АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ

АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ

Автоматизация тестирования — лучший способ повысить эффективность, покрытие тестами и скорость выполнения тестирования ПО.

Цель автоматизации — сократить количество тестовых случаев, которые нужно запустить вручную, а не полностью исключить ручное тестирование

Автоматизированное тестирование ПО важно потому, что:

- ✓ Ручное тестирование всех рабочих процессов, всех полей, всех негативных сценариев требует много времени и денег.
- ✓ Трудно вручную тестировать многоязычные сайты.
- ✓ Автоматизация тестирования в тестировании ПО не требует человеческого вмешательства. Можно запустить автоматизированный тест без присмотра (в течение ночи).
- ✓ Автоматизация тестирования увеличивает скорость выполнения тестов.
- ✓ Автоматизация помогает увеличить охват тестированием.
- ✓ Ручное тестирование может стать скучным и, следовательно, подверженным ошибкам.

Какие тестовые случаи автоматизировать?

Тестовые случаи для автоматизации можно выбрать, используя следующий критерий для повышения рентабельности инвестиций в автоматизацию:

- ✓ **Критически важные для бизнеса** тестовые случаи.
- ✓ Тестовые случаи, **которые выполняются многократно**.
- ✓ Тестовые случаи, **которые очень утомительны** или сложны для выполнения вручную.
- ✓ Тестовые случаи, **требующие много времени**.

Следующая категория тестовых случаев не подходит для автоматизации:

- ✓ Тестовые случаи, которые были разработаны заново и не были выполнены вручную хотя бы один раз.
- ✓ Тестовые случаи, требования к которым часто меняются.
- ✓ Тестовые случаи, которые выполняются на индивидуальной основе.

ПРОЦЕСС АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ

Шаг 1) Выбор инструмента тестирования

Шаг 2) Определите область автоматизации

Шаг 3) Планирование, проектирование и разработка

Шаг 4) Выполнение теста

Шаг 5) Техническое обслуживание

Шаг 1. Выбор инструмента тестирования

Выбор инструмента тестирования во многом зависит от технологии, на которой построено тестируемое приложение.

Шаг 2. Определите область автоматизации

Область автоматизации — это область вашего тестируемого приложения, которая будет автоматизирована. Следующие пункты помогают определить область:

- ✓ Функции, которые важны для бизнеса
- ✓ Сценарии с **большим объемом данных**
- ✓ **Общие функции** для всех приложений
- ✓ Техническая осуществимость
- ✓ Степень повторного использования бизнес-компонентов
- ✓ **Сложность** тестовых случаев
- ✓ Возможность использования одних и тех же тестовых случаев для кросс-браузерного тестирования.

Шаг 3. Планирование, проектирование и разработка

На этом этапе вы создаете стратегию и план автоматизации, которые содержат следующую информацию:

- ✓ Выбраны инструменты автоматизации
- ✓ Конструкция фреймворка и ее особенности
- ✓ Элементы автоматизации, входящие и выходящие за рамки
- ✓ Подготовка испытательного стенда автоматизации
- ✓ График и хронология написания и выполнения сценариев
- ✓ Результаты автоматизированного тестирования

Шаг 4. Выполнение теста

На этом этапе автоматизации выполняются скрипты.

Скриптам нужны входные тестовые данные, прежде чем они будут установлены для запуска. После выполнения они предоставляют подробные тестовые отчеты.

Выполнение может осуществляться напрямую с помощью инструмента автоматизации или через инструмент управления тестированием, который вызовет инструмент автоматизации.

> *Пример: Центр качества — это инструмент управления тестированием, который, в свою очередь, вызывает QTP (*QuickTest Professional*) для выполнения скриптов автоматизации.*

Скрипты могут быть выполнены на одной машине или на группе машин.

Выполнение может быть выполнено ночью, чтобы сэкономить время.

Шаг 5. Тестирование подхода к обслуживанию автоматизации

Подход к обслуживанию автоматизации тестирования — это фаза тестирования автоматизации, проводимая для проверки того, нормально или нет работают новые функции, добавленные в программное обеспечение.

Обслуживание в тестировании автоматизации выполняется, когда добавляются новые сценарии автоматизации и их необходимо пересматривать и поддерживать, чтобы повысить эффективность сценариев автоматизации с каждым последующим циклом выпуска.

