Test automation guide

A guide to test automation principles from a business viewpoint

Why?

This guide will help you understand the fundamentals of test automation and to find out different aspects of automation that you should consider before investing in it.

Teemu Pesonen, Director, QA & Automation
VALA Group

CONTENT

1. What is test automation?
2. Pros and cons of test automation
3. Is test automation suitable for you?
4. What to consider before implementation
5. Things to consider when choosing the tools
6. Our suggestions for various environments
7. What happens after the implementation
8. Conclusions

1. What is test automation?

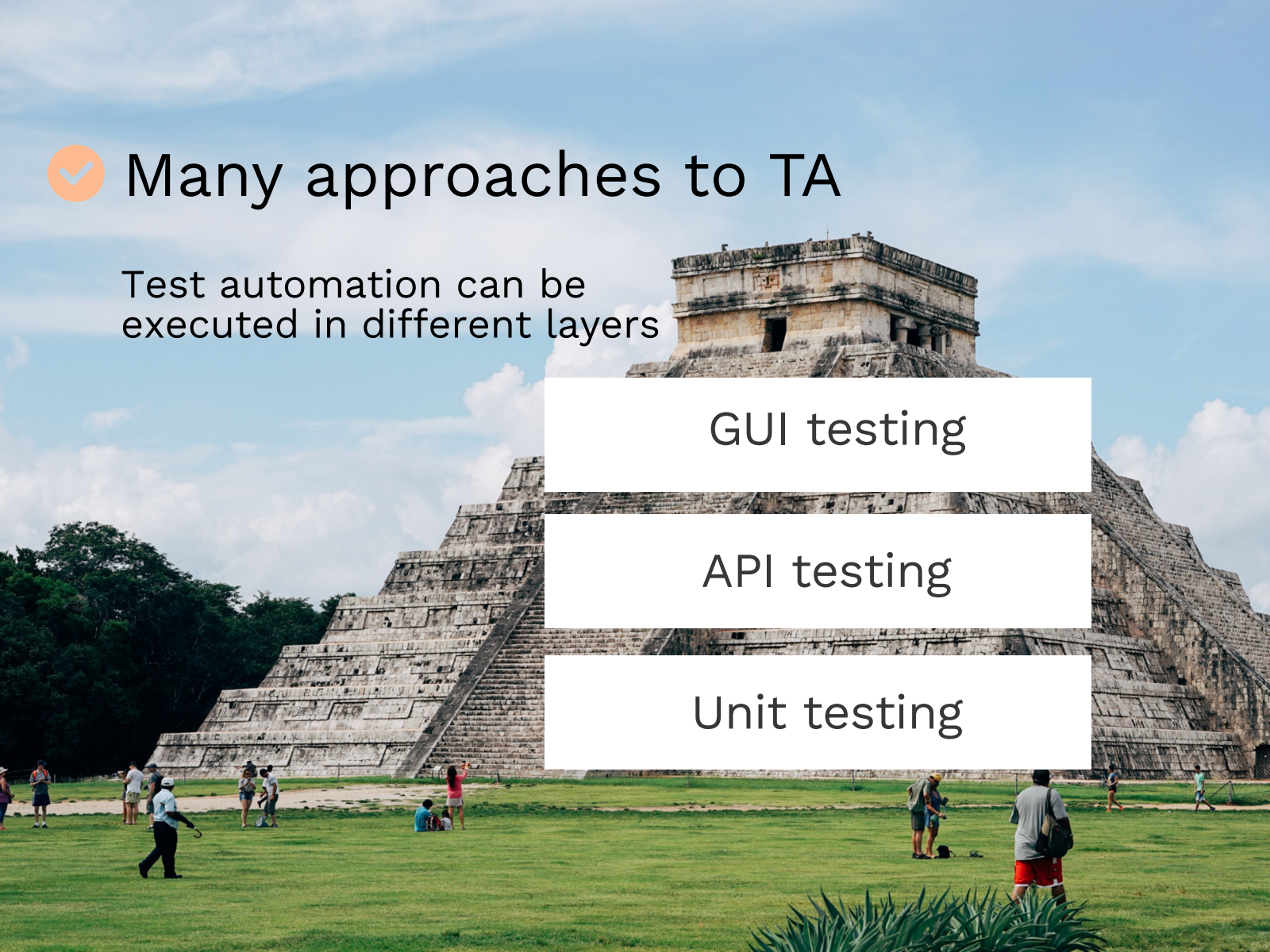
Test automation (TA) is the process of using dedicated tools to create means of executing test scenarios using computers and without human intervention in the execution phase.

