**1. ПРИМЕНЕНИЕ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ**

При ручном подходе тест-кейсы запускаются вручную без использования программных средств. Собственно, это понятно из названия. При автоматизированном тестировании запуск тест-кейсов осуществляется при помощи специально разработанных скриптов.

Для краткосрочных проектов с малым количеством релизов выгоднее будет несколько раз провести ручное тестирование.

Чем более долгосрочный проект, чем более частые предполагаются релизы - тем более выгодным становится применение автоматизированного тестирования.

### Какие виды тестирования рекомендуется автоматизировать?

* **Регрессионное тестирование.** Данный вид тестирования – первый кандидат на автоматизацию из-за регулярного запуска тестов. На долгосрочных проектах автоматизация позволяет значительно сократить затраты на обеспечение качества продукта.
* **Нагрузочное тестирование.** Автоматизация нагрузочного тестирования позволяет быстрее получать результаты, экономить на мощностях и стоимости инструментов.
* **Тестирование локализации.** Если продукт будет выводиться на мировой рынок, его необходимо адаптировать к культурным аспектам разным странам. Локализация включает в себя перевод всех элементов интерфейса, служебных элементов, адаптацию режима отображения даты, времени, единиц измерения, валюты.

При этом если продукт регулярно обновляется, добавляется новая функциональность, то тестирование локализации выгоднее проводить в автоматическом режиме.

Нередко наиболее выигрышным сценарием является сочетание двух подходов. При этом доля автоматических и ручных тестов будет варьироваться в зависимости от требований проекта, бюджета, сроков, в которые должно уложиться тестирование, экспертизы команды.

тестовые случаи для автоматизации можно выбрать по следующим критериям:

* Высокие риски и сбои недопустимы – крайне актуально для банковской сферы.
* Тестовые сценарии, которые регулярно повторяются.
* Тестовые сценарии, которые очень сложны и утомительны для выполнения вручную.
* Тестовые примеры, отнимающие много времени.

Следующая категория тестовых случаев не подходит для автоматизации:

* Новые тестовые примеры, которые не выполнялись вручную хотя бы один раз.
* Сценарии тестирования, требования к которым часто меняются.
* Тестовые примеры, которые выполняются на разовой основе.

Автоматизированные скрипты позволяют получать более точные результаты, они доступны для повторного использования. Вручную можно протестировать практически любое приложение, иногда даже без предварительной подготовки.

Программное обеспечение для автоматизации тестирования также может вводить тестовые данные в тестовую среду, сравнивать ожидаемые и фактические результаты и создавать подробные отчеты о тестах.

**2. ПРОЦЕСС АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ**  
В процессе автоматизации выполняются следующие шаги:

1) **Выбор тестового инструмента**

2) **Определение объема автоматизации**

3) **Планирование, дизайн и разработка**

4) **Выполнение теста**

5) **Техническое обслуживание**

**1) Выбор тестового инструмента**Selenium Webdriver сегодня является наиболее популярным инструментом для автоматизации выполнения тестов в браузерах. Это инструмент тестирования программного обеспечения, используемый для регрессионного тестирования. Это инструмент тестирования с открытым исходным кодом, который предоставляет возможность воспроизведения и записи для регрессионного тестирования. Он также поддерживает большинство популярных языков программирования (Java, C#, Python, Ruby, JavaScript.)

**2) Определение объема автоматизации**  
Объем автоматизации – это область тестируемого приложения, которая будет автоматизирована. Его помогают определить следующие пункты:

* Функции, важные для бизнеса
* Сценарии с большим объемом данных
* Общие функции приложений
* Техническая осуществимость
* Частота повторного использования бизнес-компонентов
* Сложность тестовых случаев
* Возможность использовать одни и те же тестовые сценарии для кросс-браузерного тестирования

**3)** **Планирование, проектирование и разработка**  
На этом этапе вы создаете стратегию и план автоматизации, которые содержат следующие детали:

* Выбранные инструменты автоматизации
* Конструкция каркаса и его особенности
* Входящие и выходящие за рамки элементы автоматизации
* Подготовка стендов автоматизации
* График и временная шкала сценариев и выполнения
* Результаты тестирования автоматизации

4) Выпо**лнение теста**  
На этом этапе выполняются сценарии автоматизации. Сценариям необходимо ввести тестовые данные, прежде чем они будут запущены. После выполнения они предоставляют подробные отчеты об испытаниях.

