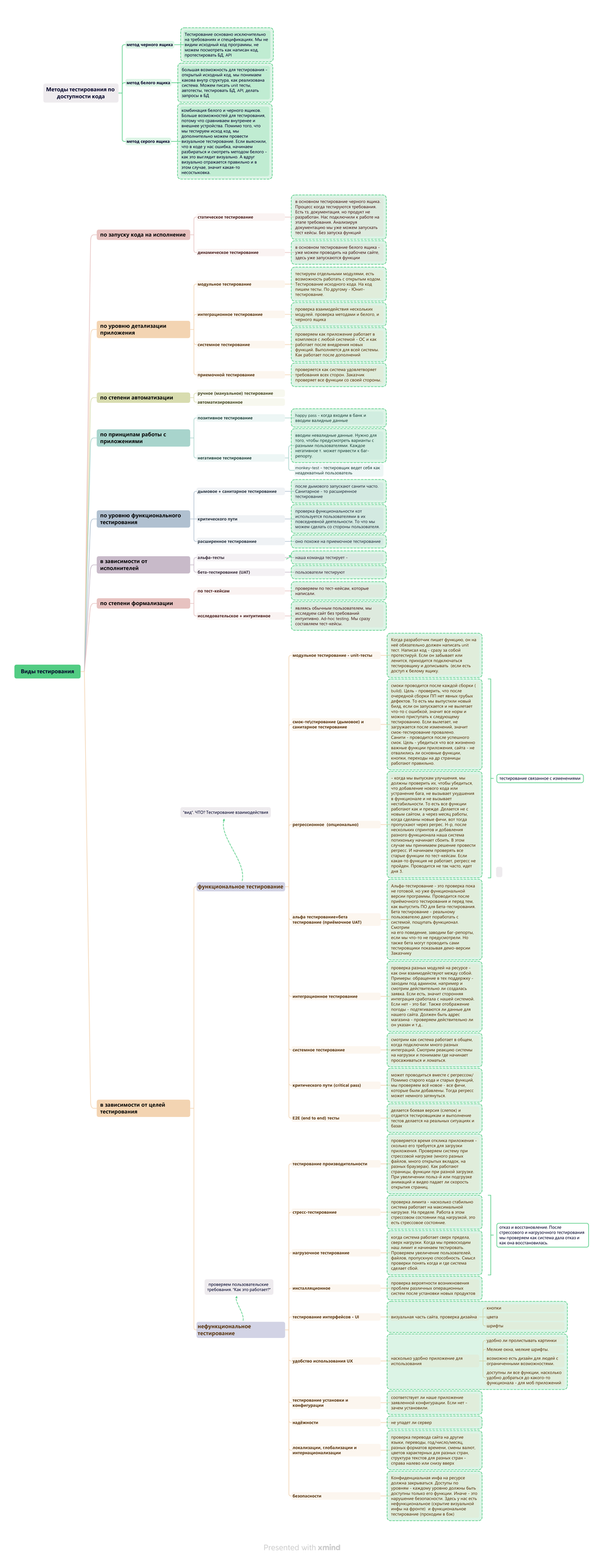
Методы тестирования по доступности кода
метод черного ящика
Тестирование основано исключительно на требованиях и спецификациях. Мы не видим исходный код программы, не можем посмотреть как написан код, протестировать БД, API
метод белого ящика
большая возможность для тестирования - открытый исходный код, мы понимаем какова внутр структура, как реализована система. Можем писать unit тесты, автотесты, тестировать БД, API, делать запросы в БД
метод серого ящика
комбинация белого и черного ящиков. Больше возможностей для тестирования, потому что сравниваем внутреннее и внешнее устройства. Помимо того, что мы тестируем исход код, мы дополнительно можем провести визуальное тестирование. Если выяснили, что в коде у нас ошибка, начинаем разбираться и смотреть методом белого - как это выглядит визуально. А вдруг визуально отражается правильно и в этом случае, значит какая-то несостыковка.
Виды тестирования
по запуску кода на исполнение
статическое тестирование
в основном тестирование черного ящика. Процесс когда тестируются требования. Есть ТЗ, документация, но продукт не разработан. Нас подключили к работе на этапе требования. Анализируя документацию мы уже можем запускать тест кейсы. Без запуска функций
динамическое тестирование
в основном тестирование белого ящика - уже можем проводить на рабочем сайте, здесь уже запускаются функции
по уровню детализации приложения
модульное тестирование
тестируем отдельными модулями, есть возможность работать с открытым кодом. Тестирование исходного кода. На код пишем тесты. По другому - Юнит-тестирование.
интеграционное тестирование
проверка взаимодействия нескольких модулей. проверка методами и белого, и черного ящика
системное тестирование
проверяем как приложение работает в комплексе с любой системой - ОС и как работает после внедрения новых функций. Выполняется для всей системы. Как работает после дополнений
приемочной тестирование
проверяется как система удовлетворяет требования всех сторон. Заказчик проверяет все функции со своей стороны.
по степени автоматизации
ручное (мануальное) тестирование
автоматизированное
по принципам работы с приложениями
позитивное тестирование
happy pass - когда входим в банк и вводим валидные данные
негативное тестирование
вводим невалидные данные. Нужно для того, чтобы предусмотреть варианты с разными пользователями. Каждое негативное т. может привести к баг-репорту.
monkey-test - тестировщик ведет себя как неадекватный пользователь
по уровню функционального тестирования
дымовое + санитарное тестирование
после дымового запускают санити часто. Санитарное - то расширенное тестирование
критического пути
проверка функциональности кот используется пользователями в их повседневной деятельности. То что мы можем сделать со стороны пользователя.
