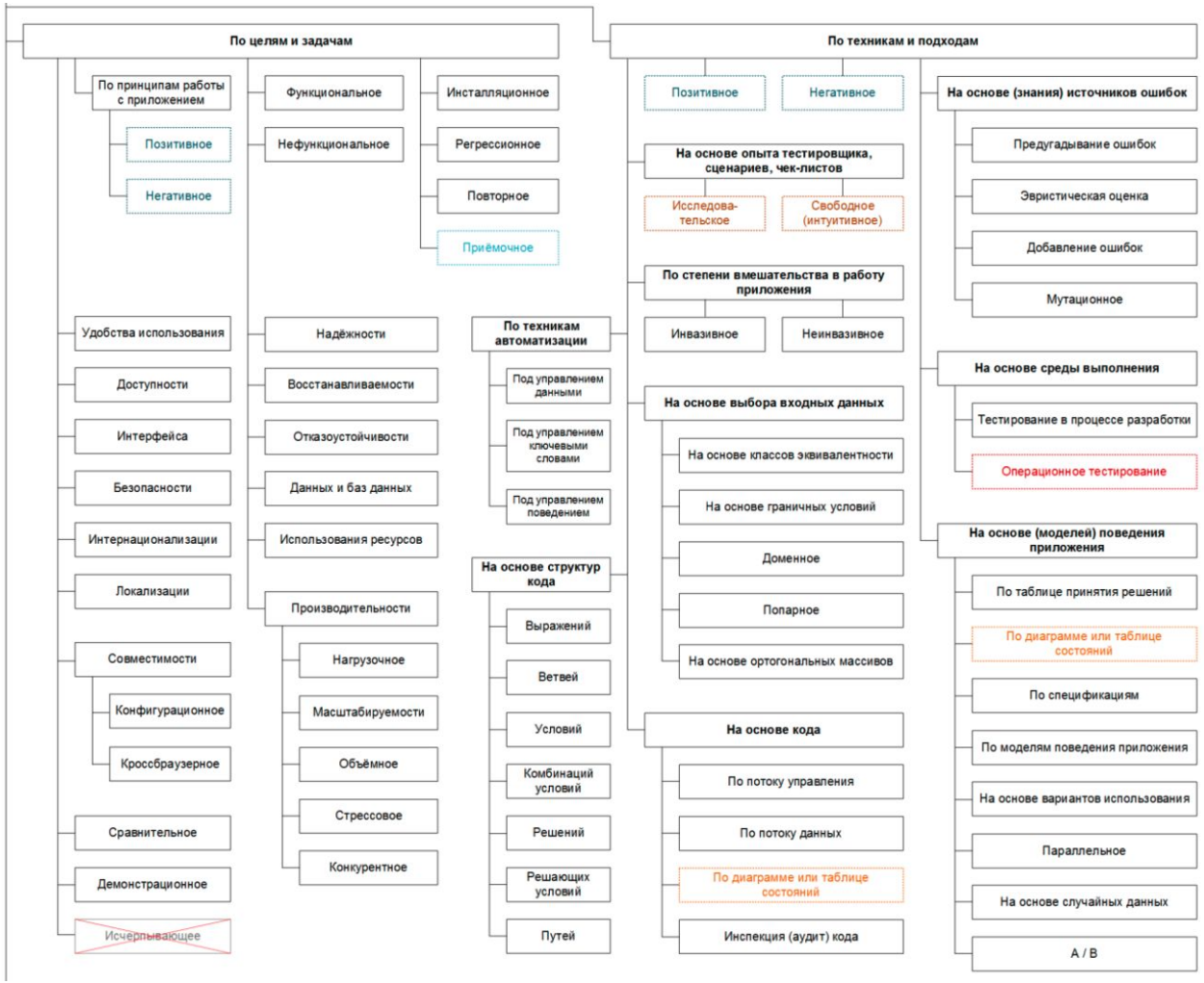
Классификация методов тестирования


Классификация тестирования
По запуску кода на исполнение
Статическое тестирование
Динамическое тестирование
По уровню детализации приложения (по уровню тестирования)
Модульное тестирование
Интеграционное тестирование
Системное тестирование
По фокусировке на уровне архитектуры приложения
Уровень представления
Уровень бизнес-логики
Уровень данных
По доступу к коду и архитектуре приложения
Метод белого ящика
Метод чёрного ящика
Метод серого ящика
По (убыванию) степени важности тестируемых функций (по уровню функционального тестирования)
«Дымовое» («смоук»)
Критического пути
Расширенное
По привлечению конечных пользователей
Альфа-тестирование
Бета-тестирование
Гамма-тестирование
По степени автоматизации
Ручное
Автоматизированное (и «автоматическое»)
По природе приложения
Веб
Мобильное
Настольное
...
По степени формализации
На основе тест-кейсов
Исследова-тельское
Свободное (интуитивное)


По целям и задачам
По принципам работы с приложением
Позитивное
Негативное
Функциональное
Нефункциональное
Инсталляционное
Регрессионное
Повторное
Приёмочное
Удобства использования
Доступности
Интерфейса
Безопасности
Интернационализации
Локализации
Совместимости
Конфигурационное
Кроссбраузерное
Сравнительное
Демонстрационное
Исчерпывающее
Надёжности
Восстанавливаемости
Отказоустойчивости
Данных и баз данных
Использования ресурсов
Производительности
Нагрузочное
Масштабируемости
Объёмное
Стрессовое
Конкурентное
По техникам и подходам
Позитивное
Негативное
На основе опыта тестировщика, сценариев, чек-листов
Исследова-тельское
Свободное (интуитивное)
По степени вмешательства в работу приложения
Инвазивное
Неинвазивное
На основе выбора входных данных
На основе классов эквивалентности
На основе граничных условий
Доменное
Попарное
На основе ортогональных массивов
На основе кода
По потоку управления
По потоку данных
По диаграмме или таблице состояний
Инспекция (аудит) кода
