Introduction: Today’s testing landscape

If we consider recent history, we can see that we’ve shifted the goal posts from software delivery cycles that span months, or longer, to agile delivery which has made two-week iterations the norm, to continuous development cycles. We’ve also therefore had to move to continuous testing cycles and involve test automation to address increased software complexity and a modern pace of application development and delivery.

Nevertheless, when we look into the future, it’s clear that even continuous testing will not be sufficient. In the last couple of years, applying AI techniques to software testing has become feasible – it provides an exciting opportunity to improve existing processes, but we need it to meet the quality requirements of a future driven by IoT, robotics and quantum computing.

Recent years have shown that machine learning, data mining, knowledge representation, constraint optimisation, planning, scheduling, multi-agent systems, etc. have real potential to positively impact software testing. The last couple of years have seen a rapid growth in testing AI applications as well as the application of AI techniques in software testing – so it’s not just future-gazing.

How do AI and QA interact?

The use of skilled engineers to apply monitoring techniques that search for faults, engage in root-cause analysis, and execute appropriate recovery strategies remains the de facto standard of most professional organisations. Testing in the main is still geared toward manual testing, or the manual creation of "automated" test scripts, and most A-QA testing activities are focused on only a single, or a small number of applications.

Automated test scripts tend only to target functional, structural and performance issues, and testing concerns like usability and accessibility are often deemed too difficult to automate. In the same way, understanding the reasons for test failures or even failure prevention is also impossible for automated scripts.

The perfect solution is true automation without human intervention and systems that deliver better testing than application test teams – and this is where AI plays its part. To take it a step further - imagine a world where software can test, diagnose, and heal itself, and all we have to do is continue to train it.
Self-healing software

Self-healing platforms are capable of identifying and fixing operational issues automatically, before they cause full-scale systems failures. The technology monitors how systems normally function and flags any deviations from those patterns as potential errors. If the system identifies a certain problematic pattern, it automatically makes adjustments to restore normal operations.

A test failure could be a sign that the latest application changes broke business-critical functionality...or it could stem from a number of other issues. For example, a dependent system (e.g., a third-party application or API) may be temporarily unavailable, broken, or so slow, causing the test to time out. A simulated test environment might not be available or functioning correctly. Or, the test data provisioning process might be feeding the test expired or inappropriate data.

By watching the system’s back-end behaviour during both passed and failed tests, such systems learn which variations are acceptable and which are problematic. They then use this insight to predict the root cause of a test failure. If a test is failing due to an issue with the application under test, they provide developers the most critical details for debugging and resolving the issue.

In many cases, they can also automatically “heal” the root cause of a test failure. For example, assume that a test is failing because it’s trying to interact with a third-party API that’s temporarily unavailable. They can automatically simulate the behaviour of that API based on previous successful tests runs—enabling testing to proceed as normal. Which is impossible in the A-QA world.

How can you integrate AI into your testing approach?

Artificial intelligence testing tools can work side by side with the software testers in order to achieve improved quality in software testing and make the delivery faster. It’s a good idea to start looking at existing applications that can significantly simplify a test engineers’ life. Most are based on their purposes such as:

- Maintaining automation tests
- Test cases generation based on user behavior
- Mobile test automation
- Visual testing
- Code analysis and tests run which cover the impacted area

Attention must also be paid to the necessary technical skills required for involving AI in the testing approach. There are a lot of free internet courses and conferences that provide a good start.

We’re only at the beginning! After that I hope this initiative will lead to project by project introduction to AI in QA practices and common usage of the best tools with embedded AI that can facilitate the testing process within projects.

Godel’s work on AI within QA

Godel has already started a series of education meetups to gather interested parties together and discuss how we can improve our software testing activities.

We have discussed what AI and its basics such as machine learning, natural language processing, vision, expert systems, planning, scheduling systems, robotics and self-healing systems are.

We will also be discussing existing tools with embedded AI technology that help testing as well as best practices for testing systems that are built with AI.
Impact on QA Engineers

As with all technology, some prophesize that AI might put humans out of jobs. I don’t agree. Even with AI performing all the basic tasks, human expertise would always be required to oversee the testing process, so skilled and specialised QA testers face no threat. Even when AI detects a deviation while performing testing, a human expert would still need to go in and validate the issues discovered.

8 of today’s AI-driven testing capabilities:

- **Exploring application behaviour**
  Test bots generate actions such as filling out a form on a screen, clicking submit, and checking for an appropriate response. This may be based on historical data collected from human-present software testing sessions.

- **Detecting failures and system changes**
  Test bots use image recognition and other techniques to determine when a failure occurs, or to detect legitimate changes in the current version of the application. AI has enabled visual UI test automation.

- **Discovering application structure**
  Test bots can perceive the different screens and widgets in an application and classify them correctly.

- **Learning from tests or user traces**
  Test scripts and user traces have concrete examples of interesting paths that human testers and end users cover when exploring an application. Test bots trained on these examples can generalise them to new cases.

- **Adaptive testing**
  Test bots can modify their behaviour at runtime based on feedback. This is generally achieved through a machine learning technique known as reinforcement learning. In reinforcement learning, positive outcomes are rewarded, and negative outcomes are punished, allowing the bots to improve over time.

- **Faster manual testing**
  AI will make developers more time efficient, as script writing and analysing large amounts of data sets will become faster and less error-prone.

- **Increase in test coverage**
  AI can find answers to detailed questions in a matter of seconds. Testers can use this information to decide whether coding changes are required to prevent program errors.

- **Automating the testing process**
  The more tests you conduct, the more labour-intensive and expensive their support becomes. Since bots are not fully encoded, they adapt and learn to find new application functions themselves - automatically estimating to decide whether it is a new functionality or defect.
What should QA engineers do to stay ahead?

All in all, AI in QA only helps the testing process - it requires additional skills from QA engineers.

To ensure quality in a machine learning project, the team's AI quality engineering needs an extended set of skills. The team should have expertise in A/B testing and metamorphic testing, amongst other techniques that have gained new importance.

Strong programming skills in the most prominent machine learning languages, such as Python, Scala, R, Spark, are required, as well as in languages such as Go and C++, and with open-source software libraries like TensorFlow. This isn’t just about understanding the developed software, but is also about creating a custom toolset for specific tests. These skills need to be extended by a strong understanding of the new technologies: machine learning, big-data and cloud computing.

Mathematical skills, especially in statistics, calculus, linear algebra and probability are the core to understanding machine learning. Knowledge about computer-hardware-architectures is crucial to determine the performance of a chosen model.

Just as automation in testing is nowhere close to replacing the manual tester, the same applies to AI. If you are a software tester, you shouldn’t start panicking about the possible takeover of your job by artificial intelligence.

What you should do instead, is keep yourself updated with the changing technology and keep learning.

A QA Engineer’s Toolkit* (for AI expertise)

Testing skills

Classical testing techniques, validation techniques, risk analysis, knowledge of DB/API side of applications, security testing, load testing...

Programming skills

Mathematical skills

Numerical methods, calculus, linear algebra, statistics, probability

*This is an example toolkit, just a few of the many tools, technologies and skills.