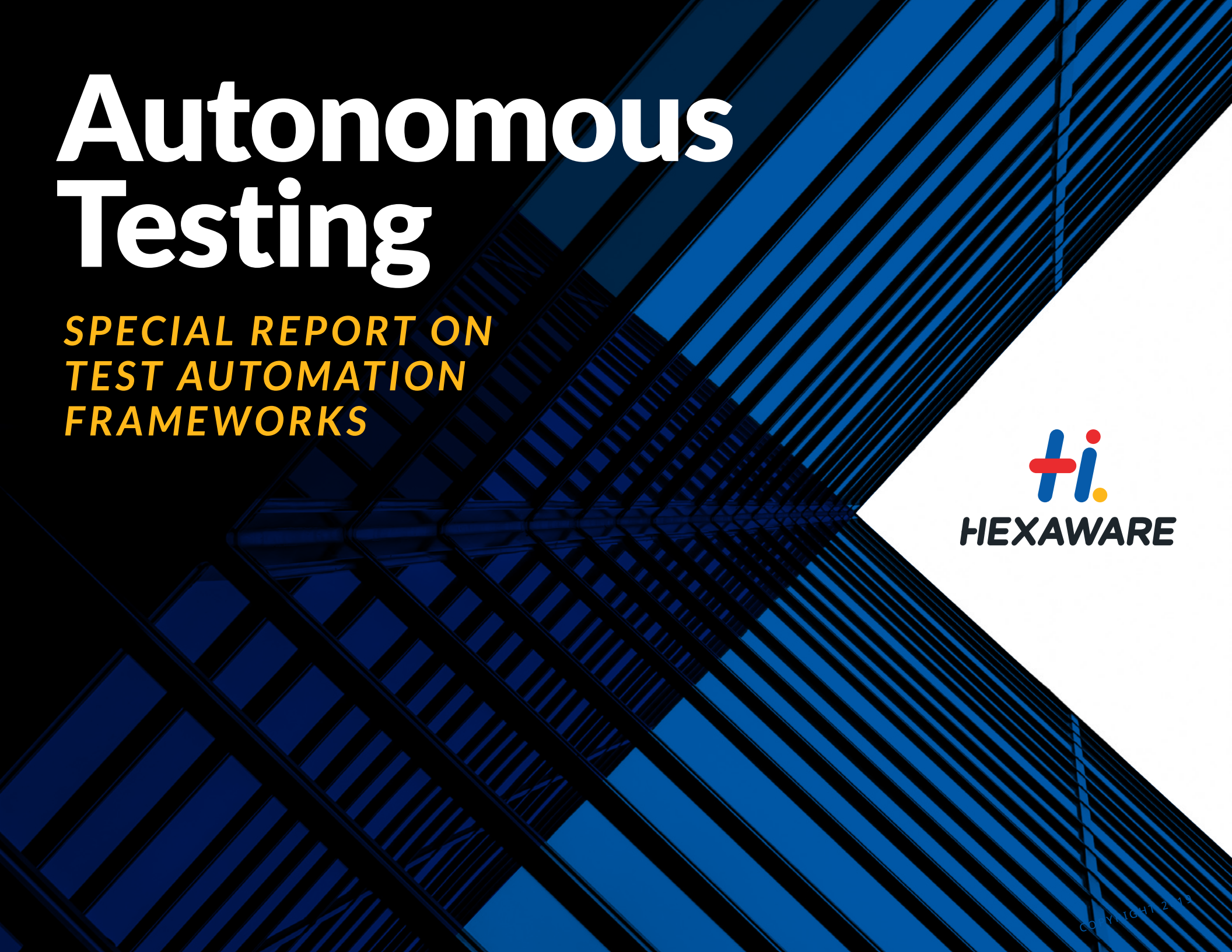
Autonomous Testing

SPECIAL REPORT ON TEST AUTOMATION FRAMEWORKS



HEXAWARE

HEXAWARE

The software industry's current shift toward continuous development and deployment has left testers struggling to keep pace with delivery while maintaining quality. The challenge is even greater in organizations with applications being developed and deployed across multiple platforms.

