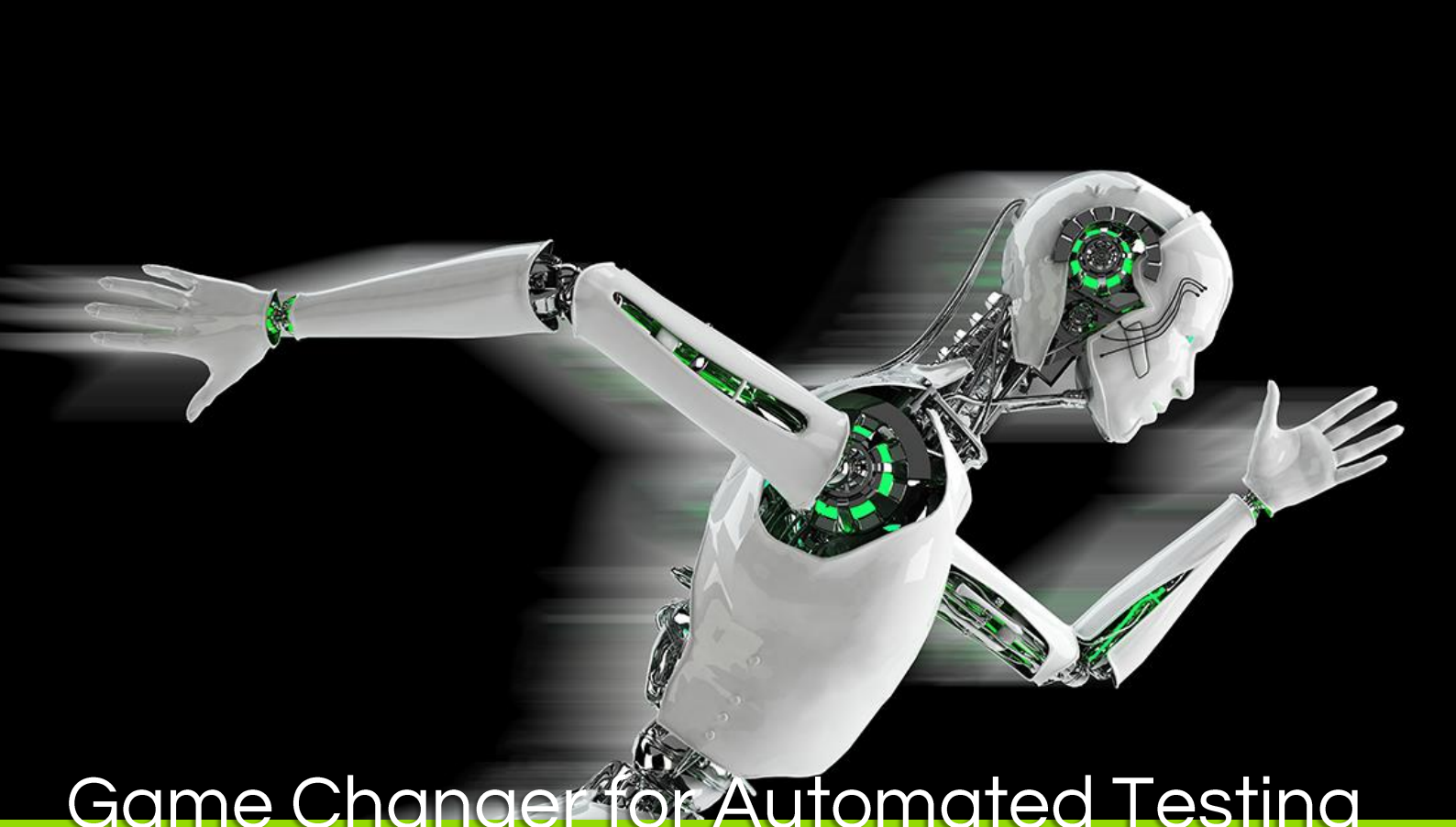


Game Changer for Automated Testing

Creating test cases is 77x faster with Testup.io Modifications and maintenance are 14.6x faster.

Any IT application requires testing before a successful Go-Live. The automation of these tasks allows the inexpensive repetition and continuous monitoring of functionalities and has become commonplace. It is an integral part of modern Continuous Integration and Continuous Delivery (CI/CD). As costs for implementation and adaptation are the driving factors in deciding whether to automate testing, this will be the focus of this paper. By hiring 12 test automation professionals where each of them has the same task but uses their respective favorite tool, we can benchmark these tools. Assuming that all tools find the correct issues, we will evaluate the tools based on the time it takes to create and adapt tests.