The most important thing about test automation is the fact that it is basically a tool. Test automation is not a testing strategy, it doesn't know about planning, test coverage and so on.

Usually, the test automation process should generate a custom test automation framework using various tools (libraries, low level test automation frameworks), that is customized for the application under test.

Common requirements for a TA solution

- Simple enough way of writing and automating new test cases
- Good logging mechanism so that automated test run results are easy to understand/debug
- Results reporting to external systems (eg. Jira, TestRail)
- Code base should be easy to extend, maintain and customize
- Unattended execution of test scenarios - continuous integration
- Capability to run on different OS (Windows, Linux, Mac & mobile platforms)
- Email notifications



Many approaches to TA

Test automation can be executed in different layers

GUI testing

API testing

Unit testing

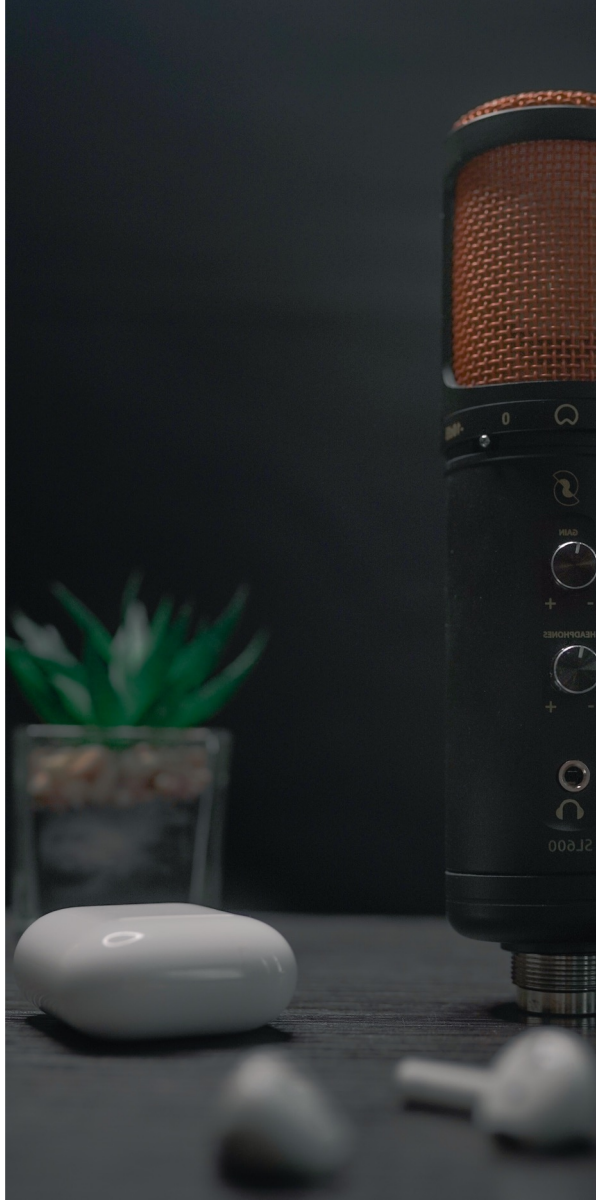
"Recording tools"

One special approach to GUI test automation is the usage of tools that are able to record human actions and generate code to re-play those actions. Recorded scripts approach is sometimes selected without good understanding of the risks and disadvantages.

Compared to a proper test automation framework, this approach has the following pros and cons.

Benefits:
- Fast and easy to implement scripts
- Allows inexperienced test engineers to create scripts

Challenges:
- Recording tool might generate more steps than needed
- Identification of GUI objects might be suboptimal
- Coding is anyways needed in some cases (for loops)
- On mid – long-term challenging to maintain the code
- More or less mechanical actions – no real understanding of how the software works
- Slightest GUI change will most likely break all the scripts that interact with that part of the GUI



2. Pros and cons of TA solution

PROS	CONS
Reliable	Longer initial development time
Fast	Requires additional tools
Precise	Analysis of a failed test case
Never forgets	Actual automated tests are production code
Increases test coverage	Requires specific skills from testers
Saves money in the long run	Fragility of test methods (especially GUI automation)
Complex, lengthy test cases can be executed unattended -> more time for test engineers to focus on other things -> happier test engineers	Benefits show in the mid to long term – most of the time initial costs are higher compared to manual testing
	Legacy code might not have been developed with automation in mind

3. Is TA suitable for you?

In most cases, if a company develops software, then test automation is used. This doesn't mean that everything should or can be automated.

In certain cases, manual testing is a better approach than test automation. Examples of these situations on the next page.