Фреймворк для автоматизации

Фреймворк — это набор руководств по автоматизации, которые помогают в:

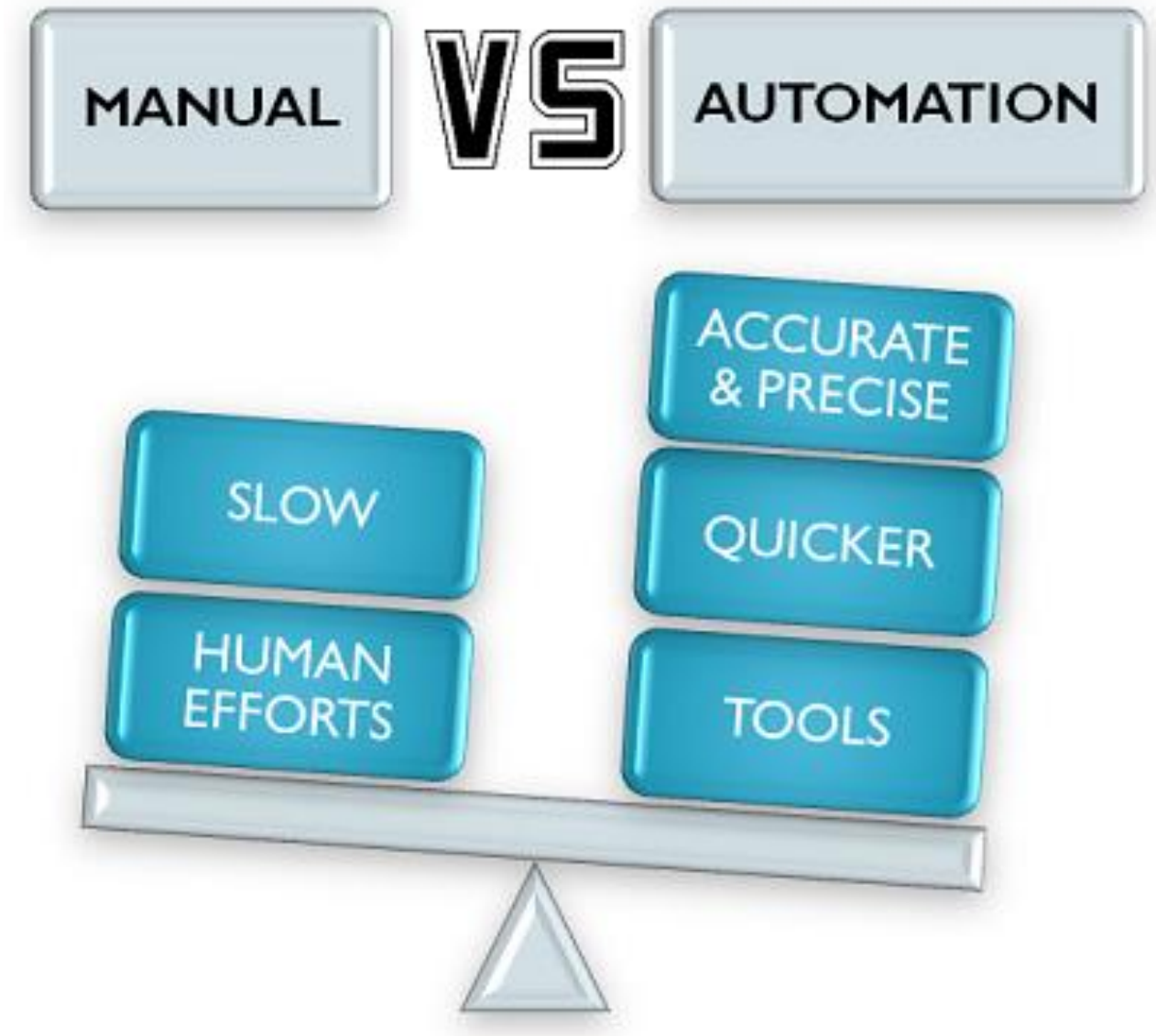
- ✓ Поддержание последовательности тестирования
- ✓ Улучшает структурирование теста
- ✓ Минимальное использование кода
- ✓ Меньше обслуживания кода
- ✓ Улучшение возможности повторного использования
- ✓ Нетехнические тестировщики могут быть вовлечены в код
- ✓ Период обучения использованию инструмента может быть сокращен
- ✓ Включает данные везде, где это уместно

