SQmag

THE VOICE OF SOFTWARE QUALITY

TEST AUTOMATION

USAGE-CENTRIC TESTING

Building a decision framework to help us design better tests.

Bruno Legeard & Joan Racenet

BEYOND THE BUZZWORDS: TEST AUTOMATION

Test automation is a valuable tool for any software development team.

Luis Francisco Contreras González & Martin Contreras Romo

RESCUING TEST AUTOMATION FROM ITS DOWNFALL

What's the insight we want our automated tests to give us?

Pieter Withaar

FLAKY TESTS: THE SILENT KILLER OF SOFTWARE QUALITY

Flaky tests are like a canary in a coal mine.

Rabeb Mnani

TEST AUTOMATION, JAPANESE SOFTWARE MARKET

All these challenges make it difficult for companies in Japan to effectively implement test automation.

Hiroshi Yokkaichi

WHY TEST AUTOMATION MATTERS

Let's see what technology thinks - meet our guest author - we highly recommend this reading.

Guest Author

#13
may 2023

A4Q Certified Selenium 4 Tester Foundation

A4Q
Selenium 4
Tester Foundation

alliancefor qualification.com

> THE FUTURE DEPENDS ON TEST AUTOMATION, TODAY, TOMORROW, AND BEYOND

Stephan Goericke

CEO, International Software Quality Institute

#13

CONTENT

DEAR READERS,

At a time when the software development industry is evolving rapidly, and software is becoming increasingly complex, test automation is what companies need to ensure their continued growth and profitability. Any company considering digital transformation must first be clear about where test automation development is headed and which tools are best suited to ensure continued development and implementation success. The advantages of test automation for software testers far outweigh sitting in front of a computer carefully going through application screens manually. Firstly, automated testing allows a tester to playback pre-recorded actions, compare results, and report the success or failure of the test result findings most efficiently. This makes automated testing a crucial feature of development practices. For significant software development, companies consider automated software testing critical; it reduces time spent on testing because the scope of software quality testing is far more accurate, high-quality, and quicker. Software testers manage a role of intense responsibility. They ensure the safe delivery of quality-assured software development products and the mobilization thereof. Software testers continually test the software for errors, bugs, or defects. In this issue, our guest author shares how test automation will perform in the future. There are several exciting developments to look forward to. As software development continues to evolve, so will test automation. We highly recommend reading this article and checking out how technology thinks. Our authors explore the topic of test automation. Further, Luis Francisco Contreras González and Martin Contreras Romo write about test automation as a valuable tool for any software development team, but also how important it is to consider the needs of the application being developed and selecting the right tool. Pieter Whithaar writes about what insights can be gained from test automation and the benefits thereof. Hiroshi Yokkaichi shares his insights on how crucial test automation is in software development and quality assurance in the Japanese software industry. Even with the unique challenges, the lack of skilled human resources, complex regulations, and high communication costs with low-skilled engineers, the need for an effective testing process is increasingly important as the demands of the market desire fast and high-quality software delivery. While Rabeb Mnani writes about Flaky tests and their detriment to software quality, Bruno Legeard writes about building a decision framework to design better tests and optimize automated test suites. That is the entire point of usage-centric testing. As always, automation technologies continue to underpin all areas of business innovation, and for the software testing sector to remain effective, agile, and flexible, knowledge is fundamental.

LET'S GO FURTHER; LET'S BECOME MORE.
Enjoy the read.

FLAKY TESTS:

The silent killer of software quality

With the evolution of web and mobile applications, the building of maintainable and reliable automated tests has become a crucial need to ensure high quality software. But what if these tests do not have the same outcome for the same code and the same environment and they are failing for no obvious reason? This behavior is called the « flakiness » issue which is one of the most challenging problems in software development as they can waste a significant amount of time and resources and impede the release of new features. They can also undermine the credibility of the test suite and decrease the confidence of developers and stakeholders in the software. According to some studies, flaky tests can consume up to 50% of a development team's time and they can cause delays in software releases and increased costs.

A frequent example of the impact of flaky tests is when a test suite that is designed to be run daily starts to fail frequently, this will slow down the development process as the developers will need to investigate and debug the test failures before they can continue working on new features. Another example is when a flaky test causes a regression in the software and it goes unnoticed, this can lead to customer complaints and the development team will have to spend extra time fixing the issue and testing the software again. The goal of this article is to explain the problem of flaky tests and provide strategies for preventing, identifying, and fixing them to ensure a stable and the reliable test suite, and to provide best practices for maintaining a stable test suite.

> FLAKY TESTS ARE LIKE A CANARY IN A COAL MINE. THEY ARE A WARNING THAT SOMETHING IS WRONG AND NEEDS TO BE ADDRESSED.

Causes of Flaky Tests

It can be hard to point out the exact cause of flaky tests as there can be a variety of reasons why these tests are failing. The common cause of flaky tests includes changes in the test environment, test design issues, and test data dependencies. The test environment can include things such as the operating system, software dependencies, and configurations. An automated test that runs correctly on Windows might fail on Linux, or an update to a dependency might cause a test that previously passed to fail.

Also, changes to the test environment can also include changes to the test infrastructure, such as updates to testing frameworks or libraries. Test design issues may cause unreliable tests that can be poorly written, tests that are tightly coupled to the product implementation or tests that make assumptions about the state of the system can fail when the product is updated with new versions or when the state of the system changes. Besides, tests should properly handle exceptions and errors to avoid test failures.

Tests that rely on external data or services, such as databases or APIs, can become flaky if that data or service changes. For example, a test that relies on a specific dataset might fail if that dataset is modified or deleted, or a test that relies on an external API might fail if the API's response format changes. Automated tests that have timing-related issues, such as tests that rely on hard-coded sleep or wait times, can lead to flaky tests as the

test results maybe dependent on the time it takes to execute the test. Additionally, the developer should avoid using a lot of sleep time and use instead implicit waits because it makes the test slower and cause it to fail when the test suite is run on a different environment with different resources. Shared resources can lead to Interference with other tests by modifying the the same resource can cause flay tests as it may be dependent on the order in which they are executed. All these causes can make automated tests flaky and unreliable, and they can be frustrating for developers to investigate but understanding the causes can help to better prevent this issue.

