eficode

EFICODE QUICK GUIDE

TEST AUTOMATION & ROBOT FRAMEWORK

WWW.EFICODE.COM

CONTENT

EFICODE QUICK GUIDE

TEST AUTOMATION

End repetitive work, do
things only once.

Test automation is not just a cost-cutter.

Test automation helps driving development more effectively, react to customer requirements better and release changes for customers much faster.

Test automation is
an inseparable part of modern
software engineering.

www.eficode.fi

In just a few years, test automation has become an essential part of functional, customer-oriented product development. In order to utilise organisations' core competence as effectively as possible, repetitive routine tasks must be handed over to computers.

Test automation is best demonstrated with a practical example. Consider an average software project with roughly 500 test cases to be executed on the browser. If each test case takes five minutes on average to execute, testing the entire software just once takes a single tester the equivalent of about one week in working hours. If the cost of the tester, a hired external consultant, is added to this, a single test execution costs approximately 2 000 euros. When we take into account that tests are done using the three most common browsers, each with its own test execution, the costs of a single, manual test can easily climb to 6 000 euros.

> 500 manual tests x 5 min = 42 hours of work

The information we have gathered from customer projects shows that the automation of a single, normal browser-based test takes two hours at most, so the automation of the test cases in the example would take roughly one thousand hours in total. Even if the test automation expert would cost twice as much as the tester, the automation costs would still be less than one hundred thousand euros, or less than twenty full, manual test executions.

> Test automation pays for itself back on average in six months.

EFICODE RESEARCH

Test automation reduces the cost of quality control more than

DURING THE PROJECT

25 %

IN THE LONG RUN

75 %

The repeated execution of automatic tests does not incur any additional costs in practice. In an average software project, test automation therefore easily pays for itself in less than six months. We have proved this in several of our automation projects.

Test automation changes the organisation's approach

Software testing has traditionally been seen as a phase in which the functionality of the implemented software is ensured using documented test cases at the end of development.

In order to create competitive, enjoyable products in addition to fulfilling the original need, it is no longer enough that the functionality of services is ensured immediately before their release. And in order to efficiently respond to error messages and change requests from users, it is no longer enough that the development proposals and change requests are consigned to the next biannual release.

If all quality assurance is done manually during the most active development phase, the software will usually have time to change significantly in the time used for testing and, at best, the implemented and previously tested features may have already been broken for several days. In modern software development, information about the quality of the product must be received much faster so that development measures can focus on producing a high-quality product more effectively and reliably.

Product development must pay even more attention on the usability of a product and user needs already before development begins and especially during the active development phase. When used appropriately, test automation offers testers better opportunities for exploratory testing where new, unidentified problems are searched for while ensuring that the product is easy and agreeable to use.

Linking automatic tests with the requirements of the implemented system helps to monitor the real-time progress of development more reliably also from a business perspective.

Test automation enables an agile organisation and real-time quality control

Test automation brings quality assurance to configuration and implementation. When the requirements set for a product are linked to automatically executed test cases, software implementation and the fulfilling of requirements can be monitored in real time. If automatic tests are also written in natural language, the entire development organisation may engage in open discussion with the help of the requirements and the related automatic tests. This reduces misunderstandings and helps to react to changes much faster.

Does maintaining tests incur costs?

Automatic test cases should be considered as part of the product and product development. Once the test cases have been created, they must be maintained in the same way as the product's software code and its other dependencies. When the product changes, the test cases related to the change must usually be updated to correspond to the new functionality.

Whether making these changes incurs extra costs as compared to updating the documentation of, for instance, manual test cases depends on the product under development. If automatic test cases have been designed well and implemented structurally correctly, the updates related to software changes must ideally only be made in one place so that all the test cases using the new feature are simultaneously updated. This is known as keyword-driven test writing and you can read more about it on the other side of this guide.

How should test automation be initialised?

