ozon{tech

Теория: краткая вводная

Небольшой экскурс в теоретические основы терминологии тестирования.

План занятия

Методология

Тестирование — комплекс мероприятий, направленный на проведение проверок на соответствие производимого продукта требованиям, к нему предъявляемым (прямым и косвенным).

Тестирование — процесс получения информации о соответствии продукта требованиям, которые к нему предъявляются.

Поиск ошибок — только как побочный эффект!

Цель тестирования — предоставление актуальной информации о соответствии производимого продукта требованиям.

Agile-процессы

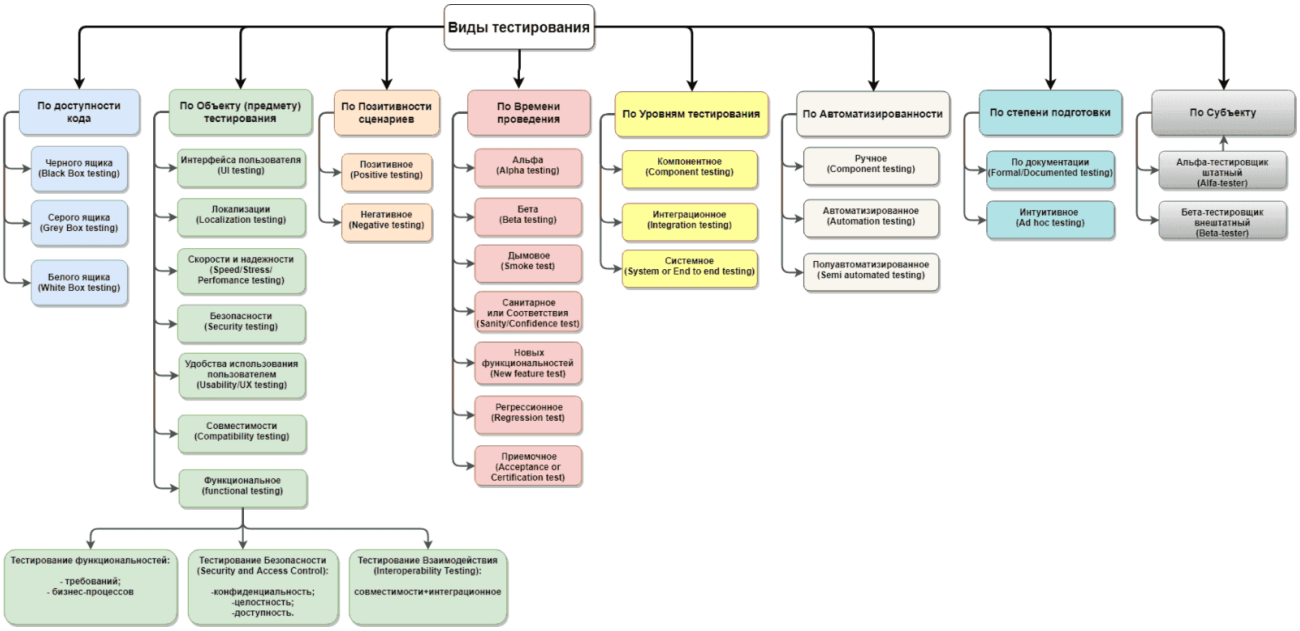
SDLC

- Анализ требований — «Какие проблемы требуют решений?»
- Планирование — «Что мы хотим сделать?»
- Проектирование и дизайн — «Как мы добьемся наших целей?»
- Разработка ПО регулирует процесс создания продукта.
- Тестирование регулирует обеспечение качественной работы продукта.
- Развертывание регулирует использование финального продукта.