The need for autonomous testing has never been greater, but many enterprises are unsure how to begin implementing test automation frameworks. This special report focuses on the basics of test automation, how to integrate it into your existing test environment, and how AI and machine learning are driving the future of software testing.

In this Test Automation Framework Special Report

Smart Testers Adopt Smart Automation

As technology continues to evolve, questions around the role of quality also continue. Is manual testing still required? What should the role of automation be? Where are we heading with quality? Smart testers hoping to develop their careers will have to brush up on their exposure and expertise and embrace automation.

5 Ways to Write Automated Tests Efficiently

When written clearly and concisely, automated tests can perform quality assessment that would take developers days or weeks to do manually. To maximize their effectiveness, you should compose these tests the best possible way. Here are five guidelines to write automated tests efficiently and increase their usefulness.

7 Essential Quality Attributes for Your Test Automation Framework

A common problem in software is that developers and designers tend to concentrate on pure functionality and neglect quality attributes. These are the famous "-abilities": usability, reliability, portability, etc. If your testing framework is suffering, you might want to check if it has these seven quality attributes.

4 Advantages of Applying AI in Software Testing

We're always looking for smarter, faster, better ways of testing. As the popularity of artificial intelligence grows, more and more testers are realizing its capacity to make cumbersome and time-consuming tasks simpler. AI is coming, so we should take advantage of it. Here are four benefits to applying AI in testing.

AI-Driven Test Automation and Your Future

Many software testers are lamenting the impending demise of their jobs thanks to artificial intelligence. But Jon Hagar thinks there's no need to panic just yet. Here, he details some capabilities he's seen in AI, relates how these can be used in software testing, and explains why he thinks most people don't have to worry—although he also explains who should! As usual, it comes down to a willingness to learn new things.

Autonomous Software Testing: Journey towards Autonomous Testing to Achieve Testing Singularity

With advances in leveraging machine learning and deep learning in software development, software built with this technology is quickly revolutionizing every aspect of work, life, and business around the world. Learn more about how autonomous software testing takes advantage of AI and machine learning to make testing more independent from human intervention by aggregating the data that it collects from the activities it performs.

Additional Test Automation Framework Resources

HEXAWARE

Smart Testers Adopt Smart Automation

By Rajini Padmanaban

As technology evolves, so do product ideas, development capabilities, and quality strategies. Users are passionate to embrace new concepts, even in risky zones, and companies and stakeholders are willing to bet on bold ideas, even if the chances of failure are high.

While all of these evolutions are taking center stage, discussions around the role of quality continue to linger. Is manual testing still required? What should the role of automation be? With the kind of technology and product evolutions we are witnessing, where are we heading with quality?

When it comes to testing, automation is undoubtedly key. But it cannot be just automation—it has to be smart automation that leverages technologies such as artificial intelligence and machine learning. Even cognitive-rich areas such as accessibility testing are benefitting from automation solutions that leverage image recognition, language processing, and machine learning APIs.

While smart automation evolves, device testing is also gaining prominence. Smart in-house testing when field testing is not feasible is also becoming a more common option, now that cloud offerings are making applications more testable.

And as owning quality becomes everyone's responsibility, testing solutions have become more widespread but dedicated testers are becoming rarer. Organizations do not want a large bandwagon of testers. Gone are the days when developer-to-tester ratios were three to one. Today, teams that have a ratio of even nine to one are not unheard of. While there is no single answer for determining the right ratios, the shrinking numbers showcase a collective ownership of quality.

A tester in such a model is not just a quality subject matter expert. They are a strategist, a technologist, a team player, a business liaison, an end-user advocate, a communicator, and a collaborator. Unless all these attributes come together, it is not going to be possible to achieve the velocities needed within the timelines of a tight sprint cycle.

With these high expectations from the quality discipline, smart testers hoping to develop their careers will have to brush up on their exposure and expertise. A high level of finesse is now being expected, even if the number of testers are fewer.

Gone are the days when developer-to-tester ratios were three to one.