Naturally, the easiest and most cost-effective way to introduce test automation is in a new product development project. However, this does not imply that test automation will also bring cost savings in the quality control of products that have already been implemented and are maintained.

When test automation is introduced to a new product development project, it should be tied in with the entire product development cycle. Test cases should be linked to requirements and written at the same time as the feature in question

is implemented. This ensures that the test is ready roughly at the same time or even before the new feature is complete and that it is implemented as was originally intended.

New development projects

- Choose a test automation tool providing test results that can be understood by all individuals involved in the project.
- Make sure that the test automation tool can be integrated with the technologies used in implementation.
- Make sure that the tests can be integrated with the requirements of the product. The fulfilment of requirements can consequently be monitored using test results.
- Make sure that the individuals responsible for writing tests are comfortable with the test automation tool in question.

Products in maintenance

- First, consider whether introduction of test automation is justified for the product in question. How much does quality assurance currently cost and how long life cycle does the product have left?
- Choose a test automation tool with which the product can be tested as effortlessly as possible through interfaces or user interfaces that have already been created.
- Try to determine the key functionalities of the product that should be tested first. In addition to gaining the most benefit from automation, you can simultaneously estimate the cost of initialising test automation in terms of the entire product.
- Make sure that the individuals responsible for further development and maintenance of the product are motivated to learn test automation and maintain test cases as part of other maintenance and development. Test automation will otherwise go to waste soon after its initialisation. ■

1. Automatic tests save time and money

Automatic testing is practically free and always faster than manual testing. Tests that have been automated once can be executed practically as often as required and in different environments, such as terminals and browser versions.

2. Test automation forms the basis of an agile organisation

Automatic test cases linked to requirements and written in natural language provide a solid foundation for software development. When the entire organisation can engage in open discussion surrounding the test cases, the number of misunderstandings decreases and it is easier for development teams to assume full responsibility for their implementation measures.

3. Automatic quality control describes the state of development in real time

Automatic tests can easily be combined with product requirements. When the tests are executed after each change made in the software, the state of product development can be monitored reliably in real time.

4. Human resources are put to better use

Once automated, the tests allow know-how to be allocated towards product development, the planning of better test cases and the testing of the usability and ergonomics of the implemented service, among other tasks.

5. Cooperation with suppliers becomes easier

Test automation covers a large share of the quality gates, which can easily be used as a starting point for negotiations concerning software acquired elsewhere. After this, it is not a matter opinion how well the agreed quality gates and metrics are implemented, but energy can be focused on constructive cooperation.

6. Different environments can be tested in one go

If tests have been automated appropriately, test cases can be executed against various environments, such as different terminals and browser versions, in one go. This guarantees that all of the environments used by customers behave similarly and no unpleasant surprises arise with e.g. new browser versions.

7. Automation facilitates difficult, lengthy testing targets

Unlike humans, automation does not get tired of executing individual test cases that last for hours or more. When a bug is found during testing, automation can dig under the surface and directly diagnose the sub-layer of the software where the problem is located. This also makes fixing the problem faster.

8. All quality reporting can be combined

There are more and more institutional and other reporting liabilities concerning software. Instead of writing reports by hand, mechanisms can be developed in automation systems to produce a report in the exact form that is required by the party in question.

9. Reported bugs are no longer lost

Bugs found by the end user can be written down as automatic tests already before they are fixed. This ensures that they are not accidentally lost or remain untested.

10. Further software development is made easier in the long run

The greatest disadvantage of large, long-term software is the unexpected effects of a new functionality in parts where it is no longer known which business-related decisions led to a particular functionality. Automation not only finds unexpected bugs already during development, but also documents business-related decisions. It is easy to see that a business-related decision is no longer valid and therefore the software can be safely edited. ■

CHALLENGE / SOLUTION

Start-up costs of test automation are high.

Test automation should always begin with the automation of some of the most important and clear test cases, so that it is easier to estimate the cost of automation and compare it to current costs.

