mabl

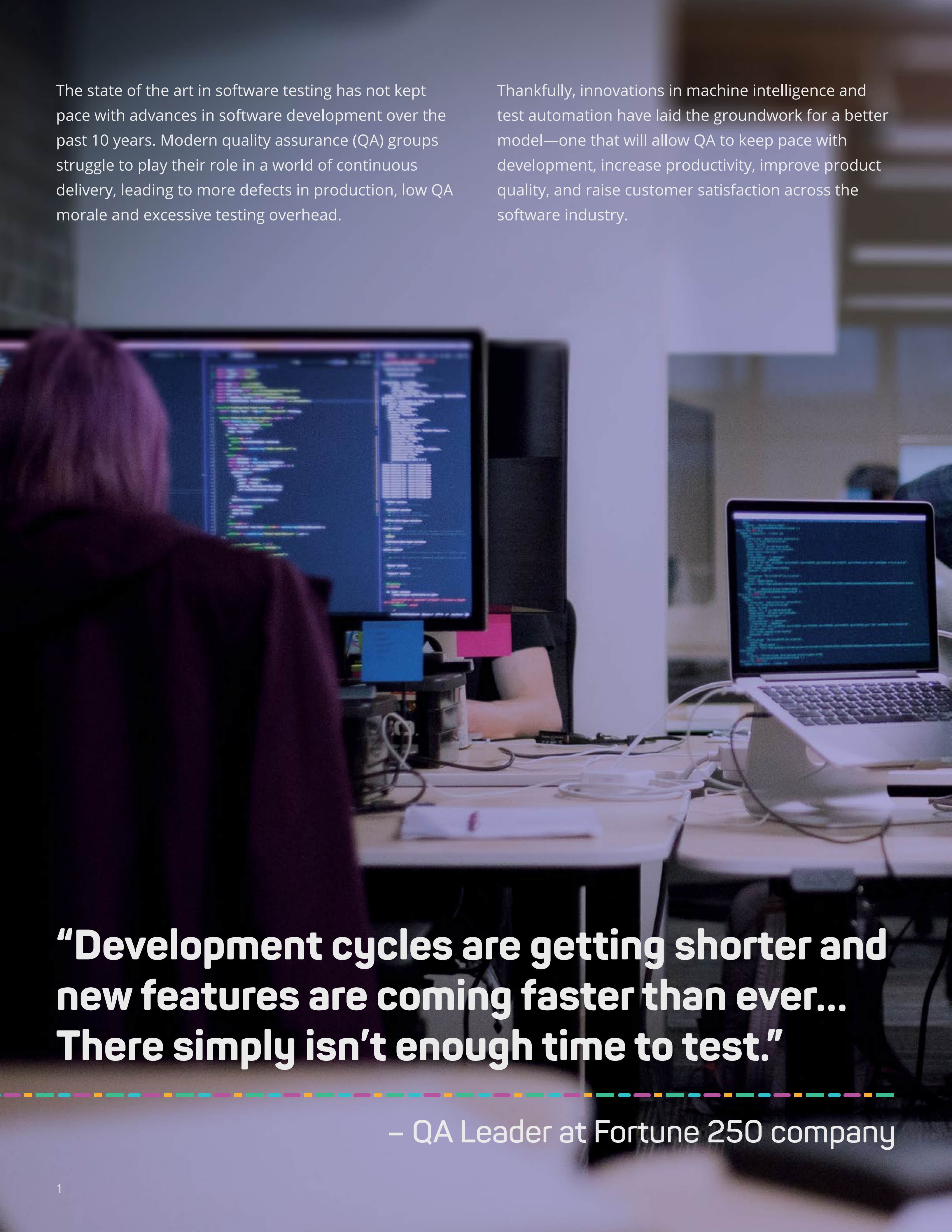
The Era of Intelligent Testing

mabl.com

May 2018

The state of the art in software testing has not kept pace with advances in software development over the past 10 years. Modern quality assurance (QA) groups struggle to play their role in a world of continuous delivery, leading to more defects in production, low QA morale and excessive testing overhead.