When manual testing

- When human validation is needed (look and feel of the GUI).
- When features are still under development (functionality changes a lot).
- When features are too complex and automation effort would be too high.

What kind of test automation?

Another challenge is choosing what kind of test automation should be implemented. As mentioned before, test automation can be executed in different layers (Unit, API, GUI).

Each layer has its own purpose and they are not mutually exclusive. In some cases a good decision is to automate all of these layers and in other cases some layers can be skipped.

Once it is decided which layer(s) to automate, choosing the tools for doing this is also critical. Selecting the right tools can make the difference between a successful effort and failure.

4.

What to consider before implementation

Test automation is not a magic button that can solve all problems when pushed. Test automation is only a tool. A powerful tool if used correctly.

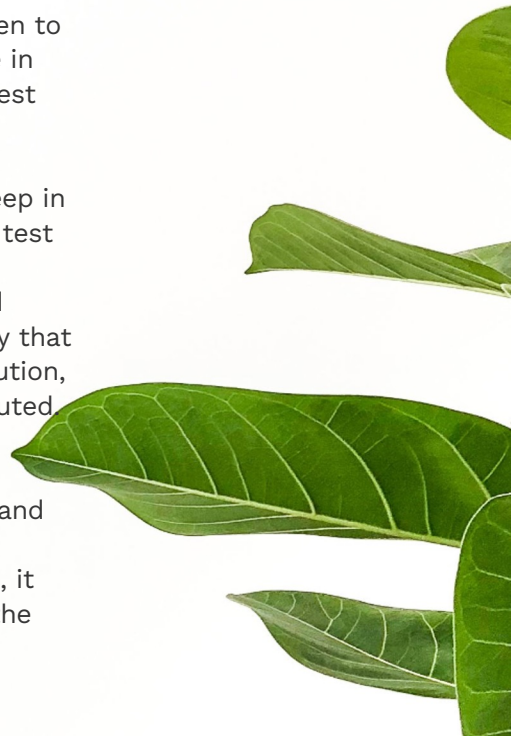
Still, if used in a wrong manner it can do more damage than good. The first things to keep in mind are the pros and cons.

TA requires time and money – when is it worth it?

TA requires longer initial development time and it is not cheap - this is one of the most important things to remember when deciding if/when to start a test automation process. Even if all other things required are in place (application can be automated, enough skilled people etc.) a test automation effort takes time to produce results.

The initial higher development cost and slowness is something to keep in mind. Test automation only shows benefits on mid-/ long-term. If a test case is properly automated the benefits of having that test case automated will only be clearly visible once the test case is executed in many regression cycles. The cost of testing the feature covered by that automated test decreases over time. Compared to the manual execution, the cost stays the same no matter how many times the test is executed.

TA requires good test planning and good regression strategy. As mentioned, TA is only a tool. What to automate needs human input and not everything should be automated. Only automate scenarios that provide good enough added value. Automating something takes time, it might not be trivial so this has to be balanced by the benefits that the automated test case brings.



TA requires time and money – when is it worth it?

Good candidates for automation can be tests that:

- Take a long time to execute
- Have lots of data validation
- Are executed in each regression cycle
- Are boring/difficult to be executed manually
- Also, in some cases, test automation is the only way that a scenario can be executed (non-functional testing for example - stability, performance, benchmarking, stress testing - these are usually done only with test automation).

TA is software development – you need skilled people

TA requires skilled engineers. One of the most important factors that can decide whether a test automation process is successful or not is the skill of the people involved in the effort. One should not expect doing good test automation without developer support.

There are tools and sales people that advertise magic solutions that require no development skills and promise great results. This has nothing to do with reality. One should be critical when hearing this kind of sales pitch, they either have no idea what they are talking about or they try to take advantage of your trust.

Test automation is closer to development than it is to testing. The test cases and the framework are actually production code. Writing and maintaining that code should be done by following development rules. In some cases automating certain scenario is not trivial and requires quite advanced development skills in order to add the support in the framework for performing such actions - again, you need developers for doing that.

TA is software development – you need skilled people

There are good frameworks available now that allow better differentiation between developers and test engineers - this way test automation tasks can be better divided to suite everyone's skills. Engineers that have more development skills can focus on developing the framework and the engineers that have more product knowledge and testing skills can use the framework & tools created by their colleagues to automate the manual test cases.

TA sometimes requires support from the teams that are developing the software under test. Some applications might not have been built with test automation in mind. In these cases small changes in the application to facilitate easier automation can help. For example, in GUI testing ensuring that there are unique identificators for certain GUI elements will help test automation efforts significantly.

5. Things to consider when choosing the tools

Most important factors:

BUDGET: This can already make certain tools unavailable due to their licensing costs.

