Quality Assurance
QA

Обеспечение качества

1. Определение и связь с тестированием

Обеспечение качества (англ. *quality assurance, QA*) — это процесс или результат формирования требуемых свойств и характеристик продукции по мере её создания, а также — поддержание этих характеристик при хранении, транспортировании и эксплуатации продукции.

QA != тестирование!
Это меры по обеспечению качества на всех жизненных циклах продукта, от анализа бизнес требований до передачи продукта заказчику.

QA появилась задолго до IT, во многих отраслях производства QA уже развита очень хорошо. В IT QA еще продолжает развиваться.

Тестирование это лишь один из методов контроля качества (QC). А QC лишь один из механизмов QA.

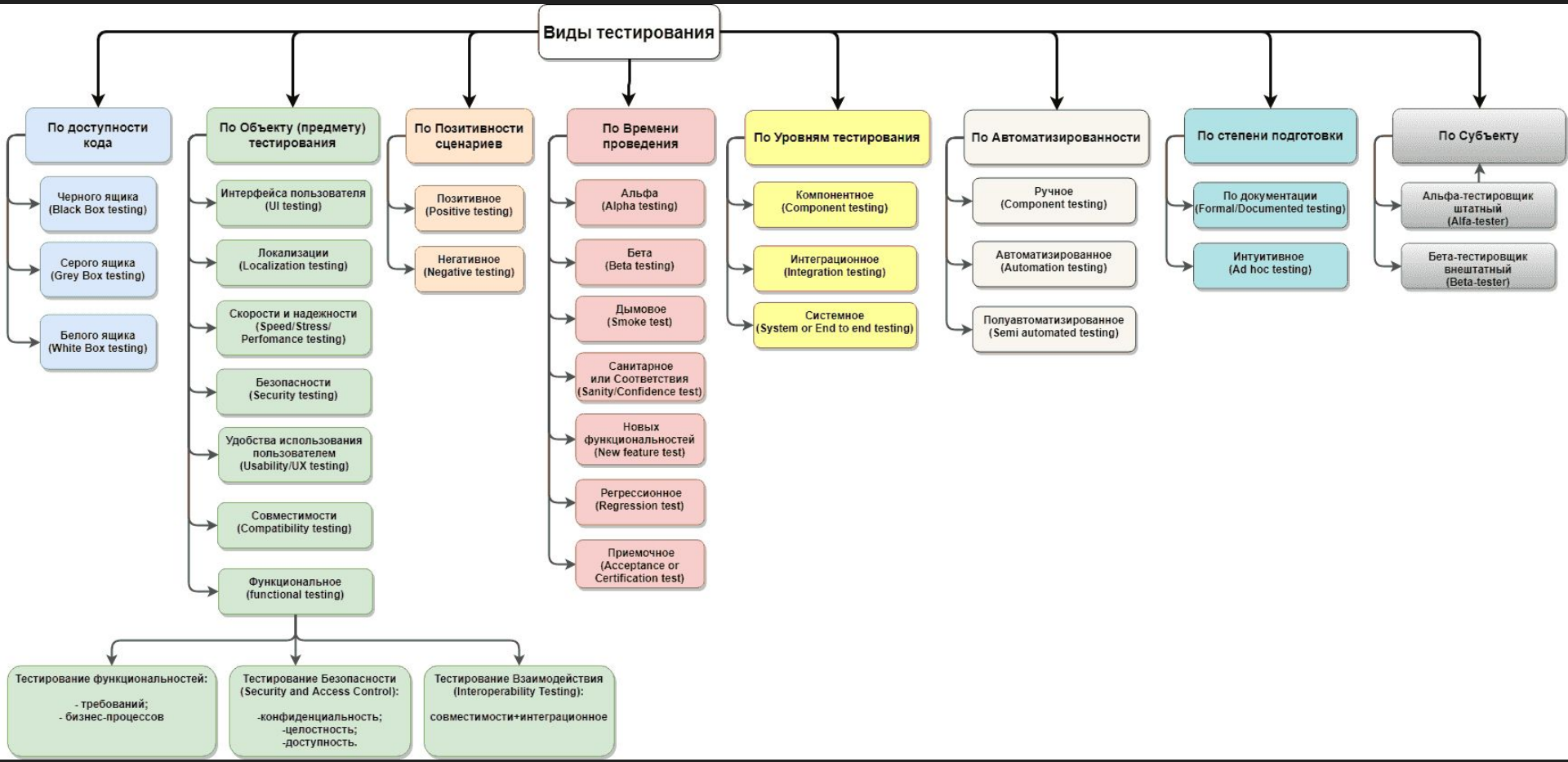
- QA - думает как обеспечить качество (опыт и квалификация)
- QC - думает как проверить качество (опыт и квалификация)
- Test-инженер - разрабатывает тест кейсы (минимальный опыт и квалификация)
- Тестировщик - выполняет поставленную задачу по тестированию (без опыта, без квалификации)