Thankfully, innovations in machine intelligence and test automation have laid the groundwork for a better model—one that will allow QA to keep pace with development, increase productivity, improve product quality, and raise customer satisfaction across the software industry.



“Development cycles are getting shorter and new features are coming faster than ever... There simply isn’t enough time to test.”

– QA Leader at Fortune 250 company

Why We Need a New QA Toolset

Existing QA solutions were built for a world where software changed infrequently and functionality was highly specified and documented. This is no longer the case.

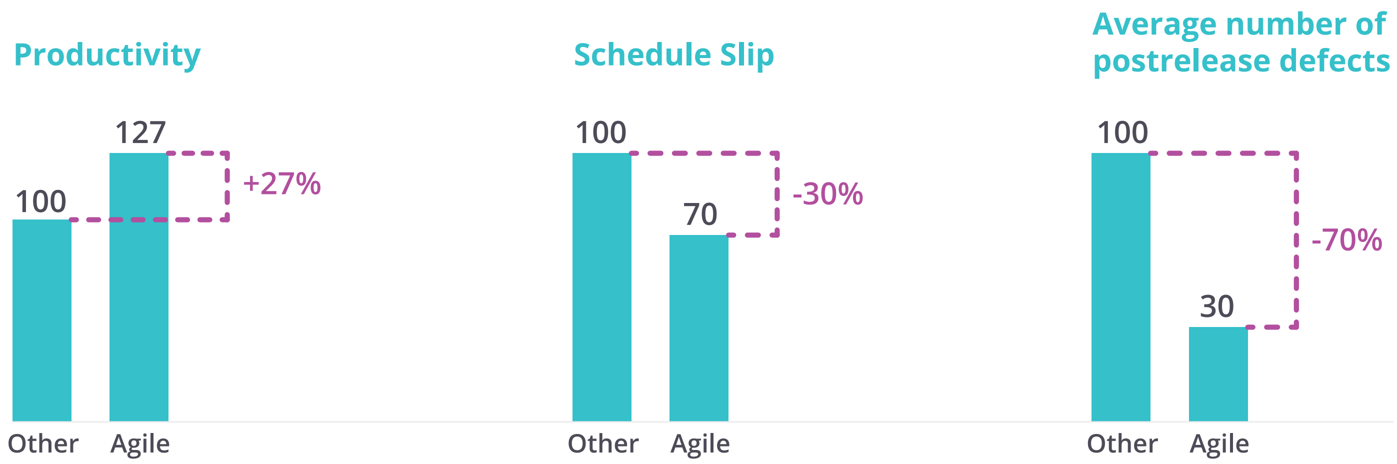
Products Are Evolving Faster Than Ever

Several forces are converging to accelerate development. Software as a service (SaaS) has removed many of the issues around operating system compatibility, packaging, deployment and versioning that consumed significant development time during the installed software era. Consider that in 2016 Amazon Web Services delivered more than 1,000 new features and services—a pace that would be impossible with packaged software.

The proliferation of open source and cloud-based features as a service has also reduced the time that developers spend building generic functionality. At mabl, for example, the thought of building our own authentication engine never crossed our minds; we use Auth0. Instead of building our own data processing pipeline, we use Apache Beam and Google Cloud Dataflow. Rather than building our own ML framework, we use Tensorflow. And the list continues; reusable components and cloud-based services save us years of engineering effort, allowing our developers to deliver product features rapidly.

Continued adoption of agile methodologies has further increased the efficiency of development teams. A recent McKinsey study of 1500 software projects demonstrated that agile teams are 27 percent more productive than those that follow methodologies such as waterfall.

Performance of teams using agile software development vs those using all other software-development methods,[1] % of 'other'



[1] Based on more than 1,500 projects in Numerics industry software database.
Source: Numerics; McKinsey analysis

An analysis of more than 1,500 projects suggests the value of agile.