RESEARCH: Do research on each possible candidate and gather information (internet is good enough) - make sure that the persons in charge of making the decision actually understand the needs and the challenges involved.

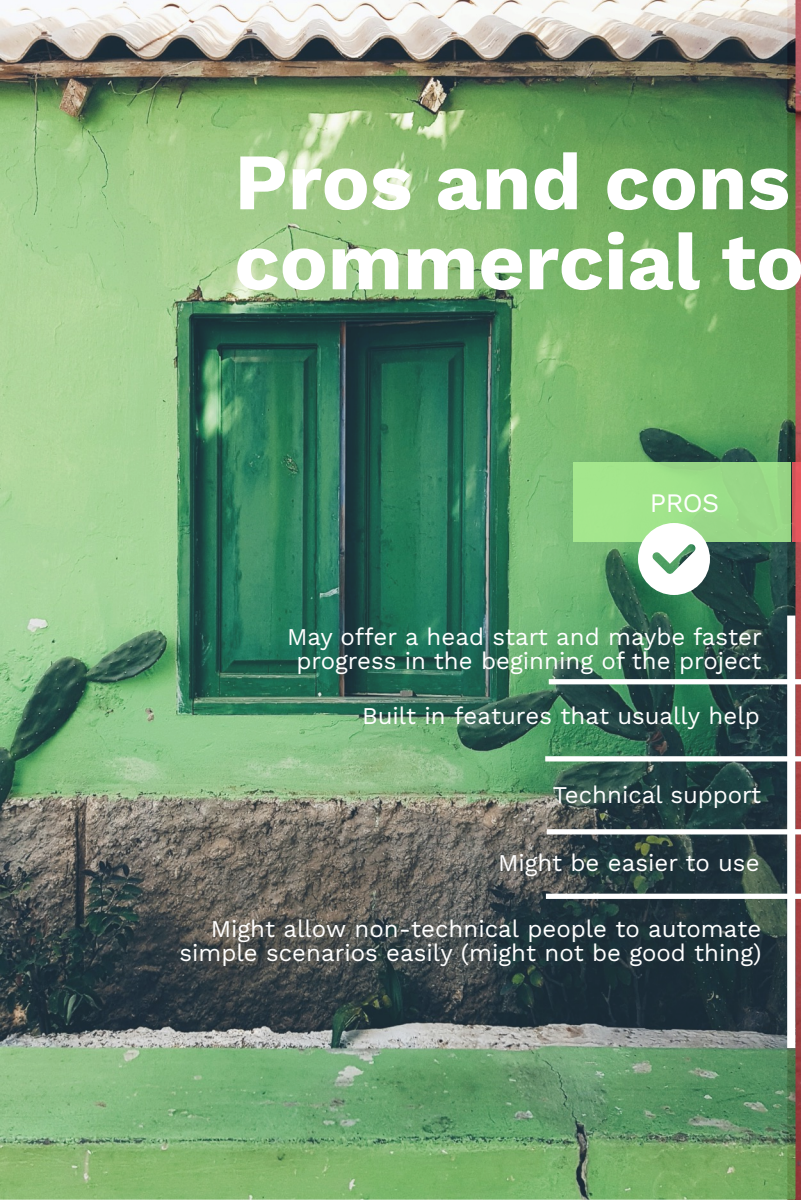
SKILLS OF THE TEAM: This can suggest whether to go for open source or a commercial product. Or the team can be expanded with right people that could handle the challenges of using open-source solutions.

TESTING: Use proper Proof Of Concepts rather than just trust & pick without evaluating.

Pros and Cons of open source tools

PROS	CONS
Flexible - with enough skills you are in control and can implement whatever features needed whenever	Required skills from the team (more techical people - might actually be a good thing)
Some solutions are widely used and communities & support is really good	No support - you are on your own (might be ok with good enough skills)
No extra costs for licensing	Extra costs for maintenance - most likely will be less than paying for licences
Access to source code	Not so many built-in features available out of the box
Most of the tools support various programming languages	Relies heavily on developer support (can also be a good thing)
Tools like Selenium, Playwright and Appium are quite mature	Fixing of bugs & changing the open-source solution might be needed

Pros and cons of commercial tools



PROS	CONS
May offer a head start and maybe faster progress in the beginning of the project	Extra costs from licensing
Built in features that usually help	Implementation of missing features is done by the provider of the solution
Technical support	Not that flexible - some solutions offer only support for single programming language
Might be easier to use	Might not be well suited for future test automation needs
Might allow non-technical people to automate simple scenarios easily (might not be good thing)	No access to source code

6. Tool suggestions for various environments

Robot Framework
Generic keyword driven framework. Easily extensible and can be integrated in most of the projects.