Of course, some areas of software products will continue to be tested manually due to varied reasons, including automation limitations, the need for cognitive manual skills, exploratory test efforts, product scope and business priorities, or constraints around cost and time. But that just means that in-demand testers will retain these manual testing skills while developing skills in automation as well.

While testing activities at a manual level will not become extinct, if you want to become indispensable, move up the quality value chain by embracing smart automation.

5 Ways to Write Automated Tests Efficiently

By Angela Baker

No matter how experienced your developers are with testing, chances are that quality assurance isn't their first priority. This makes using automated testing algorithms highly beneficial for your dev team in many ways:

- Earlier bug detection
- High return on resource investment
- Continuous testing with concrete and objective data
- Smarter resource allocation
- Reusability of new versions, hotfixes, and applications

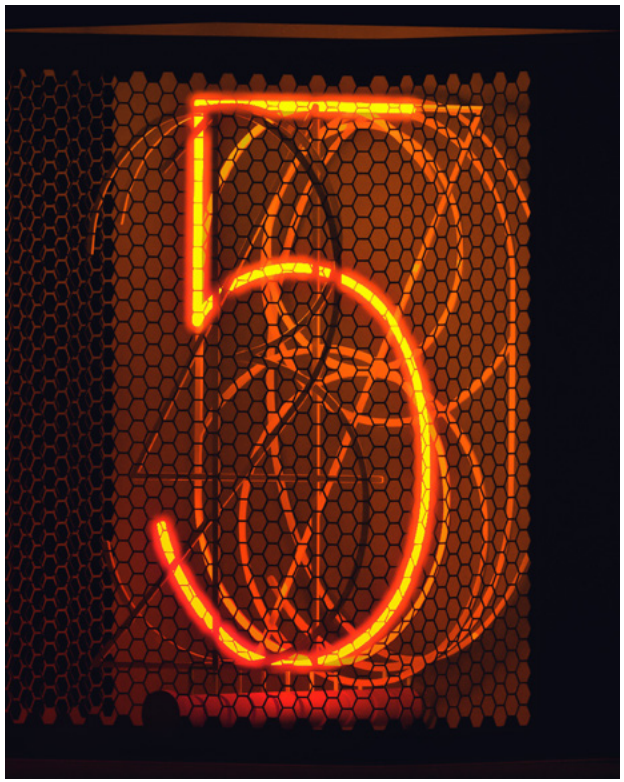
To maximize the effectiveness of automated tests, you should compose them the best possible way. Here are five guidelines that will help you write automated tests efficiently in order to grow your team's output and overall productivity.

1. Focus on Single-Result Tests

Every time you approach creating a new automated test, it's important to have a clear goal in mind. Ask yourself and your team what the point of that particular test is. Focusing on single-result tests will yield far more precise and objective results for you to use in your further development, leaving very little to chance or misunderstanding.

2. Set Test Priorities

Automated tests will always feature numerous results and parameters for you to analyze after the test is done. In order to create an understandable data hierarchy, you should pay close attention to the order in which you write your test calls. Each call should be succeeded by a new call that builds on the previous one in context to make it easier to analyze.



Every time you approach creating a new automated test, it's important to have a clear goal in mind.

HEXAWARE

Look for effective ways to integrate automated tests into your development project, and you will undoubtedly find useful applications.

3. Start Simple

Even though you may have clear goals for your automated tests early on, it's important to naturally build on simple calls and go forward from there. Start with baseline calls and analysis of your development project before slowly turning the test in the direction of your target parameters. Don't build automated tests with overly complex calls without baseline analysis as its precursor, and you will have access to much more accurate and usable data.

4. Recycle Parameters When Possible

Your automated tests and their algorithms should be as simple as possible once they're written. You can recycle parameters and calls from previous lines in order to build a cohesive test for your development project. This will keep the automated test lightweight, make it faster to execute, and allow you to easily connect related parameters in the analysis stage.

5. Rethink Before Launch

Lastly, it's essential to test your automated test algorithms before the official testing period arrives. Make sure that there are no bugs in your code and that the automated test will run as intended. Eliminate potential bottlenecks and slowdowns in the automated test during its development in order to make it accessible and usable to everyone on your development team and related company departments.