Our product cannot be tested automatically, because it is so complex.

We have not come across a single environment in our project history whose testing could not be automated with the right choice of tools.

We have no time for test automation at present.

Test automation is often neglected due to other urgent matters. It is therefore important to begin automation in an agile way and, if your own product development team is busy, choose a suitable partner for the project.

It is difficult or even impossible to set up an automatic test environment.

A product is practically always tested somewhere before it is updated in the production environment. If no other options are available, test automation may begin in this environment.

Automatic tests may not be carried out on the product, because product development has been outsourced to a subcontractor.

Who controls product development, the company or its subcontractor? Subcontractors are often ready to also develop their own business, but test automation should also be taken into account when renewing contracts in problem cases. ■

The easiest way to try out test automation is Eficode’s test automation pilot. During the pilot, our experts survey the most important test cases with you and use them to introduce test automation for your product. Product development also involves other approaches that comply with the devops-principles, such as continuous integration, which can be used to monitor the results and the development of test automation effectively in real time.

eficode

EFICODE QUICK GUIDE

ROBOT FRAMEWORK

Leave the routine testing
for Robot.

The development of Robot Framework started at Nokia Networks in 2005 and in ten years it became the standard tool for organisations during the development and testing phase.

Robot Framework provides
the easiest way to combine the software
to be tested with automatic tests written
in natural language.

www.eficode.com

ROBOT FRAMEWORK

The strength of Robot Framework lies in the fact that automatically executed test cases are written in a reusable and intelligible form.

The tests are connected to the tested system using test libraries, ensuring that test cases can always be written in the same way regardless of the environment. Thus, the entire development organisation can use the same tool for test automation and writing uniform test cases.

Test cases written with Robot Framework are based on reusable keywords. Thus, when the functionality of the software changes, there is no need to rewrite each test case, but it is enough that the keywords related to the changed parts of the software are updated to correspond to the new functionality.

Although tests are usually written in English, Robot Framework does not limit the choice of test language. Therefore, test cases may be written in e.g. Finnish, if needed.

Hundreds of thousands of test cases have been written with Robot, and it is used by thousands of organisations around the world. In addition, seven Finnish IT companies have established a foundation around Robot Framework to guarantee the development of the tool as an independent, free and open-source software. ■

WHY ROBOT FRAMEWORK?

FLEXIBLE Robot Framework can be adapted to test almost any software or product. It is ideal for testing web, mobile and integrated software and the related hardware.

NATURAL LANGUAGE Tests written with Robot can be read by everyone as they can be written in e.g. Finnish. Requirements can be seamlessly linked to the tests.

FREE As it is open-source software, using Robot Framework for testing is free. Not only do you keep your costs in check, but you can also bid farewell to vendor lock-ins.

POPULAR Many of the Fortune 500 companies, and thousands of smaller ones, already use Robot Framework. ■

EXAMPLE TEST

The example test below displays making a wire transfer in an online bank. The name of the test is given in the first row, and each indented row is one keyword. The keyword is linked via the test library to a specific action, such as entering data in the text field displayed on the browser screen or clicking a button. Keywords may also be assigned variables with which test cases can be executed using different combinations of test data.

```
Making a wire transfer between bank accounts
    Log in to netbank
    From shortcut menu select          New payment
    Insert recipients account          FI4950009420028730
    Insert recipients name             Testi Anna
    Insert message                     test payment
    Insert amount                      100
    Click continue
    Insert correct pin code
    Accept payment
    Verify payment has been transferred correctly
```

BEST PRACTICES

Write test cases together and link them with requirements

When test cases are written together, all of the individuals involved in development go through the requirements and functionalities of the implemented product together. In this way, writing test cases provides a better mutual understanding of what is being done.