Check that the old functionality still works

Test Automation is done for quality assurance in software development. As software changes throughout the releases, it is prudent not just to test the new features, but ensure that old functionalities still operate as designed. As this would result in monotonous repetition of tests for each release, ideally for each function of the application, automation quickly becomes an economic necessity.

The main entry barriers are the costs not just of implementing an automated test, but also to maintain it throughout the releases. If managed poorly, maintenance efforts can explode making test automation appear as a concurrent development to the main application itself.

Find two changes of a website

For our test, we asked our 12 test automation professionals to automate tasks on our website. The website is based on WordPress using the Elementor plugin. We required each automation professional to have at least 3 years of experience in their respective technology.

TESTUP
VISUAL TESTAUTOMATION

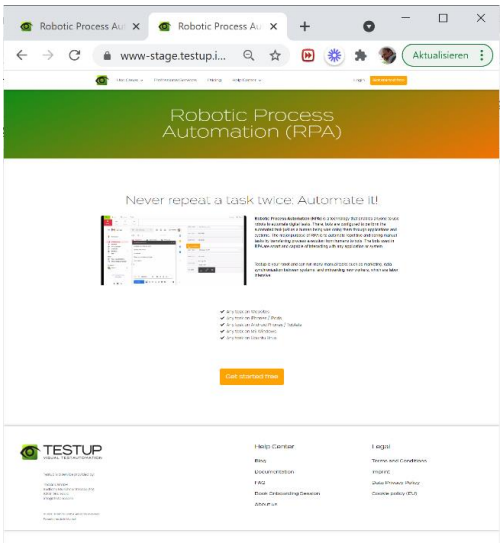
The task consisted of checking 37 Buttons, 22 Links, 2 Hover Effects, 1 Pop-up, 24 Headers, filling 5 Form Fields and checking 1 Signup-Result on multiple pages of a website.

After finishing the test automation, we changed the website in two places covered by the previous tests. We removed one header and a button, to check if this would be detected. We also made an additional modification to the website, which is not covered by the tests, to see how it would affect the automation.

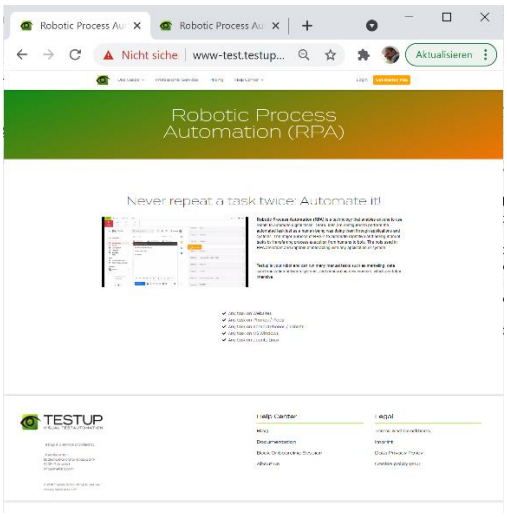
Then we let the testers run the tests, to see if the two changes would be detected and how they would cope with the third one.

Afterwards we asked them to update their tests, assuming that the changes were intentional, so that the test-cases would now pass if run again.

Here an example of a hidden orange Button:



One of the pages of the use case. (Before the change.)



The same page as above with intended change: Orange button is hidden. (After the change.)

Automation tools

To select the automation tools we benchmark, we look at commonly used frameworks. A survey which frameworks are used in practice is provided by reflect.run.

We focus on

- **Cypress** + BDD: A relatively new Open Source framework with a focus on Google Chrome.
- **Ranorex Studio** (Scripted/Recorded): A commercially available Software package capable of automating Windows as well as Web Applications.
- **Selenium** (+ Python/Java): The cross browser standard for test automation by providing an API to the browser interactions.
- **TestCafe** (+ Python/Java): Simplifies the usage of the browser interactions by eliminating selenium boiler plate code.
- **Testup.io**: A cloud service in which you record tests by interacting with your application through VNC.

Not all tool delivered correct issues

Each of the applications should find at least the two changes which were covered by the test case. The third and additional change should not be found. But, due to different implementation details, some of the implementations detected artifacts, additionally to the intended changes, so called false positives: One of the Selenium Python implementations returned non-visible changes in a Title text. Another implementation detected a change of a button, which was again not visible. Only Testup.io detected the two intended changes and did not report any false positives.