How to reduce flaky tests?

Not having reliable tests in place can be a major issue for any development the team as it affects its ability to create quality code and prevent finding bugs. To eliminate the flaky tests, we need first to isolate those tests, keep them in another automated test suite or mark them as flaky in the original test suite so that we can keep the remaining deterministic tests aside. In this case, it is better to use a test framework that provides built-in support for retrying falling tests.

Then what to do with the isolated tests? Isolating tests means decreasing the regression test coverage, so these tests should be treated quickly. Isolating test environments from external dependencies can help prevent flakily tests. This can include using virtualization or containerization technologies to create isolated test environments or using service virtualization to simulate external dependencies. In addition, Developers should consider the best practices in test design. This includes designing tests that are independent of the implementation, designing tests that handle exceptions and errors properly, and designing tests that can handle concurrency and timing issues. Managing the test data and resources in a controlled way can reduce flaky tests. This can include using test data that is less prone to change, such as data that is generated by the application or using test data management tools to version and manage test data.

Besides, using continuous integration tools and testing can help to detect flaky tests by running tests regularly and detecting failures as soon as they occur:automated tests should be run every time the test suite code is changed, or the product has been modified so that developers can detect test failures early and address them before they become more difficult to fix. Tests Reports and test logs are indispensable for investigating test failures. Sometimes, tests are not deterministic due to the user testing frameworks and libraries which are not well maintained. We should ensure that the tests are running on a stable and well-supported platform. Additionally, code review is one of the most crucial factors that can ensure a stable test suite by catching the test design issues, identifying test data, and detecting concurrency issues: Reviewers can check if the tests are properly designed to handle concurrency, and if not, they can suggest changes to fix it. Also, they can check if the tests are dependent on external data or services. By having other developers review the tests, it is more likely to catch issues early and to have a better understanding of the tests and their purpose. This can help to maintain a stable test suite and ensure the quality and reliability of the software.

Finally, the causes of flaky tests are essentially related to imperfections in the software development process which is revealed by failed tests. By investigating the cause of flakiness, we are taking a step back to call into question some practices and software processes that need to be reviewed. This is what makes flakiness an anomaly indicator for automated tests that should be carefully considered and not ignored.

It's worth mentioning that Artificial Intelligence may have the potential to help prevent flaky tests by providing new tools and techniques for tests and maybe one day.

Rabeb Mnani

I am a QA Automation Engineer with six years of software development and testing experience, specializing in automating Financial applications. I have a proven background in Software Testing in Client-Server Applications and Web-based applications using Manual Testing Techniques and Automated open source Testing Tools. I have extensive Knowledge of Quality Assurance standards, methodologies, and strategies and am certified by ISTQB® Foundation, Professional Scrum Product Owner, and A4Q Selenium.

USAGE-CENTRIC TESTING:

a shift-right approach to support E2E test automation

Introduction

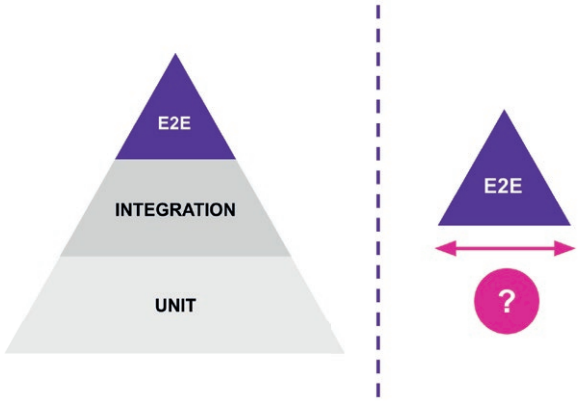
By embracing DevOps mindsets, software development teams can continuously deliver new value to their customers. The testing process followed the lead and became continuous. Teams don't just test during development but also in the context of the application in operation, the so-called shift-right.

Testing in production has many advantages. By monitoring an application in its users' hands, we can quickly discover errors, bugs, and performance issues. We can fix them before causing any more damage to our customer's satisfaction. We can also do A/B testing on a feature's design to see what works better for our users and learn how they interact with our application.

Knowing our users' behavior can help us design more accurate tests and targeted test suites. We can keep improving quality and reducing the release cycle simultaneously. In this article, we will discuss how Usage-centric Testing can strengthen our end-to-end test automation.

End-to-end test automation, a love/hate relationship

Before we dig further into Usage-centric Testing, let's recall why it is important (and complicated) to design and automate the most relevant end-to-end (E2E) tests. E2E tests focus on validating the right functioning of the system/application. This includes testing business processes (from beginning to end) under production-like circumstances from the user's perspective. E2E test automation generally aims to detect regressions on the most critical user flows. Let's position E2E testing on our good old test pyramid:



E2E test position on the pyramid hints at two facts about those tests:

- **E2E tests are expensive to produce and maintain.** First, designing them requires good business knowledge, and their automation requires some technical bootstrapping (setting up dependencies, database flushing, a set of first automated steps for the basic actions in your application like the login, etc.). Then, even if writing new test cases becomes smoother with time (once you have a proven library of automated steps), maintenance remains complicated and time-consuming. Every minor change in the UI results in tweaking the E2E test code to correct it in case of test code breakage.

- **But these tests give us reasonable confidence to release the application.** This value is the positive counterpart of the previous point: E2E tests provide us with confidence that a tested user journey won't break. E2E tests seek to simulate the real world as closely as possible to prevent anomalies during actual use.

E2E tests are expensive but provide value, so we want a few of them, but which one?