расширенное тестирование
оно похоже на приемочное тестирование
в зависимости от исполнителей
альфа-тесты
наша команда тестирует -
бета-тестирование (UAT)
пользователи тестируют
по степени формализации
по тест-кейсам
проверяем по тест-кейсам, которые написали.
исследовательское + интуитивное
являясь обычным пользователем, мы исследуем сайт без требований интуитивно. Ad-hoc testing. Мы сразу составляем тест-кейсы.
в зависимости от целей тестирования
функциональное тестирование
"вид". ЧТО? Тестирование взаимодействия
модульное тестирование - unit-тесты
Когда разработчик пишет функцию, он на неё обязательно должен написать unit тест. Написал код - сразу за собой протестируй. Если он забывает или ленится, приходится подключаться тестировщику и дописывать (если есть доступ к белому ящику).
смок-те\стирование (дымовое) и санитарное тестирование
смоки проводится после каждой сборки (build). Цель - проверить, что после очередной сборки ПП нет явных грубых дефектов. То есть мы выпустили новый билд, если он запускается и не вылетает что-то с ошибкой, значит все норм и можно приступать к следующему тестированию. Если вылетает, не загружается после изменений, значит смок-тестирование провалено. Санити - проводится после успешного смок. Цель - убедиться что все жизненно важные функции приложения, сайта - не отвалились ли основные функции, кнопки, переходы на др страницы работают правильно.
регрессионное (опционально)
- когда мы выпускам улучшения, мы должны проверить их, чтобы убедиться, что добавление нового кода или устранение бага, не вызывает ухудшения в функционале и не вызывает нестабильности. То есть все функции работают как и прежде. Делается не с новым сайтом, а через месяц работы, когда сделаны новые фичи, вот тогда пропускают через регрес. Н-р, после нескольких спринтов и добавления разного функционала наша система потихоньку начинает сбоить. В этом случае мы принимаем решение провести регресс. И начинаем проверять все старые функции по тест-кейсам. Если какая-то функция не работает, регресс не пройден. Проводится не так часто, идет дня 3.
тестирование связанное с изменениями
альфа тестирование+бета тестирование (приёмочное UAT)
Альфа-тестирование - это проверка пока не готовой, но уже функциональной версии программы. Проводится после приёмочного тестирования и перед тем, как выпустить ПО для Бета-тестирования. Бета тестирование - реальному пользователю дают поработать с системой, пощупать функционал. Смотрим на его поведение, заводим баг-репорты, если мы что-то не предусмотрели. Но также бета могут проводить сами тестировщики показывая демо-версии Заказчику
интеграционное тестирование
проверка разных модулей на ресурсе - как они взаимодействуют между собой. Примеры: обращение в тех поддержку - заходим под админом, например и смотрим действительно ли создалась заявка. Если есть, значит сторонняя интеграция сработала с нашей системой. Если нет - это баг. Также отображение погоды - подтягиваются ли данные для нашего сайта. Должен быть адрес магазина - проверяем действительно ли он указан и т.д.
системное тестирование
смотрим как система работает в общем, когда подключили много разных интеграций. Смотрим реакцию системы на нагрузки и понимаем где начинает просаживаться и ломаться.
критического пути (critical pass)
может проводиться вместе с регрессом/ Помимо старого кода и старых функций, мы проверяем всё новое - все фичи, которые были добавлены. Тогда регресс может немного затянуться.
E2E (end to end) тесты
делается боевая версия (слепок) и отдается тестировщикам и выполнение тестов делается на реальных ситуациях и базах
нефункциональное тестирование
проверяем пользовательские требования. "Как это работает?"
тестирование производительности
проверяется время отклика приложения - сколько его требуется для загрузки приложения. Проверяем систему при стрессовой нагрузке (много разных файлов, много открытых вкладок, на разных браузерах). Как работают страницы, функции при разной загрузке. При увеличении польз-й или подгрузке анимаций и видео падает ли скорость открытия страниц.
стресс-тестирование
проверка лимита - насколько стабильно система работает на максимальной нагрузке. На пределе. Работа в этом стрессовом состоянии под нагрузкой, это есть стрессовое состояние.
нагрузочное тестирование
когда система работает сверх предела, сверх нагрузки. Когда мы превосходим наш лимит и начинаем тестировать. Проверяем увеличение пользователей, файлов, пропускную способность. Смысл проверки понять когда и где система сделает сбой.
отказ и восстановление. После стрессового и нагрузочного тестирования мы проверяем как система дала отказ и как она восстановилась.
инсталляционное
проверка вероятности возникновения проблем различных операционных систем после установки новых продуктов
тестирование интерфейсов - UI
визуальная часть сайта, проверка дизайна
кнопки
цвета
шрифты
удобство использования UX
насколько удобно приложение для использования
удобно ли пролистывать картинки
Мелкие окна, мелкие шрифты.
возможно есть дизайн для людей с ограниченными возможностями.
доступны ли все функции, насколько удобно добраться до какого-то функционала - для моб приложений
тестирование установки и конфигурации
соответствует ли наше приложение заявленной конфигурации. Если нет - зачем установили.
надёжности
не упадет ли сервер
локализации, глобализации и интернационализации
проверка перевода сайта на другие языки. переводы. год/число/месяц разных форматов времени, смены валют, цветов характерных для разных стран, структура текстов для разных стран - справа налево или снизу вверх
безопасности
Конфиденциальная инфа на ресурсе должна закрываться. Доступы по уровням - каждому уровню должны быть доступны только его функции. Иначе - это нарушение безопасности. Здесь у нас есть нефункциональное (скрытие визуальной инфы на фронте) и функциональное тестирование (проходим в бэк)
Presented with xmind