**5) Обслуживание автоматизированного тестирования**  
Этот этап автоматизированного тестирования проводится для проверки того, как работают новые функции, добавленные в программное обеспечение: нормально или нет. Сопровождение в автотестировании выполняется, когда добавляются новые сценарии автоматизации, и их необходимо проверять и поддерживать, чтобы повышать эффективность сценариев автоматизации с каждым последующим циклом выпуска.

**3. ИНСТРУМЕНТ ТЕСТИРОВАНИЯ** **WEBDRIVER**

Selenium WebDriver - наиболее популярный пакет с открытым исходным кодом для автоматизации тестирования Web UI. Он позволяет тестам напрямую взаимодействовать со страницей в живом браузере. Selenium симулирует действия реального пользователя.  
По назначению Selenium WebDriver представляет собой драйвер браузера, то есть программную библиотеку, которая позволяет разрабатывать программы, управляющие поведением браузера.  
Selenium WebDriver, или просто WebDriver – это драйвер браузера, то есть не имеющая пользовательского интерфейса программная библиотека, которая позволяет различным другим программам взаимодействовать с браузером, управлять его поведением, получать от браузера какие-то данные и заставлять браузер выполнять какие-то команды.

WebDriver не имеет прямого отношения к тестированию. Он всего лишь предоставляет автотестам доступ к браузеру. На этом его функции заканчиваются.

Структурирование, группировку и запуск тестов, а также генерацию отчётов о тестировании, обеспечивает фреймворк тестирования, такой как Junit или TestNG для Java, Nunit или [Gallio](http://www.gallio.org/) для .Net, [RSpec](http://rspec.info/) или [Cucumber](http://cukes.info/) для Ruby и так далее. Разработка тестов ведётся в среде [Eclipse](http://www.eclipse.org/), [Intellij IDEA](http://www.jetbrains.com/idea/), [Visual Studio](http://www.microsoft.com/visualstudio/rus/products/visual-studio-overview), [RubyMine](http://www.jetbrains.com/ruby/) и так далее. Сборка осуществляется посредством [Maven](http://maven.apache.org/), [Gradle](http://www.gradle.org/), [Ant](http://ant.apache.org/), [NAnt](http://nant.sourceforge.net/),[Rake](http://rake.rubyforge.org/) и так далее. Запуск тестов по расписанию и публикацию отчётов выполняет сервер непрерывной интеграции – [Jenkins](http://jenkins-ci.org/), [CruiseControl](http://cruisecontrol.sourceforge.net/), [Bamboo](http://www.atlassian.com/software/bamboo/), [TeamCity](http://www.jetbrains.com/teamcity/) и так далее. И всё это – самостоятельные инструменты, не имеющие отношения к проекту Selenium.

**4. ПРИМЕРЫ ТЕСТОВ С ИСПОЛЬЗОВАНИЕМ WEBDRIVER**

В моем примере тесты написаны на Java, использовались **Intellij IDEA** + junit + Maven

В тесте orderVacuumCleaner\_1 тестируется сценарий:

перейти на домашнюю страницу интернет-магазина

>> в поле поиска ввести слово «Пылесос»

>> перейти на страницу, на которой представлены пылесосы

>> выбрать один из товаров

>> в открывшейся странице товара убедиться, что товар есть в наличии

>> отправить товар в корзину

>> перейти в корзину

>> проверить, что в корзине находится выбранный товар

>> удалить товар из корзины

>> проверить, что товар из корзины удален

>> закрыть браузер