When written clearly and concisely, automated tests are highly efficient in performing quality assessment that would take developers days or weeks to execute manually. Look for effective ways to integrate automated tests into your development project, and you will undoubtedly find useful applications.

7 Essential Quality Attributes for Your Test Automation Framework

By Iryna Suprun

A common problem in software is that developers and designers tend to concentrate on pure functionality and neglect quality attributes. These attributes are the famous "-abilities": usability, reliability, portability, and so on.

This statement is valid for test automation frameworks as well. But there is an additional issue here: People often forget that a test automation framework is still software. Moreover, it is very specific software that can be compared with medical diagnosis software, because test automation frameworks are used to diagnose the health status of the software under test.

People often forget that a test automation framework is still software.

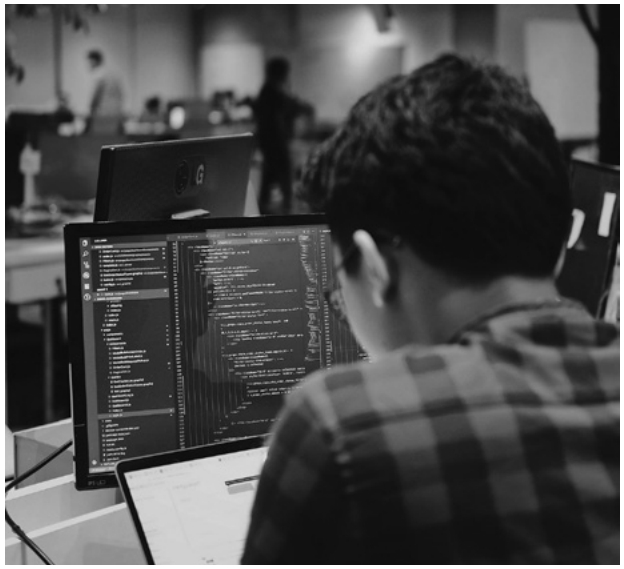
Considering this perspective, then stability, reliability, user-friendliness, communication, portability, and other quality attributes are crucial for test automation frameworks. Without them, it's dead, like a body without a soul.

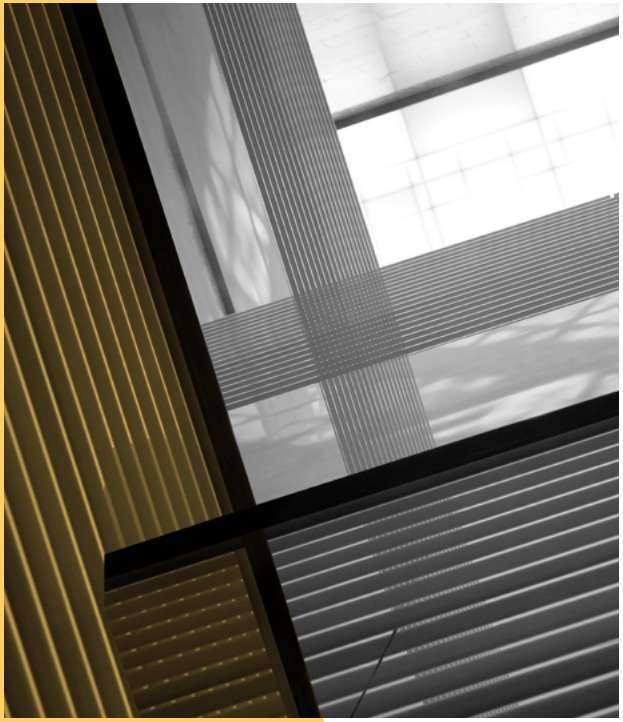
There are some of the signs that something is wrong with your test automation framework and it needs some CPR:

- Low ROI
- People in the company refuse to use it
- The number of automated tests remains low
- Quality is not improved
- There is no trust in test results

If you find that your testing framework is suffering, you might want to check whether the framework has these seven quality attributes:

1. Reliability: The test framework should be more reliable than the software under test. It should be tested with the aim of preventing it from crashing or throwing unhandled exceptions. Think about network delays, environments that are not available, and lost database connections, and ensure that the framework handles these cases.