Finally, continuous integration and continuous delivery are accelerating the pace of change in software. A recent survey by Atlassian suggests that more than 50 percent of software teams are already using continuous integration and/or delivery, and more than 32 percent have plans to move to CI/CD. The same survey indicates that more than 50 percent of organizations are able to push changes to production daily.

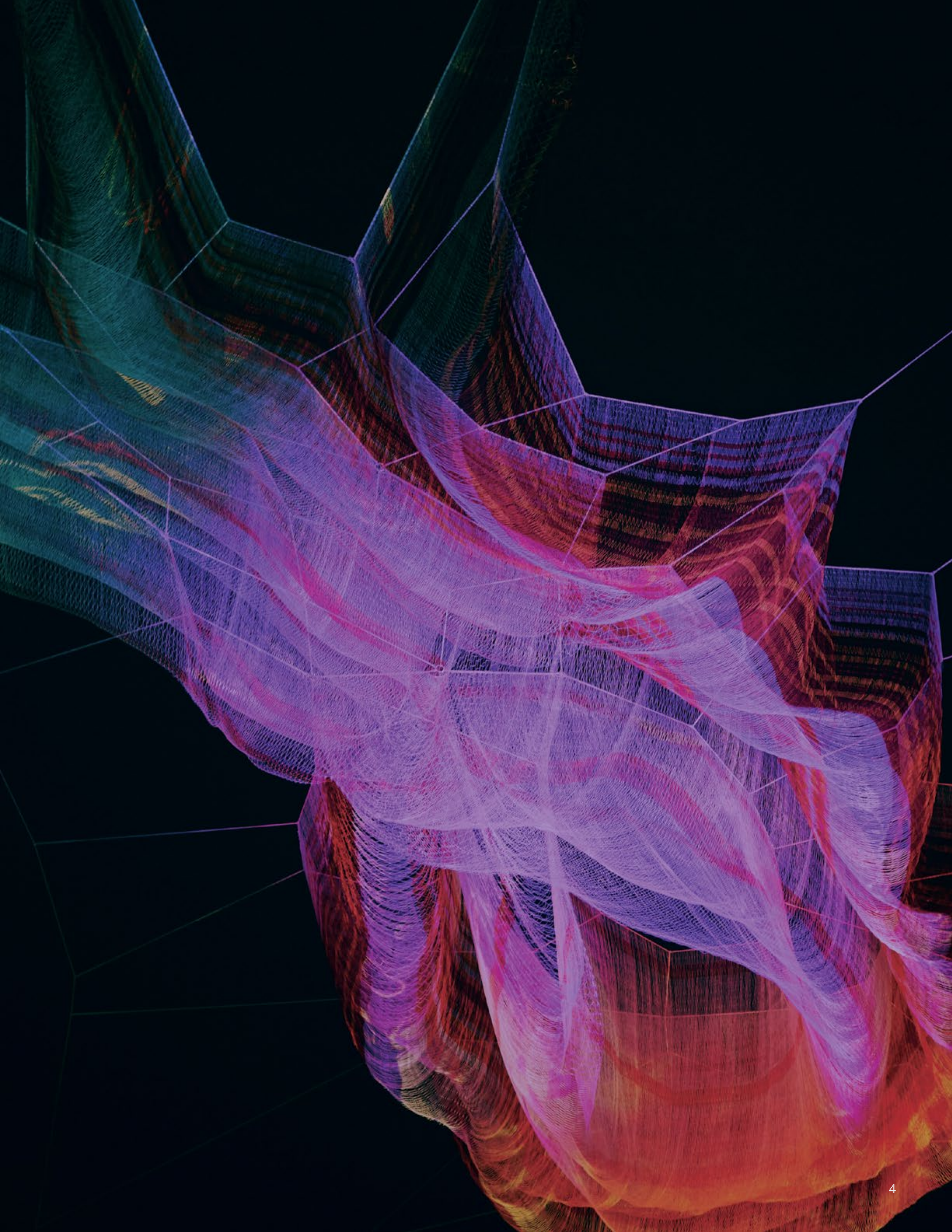
In the end, improved developer productivity means that software products are evolving faster than ever. Unfortunately, this places increased pressure on the QA function. As one QA leader from a Fortune 250 company said in a recent interview with mabl, "Development cycles are getting shorter and new features are coming faster than ever. And we have five productive developers for every QA engineer. There simply isn't enough time to test."