Execute tests after each change

When tests are executed after each change, any bugs or faults will be detected as early on as possible. Also, new functionalities are not accidentally added on top of a broken product, which might, at worst, have to be taken down later. This approach is known as continuous integration and it is one of the cornerstones of the devops methods.

Execute tests also on development branches

In order to guarantee that the combining of new features implemented on different development branches of a product into a single release would go as smoothly as possible, it is important that test cases are also executed when changes are made on development branches.

Execute parallel tests

Executing parallel tests expedites feedback and improves product architecture. Dependencies are not formed between test cases when they are executed at the same time and ideally against environments set up for testing.

Automatic test cases are part of the product

Written tests must be maintained similarly to software code. Naturally, the aim is to have test cases that are as easy to maintain as possible, but if the functionality of the tested system changes, test cases or keywords must also be changed.

Invest in good test data

At best, the test environment corresponds to the production environment so well that the test cases executed against it can be used to release the latest version for customers. An essential part of such testing is the test data used to test the system. The data determines what information is visible in the product's test environment. ■

TECHNICAL BENEFITS

1. Automatic tests guarantee up-to-date documentation

When test cases are written in natural language, they describe the properties of the implemented system in an intelligible form. At the same time, they also guarantee that the product works as desired. Successfully executed text cases therefore form an up-to-date picture of the functionalities of the system.

2. Automatic tests are always executed in the same way

When tests are executed manually, there is always a risk that some of them remain undone or the test results are unreliable due to a typing error made by the tester. Automatic tests, on the other hand, guarantee that the product functions always in the same way. If automatic tests guarantee the desired functionality sufficiently, the resources freed up from manual testing can be allocated to more productive work.

3. Bugs are discovered faster

Automatic tests are ideally executed after each change so that system failures are detected almost immediately. When bugs are found in time, development is more gratifying and the costs incurred by fixing bugs decrease significantly.

4. Greater test coverage

When a test case has been automated, it is added to a regression test set, which can be executed practically as often as required. Consequently, each automatic test case increases the test set that ensures the functionality of the implemented product.

5. Test automation makes development more secure

When automatic tests guarantee the functioning of the implemented product, development achieves a whole new sense of security. The focus of development can shift more freely to making sensible technical decisions instead of cautious changes, because the development team can be certain that an automatic, targeted error message will signal any failures. ■

```
*** Settings ***

Library      Selenium2Library timeout=15

Test Setup   Open browser and go to homepage
Test Teardown    Close Browser


*** Variables ***

${BROWSER}   firefox
${HOMEPAGE}  http://www.google.fi


*** Test Cases ***

Finding a blog article on Robot Framework
  Search google for          Robot Framework Eficode
  Click search result        Automatic testing with Robot
                             Framework pt. I ... - Eficode
  Wait Until Page Contains   short video tutorial on the
                             power of the Robot Framework and
                             Selenium


*** Keywords ***

Open browser and go to homepage
  Open Browser  ${HOMEPAGE}     ${BROWSER}

Search google for
  [Arguments]   ${search}
  Input Text         name=q          ${search}
  Click Button       name=btnG

Click search result
  [Arguments]   ${link}
  Wait Until Page Contains      ${link}
  Click Link                    ${link}
```

Robot Framework is the most common answer given by Eficode when discussing which tool to choose for the automation of acceptance testing.

It is a generic application framework designed to be easily expandable and created from the very beginning specifically for acceptance testing and its automation. Robot expresses business requirements as test cases written in natural language, and actual testing measures are carried out using libraries that integrate different testing programmes and technologies together to make the computer do what has traditionally been done manually by a tester. It is easy to accept Robot Framework as the starting point and foundation on which to build full test automation, whether the tested software is a simple mobile app or a customer relationship management system consisting of multiple different programmes.

Write intelligible test cases

Test cases in Robot Framework can be written in natural language so that, if necessary, the agent can even personally write what the tested software needs to do. However, test cases are automated by software engineering so that the tests serve as aids and stimulate essential discussion. Communication intensifies, and it can be explicitly verified whether the functionality works as desired; all you need to do is to execute a test case written in plain language and see what happens.