Sizing the E2E test suite: an arduous task

« The number of tests in our test suite grows exponentially… but the number of bugs doesn't decrease accordingly! »
« We have too many tests, and a lot of them cover functionalities that are not relevant anymore… »
« We have limited resources to develop and maintain our E2E tests; we don't know how to prioritize our test effort… »
This is the kind of statement we can hear when conducting interviews with QA engineers, test managers, and developers we get in touch with. At some point during the development of software, the number of E2E tests grows to become a bottleneck that slows down our ability to deliver quickly.

How do we fix our love/hate relationship with E2E tests, so it becomes a "love/love" relationship? Finding the proper minimal set of E2E tests is critical to balancing test suite execution speed and feature coverage. But how? By focusing our tests on our users' most essential journeys, which are, therefore, the most critical ones to our business. As we can't do this before delivering new features, we have to "shift right" a bit, so we can incrementally and iteratively improve our E2E test suite by learning from the way our application is used in production. This is precisely the Usage-Centric Testing goal.

Usage-centric Testing: a shift-right approach

Learning from our users' behavior to continuously optimize our E2E tests

The main idea of Usage-centric Testing (UcT) is to leverage users' behavioral data to continuously optimize our E2E test suite.

As we saw previously, designing a suitable E2E test suite is more complicated than it looks. Collecting anonymized user actions supports the design of E2E tests according in order to:

- **Measure and improve E2E test coverage.** User flow analysis increases the fidelity of E2E tests but also enables design according to the representativeness of test action flows. Application usage analysis enables the detection of frequent usage patterns, or critical from a business point of view, and particular cases that deserve to be tested. The collection of user actions allows a new indicator: the measurement of usage coverage by E2E tests.
- **Clean up our E2E test suite.** Learning how features are used is also valuable. It is the perfect opportunity to eliminate useless tests and reduce our E2E test suite maintenance effort.
- **Prioritize test environments.** Demographic data can also be leveraged. Devices we use to interact with software multiplied up, making testing even harsher. There are a lot of cross-browser test automation tools on the market, making cross-device testing easier. By getting insights into our users' most common environments, we can effectively prioritize the test runs for these environments and shorten our feedback loop.

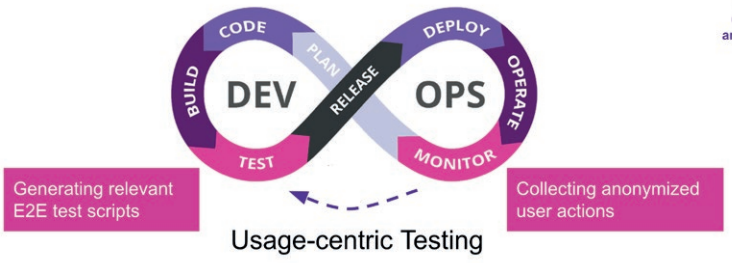
Collecting and analyzing anonymized user actions thus allows better design decisions for E2E tests and their optimization over time. Now, let's see how we can concretely do this.

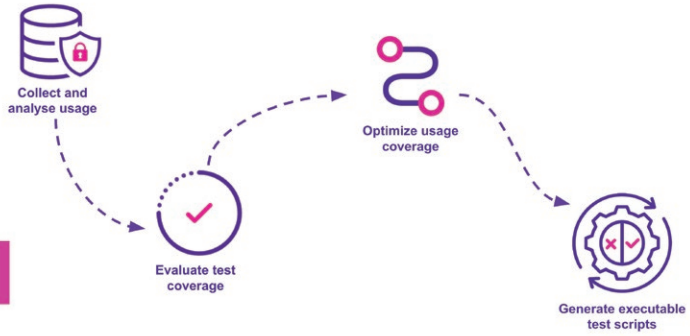
A usage-centric test automation process

Usage-centric test automation process is composed of four main activities:

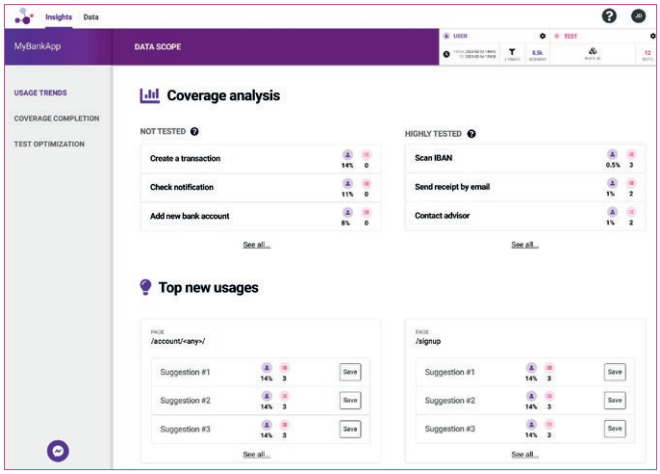
- Collect anonymous data about user actions on the application and figure out its actual usage;
- Evaluate the coverage of these usages by the current E2E tests using their execution traces;
- Determine which usages to cover and which usage sessions are relevant to increase the coverage. Identify E2E test cases with less value;
- Generate test scripts for your favorite test automation framework.

These activities must be supported by tools to be carried out effectively. The first step is tracking usage data from web analytics tools (commonly used by marketing teams). But the support is very partial, as these tools don't measure the coverage of the tests with respect to usage.



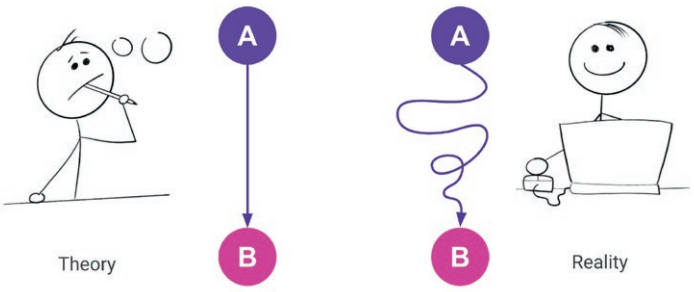


A new segment of test automation tools has thus emerged to support the UcT approach. Appvance, Datadog RUM, Gravity, and Prodperfect, for example, support a UcT-like approach. These tools respect the GPDR rules by collecting anonymized user behavior data and by generating test automation code to cover the usage. They also use Data Mining and Machine Learning techniques to provide services such as suggesting tests to be performed and analyzing test coverage on a given scope of use. The following figure shows a screenshot of the analysis of usage coverage by tests in Gravity.



