Perform Automatic Testing on a Data Processing Application

Ms. ALF. Sajeetha

Unit 20 - APDP

KEY ELEMENTS

Types of automatic testing

Tool automation parameters

Follow common testing frameworks and methodologies

Tools

Self-built testing tools

MANUAL TESTING

Manual testing is essentially self-explanatory

Testing of a web application is done manually, by human action

AUTOMATION TESTING

Automated testing uses the assistance of tools, scripts and software to perform test cases by repeating pre-defined actions

USE OF AUTOMATION IN THE FOLLOWING CONTEXT

1. Regression test
2. Data set up generation
3. Product installation
4. GUI interaction
5. Defect logging
6. Unit testing and integration testing

REGRESSION TESTING

Automated Test Scripts

Automated test scripts can be made to run regression tests after code changes. These scripts can simulate user actions, verify expected behavior, and report any discrepancies. Popular frameworks for writing test scripts include Selenium WebDriver (web applications), Appium (mobile applications), and JUnit (Java applications).

DATA SET UP AND GENERATION

Test Data Generation Tools

Tools like JMeter or Gatling can generate large volumes of test data based on defined rules and formats. This is especially helpful for load testing and performance testing. This ensures tests have access to a variety of data inputs, reduces manual effort in creating test data, and simplifies data management for different test scenarios.

PRODUCT/SOFTWARE INSTALLATION

Software Installation Scripts

Scripts can be written to automate the software installation process on different environments. Tools like Ansible or Chef can manage infrastructure provisioning and configuration, including software installations. This saves time compared to manual installation, ensures consistent configuration across different environments, and minimizes the risk of errors during installation.

GUI INTERACTION

Automated Testing Frameworks

Frameworks like Selenium WebDriver allow for simulating user interactions with the graphical user interface (GUI) of an application. These tools can click buttons, enter text, navigate menus, and verify the resulting behavior. This enables automated testing of user workflows and functionalities, frees up testers for more exploratory testing, and improves test coverage.

DEFECT LOGGING

Defect Tracking Systems (DTS)

Integrating testing frameworks with defect tracking systems like Jira or Bugzilla can automatically log defects identified during test runs. This captures detailed information about the defect and facilitates tracking its resolution. It saves time and reduces manual effort in logging defects, provides a centralized repository for managing defects, and improves communication between testers and developers.

UNIT TESTING

Testing Frameworks

Unit testing frameworks like JUnit (Java), PHPUnit (PHP), or pytest (Python) allow developers to write unit tests that focus on individual units of code (functions, classes). These tests can be automated and run frequently to ensure code quality. It promotes writing clean and modular code, catches bugs early in the development process, and facilitates code refactoring with confidence.

INTEGRATION TESTING

Continuous Integration (CI) Tools

CI tools like Jenkins or Travis CI can be used to automate building, testing, and deployment processes.

Integration tests can be integrated into the CI pipeline to ensure different modules of the application work together seamlessly.

It ensures timely detection of integration issues before code is deployed to production, improves overall software quality, and facilitates faster delivery cycles.

KEY FEATURES SUPPORTING UNATTENDED TEST RUNS

Automated Test Execution: Automated testing scripts or frameworks execute test cases without manual intervention. This allows you to schedule tests to run overnight, on weekends, or at any time that best suits your development cycle.

Test Scheduling and Management: Many testing frameworks and tools provide built-in scheduling capabilities. You can define when and how often tests should be run, ensuring regular testing without manual effort.

Continuous Integration (CI) Pipelines: CI tools integrate automated testing into the development workflow. Code changes trigger test runs, providing immediate feedback and preventing regressions before code is deployed.

Detailed Test Reports: Automated tests generate comprehensive reports that summarize test results, identify failures, and provide logs for debugging purposes. Analyzing these reports becomes your primary task after the unattended test run.

Remote Access and Monitoring: Certain frameworks allow remote access to monitor ongoing test runs and intervene if necessary. This can be helpful for debugging unexpected issues or troubleshooting failed tests.

TOOL AUTOMATION PARAMETERS

DATA-DRIVEN CAPABILITIES:

In software development, data-driven capabilities refer to the ability of a tool or framework to utilize external data sources to control its behavior or execution. This data can be used for various purposes:

- **Test data generation:** Data-driven testing frameworks allow you to define test cases with different sets of data, enabling efficient testing of various scenarios.
- **Configuration management:** Configuration files containing data can be used to define settings, parameters, and behavior of the tool or application.
- **Dynamic content generation:** Tools might use data to personalize content or customize behavior based on user information or preferences.

DEBUGGING AND LOGGING CAPABILITIES

Debugging tools and techniques help developers identify and fix errors (bugs) in code. Logging capabilities are an essential part of debugging:

- **Debugging:** Debuggers allow developers to step through code execution line by line, inspect variables, and identify where errors occur.
- **Logging:** Logging frameworks capture events and information during application execution. These logs can be used to diagnose issues, track application behavior, and analyze performance.

PLATFORM INDEPENDENCE

Platform independence refers to the ability of a tool or application to run on different operating systems (Windows, macOS, Linux) or hardware architectures without requiring significant modifications. This is often achieved through:

- **Cross-platform development tools:** Frameworks like Java, Python, or Kotlin allow writing code that can be compiled and run on various platforms.
- **Virtualization technologies:** Tools like Docker can package applications with all their dependencies, ensuring they run consistently across different environments.

EXTENSIBILITY AND CUSTOMIZABILITY

Extensibility refers to the ability of a tool or application to be extended with additional functionality through **plugins, add-ons, or custom code**. This allows users to tailor the tool to their specific needs.

Customizability refers to the ability to modify the behavior or appearance of a tool or application through configuration options or user interface settings. Users can personalize the tool to work best for their workflow.

EMAIL NOTIFICATIONS

Email notifications are a way to receive alerts and updates from a tool or application via email. This can be helpful for:

- **Test results:** Automated testing frameworks can send email notifications about test success or failure.
- **Error alerts:** Applications might send emails when critical errors occur, requiring developer attention.
- **Build status:** CI/CD tools can send emails notifying about successful or failed builds and deployments.

VERSION CONTROL FRIENDLY

Version control systems (VCS) like Git or Subversion track changes made to code and files over time. A tool or application is considered "version control friendly" if it integrates well with VCS and doesn't interfere with version tracking mechanisms:

- **Stores data outside the codebase:** Ideally, a tool should store configuration files or data outside the main codebase to avoid conflicts during version control operations.
- **Handles file updates correctly:** The tool should handle file modifications gracefully to ensure changes are tracked accurately by the VCS.

BIG DATA TESTING TOOLS:

If your data processing application deals with massive datasets, consider specialized big data testing tools. These tools can handle the scale and complexity of big data environments.

UNDERSTANDING TESTING LOGIC

- ✓ Testing logic refers to the thought process and strategies used to design and execute tests that effectively verify the functionality of your code. This involves:
- ✓ Identifying critical functionalities and edge cases to test.
- ✓ Designing test cases that simulate user interactions and data inputs.
- ✓ Verifying expected outcomes and identifying potential failures.
- ✓ A strong understanding of testing logic allows you to:
- ✓ Write cleaner and more maintainable code: Knowing what aspects are crucial to test helps you structure your code in a way that's easier to isolate and test individual units.
- ✓ Reduce bugs and improve code quality: By proactively considering potential test cases, you're more likely to write code that functions correctly under various scenarios.
- ✓ Debug code more efficiently: Understanding testing logic helps you interpret test failures and pinpoint the root cause of issues more quickly.

STUBBING AND PATCHING FOR EASIER UPDATES

Stubbing and patching are techniques used in unit testing to isolate and test specific parts of your code by replacing dependencies with controlled substitutes.

Stubbing:

- A stub is a simplified version of a dependency that provides limited functionality relevant to the test case.
- Stubs typically return pre-defined values or perform specific actions in response to calls from your code.