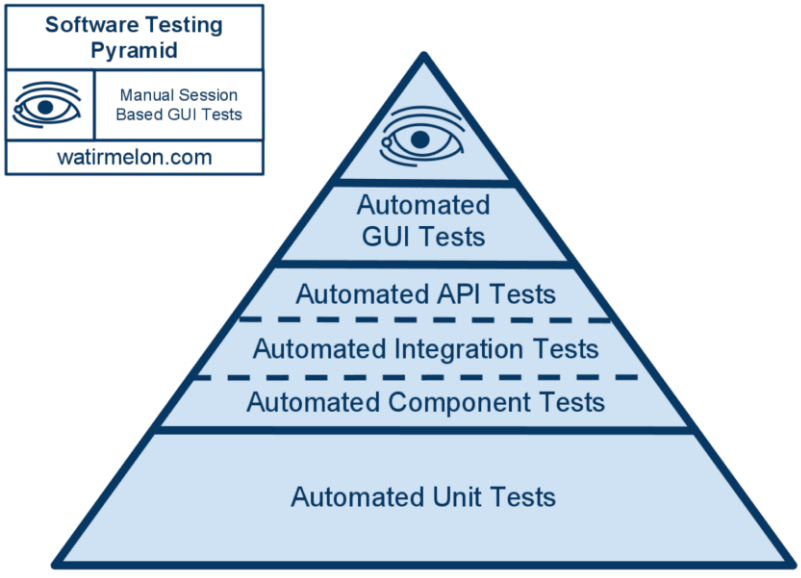
Водопадная модель

Итеративная модель

Виды тестирования



Пирамида тестирования

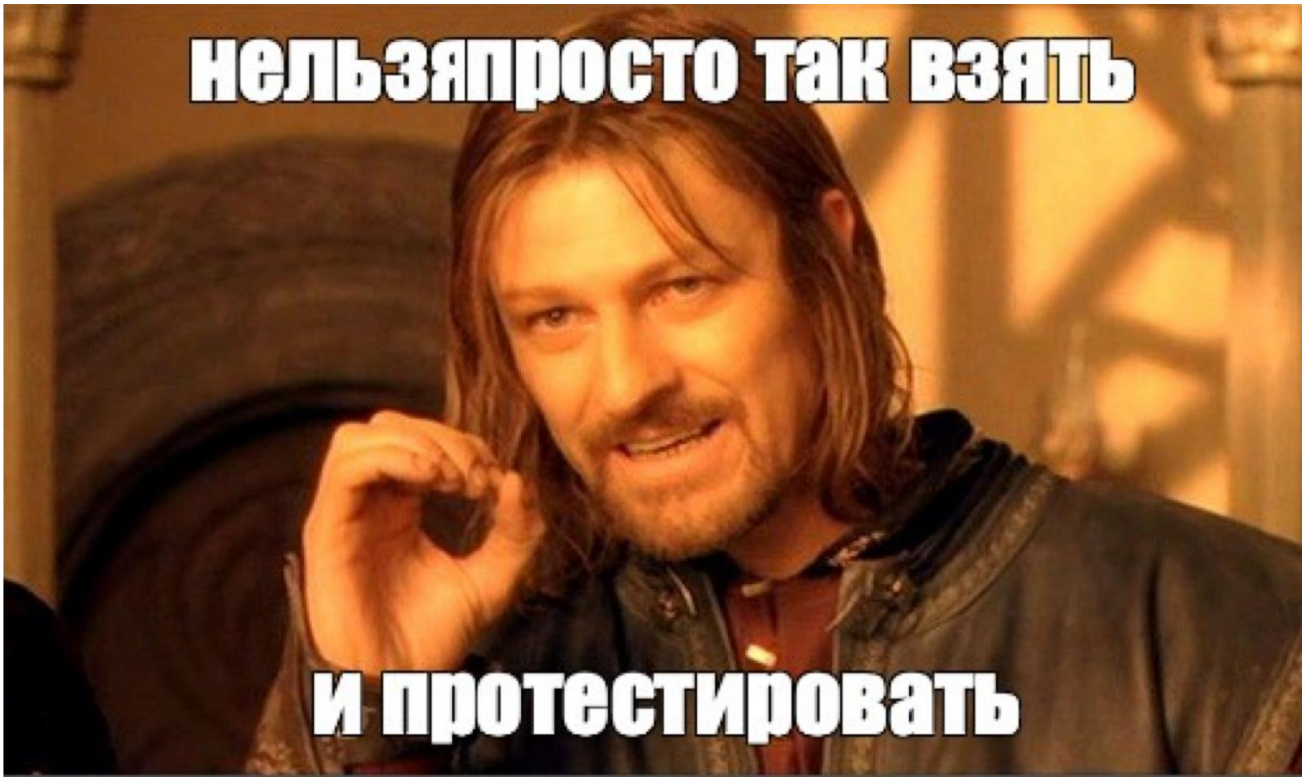


Семь основных принципов тестирования

Наличие дефектов (а не их отсутствие)

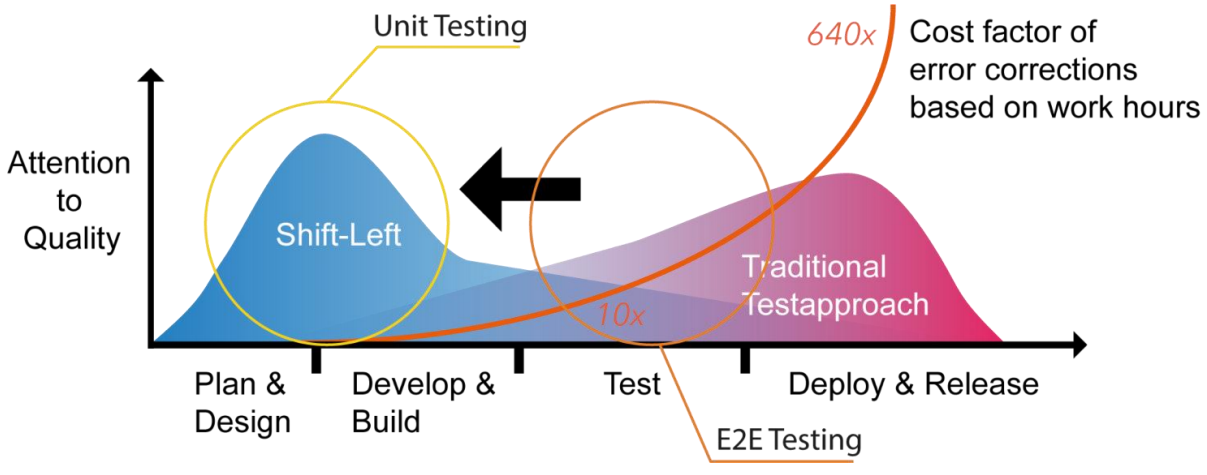
Тестирование снижает вероятность того, что в ПО останутся необнаруженные дефекты. Даже если дефекты не обнаружены, это доказывает корректность тестирования.

Исчерпывающее тестирование невозможно



Раннее тестирование (shift-left)

Чем раньше обнаружен дефект, тем дешевле его исправление.



Выводы:

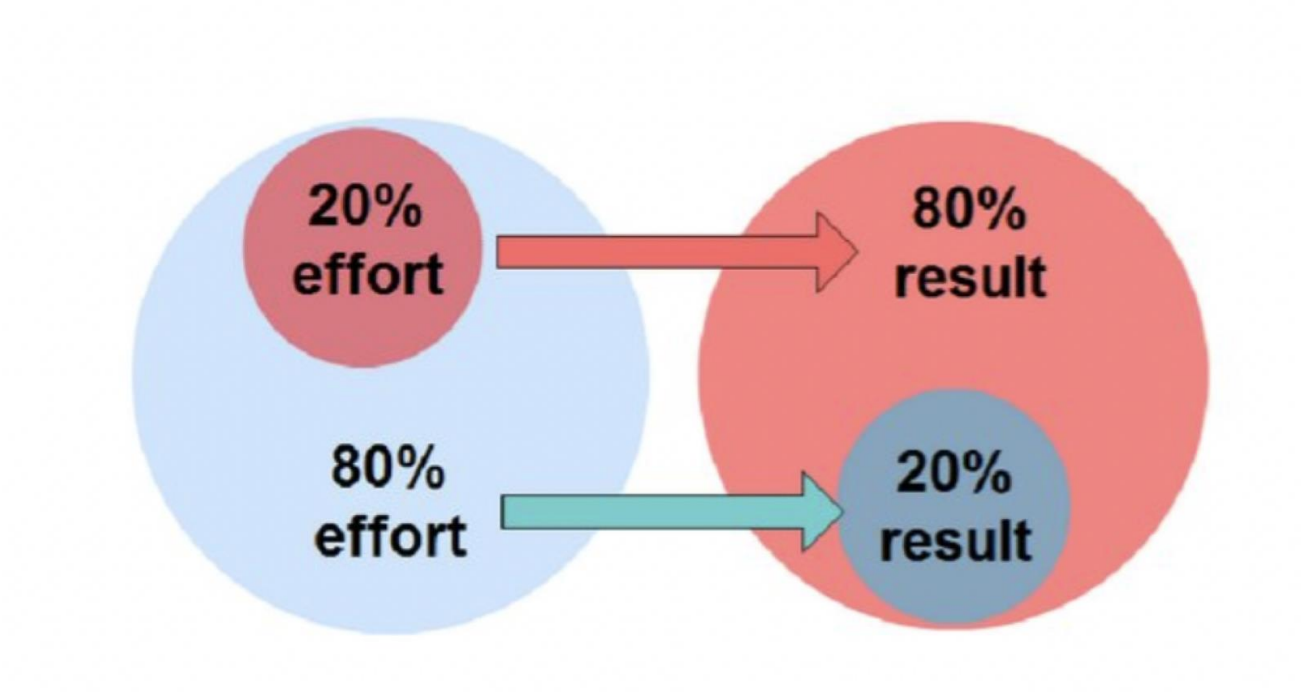
- Лучше чередовать подходы shift-left и классический подход к QA.
- Результат очевиден, если у QA хорошая экспертиза в продукте.
- Shift-left — больше аналитическая работа.
- Переосмысление всего процесса и подхода к тестированию.

Кластеризация дефектов

Большая часть дефектов, как правило, находится в небольшом количестве модулей.

Группировка дефектов по какому-то явному признаку — хороший повод продолжить исследование этой области ПО: скорее всего, именно здесь будет обнаружено еще больше дефектов.

Для эффективного тестирования следует распределить усилия с учетом реальной плотности багов в модулях продукта. Если это первое тестирование — пропорционально ожидаемой плотности.



Парадокс пестицида

Эффект, при котором при регулярном прогоне тестовых сценариев ошибки становятся устойчивыми к тестам.

Зависимость от контекста

Невозможно выработать универсальный подход на все случаи жизни. Следует учитывать как общие, так и уникальные свойства текущего проекта и соответствующе выстраивать тестирование.

Заблуждение об отсутствии ошибок

Тестирование снижает вероятность того, что в ПО останутся не обнаруженные дефекты. Но даже если дефекты не обнаружены, это не является доказательством правильности.

Отсутствие дефектов — не самоцель для QA!