Преимущества автоматизации тестирования:

- На 70% быстрее ручного тестирования
- Более широкий охват тестирования функций приложения
- Надежные результаты
- Обеспечить последовательность
- Экономит время и деньги
- Повышает точность
- Вмешательство человека во время выполнения не требуется.
- Увеличивает эффективность
- Лучшая скорость выполнения тестов
- Тестовые сценарии, пригодные для повторного использования
- Тестирует часто и тщательно
- Раннее время выхода на рынок

Минусы автоматизированного тестирования:

- Без человеческого фактора **сложно получить представление о визуальных аспектах пользовательского интерфейса**, таких как цвета, шрифты, размеры, контрастность или размеры кнопок.
- Инструменты для проведения автоматизированного тестирования могут быть дорогими, что **может увеличить стоимость проекта** тестирования.
- Инструмент тестирования автоматизации пока не является полностью надежным. **Каждый инструмент автоматизации имеет свои ограничения**, что снижает область автоматизации.
- Отладка тестового скрипта — еще одна важная проблема в автоматизированном тестировании. **Поддержка тестов обходится дорого.**


MANUAL
VS
AUTOMATION
SLOW
HUMAN EFFORTS
ACCURATE & PRECISE
QUICKER
TOOLS

Параметр	Автоматизированное тестирование	Ручное тестирование
Время обработки	Автоматизированное тестирование значительно быстрее ручного подхода.	Ручное тестирование занимает много времени и требует человеческих ресурсов.
Исследовательское тестирование	Автоматизация не допускает выборочного тестирования	Исследовательское тестирование возможно в ручном тестировании
Первоначальные инвестиции	Первоначальные инвестиции в автоматизированное тестирование выше. Хотя окупаемость инвестиций в долгосрочной перспективе выше.	Первоначальные инвестиции в ручное тестирование сравнительно ниже. Окупаемость инвестиций ниже по сравнению с автоматизированным тестированием в долгосрочной перспективе.

Параметр	Автоматизированное тестирование	Ручное тестирование
Изменение пользовательского интерфейса	Даже для незначительного изменения пользовательского интерфейса AUT необходимо изменить сценарии автоматизированного тестирования, чтобы они работали так, как ожидается.	Небольшие изменения, такие как изменение идентификатора, класса и т. д. кнопки, не помешают выполнению ручного тестирования.
Инвестиции	Требуются инвестиции в инструменты тестирования, а также в инженеров по автоматизации.	Необходимы инвестиции в человеческие ресурсы.
Экономически эффективно	Невыгодно для регрессии малого объема	Невыгодно для регрессии большого объема.