Patching:

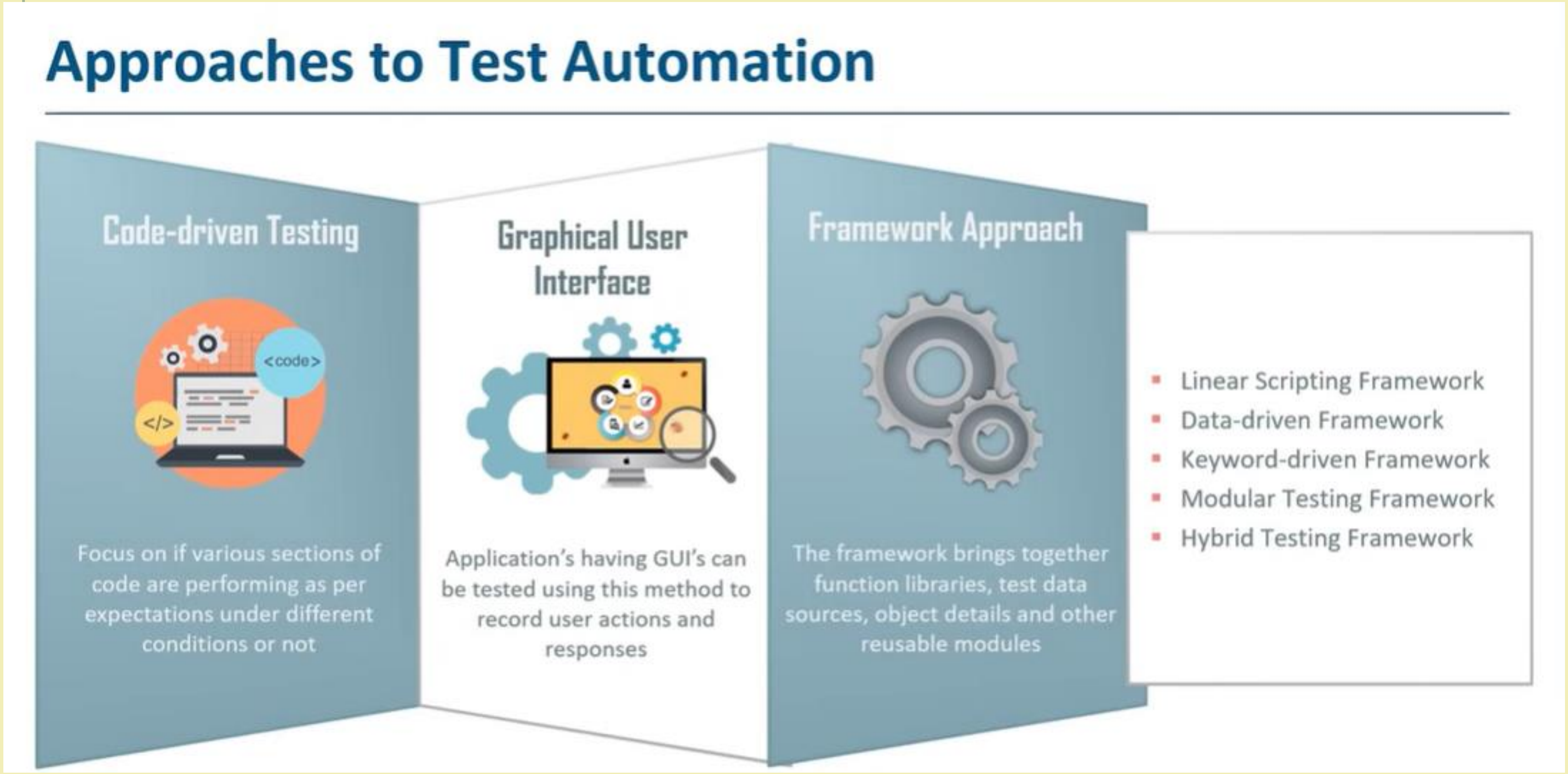
- Patching involves modifying an existing object or function to temporarily change its behavior during the test. This allows you to control the behavior of external dependencies within your test environment.

BENEFITS OF STUBBING AND PATCHING

- **Isolate code for testing:** By replacing dependencies with stubs or patches, you can focus on testing a particular unit of code in isolation from external factors. This simplifies test setup and reduces the risk of errors from external dependencies.
- **Simulate different scenarios:** Stubs and patches allow you to create controlled test environments with specific inputs and outputs, enabling you to test your code under various conditions.
- **Easier code updates:** When you stub or patch dependencies, changes to those dependencies won't affect your unit tests as long as your code interacts with them correctly. This makes it easier to update and refactor code without breaking existing tests.

TYPES OF AUTOMATIC TESTING

DIFFERENT METHODS OF IMPLEMENTING AUTOMATIC TESTING



Approaches to Test Automation

Code-driven Testing

Focus on if various sections of code are performing as per expectations under different conditions or not

Graphical User Interface

Application's having GUI's can be tested using this method to record user actions and responses

Framework Approach

The framework brings together function libraries, test data sources, object details and other reusable modules

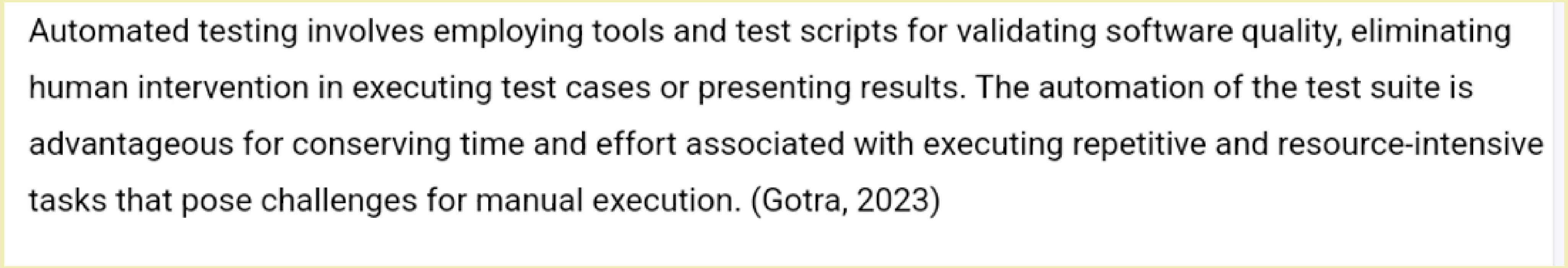
- Linear Scripting Framework
- Data-driven Framework
- Keyword-driven Framework
- Modular Testing Framework
- Hybrid Testing Framework