Common criticisms of the UcT approach

Let's have a look at some criticism of this new approach: "by collaborating with business representatives (PO / BA), I can define accurate E2E test cases, no need to analyze the usage in operation". Yes, E2E tests benefit strongly from being discussed with business representatives. But in practice, we only really know what our users do by analyzing monitored data (see Figure). UcT suggests discussing E2E test scenarios with business representatives based on monitored usage data to enable data-informed test automation choices.



Another criticism comes from the fact that the UcT approach requires usage data of the tested features. Which means that these features are already released. This is true. But let's remember that our automated tests target regression tests, i.e. the verification of the impact of changes on existing features. The UcT concept is therefore to continuously adapt the coverage of our E2E regression tests based on usage analysis.

Also, a constraint and a benefit come by design from the UcT approach. It is necessary to be able to monitor the actual usage (this is the constraint). And by this observation, we can ensure a high fidelity to the actual usage by our E2E tests (this is the benefit).

Conclusion

We build software for humans. It is now common to adopt a user-centered design approach to create useful, desirable, and delightful services. As we saw, such an approach can also be beneficial for the efficiency of our delivery process and prevent regressions in our customers' journeys. In this article, we presented Usage-Centric Testing as an answer to the tricky questions: "What should my E2E test suite cover?" and "How to decide which E2E test scenarios to automate?".

Tools supporting the UcT approach will continue to evolve. They will take advantage of collecting user actions to improve their ability to optimize and maintain E2E tests by leveraging Data Mining and Machine Learning techniques.

Bruno Legeard

is a Professor of Software Engineering at the University of Franche-Comté (France), Secretary of the ISTQB®-French Testing Board, and Scientific Advisor at Smartesting. Bruno has more than 20 years of expertise in Test Automation and Model-Based Testing. His research activities are currently focused on AI-powered test automation. He is the author of three books on test automation and Model-Based Testing in the industry. He is also the working-group leader for ISTQB® MBT and ISTQB® Acceptance Testing syllabus

Joan Racenet

is a software product manager working on a "Usage-Driven testing" platform, Gravity, developed by Smartesting. He started his career as a software developer and co-founded HipTest in 2015, later acquired by Smartbear in 2018. He worked on multiple software quality assurance tools, like Hiptest, Cucumber, and Zephyr.

BEYOND THE BUZZWORDS:

Test automation is becoming an increasingly important part of the software development process, as it allows teams to validate the functionality and performance of their applications quickly and efficiently. By automating the testing process, teams can save time and resources, reduce the risk of human error, and improve the overall quality of their products. Another advantage of test automation is the ability to run tests in parallel. This can significantly reduce the time it takes to complete a test suite, as tests can be run simultaneously on multiple devices or in various environments.

There are many different tools and approaches to test automation, and choosing the right one can be challenging. Some popular options include open-source tools like Selenium, Appium, Cypress, and others for performance, such as LoadRunner, JMeter, etc.

One thing to keep in mind is that test automation is not a substitute for manual testing. While automated tests can be a valuable addition to your testing process, they should be used in conjunction with manual testing to ensure that all aspects of the application are thoroughly tested so don´t worry; manual testing won´t die anytime soon.

There are some levels and types of tests that are more likely to be automated and others whose cost and time to automate may outweigh the benefits of doing so. Test levels that are most automatable include unit tests, integration tests, regression testing, and performance tests, just to name a few. Unit testing involves testing individual units or components of a software application to ensure they function correctly. These types of tests can be automated using tools like JUnit or TestNG. Unit testing is typically performed by the developers who are responsible for writing the code for a software application.

Integration testing involves testing the integration between different components or systems to ensure they work together correctly. This type of testing can be automated using tools such as Selenium or Jenkins. Integration testing is usually performed by a team of testers or quality assurance (QA) engineers.

Regression testing is the process of re-running previously executed test cases to ensure that changes to the software have not introduced new defects. Automation is often used for regression testing.

> “ONE THING TO KEEP IN MIND IS THAT TEST AUTOMATION IS NOT A SUBSTITUTE FOR MANUAL TESTING.

TEST AUTOMATION

Performance testing also is often used with automation to ensure that the testing process is efficient and can be repeated accurately and handle heavy requirements for requirements.

People often are afraid of automation, but just like any other thing, automation is a tool that helps to test, so it is better to look at it as an ally, and most of the time, it is only the idea that scares people. Once you start and jump into the pool things get more comfortable, it is good to have a good understanding of test automation and how it works even if you are not the one doing it yourself, but having context and background can be very useful in your career.

I recently got the ISTQB® (International Software Testing Qualifications Board) Test Automation Engineer certification, and It has a lot of great information about the strategies for test automation, what it is and the general components of it, which are great for building your foundations on the subject and be more comfortable with test automation, another thing that it is recommended is to get the fundamentals of programming, since they both go hand to hand, remember you need good foundations to build things correctly.

However, test automation is not without its challenges. Setting up and maintaining a test automation suite can be time-consuming and require a significant resource investment. It can also be challenging to keep automated tests up to date as the application changes, which can lead to false positives or false negatives if the tests are not adequately maintained.

Another thing to consider is that test automation is not a magic solution for all testing needs. It's important to carefully consider which tests should be automated and which should be left to manual testing. An example could be tests that involve a lot of visual inspection or require complex user input, which may not be best suitable for automation. Another example could be exploratory testing, which is a type of testing that involves discovering defects through open-ended exploration of the software.