Triumph for open-source software

During its over 10-year lifespan, Robot Framework has almost always been an open-source product although its development was sponsored by Nokia Networks for years. Due to its openness, there are already several free Robot Framework libraries for the most common testing targets, such as websites, interfaces, mobile apps and databases. Those who prefer open-source products also avoid expensive vendor lock-ins as both experts and competitive training services are more readily available. ■

		ROBOT FRAMEWORK	CUCUMBER	FITNESSE	SELENIUM
01	Open-source software	✓	✓	✓	✓
02	Core technology	Python	Ruby	Java	Java
03	Technologies featuring “out-of-the-box”	C#, Java	Java	C#, Python	C#, Javascript, Python, Ruby
04	Test cases are...	...in a file in natural language	...in a file in natural language	...a table on a wiki page	...software code in a file
05	“Workflow” test cases	✓	✗	✗	✓
06	Data-driven test cases	✓	✗	✓	✗
07	Gherkin test cases	✓	✓	✗	✗
08	Reporting	HTML	Output in a terminal programme	HTML	Not available
09	Expandable with what? (Programming languages)	Natural language. All.	Ruby	All	C#, Javascript, Python, Ruby ja Java
10	Usable with the most common CI servers	✓	✓	✗	✓
11	Own specialised editor	✓	✗	✗	✓
12	Optional editor	✓	✓	✗	✓

HOW DO I GET STARTED?

1. Gather and define test cases

The most important thing is to first outline the test cases that already exist to guarantee the functionality of a product and the additional test cases that are required. Without an understanding of the testing needs, implementing test automation is useless.

2. Automate a few test cases

When you begin with a few critical or easily executed test cases, you can quickly observe the challenges that may arise during the implementation of test automation.

3. Accept test automation as part of everyday life

In order to make the most of automatic tests, it is important that executing and writing tests is accepted as part of everyday product development routines. This ensures that test cases actually test that the product functions as desired and that the product's test coverage grows steadily.

4. Expand test automation to cover the entire product

Each automated test case reduces the repetitive work done in quality assurance after each change or, at the latest, before the release of a new version. The more thoroughly a product has been automatically tested, the more cost-effective its further development and maintenance are. ■

LINKS

Robot Framework's homepage:
robotframework.org

Robot Framework Screencast:
www.eficode.fi/blogi/maintainable-automatic-tests-for-your-web-application

Eficode's Robot Framework Blogs:
www.eficode.fi/tag/robot-framework

EFICODE'S ROBOT FRAMEWORK SERVICES

Eficode helps companies to implement Robot Framework in their own business. We organise Robot introductions, training sessions and longer test automation projects.

The core competence concerning automatic product testing is always passed on to our customers so that long-term product development would be as effective as possible and that vendor lock-ins threatening the core business would not be formed.

CONTACT

Marko Klemetti
CTO
marko.klemetti@eficode.com
044 522 5927

HEIKKI HÄMÄLÄINEN
Head of DevOps
heikki.hamalainen@eficode.com
+358 (0) 40 766 2610

KAJ JOKINIEMI
Vice President of DevOps
kai.jokiniemi@eficode.com
+358 (0) 40 592 6257

Tatu Kairi
Domo Arigato Mr. Roboto
tatu.kairi@eficode.com
040 533 9559

TEST AUTOMATION ROBOT FRAMEWORK

Eficode Oy
Pohjoinen Rautatiekatu 25, 00100 Helsinki
Birger Jarlsgatan 18A, 2tr, 114 34 Stockholm
Åkerlundinkatu 11 A,33100 Tampere
Uplandsgade 56, 1 sal, 2300 København S.
Marcel- Breuer- Str. 15, 80807 München
De Entree 143, 1101 Amsterdam