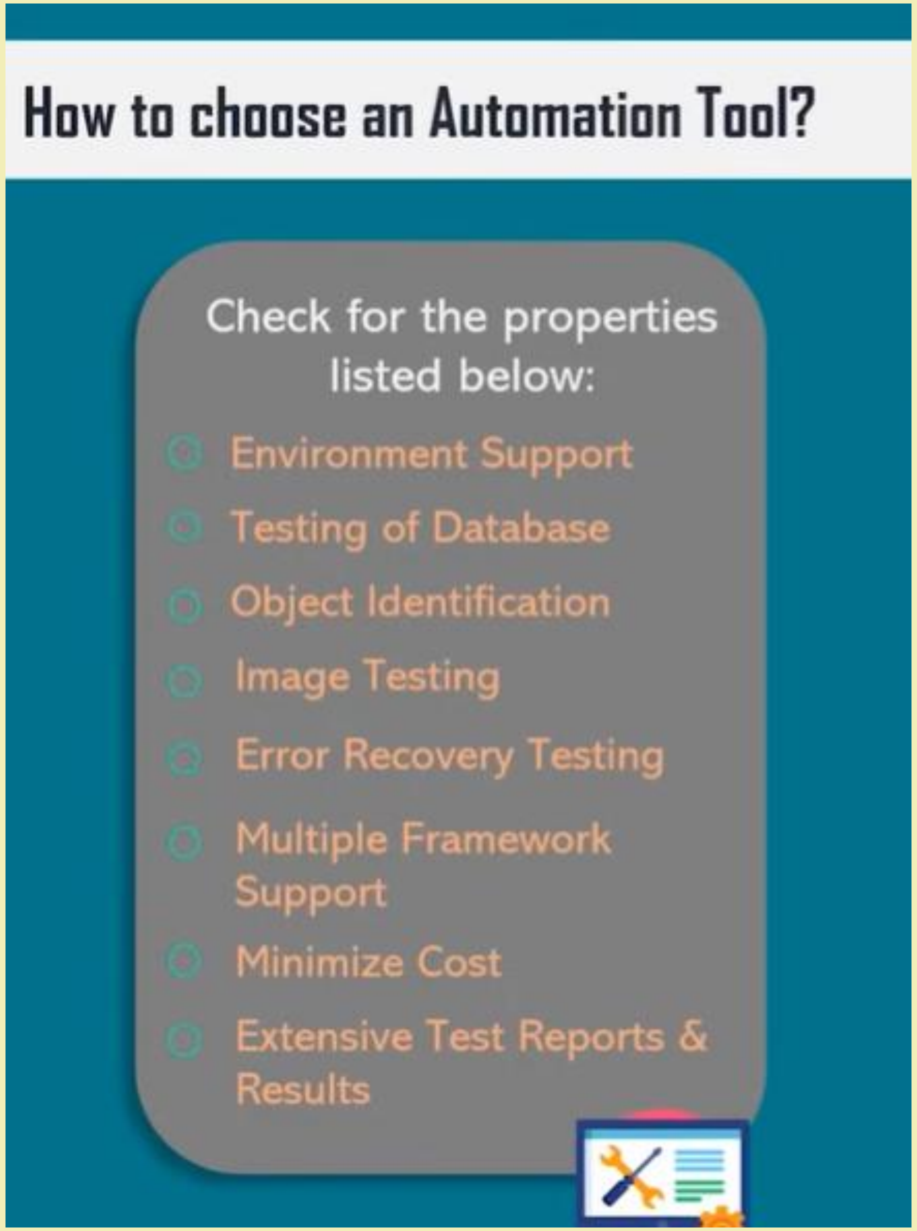
FOLLOW COMMON TESTING FRAMEWORKS AND METHODOLOGIES

Different frameworks perform best

1. Data Driven Automation Framework
2. Keyword Driven Automation Framework
3. Modular Automation Framework
4. Hybrid Automation Framework

AUTOMATION TESTING TOOLS


Automated testing involves employing tools and test scripts for validating software quality, eliminating human intervention in executing test cases or presenting results. The automation of the test suite is advantageous for conserving time and effort associated with executing repetitive and resource-intensive tasks that pose challenges for manual execution. (Gotra, 2023)

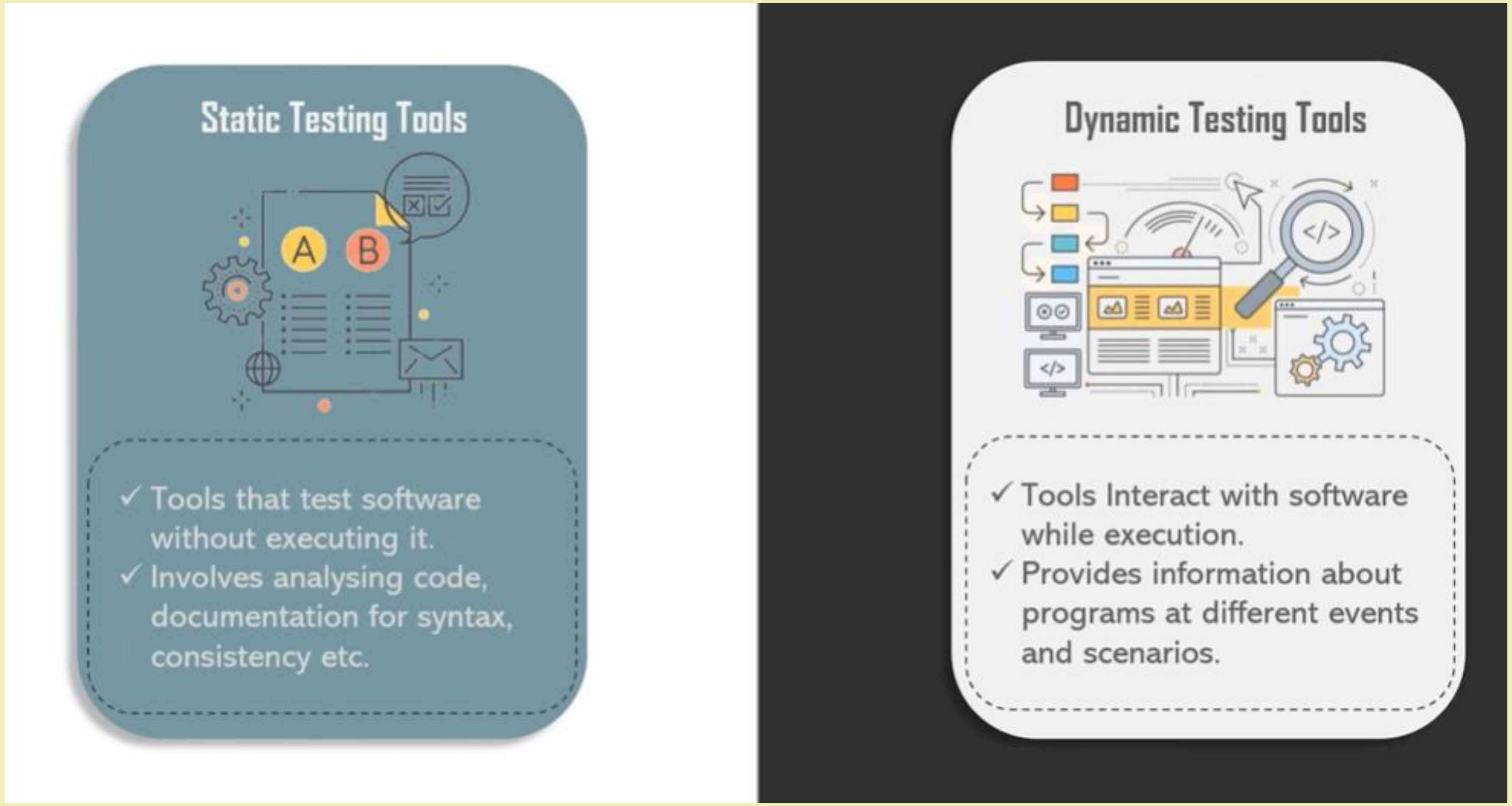


How to choose an Automation Tool?