In conclusion, test automation is a valuable tool for any software development team. Still, it is important to carefully consider the needs of your application and choose the right tool and approach for your team. It's not a one-size-fits-all solution. By properly implementing test automation, teams can save time, reduce risk, and improve the overall quality of their products.

Luis Francisco Contreras Gonzalez, Software Test Lead

Highly motivated, results-oriented Test Lead and consultant who strives for excellence in delivering business solutions with many years of experience working in the field of Information Systems and clients support for different companies in their technology departments with strong analytical and problem-solving skills.

Enthusiastic professional as a software testing specialist and consultant qualified for every job. Also, personal qualities such as commitment, honesty, persistence, reliability, fairness, working under pressure, creativity, team worker, adaptability, self-motivated, quick learning, and I am 5X ISTQB®/ ISQI certified.

Martin Contreras Romo, Senior Software Engineer QA & Software Testing

Electronics engineer specialized in programming with more than 30 years of experience in systems design and development for private companies and government agencies, professor of programming and robotics, ISTQB® Certified Tester, and Scrum certified.

HAVE YOU

THE BIGGEST CLUB ON EARTH IN THE FIELD OF SOFTWARE QUALITY

In skillsclub you can interact with the largest global network of certified experts and join high-class events, benefit from relevant content and exclusive workshops, and much more

THE BENEFITS OF SKILLSCLUB

Inspire, support, and help you develop your technical skills and theoretical knowledge and excel in your career! This club is a space where you can share experiences, access exciting and relevant content, and collaborate with and meet like-minded people to grow personally and professionally in your field.

Relevant Content
Receive access to all of our online conferences and enjoy exclusive access to recordings and presenations

Break Language Barriers
Automatically detect and translate captions into your language.

Network with Specialists
Engage within the biggest club on Earth in the field of Software Quality

Free membership
Receive a free membership to SkillsClub if you are an iSQI, GASQ or A4Q certificate holder. Other certificates may be recognised by request

Add Free Content
Have the ability to focus on content without the frustration of paid advertising

No more fraud
Receive your forgery-proof digital badge with blockchain technology and share your credentials in a very secure way.

Build Connections
Interact with other members about current topics in your field

Moderation
Feel safe knowing that our Moderator keep a close eye to remove content that goes against our club rules.

skillsclub

SKILLSCLUB FEATURES

WHAT MAKES SKILLSCLUB DIFFERENT FROM OTHER CLUBS?

Build connections in your field. Through activities, events, and discussions, you can build connections in a club filled with like-minded, certified, and positive people. Share your challenges & ask for help. Answer questions & support others. Share helpful resources.

Message your Mentors
Have your helping hand at your fingertips.

Video Call
Communicate face with members across the globe

Get answers to your questions
Help others solve their challenges

Share Resources & Tips
Have access to other members' findings.

Make Appointments
Schedule appointments and calls with other members

Data Security
We will keep all your personal information confidential.

OUR AFFILIATED PARTNERS

GASQ iSQI SUCCESS IS YOURS A4Q Alliance for Qualification

SEEN ...

HIROSHI YOKKAICHI

TEST AUTOMATION

JAPANESE SOFTWARE QUALITY MARKET

Test automation is a crucial aspect of software development and quality assurance in the Japanese market. With the increasing demand for mobile applications and Microservices, the need for efficient and effective testing is greater than ever. However, the Japanese software market faces unique challenges in implementing test automation, such as a lack of skilled human resources, complex regulations, and high communication costs with low-skilled engineers. Furthermore, Japanese IT market technology tends to lag behind Europe and the U.S. by more than 5 years and most test automation tools are occupied by no-code or low-code tools, which have limitations in maintainability and scalability, leading to repeated failures in test automation implementation. Additionally, the mainstream of quality assurance in Japan is outsourcing manual testing, which leads to hyperinflation of communication costs and human errors. Despite these challenges, test automation is becoming increasingly important in the Japanese software market as a means to improve software quality and speed up delivery times especially with the introduction of digital transformation and Agile development. Companies in Japan are looking for ways to overcome these challenges and implement test automation to achieve a more efficient and effective testing process, and to meet the demands of the market for fast and high-quality software delivery.

Current Status and Issues of Test Automation in Japan

a. Excessive Expectations of Test Automation and Failure to Implement

Test automation is often seen as a panacea for improving software quality and reducing costs, but in practice, many organizations in Japan have struggled to achieve these benefits. One reason for this is that the expectations for test automation are often too high, and organizations do not fully understand the effort and resources required to implement and maintain test automation. Additionally, many organizations in Japan have failed to implement test automation due to a lack of knowledge and skills in test automation, as well as a lack of clear goals and strategies for test automation.

b. IT industry lags behind Europe and the U.S. by more than 5 years

The Japanese IT industry has traditionally lagged behind Europe and the U.S. by more than 5 years, and this gap is particularly pronounced in the area of test automation. Many test automation frameworks that are popular in Europe and the U.S. have not yet been widely adopted in Japan, and many Japanese organizations are still using manual testing methods. Furthermore, the language barrier has made it difficult for Japanese organizations to keep up with the latest developments in test automation, and as a result, many organizations in Japan are still relying on no-code or low-code tools that do not provide the same level of functionality as more advanced tools.

c. Lack of resources, especially test automation engineers

According to the IT Human Resources Supply and Demand Survey Report (2019), the supply of IT human resources is expected to increase until 2030, but demand is also growing and supply will not be able to keep up with demand until 2030. supply will continue to grow until 2030. A study by the Ministry of Economy, Trade and Industry (March 2019) estimates that by 2030 there will be a shortage of up to approximately 790,000 IT personnel. This shortage of IT personnel, particularly in the area of test automation, has made it difficult for Japanese organizations to implement test automation successfully.

https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf

d. Outsourcing manual testing being mainstream of quality assurance

Outsourcing manual testing to reduce costs has been a common practice in Japan, but this has led to hyperinflation of communication costs and a high rate of human errors. In addition, outsourcing manual testing does not address the underlying problems of software quality, and as a result, many organizations in Japan are still struggling with poor software quality despite outsourcing manual testing. All these challenges make it difficult for companies in Japan to effectively implement test automation, but with the right strategy, resources and tools, they can overcome these challenges and achieve success in test automation.