2. Usability: To be widely adopted, the framework should be easy to use, starting from installation and ending with user-friendly test reports.

3. Communication: All the changes in the framework should be clearly communicated to the teams that use it. And don't forget about the release notes.

4. Test data: The ability to create test data is functionality, but how easy and fast we can do it is a quality attribute. Provide tools for users that allow them to handle the management of test data in a simple and fast way.

5. Portability: Different members of the team might use different OSes, and all of them should be able to run tests on their machines, in the cloud, and in the build pipeline. The framework must be usable in all these environments, or else its value significantly decreases.

6. Reporting: Reports are one of the most important parts of a test framework's functionality. Test reports that are unreadable, uninformative, hard to find, or vague, that do not provide clear information about test failures, make even the best test cases useless.

7. Integration: It should be easy and fast to integrate the test automation framework with other tools. Think about build/deploy pipelines, APIs, and a convenient command-line interface.

Of course, there are many other factors that impact the quality of test automation frameworks. There is no question that your framework should have all the required functionality as well. However, remembering that the framework is still software—more specifically, a "diagnosis software"—and that it must be "alive" will help you build a robust automation solution, as well as save a lot of time and money in the end.

To be widely adopted, the framework should be easy to use, starting from installation and ending with user-friendly test reports.

HEXAWARE

4 Advantages of Applying AI in Software Testing

By Ester Brierley

Since the effectiveness and efficiency of your testing process are crucial to the success of your product, there's no surprise that we're always looking for smarter, faster, better ways of testing. As the popularity of artificial intelligence grows, more and more testers are realizing its capacity to make cumbersome and time-consuming tasks simpler.

Artificial intelligence is coming, so we should take advantage of its abilities. Here are four benefits to applying AI in software testing.

1. Unfailing Accuracy

Even the most experienced testers sometimes make mistakes, especially when dealing with repetitive tasks. This is the main reason automation became so popular. Unlike humans, AI always performs the necessary tasks exactly as intended, completing the same repetitive tasks successfully, time after time. While AI works on repetitive tasks, testers are able to focus on creating effective automation solutions and on exploratory activities that only humans can perform.

2. Improved Flexibility

Even the simplest changes in an application can lead to test failures in automation tools because traditional testing scenarios consider a singular selector or path. Therefore, such testing approaches are somewhat rigid. Machine learning and AI allow for a more flexible testing process, learning relationships between various segments of documentation. Such systems can automatically adapt to any changes in real time, being both flexible and reliable.

3. Increased Test Coverage Overall

AI allows you to increase the scope and depth of tests significantly. It can check the file contents, memory, data tables, and internal program states, being able to quickly determine whether or not the program works as intended. AI-powered test automation allows for executing over a thousand test cases in one test run, which is impossible through manual testing.

4. Visual Validation

Pattern recognition and image recognition enable AI to detect visual bugs by performing visual testing of applications and making sure that all the visual elements look and function properly. AI can recognize dynamic UI controls regardless of their size and shape, analyzing them on a pixel level.

Artificial intelligence is coming, so we should take advantage of its abilities.

Even though AI still cannot perform software testing with no help from humans, it is already capable of improving the testing process significantly. The main advantage of AI is that it takes automation to a new level so that testers don't need to deal with repetitive tasks anymore, but it also improves the flexibility and accuracy of software tests and, through pattern recognition and machine learning, allows computers to perform tasks that used to require human work. That leaves us more time to do the creative, exploratory aspects of testing.

AI-Driven Test Automation and Your Future

By Jon Hagar

"Testing is dead."

"AI bots will replace us in testing jobs."

"The end is near!"

Have you heard fellow testers say these things? I have—some for almost forty years. As Mark Twain once said, "The reports of my death are greatly exaggerated."