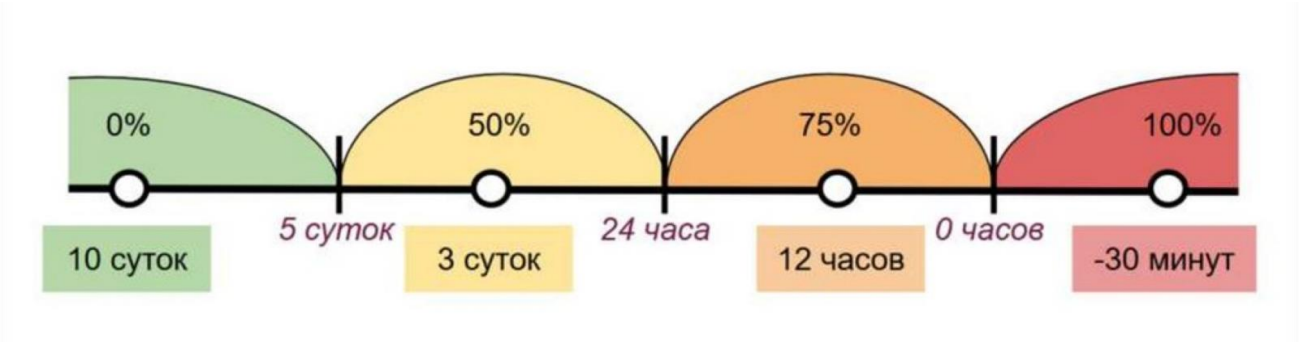
Тест-дизайн

Классы эквивалентности

Разбиение всего набора тестов на диапазоны значений ради сокращения количества тестирования без потери качества.

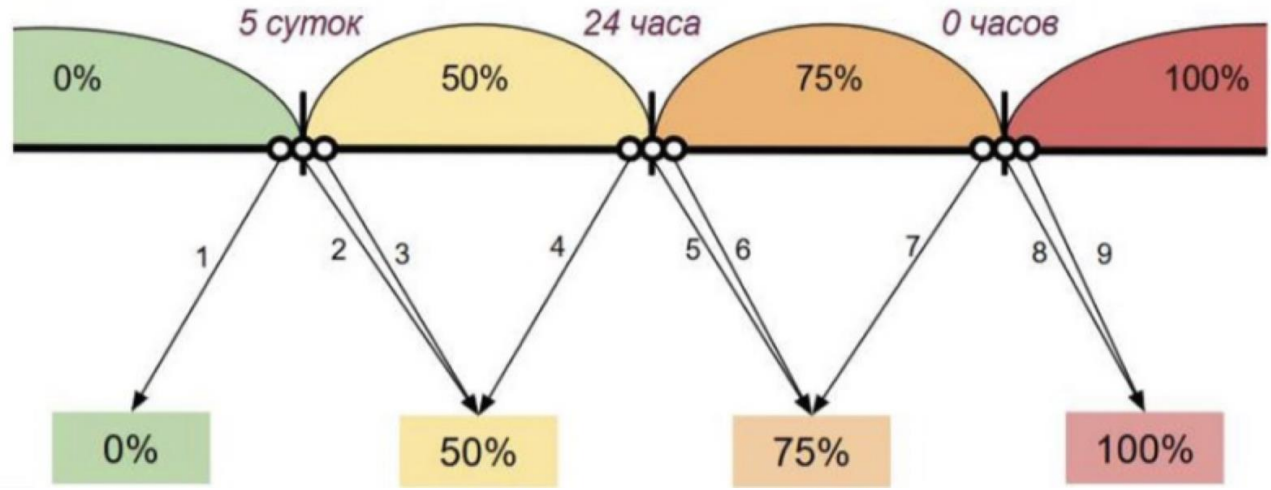
Классов не должно быть слишком много, иначе тесты станут избыточными.

Если классов недостаточно, растет риск пропустить ошибку.



Граничные условия

Используется как логическое продолжение идеи разбиения диапазона на классы. Даже если классы эквивалентности выбраны верно, зачастую неверно толкуется принадлежность значений на границах класса.



Причина-следствие

Проверка базовых действий и их результата.

- Ввод имени и номера телефона.
- Причина: нажатие на кнопку «Добавить».
- Следствие: добавление данных в БД.

Таблица принятия решений

Метод хорош тем, что наглядно показывает сразу все возможные сценарии.