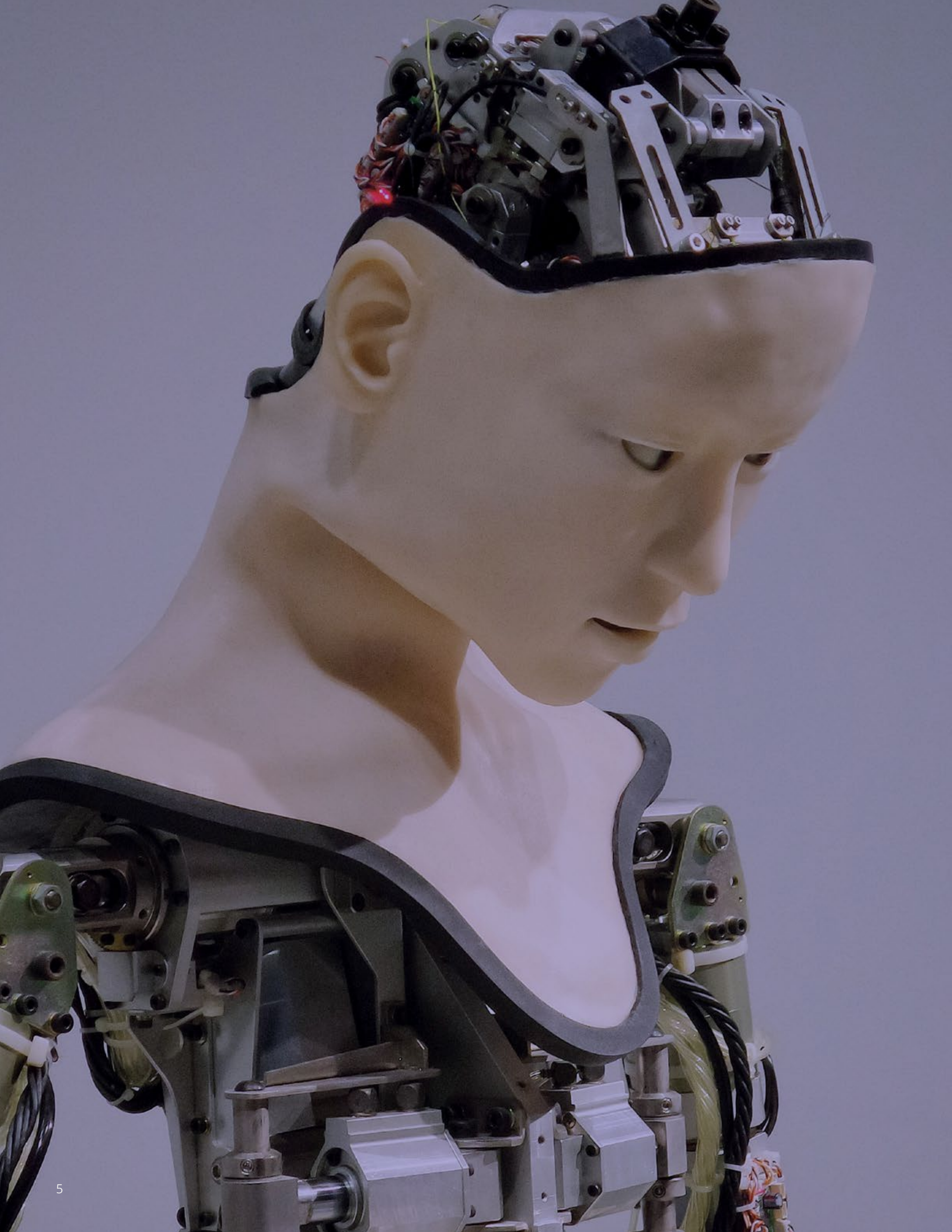
Existing Automation Frameworks Impose Too Much Overhead