Even though I have seen many changes throughout my software and testing career, I do believe AI will bring change that, if you are not ready for it, may limit your future.

I have supported AI and played with it in the past, with interesting results. It has great power that must be respected and understood. The AI bot I taught learned that code metrics are not much use as an indicator of quality, but that programmers who write more comments (not necessarily better comments, just more of them) make fewer coding errors. It may sound strange, but this was backed up by other research I have read.

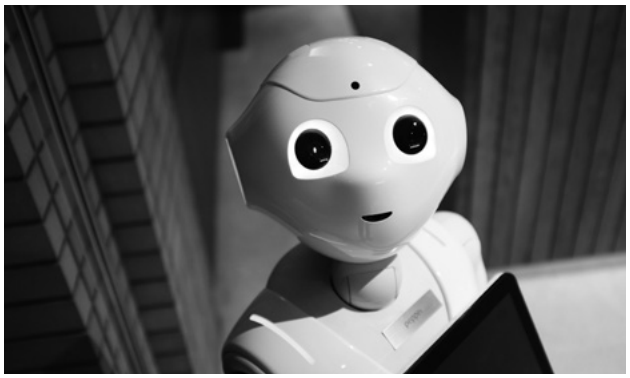
Here are some other impending changes due to AI that I think you may want to consider and work with.

Update Your Skills

First, if you are a manual tester, running tests mostly by hand and following written test procedures with little or no thought, you should be afraid. Test automation supported by AI will likely replace you. This is true of any job in any industry where there are repetitive tasks that a robot or computer can do faster than you can, and cheaper, too.

It has been happening for years, so you should take actions now to start updating your knowledge and skills.

One skill to update is creative thinking demonstrated through exploratory testing. Exploratory testing is something humans still excel at.



Next, if you are doing some kind of test automation, you need to be paying attention to AI and AI-supported test automation tools. These tools are evolving, and you need to understand this integration change. It is likely the AI bots will take over this generation, and then execution will happen using automated test tools running hundreds, thousands, and even millions of tests. If you are not doing test automation but are thinking about it, you should pay attention to AI automation trends.

Additionally, if you do not think about things like test management, planning, strategy, architecture, and

improvement, you are missing a major boat that will set sail on the AI winds without you. If these terms are unknown to you, you have another learning assignment.

The time is now to do some reading and research on these topics. Learn new things and build added value in your skills. I have spent my software testing life constantly learning and building new skills. If you are not learning new things, you will become a dinosaur quickly in the software world.

Evolve with the Technology

Although these changes driven by AI may seem intimidating, the good news is that I think we may all get to work on more fun things—and possibly also have more days off. Who would not want to work three or four days a week, monitor what the AI system is doing while we play, and get better software, all at the same time?

Software is going to be everywhere and in everything. This means we will need a lot of quality testing done by highly skilled, critical-thinking engineers.

I teach people around the world who want the easy answer and a "silver bullet" for testing. They even get mad when I tell them they must learn to think and grow. These types of people are the ones who are endangered.

Agile was going to be the silver bullet. Automation of testing was the easy answer. Neither was totally perfect, easy, or any kind of panacea. Now it is AI's turn to save the industry. Ha!

I believe we must change with technology's evolution. I am no longer using punch cards, and I now program parts of my life by talking to my phone. Times change. We must change with them.

Am I an optimist? Sure, but in the last forty years, my optimism has proven right more often than not. Who wants boring "follow-the-script" manual test jobs?

In terms of test planning, strategy, and upfront architecture, I personally gravitate toward model-based testing to drive test automation that is interfaced with AI. In these test plans, my strategy will be to create complex models in languages such as Unified Modeling Language Testing Profile (UTP) that can be expanded, reused, changed, and refactored to generate many tests within a smart, model-driven test environment.

I have tested critical software systems where the testing was driven by keywords and models that functioned in real time to allow high degrees of automation while collecting terabytes of test information. A tester would start a system test that would run thousands of automated test cases over the software under test in real time. During this time the tester could enjoy thinking about new models and strategies.

The use of planning, strategy, and model architectures will not be the only answers when dealing with the AI world, but they are the ones I am comfortable with and confident in.