Check for the properties listed below:

- Environment Support
- Testing of Database
- Object Identification
- Image Testing
- Error Recovery Testing
- Multiple Framework Support
- Minimize Cost
- Extensive Test Reports & Results

CATEGORIZING SOFTWARE AUTOMATION TOOLS



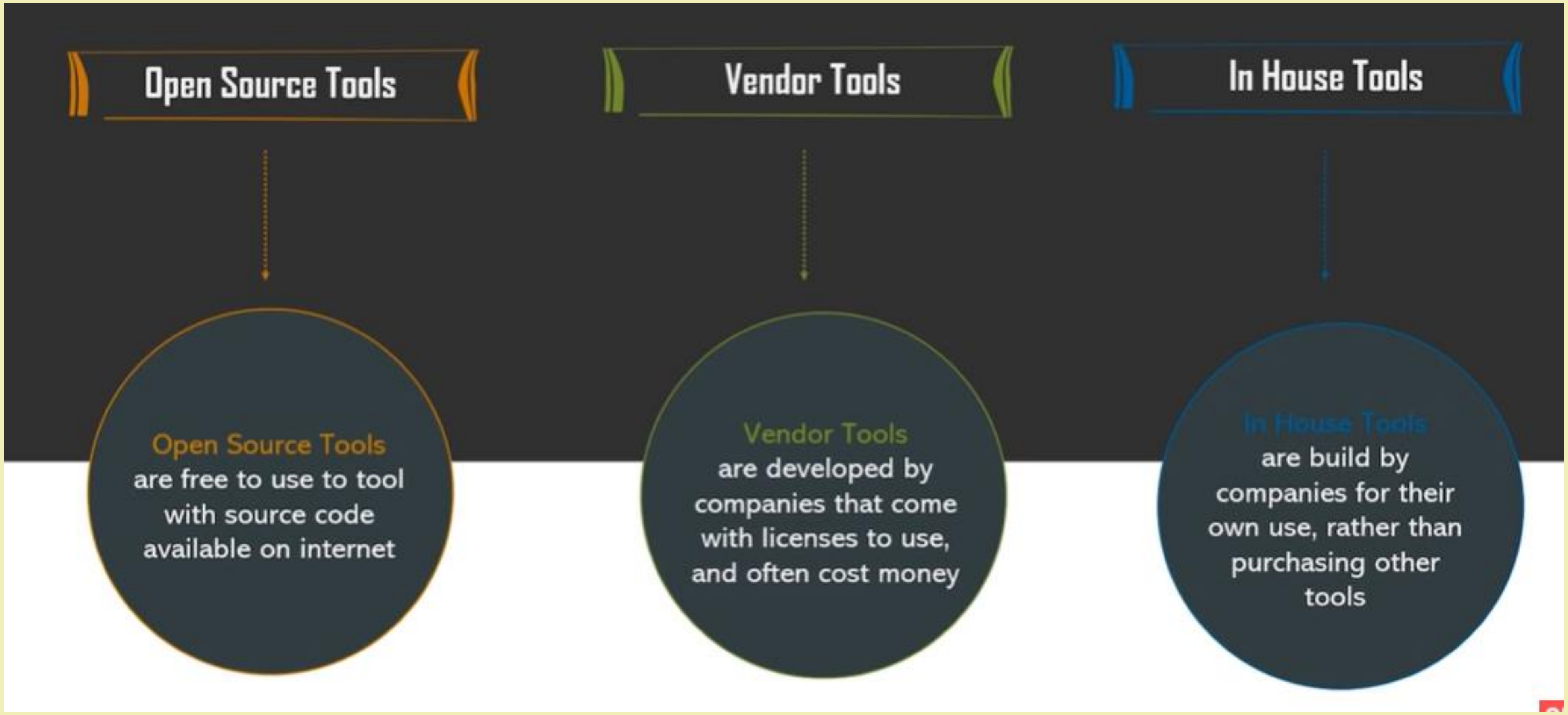
Static Testing Tools

- ✓ Tools that test software without executing it.
- ✓ Involves analysing code, documentation for syntax, consistency etc.

Dynamic Testing Tools

- ✓ Tools Interact with software while execution.
- ✓ Provides information about programs at different events and scenarios.

CATEGORIZING SOFTWARE AUTOMATION TOOLS


Open Source Tools

Vendor Tools

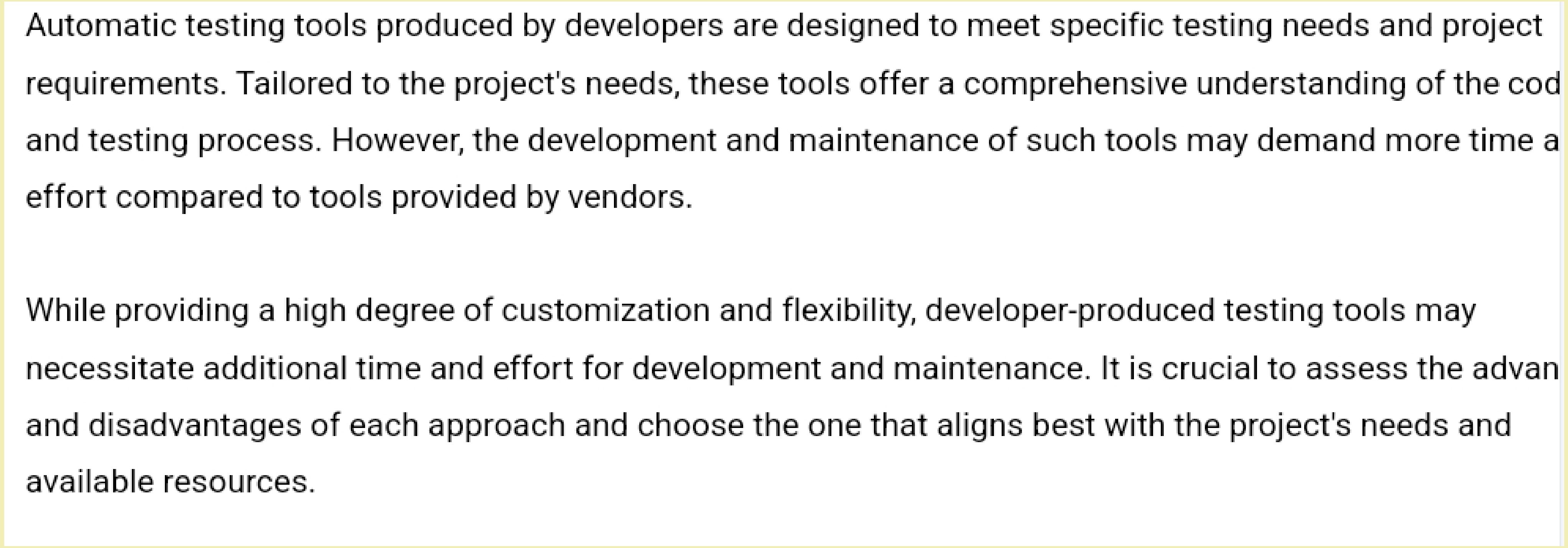
In House Tools

Open Source Tools are free to use to tool with source code available on internet

Vendor Tools are developed by companies that come with licenses to use, and often cost money

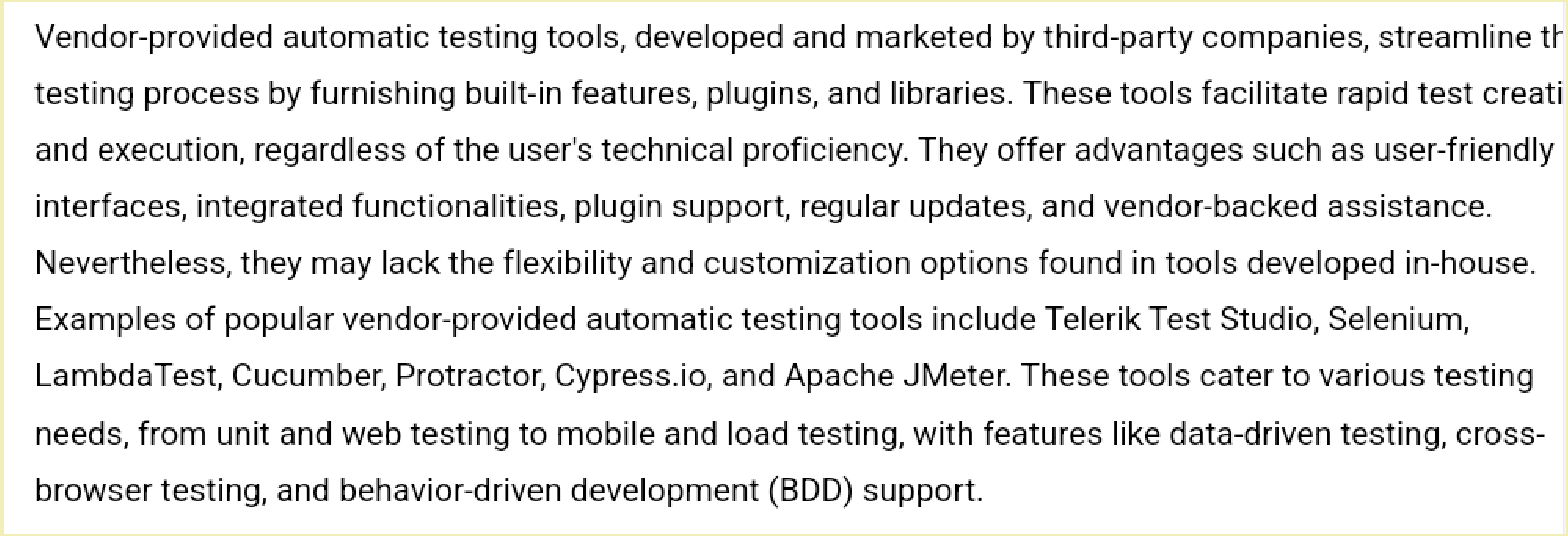
In House Tools are build by companies for their own use, rather than purchasing other tools