QA rightfully relies on automation frameworks to improve productivity; tools such as Selenium, Appium and JUnit now enjoy widespread adoption. More than 80 percent of the respondents to a recent mabl survey use Selenium today, and QA managers are under constant pressure to increase automation. Unfortunately, while Selenium and other frameworks have helped many companies increase their QA velocity as compared to manual testing, these frameworks have significant flaws:

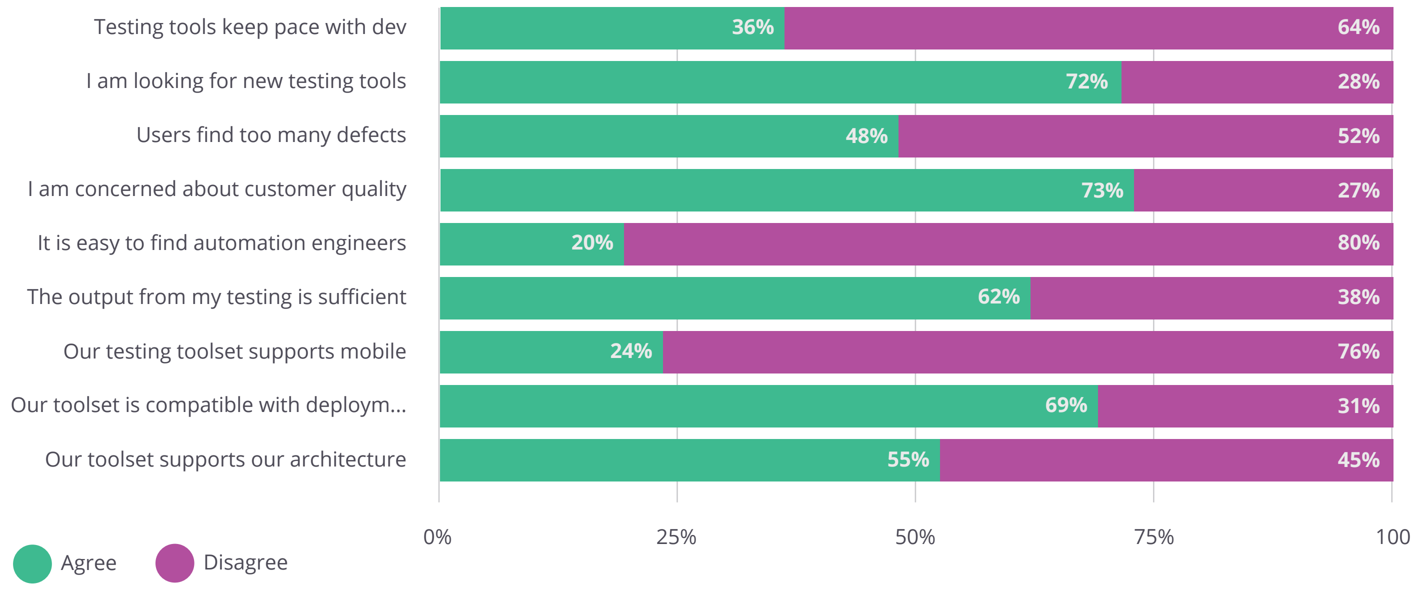
- They all require specialized scripting expertise, and given the lack of available talent, teams have very limited capacity for QA automation.
- They provide limited context in test results, which leaves teams with very little information to help them triage and address issues.
- At scale, they require their own infrastructure, which takes time to provision, operate and scale.
- They are tightly coupled to attributes of the product under test that change frequently (xPath, CSS, etc.), resulting in constant maintenance and false positives.

These flaws only become more painful as the pace of development and change accelerates, compelling many teams to limit their use or discard them altogether, resulting in lower product quality.