По техникам автоматизации
Под управлением данными
Под управлением ключевыми словами
Под управлением поведением
На основе структур кода
Выражений
Ветвей
Условий
Комбинаций условий
Решений
Решающих условий
Путей
На основе (знания) источников ошибок
Предугадывание ошибок
Эвристическая оценка
Добавление ошибок
Мутационное
На основе среды выполнения
Тестирование в процессе разработки
Операционное тестирование
На основе (моделей) поведения приложения
По таблице принятия решений
По диаграмме или таблице состояний
По спецификациям
По моделям поведения приложения
На основе вариантов использования
Параллельное
На основе случайных данных
A / B


По моменту выполнения (хронологии)
Общая универсальная логика последовательности тестирования
Позитивное (простое)
Негативное (простое)
Позитивное (сложное)
Негативное (сложное)
По иерархии компонентов
Восходящее
Нисходящее
Гибридное
По концентрации внимания на требованиях и их составляющих
Тестирование требований
Тестирование функциональных составляющих
Тестирование нефункциональных составляющих
Типичные общие сценарии
Типичный общий сценарий 1
«Дымовое» («смоук»)
Критического пути
Расширенное
Типичный общий сценарий 2
Модульное тестирование
Интеграционное тестирование
Системное тестирование
Гибридное тестирование
Типичный общий сценарий 3
Альфа-тестирование
Бета-тестирование
Гамма-тестирование
Приёмочное тестирование
Операционное тестирование

Классификация по запуску кода на исполнение

- Статическое тестирование (без запуска кода). Тестированию подвергаются:
  - Документы (требования, тест-кейсы, схемы БД и т.д.)
  - Графические прототипы
  - Код приложения (review, статический анализ)
  - Параметры (конфигурация) среды исполнения
  - Подготовленные тестовые данные
- Динамическое тестирование (с запуском кода).
  - Системное тестирование (всё приложение целиком)
  - Интеграционное тестирование (код нескольких взаимосвязанных частей)
  - Модульное (компонентное) тестирование (код отдельных частей)