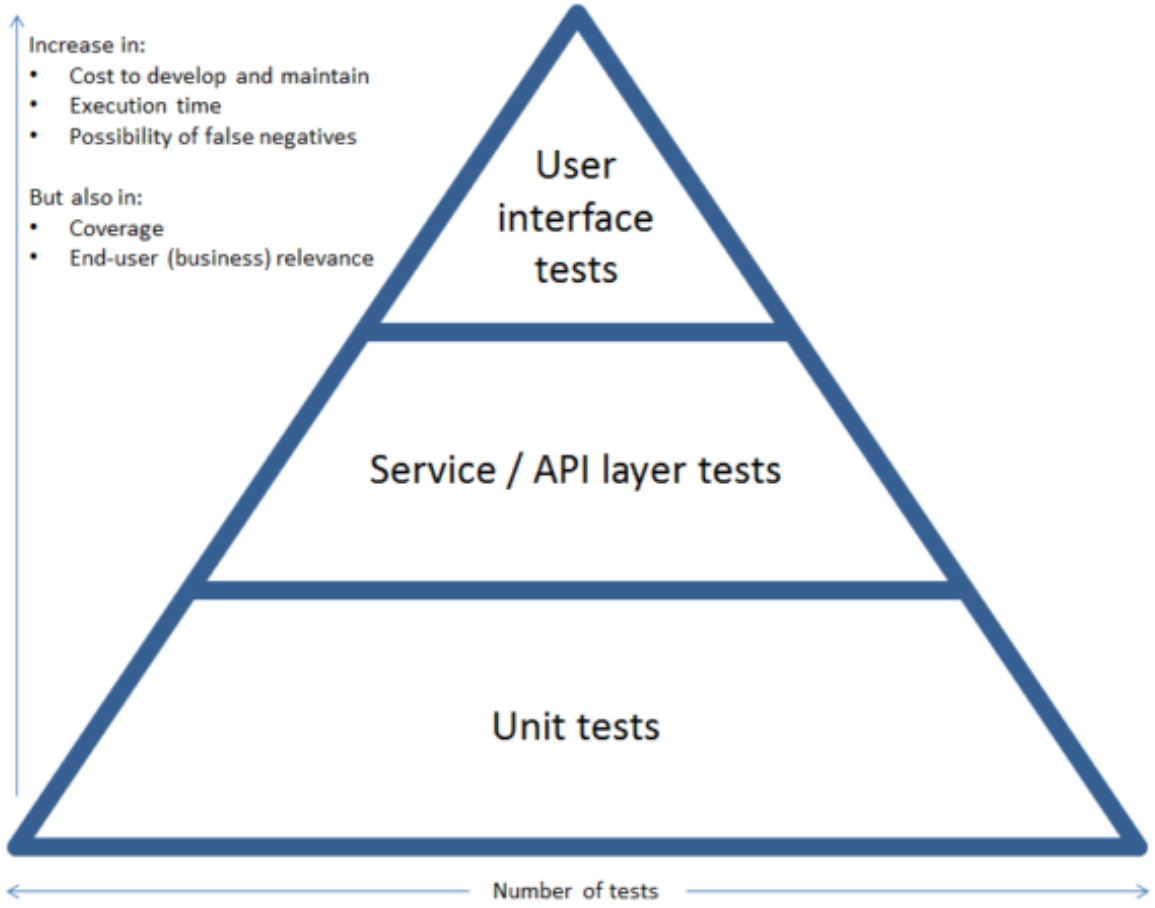
Параметр	Автоматизированное тестирование	Ручное тестирование
Тестирование производительности	Тесты производительности, такие как нагрузочное тестирование, стресс-тестирование, пиковое тестирование и т. д., должны в обязательном порядке тестироваться с помощью инструмента автоматизации.	Тестирование производительности невозможно выполнить вручную.
Параллельное выполнение	Это тестирование можно выполнять на разных операционных платформах параллельно, что сокращает время выполнения теста.	Ручные тесты можно выполнять параллельно, но для этого потребуется увеличить человеческие ресурсы, что является дорогостоящим.

Параметр	Автоматизированное тестирование	Ручное тестирование
Знание программирования	Знание программирования является обязательным условием при автоматизированном тестировании.	Нет необходимости в программировании при ручном тестировании.
Документация	Автоматизированные тесты действуют как документ, предоставляющий обучающую ценность, особенно для автоматизированных тестовых случаев. Новый разработчик может изучить тестовые случаи и быстро понять кодовую базу.	Ручные тестовые случаи не представляют никакой обучающей ценности

УРОВНИ АВТО-ТЕСТИРОВАНИЯ

- ✓ Уровень модульного тестирования (unit tests);
- ✓ Уровень функционального тестирование (non-UI tests);
- ✓ Уровень тестирования через пользовательский интерфейс (UI tests).

Пирамида автоматизации тестирования



МОДУЛЬНОЕ ТЕСТИРОВАНИЕ

> Процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы (Unit –тесты).

Идея состоит в том, чтобы писать тесты для каждой нетривиальной функции или метода.

Это позволяет достаточно быстро проверить, не привело ли очередное изменение кода к регрессии, то есть к появлению ошибок в уже оттестированных местах программы, а также облегчает обнаружение и устранение таких ошибок.

```
class CalculatorTests
{
        public void Sum_2Plus5_7Returned()
        {
                // arrange
                var calc = new Calculator();
                // act
                var res = calc.Sum(2,5);
                // assert
                Assert.AreEqual(7, res);
        }
}
```

ФРЕЙМВОРКИ для UNIT-ТЕСТОВ

NON-UI ТЕСТИРОВАНИЕ

> Процесс в программировании, позволяющий проверить работоспособность приложения используя программный интерфейс приложения (application programming interface, API).