Don't Fear the Future

Elon Musk says we should be worried about the AI threat. I say you can be better than the AI, but you should respect it.

Some people say that AI will take over jobs in the future. I say, cool—most of those jobs were boring anyway, and I enjoy creative, challenging tasks that AI seems to be bad at.

Many users complain about smart AI devices and their lack of safety. I say it sounds like a job for testers using AI automation to me. I wouldn't mind that three-days-a-week job hacking AI device security.

Mostly, I say that with the right mindset and the right skills, you can have fun in the robotic AI future—and maybe even work less and play more.

Autonomous Software Testing: Journey towards Autonomous Testing to Achieve Testing Singularity

By Hexaware

With advances in leveraging machine learning (ML) and deep learning (DL) in software development, humanity is reaching new heights every day. Software built with ML and DL revolutionizes every aspect of work, life, and business around the world.

This technology seems to be pulled from science fiction, like something you'd find in a Star trek episode.

Like a self-driving vehicle powered by artificial intelligence (AI), autonomous software testing takes advantage of AI and ML to make testing more independent from human intervention and self-learning by aggregating the data that it collects from the activities it performs.

This technology seems to be pulled from science fiction, like something you'd find in a Star trek episode. But the latest advancements in AI and cloud are enabling software testers to think like never before and implement use cases across:

- Autonomous requirement analysis and effort estimation
- Autonomous test design and maintenance
- Autonomous test selection
- Autonomous provisioning of test data and environment
- Autonomous test execution
- Autonomous test reporting and decision making

Hexaware is embarking on the autonomous software testing journey by developing our own autonomous test platform we call Autonomous Test Orchestration Platform (ATOP). ATOP incorporates ML, DL algorithms, and cutting-edge natural language processing (NLP) techniques to execute autonomous testing use cases.

Autonomous agents can continuously gather data from an agile management tool showing the completed features and the source code repositories.

Here's an in-depth look into how various autonomous testing use cases help QA teams ensure quality in high-speed development environments:

Autonomous Requirement Analysis and Effort Estimation

Assessment and estimation can easily be automated by consolidating historic data from requirements management systems, project management systems, and defect management systems. The estimates can be continually revised based on the actual changes and defects occurring in the due course of the project. The data can then be used to make autonomous change requests, approval requests, or backlog updates.

The natural language understanding and DL algorithms, such as convolutional neural network and recurrent neural network, are leveraged for requirement analysis and effort estimation by going through user stories, acceptance criteria, or feature files like Voice of Business.

Autonomous Test Design and Maintenance

In any software development lifecycle, it is expedient for feedback to be converted to test cases and scripts then executed to provide insights to developers and product owners. Otherwise, requirement drift, which is usually invisible to human eyes, causes features to be left out then recorded as production defects.

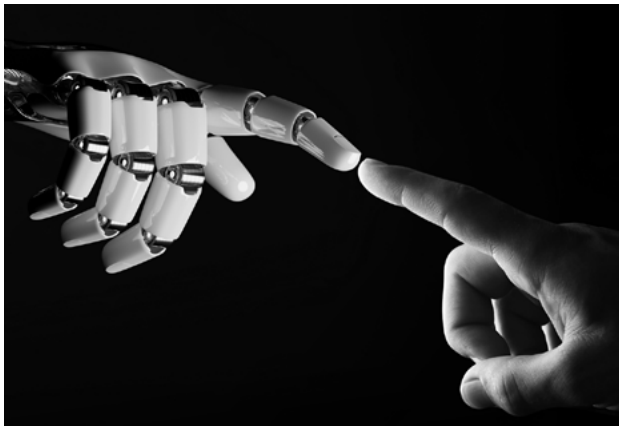
Hexaware's autonomous engine offers 360-degree coverage to combat any drifts, while autonomously generating test cases and automation test scripts by analyzing data from requirement management tools. NLP techniques such as BET or libraries such as Python NLTK, help us achieve this use case.