2. QA в IT

Если мы хотим добиться высокого качества продукта, то следует подключать QA как можно раньше, обеспечение качества должно быть на всех этапах.

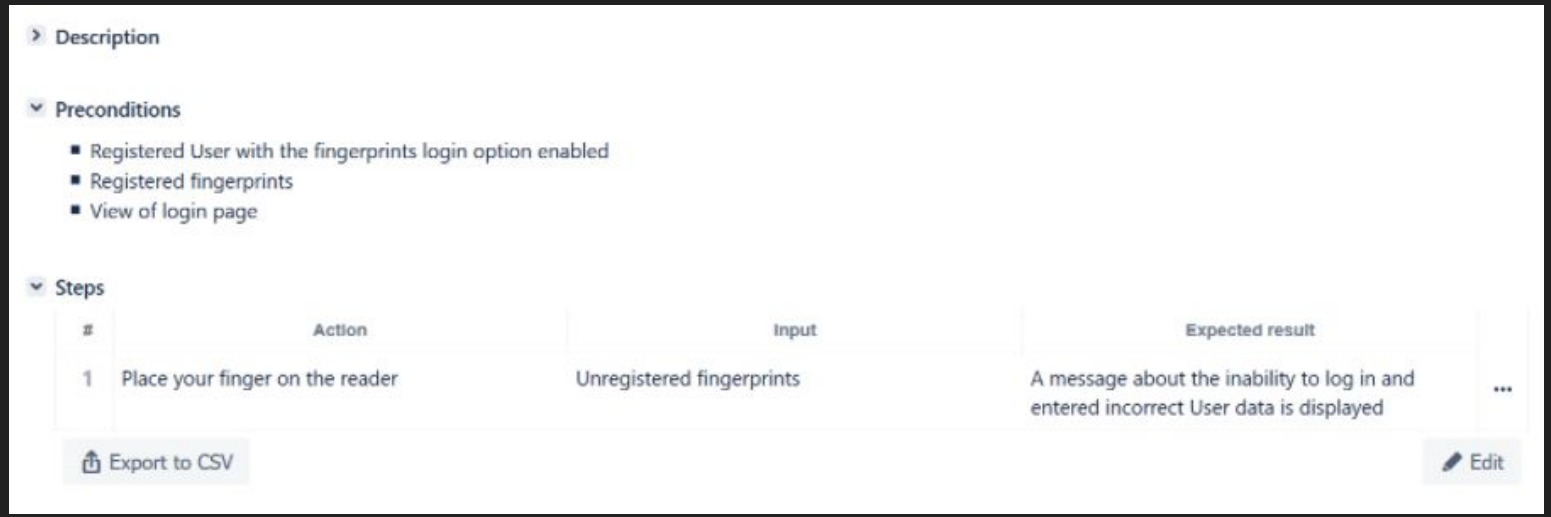
- Анализ и формирование требований => QA валидирует SRS, BRD (полнота, однозначность, непересекаемость);
- Проектирование и архитектура => QA вникает в архитектуру, ищет узкие места, тестирует прототипы (тестовые планы, приемочное тестирование);
- Разработка => QA внедряет меры по педупреждению дефектов (code-review, unit-test);
- Разработка => QA тестирует отдельные готовые части, готовит документацию, определяет метрики (тест кейсы, тестовая документация);
- Выпуск release => QA принимает решение готов ли продукт к выпуску!

QA может менять и корректировать бизнес-процессы и инструменты, если это поможет повысить качество.