Не всегда всю бизнес логику приложения можно протестировать через GUI слой. Это может быть особенностью реализации, которая прячет бизнес логику от пользователей. Именно по этой причине для команды тестирования может быть реализован доступ напрямую к функциональному слою, дающий возможность тестировать непосредственно бизнес логику приложения, минуя пользовательский интерфейс.

```
package CountriesRestTests;

…
public class GetTest {

@Test
public void getRequestFindCapital() throws JSONException {

// выполняем запрос get для доступа ко всем параметрам ответа
Response resp = get("http://restcountries.eu/rest/v1/name/belarus");

JSONArray jsonResponse = new JSONArray(resp.asString());

// получение параметра capital (столицы Беларуси)
String capital = jsonResponse.getJSONObject(0).getString("capital");

// проверка, что столицей является Минск
AssertJUnit.assertEquals(capital, “Minsk");
}
}
```

NON-UI ТЕСТИРОВАНИЕ: ИНСТРУМЕНТЫ

SoapUI

REST-assured

UI ТЕСТИРОВАНИЕ

Процесс в программировании, позволяющий проверить работоспособность и внешний вид приложения используя графический интерфейс приложения.

SELENIUM

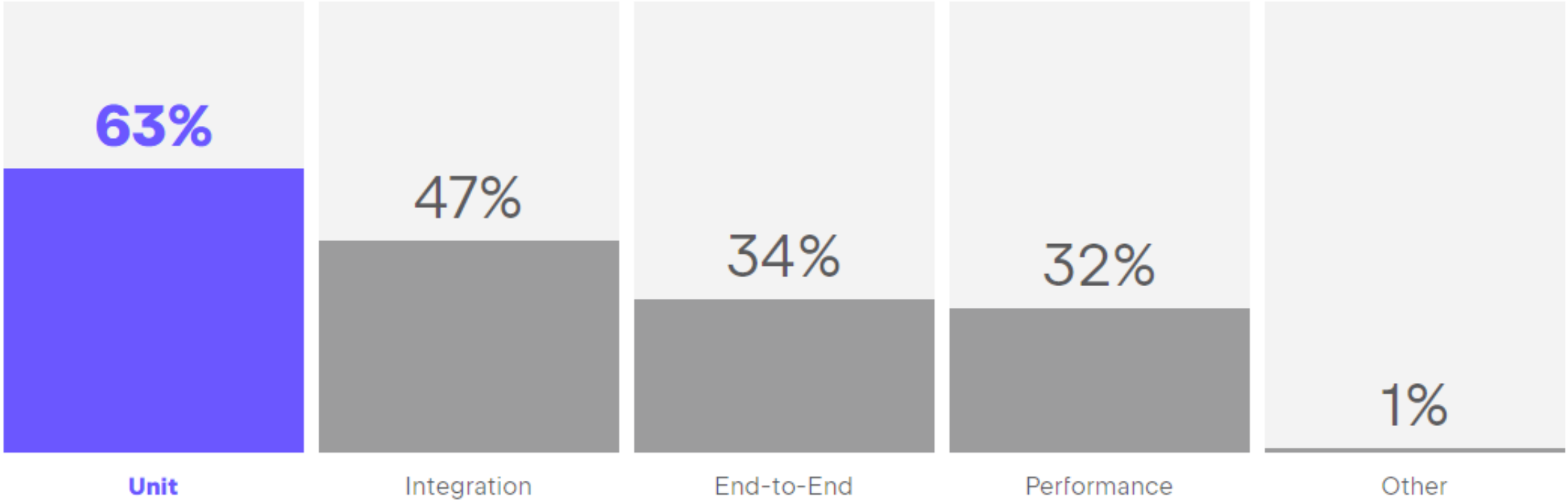
```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;
public class Example  {
    public static void main(String[] args) {
        WebDriver driver = new HtmlUnitDriver();
        driver.get("http://www.google.com");
        WebElement element = driver.findElement(By.name("q"));
        element.sendKeys("Lecture for automated testing!");
        element.submit();
        System.out.println("Page title is: " + driver.getTitle());
        driver.quit(); } }
```