“Please indicate the extent to which you agree with the following statements”



mabl survey - satisfaction with existing testing tools and processes (n = 104)

Time Pressures Lead to Unhealthy Team Dynamics

People in testing roles are increasingly at odds with their peers in development. Unable to keep up with the continuous pace of development, QA becomes a bottleneck to product velocity, leading to tension with their peers in development. This constant pressure also starves QA of its ability to make the very investments in automation and process improvements that are needed to improve throughput. The feeling of always being behind with little hope for improvement often leads to low morale within the team.

Gaining consensus on the standard of quality has also become difficult for teams, since agile teams often steer clear of detailed functional specifications for each release, preferring instead to focus on user goals and requirements (see: Agile Manifesto). Because user goals and requirements are open to more interpretation, QA and development can disagree on not only the severity of defects but also the validity of the tests themselves.

Modern software teams are also more open to evolving requirements during a given release cycle (see: “Responding to change” in the Agile Manifesto). Evolving requirements should drive updated test cases, but this requires deep engagement and coordination for already stretched personnel. Rather than testing for quality versus specified behavior, we rely much more on ad hoc and manually intensive exploratory testing to look for defects.

There is Hope for QA

While faster feature development, shorter release cycles, evolving requirements and unclear accountability for end-to-end quality all present real challenges for QA, a new generation of QA tools are emerging to confront these challenges head on.

It Will Be Easy to Create and Maintain Tests

Unlike existing test automation frameworks, next-generation QA tools will not require specialized scripting expertise. Instead, they will offer intuitive interfaces that allow anyone to quickly create and manage tests. This will help teams expand their test coverage and evolve their tests quickly as their products evolve.

Tests Will Adapt Seamlessly to Change

Because they use much more sophisticated, durable methods of simulating user behavior in automated tests, next-generation QA tools will not suffer from the brittleness associated with existing automation frameworks. Tests will not be tightly bound to individual elements within the front-end codebase that change frequently. Rather, they will use machine intelligence to create and maintain sophisticated models of tests, enabling tests to adapt the tests rather than fail when xPaths and other locators change.

QA Will Run in the Cloud

Today, teams struggle with the performance, cost and operational overhead of on-premises testing systems. Next-generation QA tools will take advantage of on-demand cloud computing resources to execute tests faster and more efficiently. They will tap into powerful data processing and analytics services to analyze test results and deliver insights. They will be delivered completely as a service, offloading the management and operational burden to tool vendors.