How to successfully implement test automation in Japan

In order to achieve success in implementing test automation in Japan, several steps can be taken. Firstly, it is important to have a clear understanding of the specific needs and goals of the organization in terms of test automation. This includes identifying which areas of the software development process will benefit most from automation and determining the resources that will be required to implement and maintain the automation.

Another important step in achieving success in test automation in Japan is to ensure that the right tools and technologies are being used. This includes researching the latest test automation tools and technologies, as well as considering the specific requirements of the organization, such as support for multiple languages and platforms. It is also important to ensure that the tools and technologies being used are highly maintainable, as this will helpto reduce the risk of repeated automation failures.

Having a skilled and experienced test automation team is also crucial for success. This includes not only having a team of test automation engineers, but also ensuring that the team members have the necessary skills and knowledge to effectively implement and maintain the test automation. This can be achieved through regular training and professional development opportunities, as well as by recruiting and hiring top talent in the field.

Finally, effective communication and collaboration between the test automation team and other stakeholders in the organization is essential for success. This includes ensuring that there is clear communication and coordination between the test automation team and the development team, as well as involving key stakeholders in the process of implementing and maintaining test automation. This can help to ensure that the test automation is aligned with the overall goals and objectives of the organization and that any issue or challenge that arises can be addressed in a timely and effective manner.

What I have written here are the obvious and necessary steps to take when it comes to test automation implementation. But it is not yet common place in Japan.

Future outlook of test automation in the Japanese SOFTWARE QUALITY MARKET

In the future, the Japanese software market is expected to see a continued trend towards high-speed delivery and agile development methodologies. This will lead to an increasing demand for test automation, particularly for mobile applications and microservices. To meet this demand, companies will need to focus on building a solid test automation strategy. This includes not only selecting the right tools and technologies, but also investing in the necessary resources and expertise to ensure successful implementation.

One key driver of this trend is the increasing adoption of DevOps and continuous integration/continuous delivery (CI/CD) practices in Japan. As more companies move towards these methodologies, the need for automated testing will become even more critical. This is because DevOps and CI/CD rely heavily on automation to ensure that code is tested, integrated, and deployed quickly and efficiently. Without test automation, it would be difficult for companies to keep up with the fast pace of development and delivery.

Another important factor driving the future outlook of test automation in Japan is the growing popularity of mobile applications and microservices. These types of applications are becoming increasingly prevalent in the market, and they require a different approach to testing than traditional desktop or web applications. Mobile applications, in particular, require more extensive testing to ensure that they work correctly on a wide range of devices and platforms. Microservices, on the other hand, require more extensive testing to ensure that they work correctly when integrated into a larger system.

Overall, the future outlook for test automation in the Japanese software market is positive. Companies that invest in building a solid test automation strategy and keep up with the latest trends and technologies will be well-positioned to succeed in this competitive market.

> IN THE FUTURE, THE JAPANESE SOFTWARE MARKET IS EXPECTED TO SEE A CONTINUED TREND TOWARDS HIGH-SPEED DELIVERY AND AGILE DEVELOPMENT METHODOLOGIES.

Hiroshi Yokkaichi, SDET (Software Development Engineer in Test) Lead / Quality Engineering and Assurance *is an experienced SDET lead with over 20 years of software testing and quality engineering expertise. Hiroshi has a deep understanding of Agile methodologies, DevOps, and test automation and has led QA teams, designing and building several types of test automation architectures and optimizing the efficiency of software development processes. Hiroshi is a strong communicator who can consult with stakeholders on test strategies. His passion for evaluating and introducing the latest technologies and tools has enabled them to adapt to different business areas quickly. Hiroshi is a self-starter who can manage and complete required activities under time constraints and have experience managing multicultural teams and implementing test automation.*

GOING WITH THE TRENDS

Let's see what technology thinks

WHY TEST AUTOMATION MATTERS:

advantages and future developments

As a former QA professional and current CTO, I've seen firsthand how test automation has revolutionized the software development industry. In today's fast-paced tech landscape, the ability to test software quickly, accurately, and reliably is essential to delivering high-quality products on time and on budget.