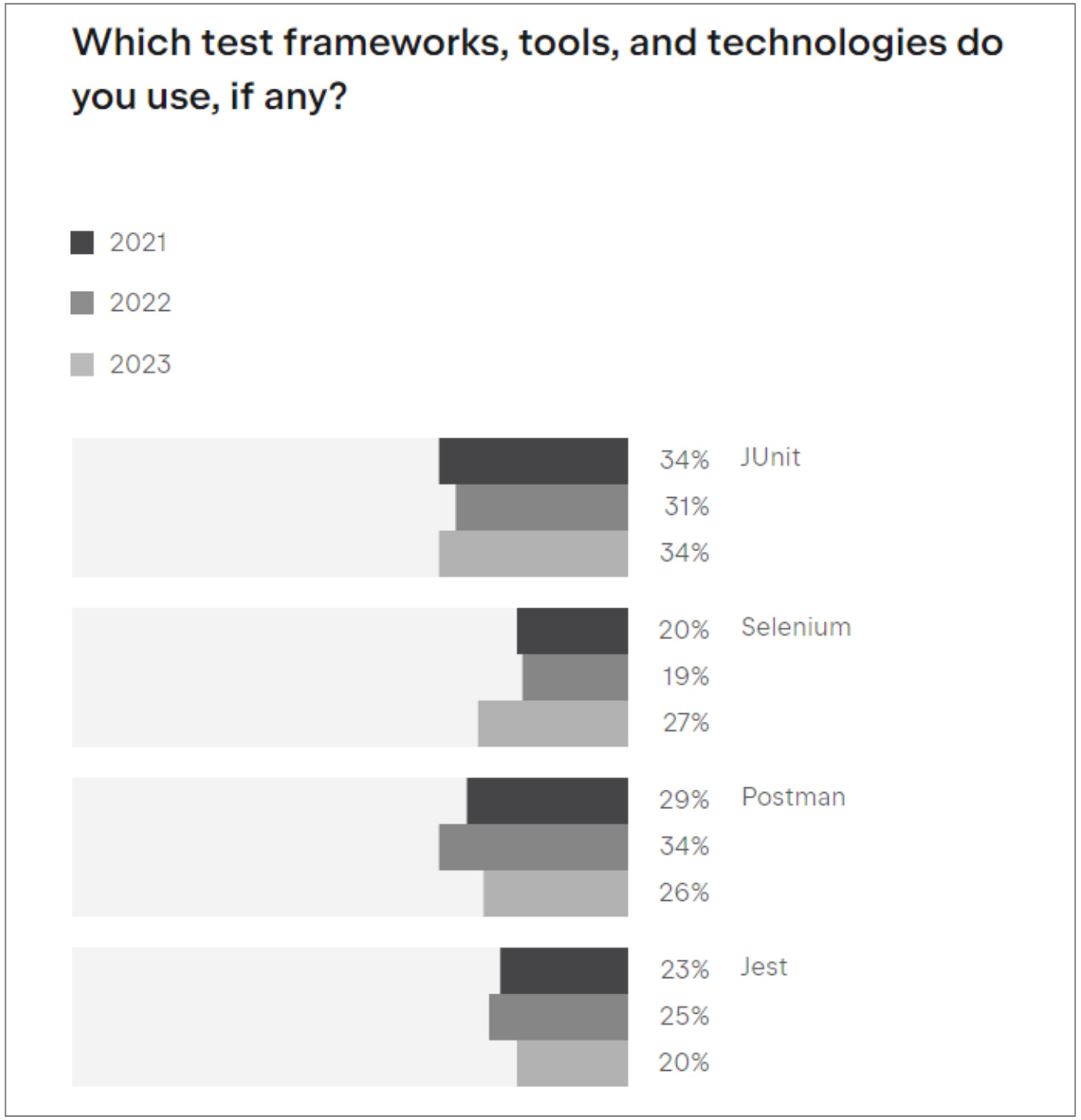
ИНСТРУМЕНТЫ UI АВТОМАТИЗАЦИИ

Исследования JetBrains 2023

https://www.jetbrains.com/lp/devecosystem-2023/testing/

What types of tests do you have in your projects?





Which programming languages do you use for test automation in your project?