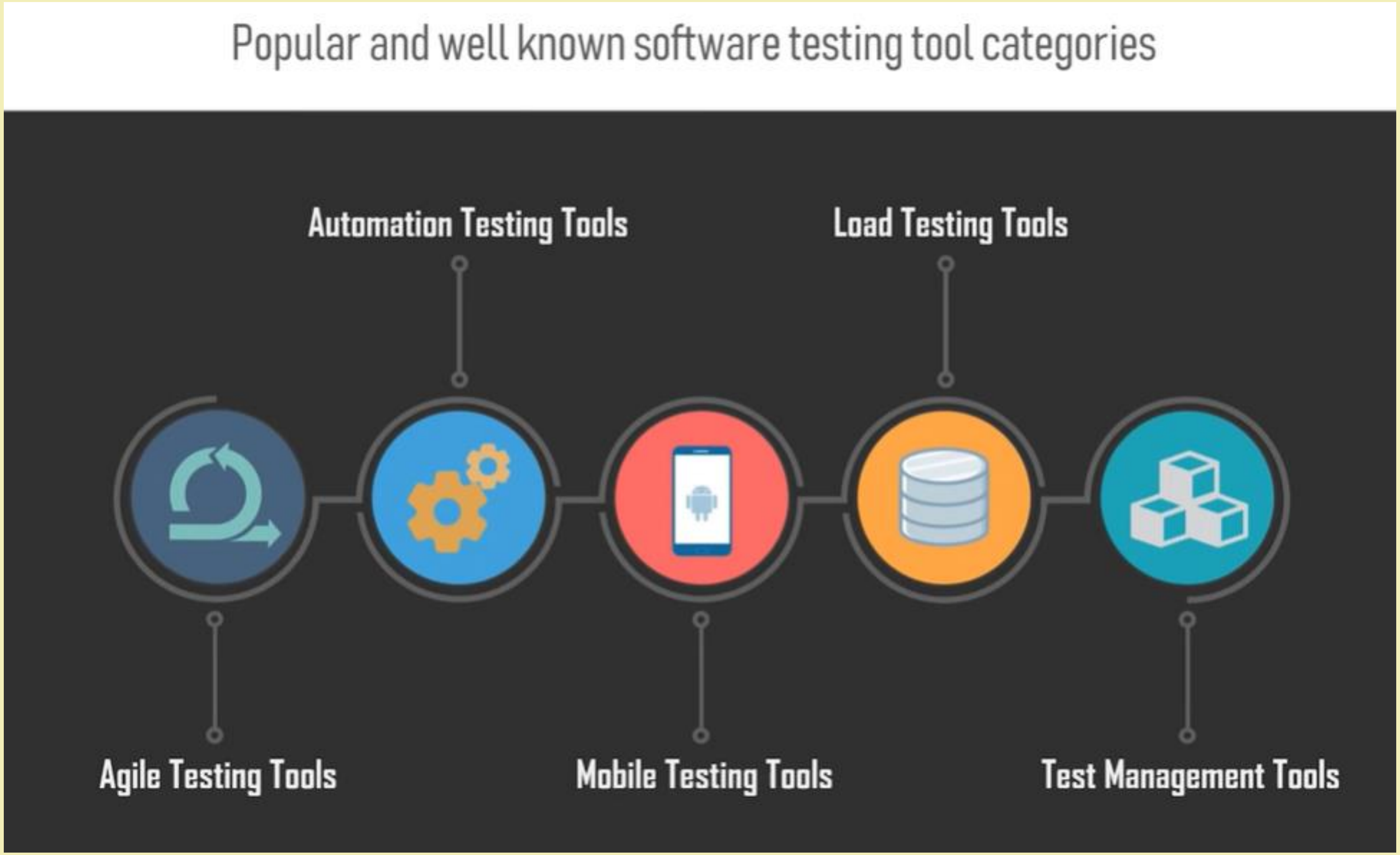
DEVELOPER PRODUCED AUTOMATIC TESTING TOOLS


Automatic testing tools produced by developers are designed to meet specific testing needs and project requirements. Tailored to the project's needs, these tools offer a comprehensive understanding of the cod and testing process. However, the development and maintenance of such tools may demand more time a effort compared to tools provided by vendors.

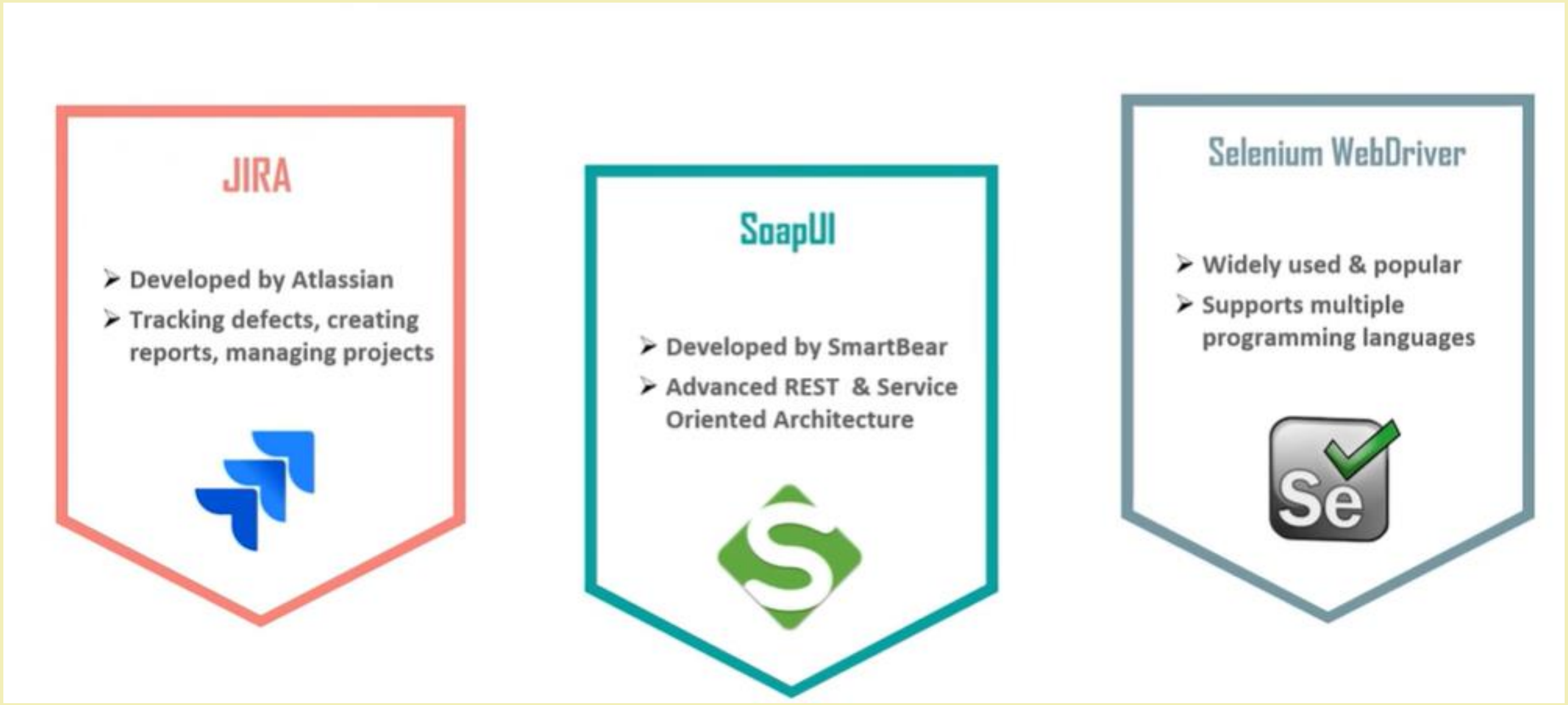
While providing a high degree of customization and flexibility, developer-produced testing tools may necessitate additional time and effort for development and maintenance. It is crucial to assess the advan and disadvantages of each approach and choose the one that aligns best with the project's needs and available resources.