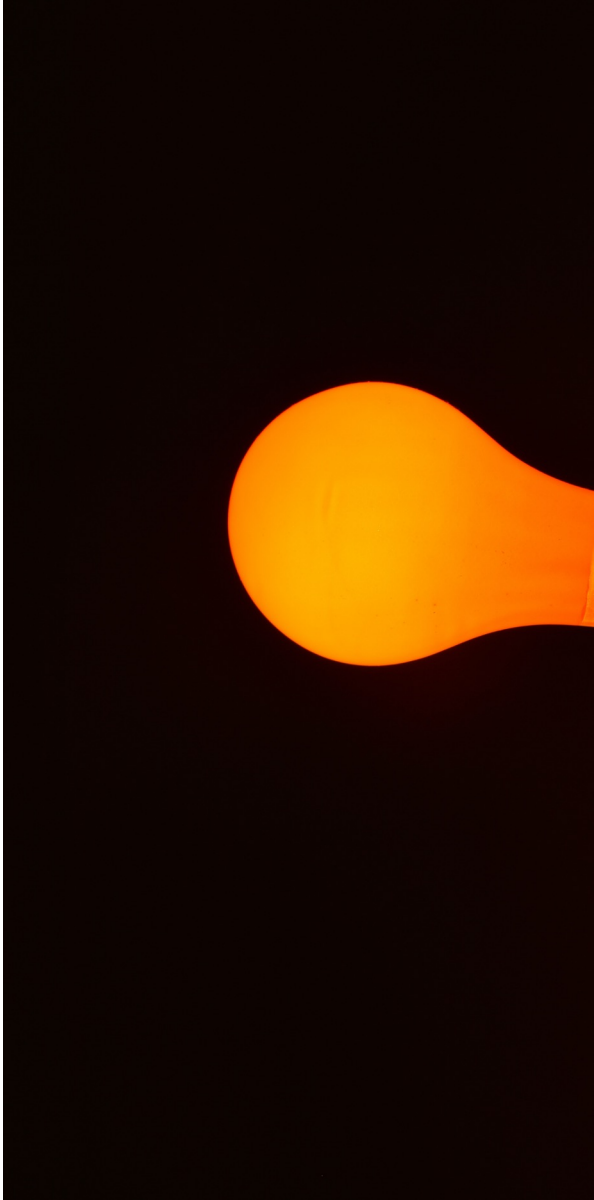
Selenium & Playwright
Widely used open-source tools for web testing.

Appium
For mobile test automation. Can help test case reusage between iOS and Android. Might be slower than dedicated tools. Well suited for blackbox testing of mobile applications.

Espresso
Dedicated tool for Android test automation. Well suited for white-box testing (access to source code of the application is needed). Faster than Appium, closer to development than testing. Doesn't support iOS.

XCTest
Same as Espresso, but for iOS.

OpenCV
Open-source library useful for test automation efforts where image recognition is needed.



7. What happens after the implementation?

- Define processes on how to do test automation:
  - Which manual test cases should be automated and when
  - How test automation should be used in regression cycles
- Focus on company-wide adoption of test automation and train new people to use the test automation tools and framework.
- Build automated regression cycles using the tests that have been automated.
- Implement automated reporting & KPIs extraction etc. for better regression results visibility and progress monitoring by management.
- Add new functionalities for the framework.
- Add new test cases for new features.
- Maintain existing codebase.
- Build continuous integration pipelines that run automated tests against new versions of the software so that bugs are found fast.

** Not all options above might be suited for all projects*

8. Conclusions

- Test Automation is not a magic button. It is a tool that can produce significant benefits, when used correctly.
- Like computers in general, TA is reliable, fast and precise. With these qualities it can save money and improve the quality of testing, as long as you are patient.
- Cons of TA include the costs of the initial implementation and the need for skilled technical experts.
- Not everything should be automated. There are many things that should be tested by humans in the future too.
- Good candidates for automation are tests that: are repeated often, take a lot of time or are difficult or boring for humans. Also, some things can not be tested without automation (like performance testing).
- Proper tools are essential in a successful test automation. Be patient and take your time when selecting the tools.
- Open-source tools can be more flexible and enabling than commercial tools. On the other hand, open-source tools require more skilled experts.

Thank you!

Contact us if you want to know more.

Text content: Dragos Guberna
Layout: Toni Roschier

www.valagroup.com

Teemu Pesonen, Director:
Quality & Automation

+358 400 513 514

teemu.pesonen@valagroup.com

VALA