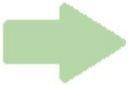
Условия	Значения	Правила				
Логин	Верный, Неверный	Верный	Верный	Верный	Неверный	Неверный
Пароль	Верный, Неверный	Верный	Верный	Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Неверный	Не пришел	Не пришел	Не пришел
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

Условия	Значения	Правила			
Логин	Верный, Неверный	Верный	Неверный	Верный, Неверный	Верный
Пароль	Верный, Неверный	Верный	Верный, Неверный	Неверный	Верный
Код	Верный, Неверный, Не пришел	Верный	Верный, Неверный, Не пришел	Верный, Неверный, Не пришел	Неверный
Действия	Пользователь вошел на сайт Пользователь не авторизован	Пользователь вошел на сайт	Пользователь не авторизован	Пользователь не авторизован	Пользователь не авторизован

Попарное тестирование

В качестве гипотезы принимается идея о том, что баг возникает на пересечении не n-факторов, а только двух. Так, можно значительно уменьшить тестовое покрытие без потерь в качестве.

Google Chrome	Windows	RU
Google Chrome	Windows	EN
Google Chrome	Linux	RU
Google Chrome	Linux	EN
Opera	Windows	RU
Opera	Windows	EN
Opera	Linux	RU
Opera	Linux	EN



Google Chrome	Windows	RU
Google Chrome	Linux	EN
Opera	Windows	EN
Opera	Linux	RU

Предугадывание ошибок

Эмпирический опыт на основе знания продукта и собственной экспертизы QA-инженера.

Пример: тестирование карандаша

Терминология

- Объект тестирования: карандаш с ластиком, для рисования
- Сроки: 1 час на тесты
- Ресурсы: 1 QA, бумага, точилка
- Артефакты: стратегия/план тестирования, отчет, баг-репорты

Уточняем требования

- Уровень твердости (L/X/M)
- Наличие ластика (функциональный)
- Основа для рисования (бумага)
- Повторное использование (можно затачивать при потере функций)

Дополнительно

- Нефункциональные требования (условия отказа)

Описываем методологию тестирования

- Функциональное
- Модульное (грифель пишет, ластик стирает)
- Интеграционное (компоненты работают совместно)
- UX/Usability (удобство пользования)
- Тестирование совместимости (ластик подходит карандашу)

- Нефункциональное (performance) (быстро испишется, часто придется точить)

Анализируем результаты, отвечая на такие вопросы

- Какие сценарии/виды тестирования были проведены?
- Готов ли продукт для эксплуатации заказчиком?
- Насколько созданный продукт соответствует требованиям?
- Какие баги (с учетом критичности) были найдены?
- Мешают ли найденные ошибки пользоваться продуктом?
- Необходима ли дальнейшая доработка/исправление багов?

Заключение

Мы кратко прошлись по основной теории, используемой в тестировании: обсудили терминологию, основные принципы тестирования, трансформацию тестирования как процесса в современных реалиях. Рассмотрели на примерах основные принципы тест-дизайна: что помогает в проектировании собственных тестов, упрощая разработку без потерь в качестве покрытия.

Полученные знания пригодятся при непосредственном создании автотестов. Далее поговорим о настройке окружения на примерах IDE PyCharm (JetBrains).

Материалы

- https://svyatoslav.biz/software_testing_book
- https://www.gasq.org/files/content/gasq/downloads/certification/ISTQB/Foundation%20Level/ISTQB_CTFL_Syllabus_2018-RU.pdf
- https://highload.today/vot-karandash-testiruj-pochemu-na-sobesedovanii-prosyat-protestit-bytovoj-predmet-i-kak-eto-sdelat
- https://www.simbirsoft.com/blog/tekhniki-test-dizayna-i-ikh-prednaznachenie/
- https://habr.com/ru/company/dcmiran/blog/521718/
- https://vladislaveremeev.gitbook.io/qa_bible