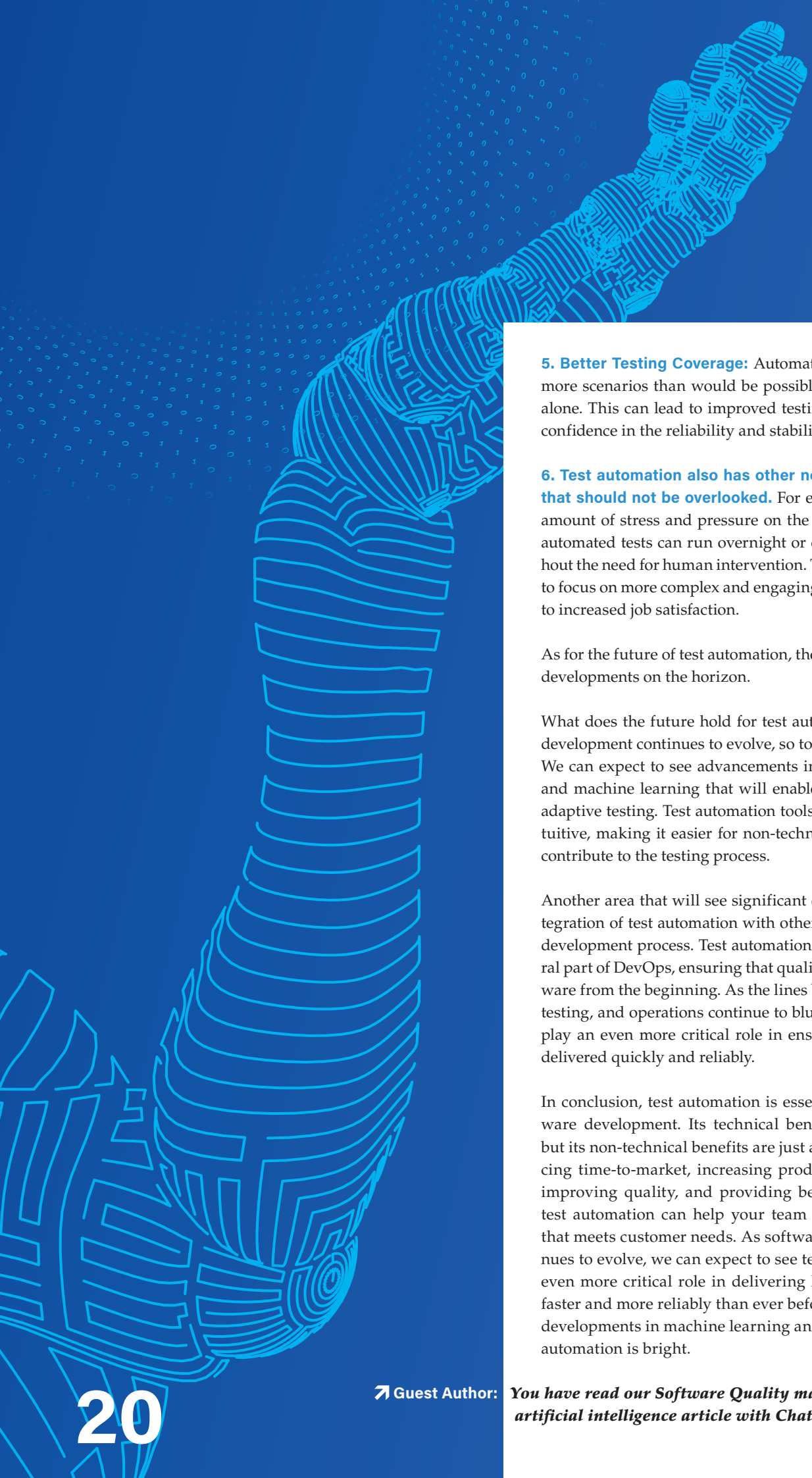
But the benefits of test automation go beyond just technical efficiency. Here are some non-technical benefits that make test automation a critical component of modern software development:

1. Faster Time to Market: By automating testing, you can reduce the time it takes to test software, allowing your team to move faster and get products to market more quickly. This can be a game-changer in competitive industries where time-to-market can mean the difference between success and failure.

2. Increased Productivity: Test automation frees up developers and testers from the tedious and time-consuming task of manual testing, allowing them to focus on more creative and value-added activities. This can lead to increased productivity and job satisfaction among your team.

3. Cost Savings: Automation can help you reduce costs by minimizing the need for manual testing, which can be expensive and error-prone. Automated tests can be run repeatedly with minimal additional cost, making it a cost-effective solution in the long run.

4. Improved Quality: Automated testing can catch defects that might be missed by manual testing, leading to higher quality products. This can help you build better software that meets customer needs and avoids costly rework and customer dissatisfaction.



5. Better Testing Coverage: Automation allows you to test more scenarios than would be possible with manual testing alone. This can lead to improved testing coverage and more confidence in the reliability and stability of your software.

6. Test automation also has other non-technical benefits that should not be overlooked. For example, it reduces the amount of stress and pressure on the development team, as automated tests can run overnight or over the weekend without the need for human intervention. This allows developers to focus on more complex and engaging tasks, which can lead to increased job satisfaction.

As for the future of test automation, there are several exciting developments on the horizon.

What does the future hold for test automation? As software development continues to evolve, so too will test automation. We can expect to see advancements in artificial intelligence and machine learning that will enable more intelligent and adaptive testing. Test automation tools will become more intuitive, making it easier for non-technical team members to contribute to the testing process.

Another area that will see significant development is the integration of test automation with other parts of the software development process. Test automation will become an integral part of DevOps, ensuring that quality is built into the software from the beginning. As the lines between development, testing, and operations continue to blur, test automation will play an even more critical role in ensuring that software is delivered quickly and reliably.

In conclusion, test automation is essential for modern software development. Its technical benefits are well-known, but its non-technical benefits are just as important. By reducing time-to-market, increasing productivity, saving costs, improving quality, and providing better testing coverage, test automation can help your team build better software that meets customer needs. As software development continues to evolve, we can expect to see test automation play an even more critical role in delivering high-quality software faster and more reliably than ever before. And with exciting developments in machine learning and AI, the future of test automation is bright.

Guest Author: ***You have read our Software Quality magazine's first artificial intelligence article with Chat GPT.***

RESCUING TEST AUTOMATION FROM ITS DOWNFALL

Engineers are sitting around a cramped table in the middle of a small meeting room. The larger rooms were already fully booked. Cables from the screen on the wall hang loose, and the whiteboard contains some diagrams from a meeting a long time ago. Everyone sits in silence, waiting for someone to break the ice.

One of them is Henry, a tester who once loved his job but who is now struggling. It all started when management decided there must be more test automation. Now a year later he is sitting here, at this table. It has been a strange rollercoaster. In the first instance, the tool selected needed to be one that the testers could use. Henry had put some of the existing test scripts into the new tools, and to everyone's delight, it was a great success! The automated tests ran smoothly, and targets were quickly met. In the months following, more and more test scripts were automated. Targets continued to be achieved, and everyone was celebrating. However, everything took a sudden turn for the worse. Now, they found themselves in this situation, unsure of when or how things had gone wrong. It was challenging to determine the exact cause.