Continuous changes to user stories can be incorporated back into the test cases and scripts once the changes are made in the requirement management tools. This is an independent process that happens outside of the continuous integration pipeline and the newly created or updated scripts can be selected for execution in the build pipelines.

Autonomous Test Selection

The test cases and scripts created autonomously must be executed by specifically targeting the functionalities developed as part of the release, sprint, or check-in. Since a human software development engineer in tests (SDET) working in a regression team is not always kept in the loop, they end up spending a lot of time analyzing the changes before they can select the tests. Accuracy is often lost in the process, resulting in defect leakage or bloated test suites with very low defect density.

Autonomous agents can continuously gather data from an agile management tool showing the completed features and the source code repositories. This data is used to precisely select tests, which are then configured and executed on their own in a pipeline. Test selection can also be contextualized precisely by grading the current code changes against the production fallouts, and relevant tests can be selected specifically for smoke, sanity, and regression execution.



Autonomous Provisioning of Test Data and Environment

Often, tests are required to be executed with multiple sets of data and with subtle variations that are mostly decided by the human SDET based on his application knowledge. Hexaware's autonomous engine ensures that the edge cases and boundary conditions, along with multiple flows of tests, can be automatically decided using various ML algorithms, such as Random Forest, and then auto-configured.

The autonomous system can learn from the way the human SDET organizes and configures the tests for execution across multiple targets, such as end points, mobile devices, and browsers. The system then autonomously predicts the tests for similar data gathered from agile management tools and source code repositories.

Based on the type of tests selected for a change, the required environment scripts can be autonomously executed to prepare the environments. These environments can be on-premise or in containers in the cloud.

Autonomous Test Execution

Tests often fail during execution because of developers making changes to the DOM code, such as renaming a field, changing an id, or adding or removing the frames. Test scripts that fail due to network and application slowness often require re-execution, resulting delays in test completion and manual intervention to execute the pre-requisite test scripts.

The autonomous platform can ensure totally independent and resilient execution, which can dynamically prioritize, guess failures, and take necessary steps to avoid the failures. The platform also provides accurate insights on the execution and quality of the system to the project team in the form of dashboards.

The test execution status is reported back to the test management systems and project management systems, so SDETs spend limited time on execution and they can focus on higher order tasks, such as coaching ATOP for accuracy.

Autonomous Test Reporting and Decision Making

Typically, the production release decision is a collaboration between the product owner who represents the business, the Scrum Master who represents the IT team, and the SDET who represents QA. The challenge arises when these key stakeholders communicate in different languages. While business speaks the language of business metrics, the Scrum Master works with a functional requirement language, and the SDET uses test case/scripts language.

Hexaware's autonomous platform helps tackle these challenges by mapping functional requirements to tests to business metrics. The platform can translate the time series data from the automated test executions and correlate them to the business metrics. The priority of the business metrics, based on current production fallouts, can also be used as input to prioritize the tests for a release regression. This input is very valuable to an organization that also receives inputs from the sales and marketing divisions.

To learn more about how autonomous testing helps QA teams ensure quality in high-speed development environments, visit https://hexaware.com/services/digital-assurance/autonomous-software-testing/.

Additional Resources

MORE INFORMATION FOR SOFTWARE PROFESSIONALS

SERVICES PAGES

- Hexaware's Test Automation Platform—TALOS
- Autonomous Software Testing
- Continuous Testing Services
- API Testing & Service Virtualization
- Multi-Channel Test Automation

SUCCESS STORIES

- Automated Testing Solution for a Leading American Transportation and Logistics Firm
- Automated Testing Solution for a Leading Insurance Company
- Mobile Test Automation for Largest U.S. Wireless Communications Service Provider
- Automated Testing Solution for a Leading Travel Technology Company
- Mobile Test Automation Validation & Production Monitoring Framework Implementation
- Mobile App Automation Testing for a Leading Wireless Service Provider

RELATED VIDEOS

- Hexaware's Multi-Channel Test Automation Solution
- Hexaware's Integrated2Design Solution for Test Automation

WEBCAST

- Continuous Testing for Multi-Channel Apps over Cloud