Виды тестирования
По доступности кода
Черного ящика (Black Box testing)
Серого ящика (Grey Box testing)
Белого ящика (White Box testing)
По Объекту (предмету) тестирования
Интерфейса пользователя (UI testing)
Локализации (Localization testing)
Скорости и надежности (Speed/Stress/ Perfomance testing)
Безопасности (Security testing)
Удобства использования пользователем (Usability/UX testing)
Совместимости (Compatibility testing)
Функциональное (functional testing)
Тестирование функциональностей:
- требований;
- бизнес-процессов
Тестирование Безопасности (Security and Access Control):
-конфиденциальность;
-целостность;
-доступность.
Тестирование Взаимодействия (Interoperability Testing):
совместимости+интеграционное
По Позитивности сценариев
Позитивное (Positive testing)
Негативное (Negative testing)
По Времени проведения
Альфа (Alpha testing)
Бета (Beta testing)
Дымовое (Smoke test)
Санитарное или Соответствия (Sanity/Confidence test)
Новых функциональностей (New feature test)
Регрессионное (Regression test)
Приемочное (Acceptance or Certification test)
По Уровням тестирования
Компонентное (Component testing)
Интеграционное (Integration testing)
Системное (System or End to end testing)
По Автоматизированности
Ручное (Component testing)
Автоматизированное (Automation testing)
Полуавтоматизированное (Semi automated testing)
По степени подготовки
По документации (Formal/Documented testing)
Интуитивное (Ad hoc testing)
По Субъекту
Альфа-тестировщик штатный (Alfa-tester)
Бета-тестировщик внештатный (Beta-tester)

4. Тестовая документация (входящая)

- Тест-план (что, где, когда и как тестируем)
- Тест-кейс (предусловие, шаги, ожидаемый результат, послесловие). Атомарный и автоматизируемый.



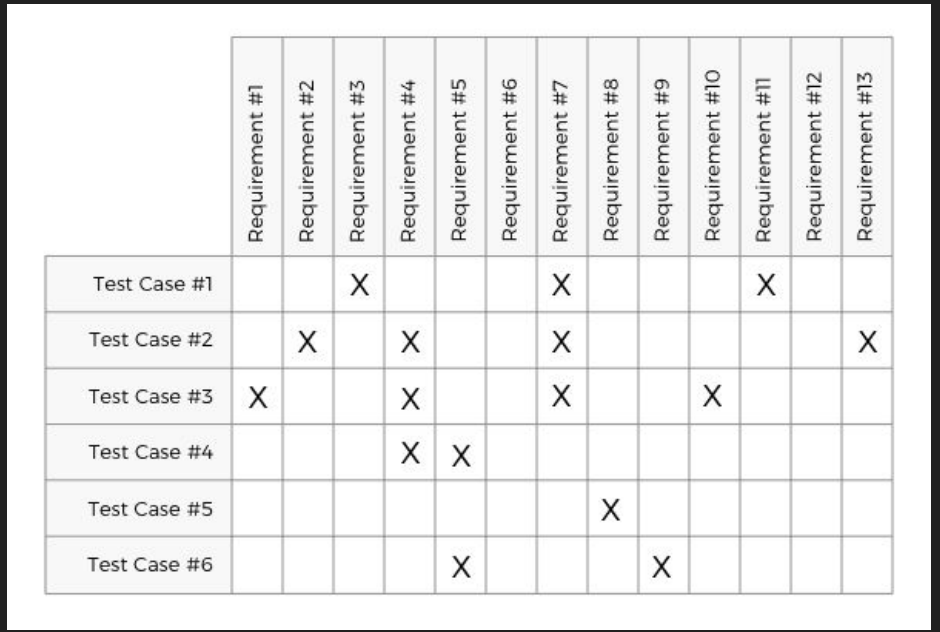
5. Документация после тестирования

- Bug report (тест-кейс с фактическим результатом или описание по воспроизведению)
- Отчет по тестированию (что, где, когда и как тестировалось). Может быть сформирован из тестового плана.
- TRM и Code Coverage

6. Метрики

Code Coverage - покрытие тестами строк кода,чаще используется для Unit тестов. Рассчитывается как процент отношения строк кода покрытых тестами / количеству строк кода.

Requirements Coverage - покрытие тестами требований к системе. Строится RTM (requirement traceability matrix).