Классификация по доступу к коду и архитектуре приложения

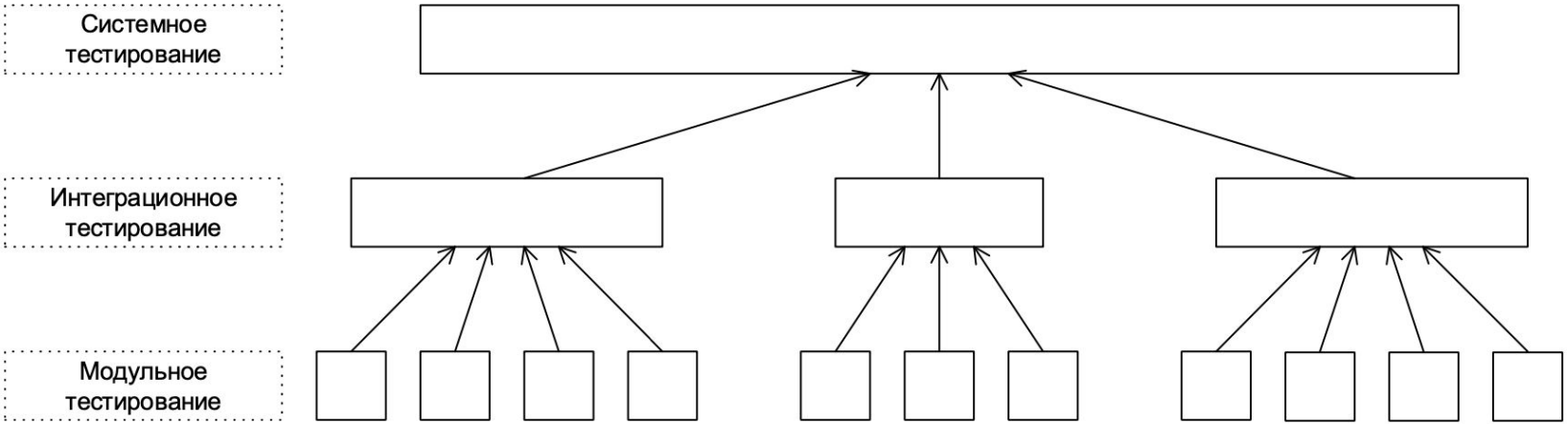
- Метод “белого ящика” - у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.
- Метод “черного ящика” - у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к ним в процессе тестирования.
- Метод “серого ящика” - комбинация методов белого ящика и чёрного ящика, состоящая в том, что к части кода и архитектуры у тестировщика доступ есть, а к части — нет.

	Преимущества	Недостатки
Метод белого ящика	• Показывает скрытые проблемы и упрощает их диагностику. • Допускает достаточно простую автоматизацию тест-кейсов и их выполнение на самых ранних стадиях развития проекта. • Обладает развитой системой метрик, сбор и анализ которых легко автоматизируется. • Стимулирует разработчиков к написанию качественного кода. • Многие техники этого метода являются проверенными, хорошо себя зарекомендовавшими решениями, базирующимися на строгом техническом подходе.	• Не может выполняться тестировщиками, не обладающими достаточными знаниями в области программирования. • Тестирование сфокусировано на реализованной функциональности, что повышает вероятность пропуска нереализованных требований. • Поведение приложения исследуется в отрыве от реальной среды выполнения и не учитывает её влияние. • Поведение приложения исследуется в отрыве от реальных пользовательских сценариев.
Метод черного ящика	• Тестировщик не обязан обладать (глубокими) знаниями в области программирования. • Поведение приложения исследуется в контексте реальной среды выполнения и учитывает её влияние. • Поведение приложения исследуется в контексте реальных пользовательских сценариев. • Тест-кейсы можно создавать уже на стадии появления стабильных требований. • Процесс создания тест-кейсов позволяет выявить дефекты в требованиях. • Допускает создание тест-кейсов, которые можно многократно использовать на разных проектах.	• Возможно повторение части тест-кейсов, уже выполненных разработчиками. • Высока вероятность того, что часть возможных вариантов поведения приложения останется не протестированной. • Для разработки высокоэффективных тест-кейсов необходима качественная документация. • Диагностика обнаруженных дефектов более сложна в сравнении с техниками метода белого ящика. • В связи с широким выбором техник и подходов затрудняется планирование и оценка трудозатрат. • В случае автоматизации могут потребоваться сложные дорогостоящие инструментальные средства.