QA Will Be Part of the Delivery Pipeline

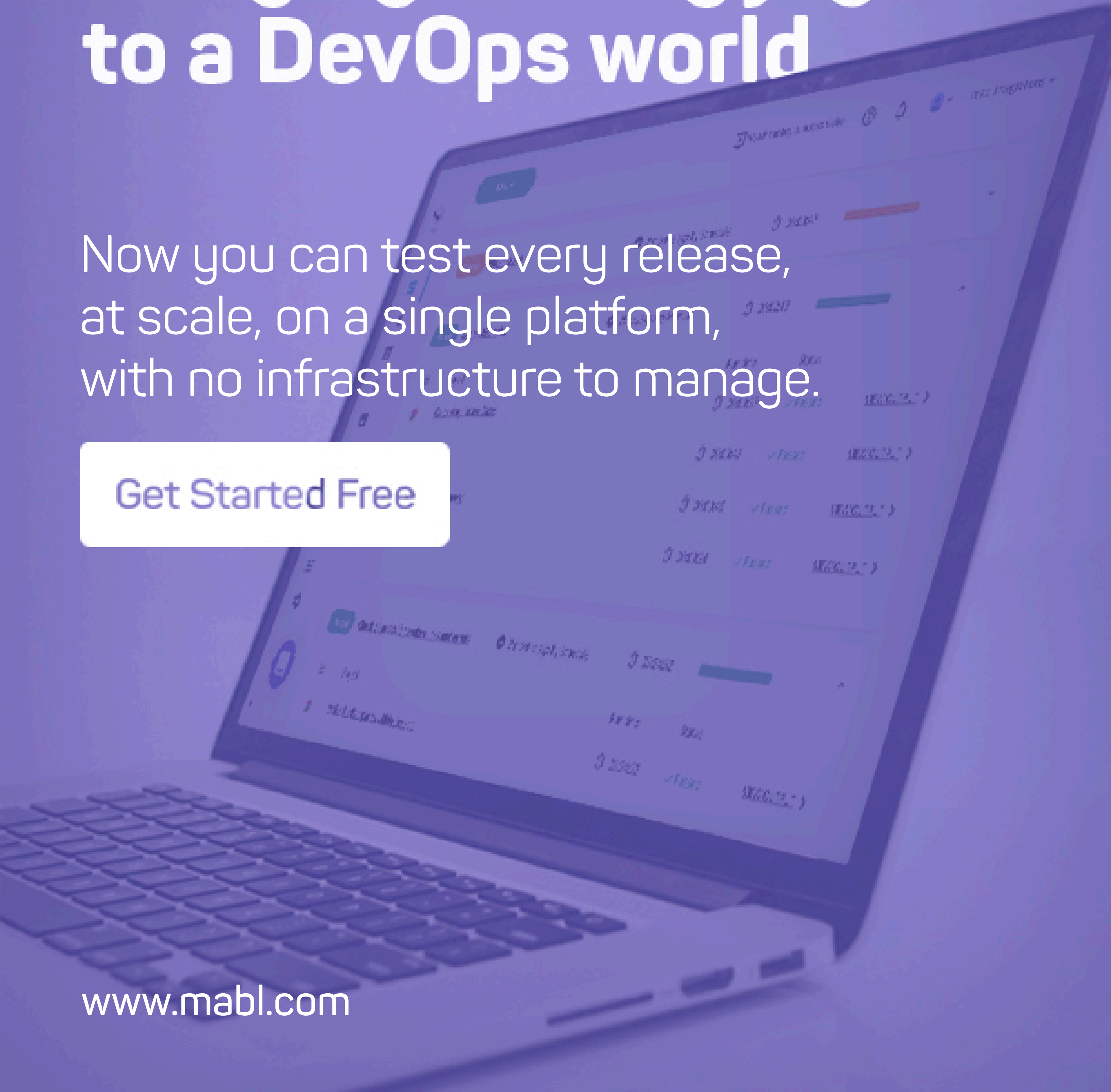
Next-generation QA tools will be tightly integrated into automated CI/CD pipelines orchestrated by Jenkins, Spinnaker, CircleCI and CodeShip. Tests will trigger automatically whenever teams make changes to the product under test and on-demand. The tools will notify team members when there are potential issues so that they can address them before they affect users. It will be simple to run tests and compare results across builds and environments. Decisions around releases, deployments, promotions and rollbacks will be largely automated.

The Definition of Quality Will Evolve

Beyond validating specific assertions in tests, the next generation of QA tools will use machine intelligence to automatically detect and highlight potential regressions in applications. As a result, we'll think less about tests “Passing” and “Failing” and more about the extent to which changes improve or degrade the user experience. We will focus less on code coverage and more on product coverage. We'll look beyond basic counts of passing and failing tests in favor of a more holistic approach to assess the risk associated with a given build or deployment.

Tools Will Be Deeply Embedded Within Our Teams

Next-generation QA tools will effectively serve as extensions to development teams. They will share test results and insights through Slack, Jira email and other common communications channels. They will take feedback and training from the entire team—including product, development, QA and customer success—and incorporate what they learn into their tests. Most importantly, next-generation tools will describe tests and results in plain language, and they will provide useful context to help developers reproduce and fix issues, just as an effective QA teammate would.