	Requirement #1	Requirement #2	Requirement #3	Requirement #4	Requirement #5	Requirement #6	Requirement #7	Requirement #8	Requirement #9	Requirement #10	Requirement #11	Requirement #12	Requirement #13
Test Case #1			X				X				X		
Test Case #2		X		X			X						X
Test Case #3	X			X			X			X			
Test Case #4				X	X								
Test Case #5								X					
Test Case #6					X				X				

7. Тест дизайн

Для эффективности проводимого тестирования, следует использовать несколько техник:

1) Эквивалентных классов (Equivalence)
2) Граничных значений (Boundary value)
3) Исследовательское тестирование (Exploratory)
4) Предугадывание ошибки (Error guessing)
5) Отбор уникальных пар (Pair Wise)
6) Построение диаграммы состояний (State transition Tech)

Акция	Предзаказ	Ед. изм.
Да	Да	шт.
Нет	Нет	кг.

1) Да - Да - шт.
2) Да - Да - кг.
3) Да - Нет - шт.
4) Да - Нет - кг.
5) Нет - Да - шт.
6) Нет - Да - кг.
7) Нет - Нет - шт.
8) Нет - Нет - кг.

8. Инструменты тестирования

Багтрекинг:

- Jira
- Bugzilla
- Mantis

….

Управление тестами:

- Jira
- TestRail
- Zephyr

….

9. Автоматизация тестирования

Инструменты:

- Selenium (WebDriver)
- Sikuli
- Watir

Языки:

- JUnit
- NUnit
- PyUnit

Отчеты:

- Serenity

Selenium

- WebDriver - доступ к работе с браузером.
- Selenium IDE для записи теста (record and play для новичка).
- Selenium Server для прогона теста на удаленной машине.
- Selenium Grid для распределения тестов между машинами.

10. Пример Selenium теста

https://git.sebekon.ru/tokarev/selenium

Тест-кейс:
Блокирование входа без пароля

Шаги:
1) открываем страницу авторизации
2) Вводим в поле логина значение: "admin"
3) Оставляем поле пароля - пустым
4) Нажимаем кнопку "Войти"

Ожидаемый результат:
Вход не происходит, на экране выводится ошибка "Неправильный логин/пароль"

```
public function testEmptyCredentials()
    {
        // 1) открываем страницу авторизации
        $this->wd->get('https://crm.dryclean.ru/login.php');

        // 2) Вводим в поле логина значение: "admin"
        $loginInput = $this->wd->findElement(WebDriverBy::cssSelector('input#login'));
        $loginInput->sendKeys("admin");

        // 4) Нажимаем кнопку "Войти"
        $loginBtn = $this->wd->findElement(WebDriverBy::cssSelector('table.login +
table'));
        $loginBtn->click();

        // Ожидаемый результат: Вход не происходит, на экране выводится ошибка
"Неправильный логин/пароль"
        $errMess = $this->wd->findElement(WebDriverBy::cssSelector('div.errors'));
        $this->assertEquals('Неправильный логин/пароль', $errMess->getText());
    }
```

11. Как в Agile?

Документация:

- Чек листы вместо тест планов.
- Пишем тест-кейсы по необходимости (как минимум для Smoke)
- Test-item вместо тест-кейсов

Автоматизация:

- Только то, что возможно или важно (обычно Smoke)

Итог

Качество это не только тестирование.

Чем раньше найден дефект тем лучше. Тем дешевле будет его устранить.

Самые дорогие дефекты рождаются в бизнес требованиях и техническом задании.

Поэтому очень важной процедурой повышения качества является упреждение дефектов:

- Валидация требований
- Code-review
- Unit-test

Для тестирования хорошо бы вести чек-листы с test-item и тест-план со smoke с возможностью автоматизации.

QA о себе)

QA отдел нужно делать независимым от остальных, не над, не под, а как отдельный измерительный прибор, который можно подключить к любому этапу жизненного цикла. Это как совесть, которая будет капать на мозг, но при этом не позволит совершать глупости.

Для того, чтобы QA мог обеспечить качество, он должен участвовать на всех этапах разработки. Если он подключается лишь в конце - то это тестировщик.

Обычно отношение QA к Dev - это 1 к 3. На 10 разработчиков 3-4 QA.

Чем хуже документация, тем более первоклассные и опытные QA нужны.

Нужно дать возможность выстроить процесс обеспечения качества.

Самое главное - QA должны решать выпускать или не выпускать продукт на приёмку заказчиком.