Benchmark results

Test Creation

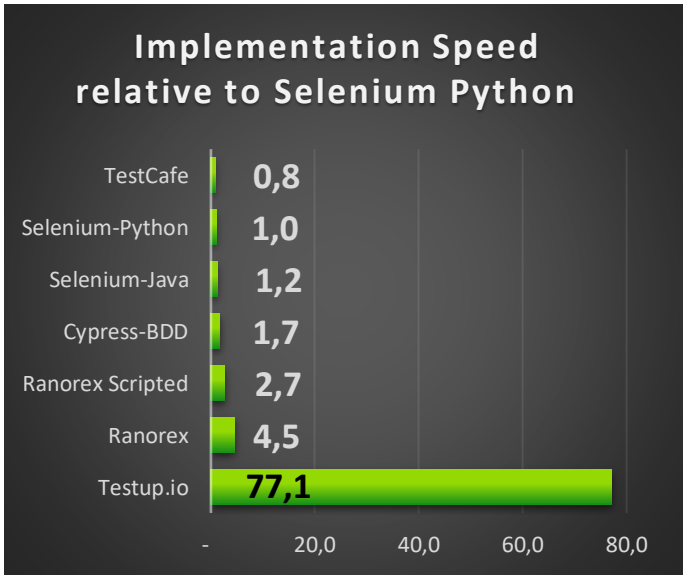
The creation took our automation professionals the following average times:

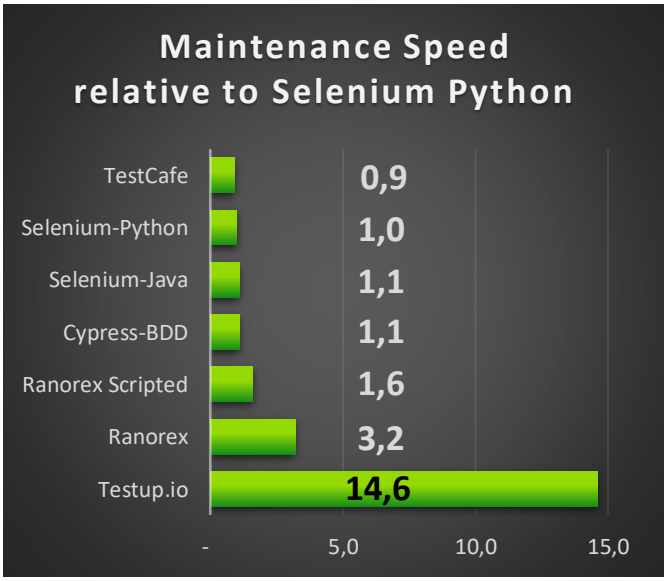
Cypress – BDD	5 Hours
Ranorex Recorded	3 Hours
Ranorex Scripted	8 Hours
Selenium Java	11 Hours 30 Min
Selenium Python	13 Hours 30 Min
TestCafe	16 Hours
Testup.io	**10 Minutes**

Test Modification

Then we went on to measure the time it took, to update the respective test-cases:

Cypress – BDD	1 Hour 30 Min
Python-Java	1 Hour 30 Min
Python-Selenium	1 Hour 35 Min
Ranorex	30 Min
Ranorex Scripted	1 Hour
TestCafe	1 Hour 45 Min
Testup.io	**6 Min**





Visualizing the same data as above, we see that Testup.io is 77x faster than current best practices for test implementation.

Testup.io is 14.6x faster for test maintenance than Selenium Python.

Testup.io is an order of magnitudes faster than the other tools. How is that possible?

Test automation is a task which closely emulates user behavior. Using a programming framework, this task requires intricate knowledge of the browser, especially the internal data structures of the application like the Websites DOM and JavaScript. On the other hand mimicking the user interactions based on image recognition allows to automate tests without this internal knowledge. Testup.io is the only application following this path consistently.

Ranorex is the only tool in the set of tools we considered in this benchmark which allows test creation and test maintenance by capturing user interactions. But, checking if the website is in the correct state still requires internal knowledge like checking if an xpath expression is available in DOM. Instead, Testup.io is based purely on visual checks. It loads the website and displays it, while clicks and checks are performed based on image recognition. Thus, defining a check is significantly faster than in Ranorex. A user can create test-cases without any programming knowledge by simply clicking through as they would in a manual test run.

At the same time, as Testup.io is limited to what a user sees visually, it cannot inspect or check hidden states easily. For these kinds of checks, a Testup.io user would record test steps to open the browser's developer console and perform the check there. This would be as slow as the traditional code based approach. But access to the developer's console is possible in Testup.io in contrast to most competitors like selenium and cypress cannot.

Savings by Introducing Testup.io are Significant

Benchmarking the new Testup.io with state of the art tools for automated end-2-end testing results is a significant edge for Testup.io. Test creation and Test maintenance are an order of magnitude faster with Testup.io than with Cypress, Ranorex, Selenium or TestCafe. Testup.io removes the two main cost hurdles for test-automation, as both creation and adaptation can be done quickly and with ease.