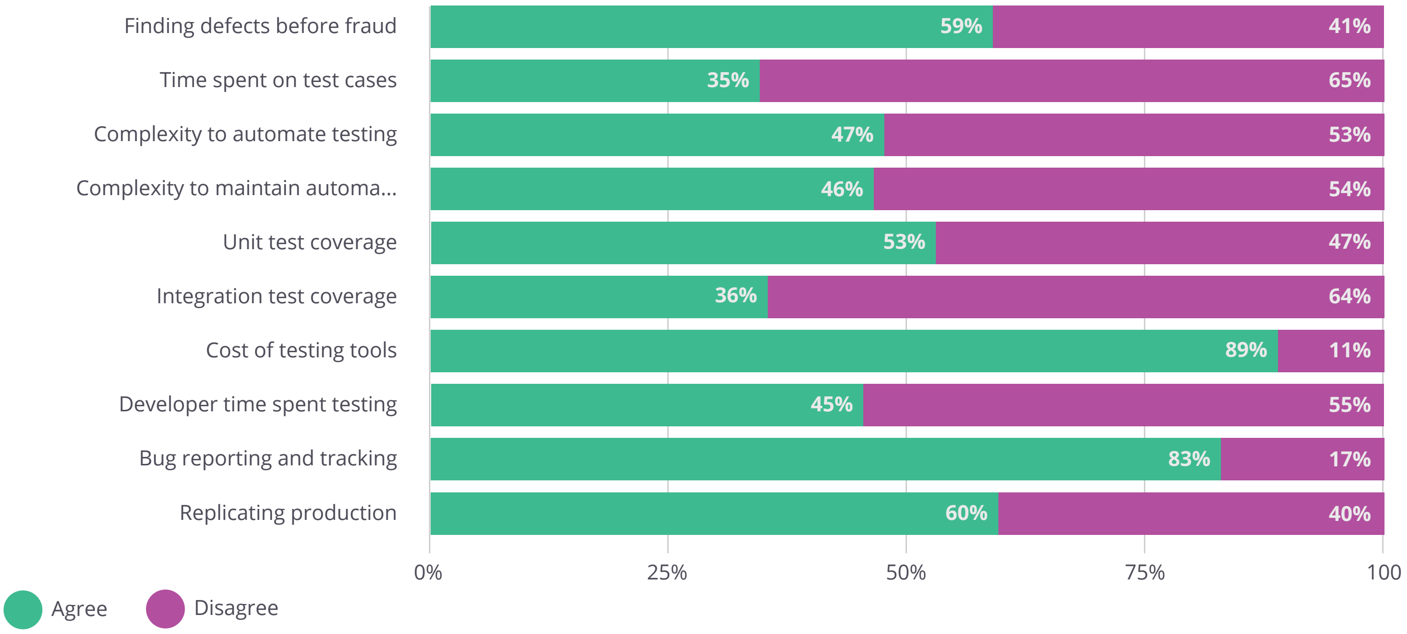


The Era of Intelligent Testing

Software quality assurance has never been easy, and the accelerating pace of software development in recent years has made it even more challenging.

Thankfully, a new generation of tools are emerging to address this challenge. Incorporating cutting-edge machine intelligence, delivered from the cloud and embedded into the modern development workflow, these tools will dramatically improve the effectiveness of QA, ushering in a new era of efficiency and innovation across the software industry.

"How satisfied are you with the following aspects of the testing process at your company?"



mabl survey – user sentiment (n = 104. For simplicity, combined "Agree" and "Strongly Agree" into "Agree" category, combined "Disagree", "Strongly Disagree" into "Disagree" category.)

About the Author

Dan Belcher
mabl Co-founder

Dan has spent his career building products to make life easier for software teams. He is a co-founder at mabl, a venture-backed startup that is using AI to make end-to-end testing easier. Prior to mabl, Dan was the Lead Product Manager for monitoring and logging at Google, which he joined through its acquisition of Stackdriver, a company that he also-co-founded. Dan spent his early career in product and technical leadership roles at VMware, Sonian, and Microsoft.

About mabl

Mabl is the ML-driven end-to-end testing service designed for modern developers and QA/Automation engineers. With the increasing speed of software delivery cycles and adoption of DevOps practices, legacy tools for software testing have been left behind. Mabl is an automated testing service which allows developers to ship software faster while maintaining an optimal customer experience. The service uses machine driven regression testing to train machine learning models that are used to quickly surface insights related to assertions, javascript errors, visual changes, broken links, and more. To learn more about mabl visit mabl.com.

mabl

141 Tremont St, Boston, MA 02111