VENDOR PROVIDED AUTOMATIC TESTING TOOLS


Vendor-provided automatic testing tools, developed and marketed by third-party companies, streamline th
testing process by furnishing built-in features, plugins, and libraries. These tools facilitate rapid test creati
and execution, regardless of the user's technical proficiency. They offer advantages such as user-friendly interfaces, integrated functionalities, plugin support, regular updates, and vendor-backed assistance. Nevertheless, they may lack the flexibility and customization options found in tools developed in-house. Examples of popular vendor-provided automatic testing tools include Telerik Test Studio, Selenium, LambdaTest, Cucumber, Protractor, Cypress.io, and Apache JMeter. These tools cater to various testing needs, from unit and web testing to mobile and load testing, with features like data-driven testing, cross-browser testing, and behavior-driven development (BDD) support.


Popular and well known software testing tool categories
Automation Testing Tools
Load Testing Tools
Agile Testing Tools
Mobile Testing Tools
Test Management Tools

AGILE TESTING TOOLS



JIRA

- ➢ Developed by Atlassian
- ➢ Tracking defects, creating reports, managing projects

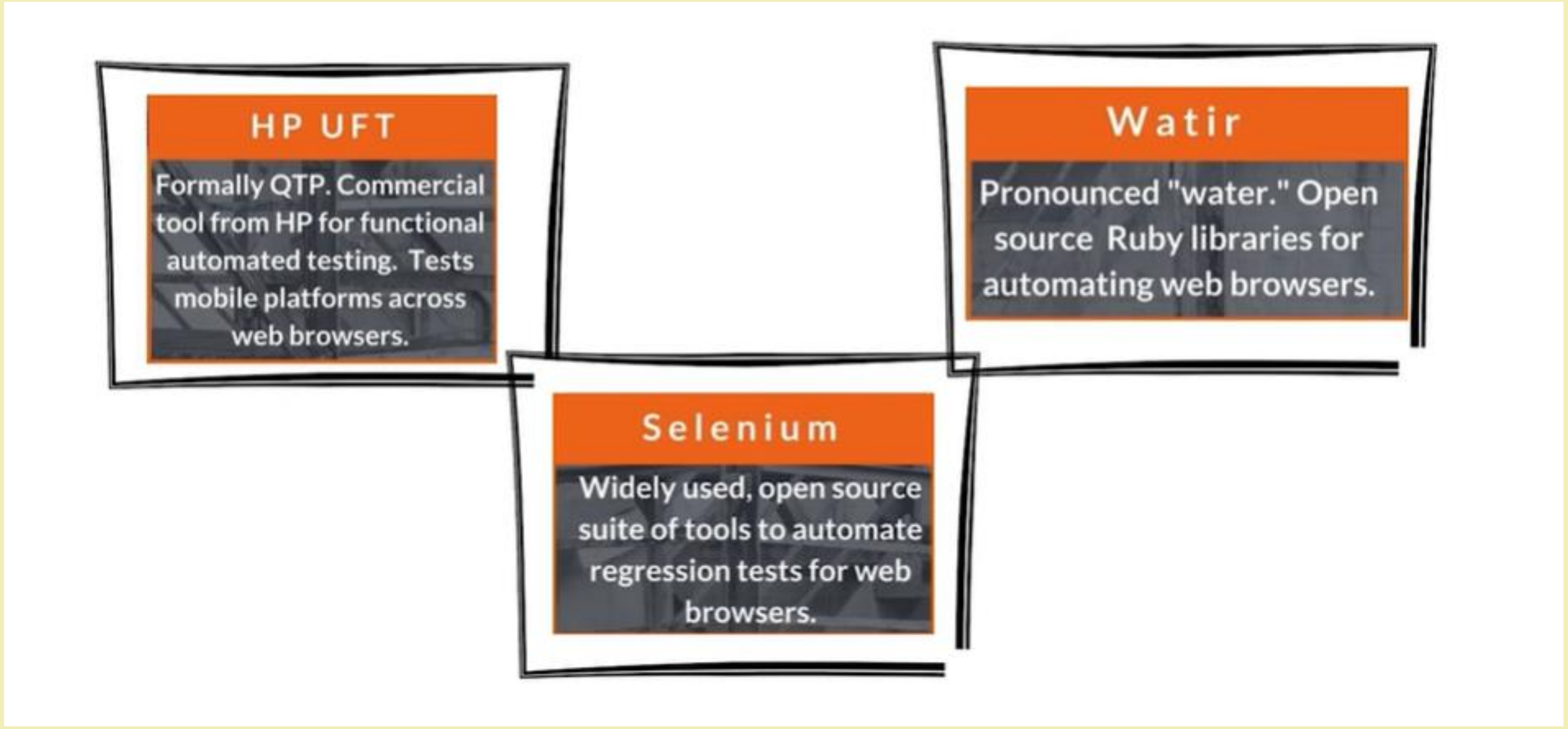
SoapUI

- ➢ Developed by SmartBear
- ➢ Advanced REST & Service Oriented Architecture

Selenium WebDriver

- ➢ Widely used & popular
- ➢ Supports multiple programming languages

AUTOMATION TESTING TOOLS



HP UFT

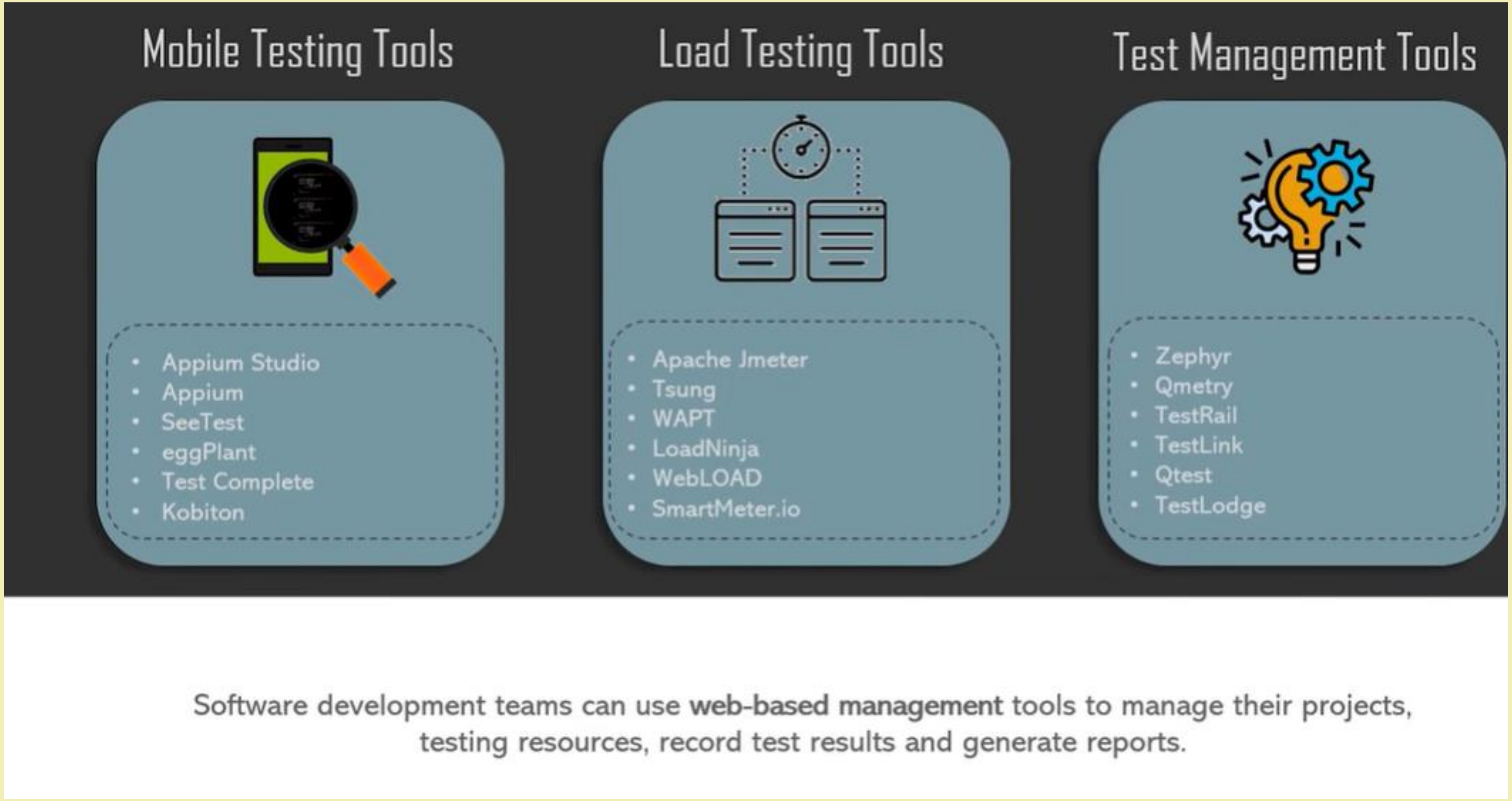
Formally QTP. Commercial tool from HP for functional automated testing. Tests mobile platforms across web browsers.

Watir

Pronounced "water." Open source Ruby libraries for automating web browsers.

Selenium

Widely used, open source suite of tools to automate regression tests for web browsers.



Mobile Testing Tools

- Appium Studio
- Appium
- SeeTest
- eggPlant
- Test Complete
- Kobiton

Load Testing Tools

- Apache Jmeter
- Tsung
- WAPT
- LoadNinja
- WebLOAD
- SmartMeter.io

Test Management Tools

- Zephyr
- Qmetry
- TestRail
- TestLink
- Qtest
- TestLodge

Software development teams can use **web-based management** tools to manage their projects, testing resources, record test results and generate reports.

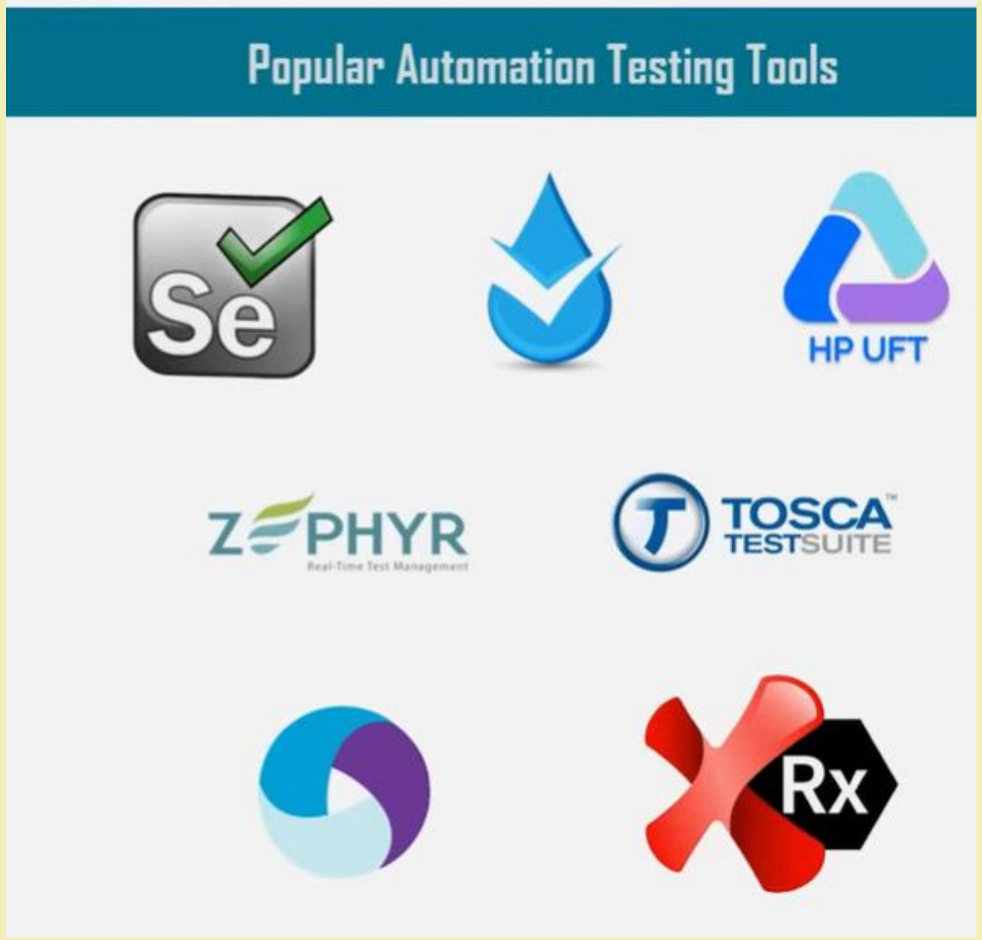
BIG DATA TESTING TOOLS:

Suitable Tools:

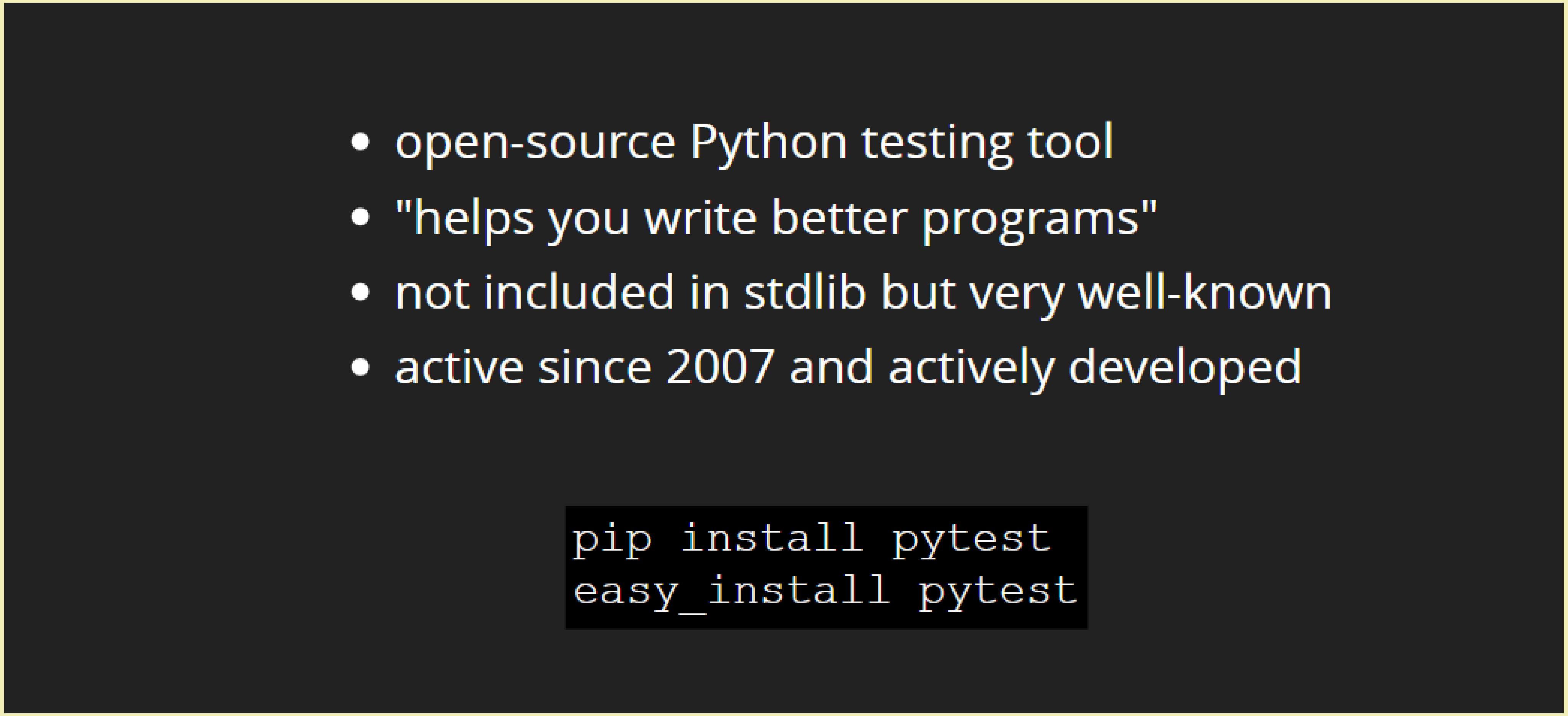
- **Apache Spark:** Offers testing libraries like Spark TestKit for unit and integration testing of Spark applications.
- **MapReduce Testing Frameworks:** Tools like MRUnit (Java) facilitate testing MapReduce jobs.
- **Streaming Data Testing Tools:** Tools like Apache Flink TestKit can help test streaming data processing pipelines.

POPULAR TOOLS FOR AUTOMATION TESTING IN SOFTWARE DEVELOPMENT

1. TestProject
2. Microfocus UFT
3. RFT
4. Selenium
5. AccelQ
6. TestCraft
7. Subject7
8. LambdaTest
9. Cucumber
10. Appium



PYTEST



- open-source Python testing tool
- "helps you write better programs"
- not included in stdlib but very well-known
- active since 2007 and actively developed

```
pip install pytest
easy_install pytest
```

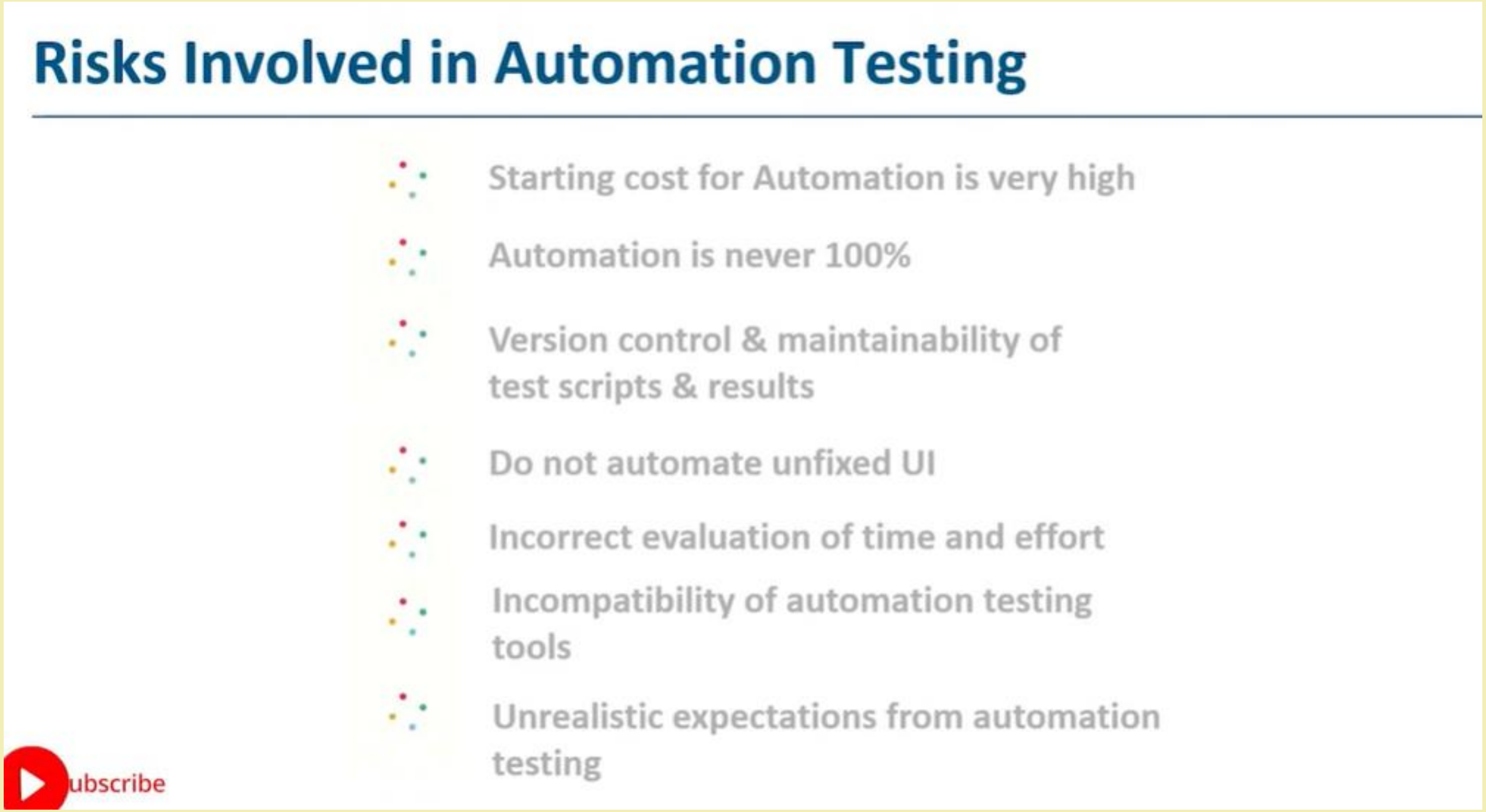
SELENIUM

JMETER

JIRA

POPULAR PYTHON LIBRARIES IN DATA SCIENCE

Library Name	Description
Pandas	Data Manipulation and Analysis
Numpy	Numerical Computing with arrays and matrices
SciPy	Scientific computing, including statistic
Matplotlib	Plotting and Visualization
Seaborn	Statistical data visualization
Scikit - learn	Machine Learning and predictive data analysis
TensorFlow	Deep Learning and Neural Networks
Keras	High- level neural networking API
PyTorch	Deep Learning and tensor computation with autograd
Plotly	Interactive graphing and analysis



Risks Involved in Automation Testing

- Starting cost for Automation is very high
- Automation is never 100%
- Version control & maintainability of test scripts & results
- Do not automate unfixed UI
- Incorrect evaluation of time and effort
- Incompatibility of automation testing tools
- Unrealistic expectations from automation testing

ubscribe