The ISTQB® Test Automation Engineer course starts with a simple question; "What are the activities we think of when we talk about Test Automation?" Participants often initially associate it with the automated execution of tests. However, they soon realize that Test Automation involves generating test ideas, defining tests, overcoming technical obstacles, working closely with software engineers to enable testability, and much more. Of course, the execution of tests is automated, but it's not the main focus of successful Test Automation. Over the past year, Henry's role has changed dramatically. He used to frequently interact with the system, discovering a variety of issues such as inconsistencies, bugs, loads of "smoke", configuration errors, simple mistakes with unexpected consequences, and more. But now, his focus is on automation and maintaining the automated test cases.

Keeping up with the fast pace of development was challenging, and the new tests were quickly limited to checking acceptance criteria within the automation toolset. The insights he used to gain were now based on a series of green checkmarks or red crosses. However, most of the time, a red cross indicated an issue with the automated test, not with the functionality it was supposed to check. The goal of any test should be to gain more insight into the system's status. A tester analyses, selects, focuses, gains, collects, and reports insights to decision-makers to assess quality. Automation can help gain those insights more efficiently, frequently, and quickly.

Unfortunately, for Henry, the situation continued to deteriorate. The gap between Henry and the developers became increasingly pronounced. They were less and less willing to provide support, and many of the insights Henry reported were found to be errors in the tests or test execution. However, in order to identify these mistakes, Henry needed the technical expertise of the developers. When organizations aim for more automated tests, testers are often the first to be considered, but the mistake of equating automated tests with testers' work should be avoided. Moving towards increased test automation requires a major shift in interactions, tooling, software, and system architecture, way of working, policies, psychological contracts, and more. To effectively handle this complex change, organizations take a structured approach, treating it as a complex change program. This approach should consider factors such as timing, people, processes, technology, capability preservation, learning, development, and scope.

Henry's craftsmanship has been successfully diminished by the organization. Others have been affected also. The pressure on the developers to deliver new functionality has been increasingly high. The urge to make quick and "dirty" solutions increases. Software craftsmanship, making clean code, has been neglected over time making it harder to add, test, remove, or change functionality. The delays caused by this created more pressure on the delivery of new functionality, a vicious circle.

These vicious circles can persist for some time, and the tool is often the ultimate sacrifice. The tool is blamed for poor automation quality, fragility, and excessive maintenance efforts. Management may not have the understanding to recognize this and will often agree to replace the tool. In our advisory work, the question that we most often get asked if whether we can help with this process, with selecting a new tool or with tool training to make automation go smoothly. And whilst there seems to be an unwritten bias or rule that every new consultant introduces a new tool, the focus always shifts quickly to the real issues.

A well-written automated test is like a paragraph in a book that tells the story of how, what, and why it was tested. The book should be consistent, with chapter titles summarizing the contents. It does not necessarily have to be readable for everyone. But every line in the book should add value to the story. In the case of test automation, the book is never finished; chapters may become irrelevant or need to be rewritten, but all chapters should contribute to the book's goal; give a valuable insight into the status of the system so that a tester can use this to compel the full story.

A couple of weeks ago, Henry followed the ISTQB® Certified Tester - Test Automation Engineer training course. He learned that to create such a book and that he needs to focus on the value of the automated tests, the way they are automated or adapted to the system, and the way they are generated, defined and executed. This is all reflected in the generic Test Automation Architecture that is a major part of the ISTQB® syllabus. This architecture offers a clear distinction between the generation of test ideas, the definition of test cases, the execution of tests, and adaptation towards the system under test. Many participants come to the course thinking they will learn about test execution and tools, they return to their teams with insights about their craftsmanship. This is not to create many test cases or just to automate them; it is to make sure the team writes a book that adds value in the most efficient way possible. Even if that means rewriting a paragraph or chapter over and over, because the context changes. It is about the quality of tests over quantity.

Henry looks up and glances around the room and opens the dialog with one simple question: "What's the insight we want our automated tests to give us?"

Pieter Withaar, Managing Director Improve Quality Services
is a highly experienced and accomplished test automation enthusiast, consultant, and trainer with a proven track record spanning over 15 years. He is passionate about helping individuals, teams, and organizations achieve their full potential, and his strong technical drive and strategic focus enable him to do just that. Pieter's background in computer science and his current focus on executive MBA studies give him a unique perspective, allowing him to approach challenges with a comprehensive understanding of both the technical and strategic implications.

SQ mag #14

COMING OCT 2023

PREVIEW

What´s in the next issue of SQ mag?

Please share your experiences with us and send your article to **contact@sq-mag.com**

PUBLISHER

A4Q Global Services GmbH
Bauerngasse 30
90443 Nümberg
info@a4q.gs

EDITOR

Editor: Stephan Goericke
Editor-in-Chief: Julia Jasper
Editorial Team:
Atefe Abouata Amlashi,
Debbie Archer,
Liesl Hartje

contact@sq-mag.com
Friedrich-Engels-Str. 24
14473 Potsdam
(Germany)

PHOTO CREDITS

p. 4: @ Omar Prestwich (unsplash)
p. 7: @ thisisengineering raeng (unsplash)
p. 11 - 12: @ rawpixel.com (freepik)
p. 13 - 14: @ Victor Rodriguez (unsplash)
p. 16 - 17: @ Note Thanun (unsplash)
p. 19 - 20: @ iuriimotov (freepik)
p. 21 - 22: @ memento media (unsplash)

ADVERTISEMENTS

Detailed metrics for the online advertising formats are available. Please send your request to:
contact@sq-mag.com

METRICS

21 cm × 29,7 cm
(DIN A4, 4-color (euro scale))
Live area: 170 mm (B) × 242 mm (H)
E-Paper and PDF-Download on
www.sq-mag.com

LAYOUT

Format Druck und Medienservice GmbH
www.format.berlin

A4Q GLOBAL SERVICE www.a4q.gs

You can compare A4Q with an airport of ideas. It is the best place from which new ideas can start – independently from which organization they belong to.

A4Q Alliance for Qualification

allianceforqualification.com