Классификация по степени автоматизации

- Ручное тестирование — тестирование, в котором тест-кейсы выполняются человеком вручную без использования средств автоматизации.
- Автоматизированное тестирование — набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования.

Классификация по уровню детализации приложения



Классификация по уровню тестирования

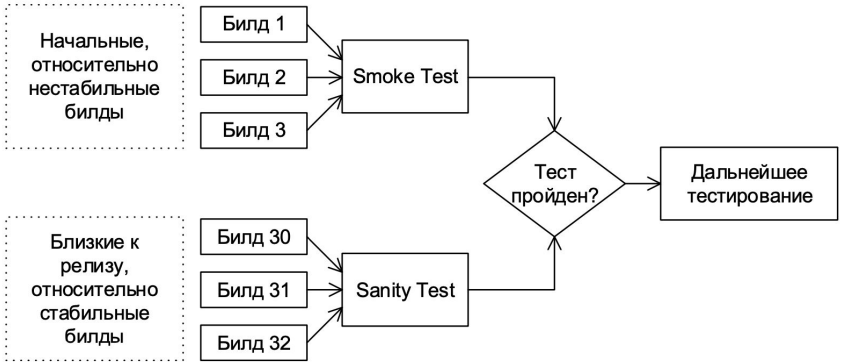
Alpha vs Beta тестирование

- Why do u call this version “beta”?
- Cuz it’s beta than nothin’!

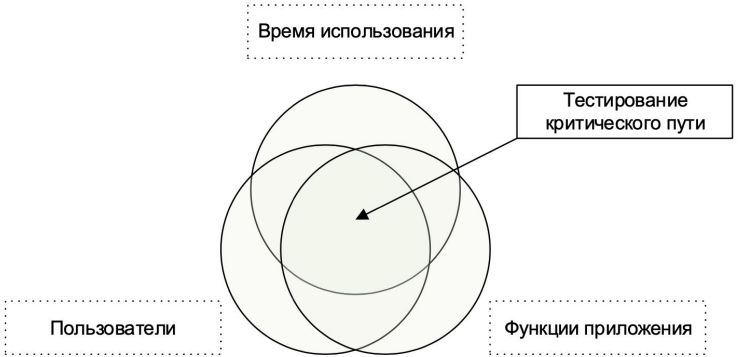
	Альфа-тестирование	Бета-тестирование
Кто	Тестировщики и разработчики.	Клиенты/заказчики или конечные пользователи.
Когда	Ближе к концу процесса разработки, продукт в почти полностью работоспособном состоянии.	Ближе к релизу.
Сколько	Может потребоваться долгий цикл. Альфа-тестирование нередко в 3-5 раз длительнее бета-тестирования.	Несколько недель (иногда месяцев) с небольшим количеством основных итераций.
Цель	Повысить качество продукта и обеспечить готовность к бета-тестированию.	Повысить качество продукта и проверить готовность продукта к использованию реальными пользователями.
Требования к среде	Требует лабораторной и тестовой среды.	Не требует лабораторной и тестовой среды, проводится в режиме реального времени.
Методы тестирования	Используются все известные и актуальные для данного продукта методы тестирования.	Тестируют с применением реальности и воображения. Исследуют каждый элемент продукта в своей родной среде.
Фиксы (устранение багов)	Большинство известных критических проблем исправлены, некоторые функции могут быть изменены или добавлены в результате ранней обратной связи.	Большая часть собранной обратной связи рассматривается и/или применяется в будущих версиях продукта. Выполняются только важные/критические изменения.
Результат	Имеем представление о качестве продукта и соответствует ли он документации.	Узнаем, что пользователи думают о продукте, в чем видят его улучшения.

Классификация по (убыванию) степени важности тестируемых функций (1)

- **Дымовое тестирование** (*smoke test, intake test, build verification test*) направлено на проверку самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения (или иного объекта, подвергаемого дымовому тестированию).



- **Тестирование критического пути** (*critical path test*) направлено на исследование функциональности, используемой типичными пользователями в типичной повседневной деятельности.
- **Расширенное тестирование** (*extended test*) направлено на исследование всей заявленной в требованиях функциональности — даже той, которая низко проранжирована по степени важности.



Классификация по (убыванию) степени важности тестируемых функций (2)