saralam

Top Challenges with Software Test Automation

And How The Right AI Solution Can Help Overcome Them

TABLE OF CONTENTS

Introduction

Over the years, technology and engineering organizations have focused on building scalable and resilient test automation to keep up with the agile pace of software delivery. However, a few challenges persisted throughout the pursuit of end-to-end test automation.

In this paper, we would like to share the challenges that we have seen first-hand, heard from technology leaders, and gathered through industry research while suggesting practical solutions for them. Particularly, with the recent advances in AI, we believe that there are effective ways to address them.

Software Testing Lifecycle

Our goal is to provide you a clear line of sight into various AI-driven approaches to help you evaluate the best approach for addressing your needs.

Top Test Automation Challenges

1 Test Automation doesn't scale[1]

This is at the top of engineering leaders' list. Without a huge time and resource investment, they cannot achieve their desired test automation coverage.

2 Automated tests are fragile[2]

Even after achieving meaningful test automation, the next challenge is to maintain it continuously. With today's rapid pace of software delivery, automated tests often break.

3 The "automation catch-up" tax

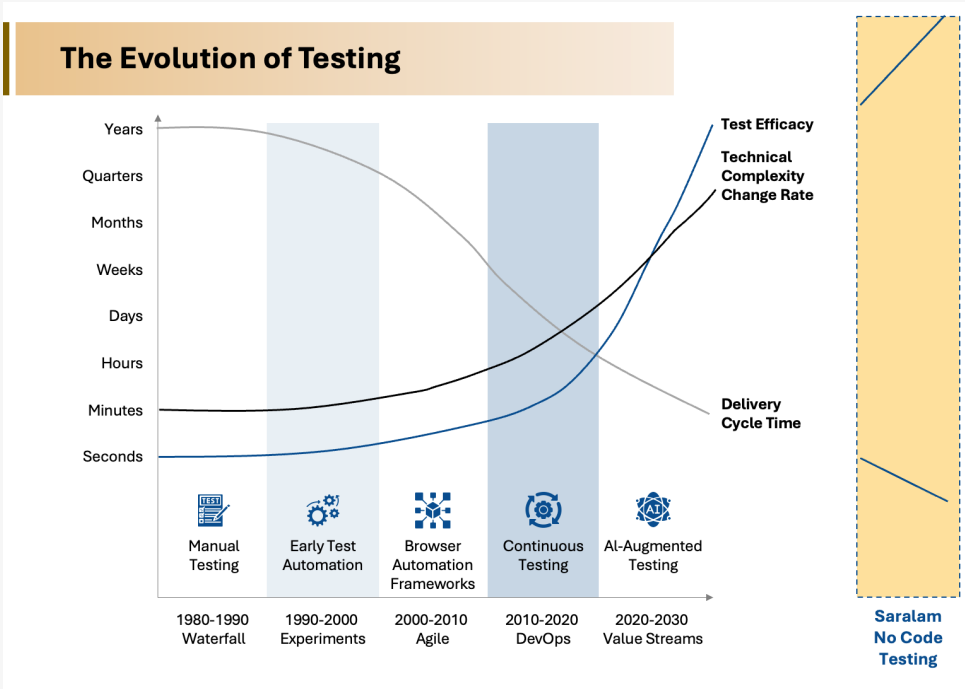
Typically, test automation lags development by 2 to 3 sprints due to feature availability. This ultimately results in issues reported later and developers needing to switch context to fix them.

4 Tools don't automate every testing phase[3]

Traditional automation tools do not automate requirement analysis, test plan design, writing test scripts, and test maintenance. Thes tasks are mostly manual today.

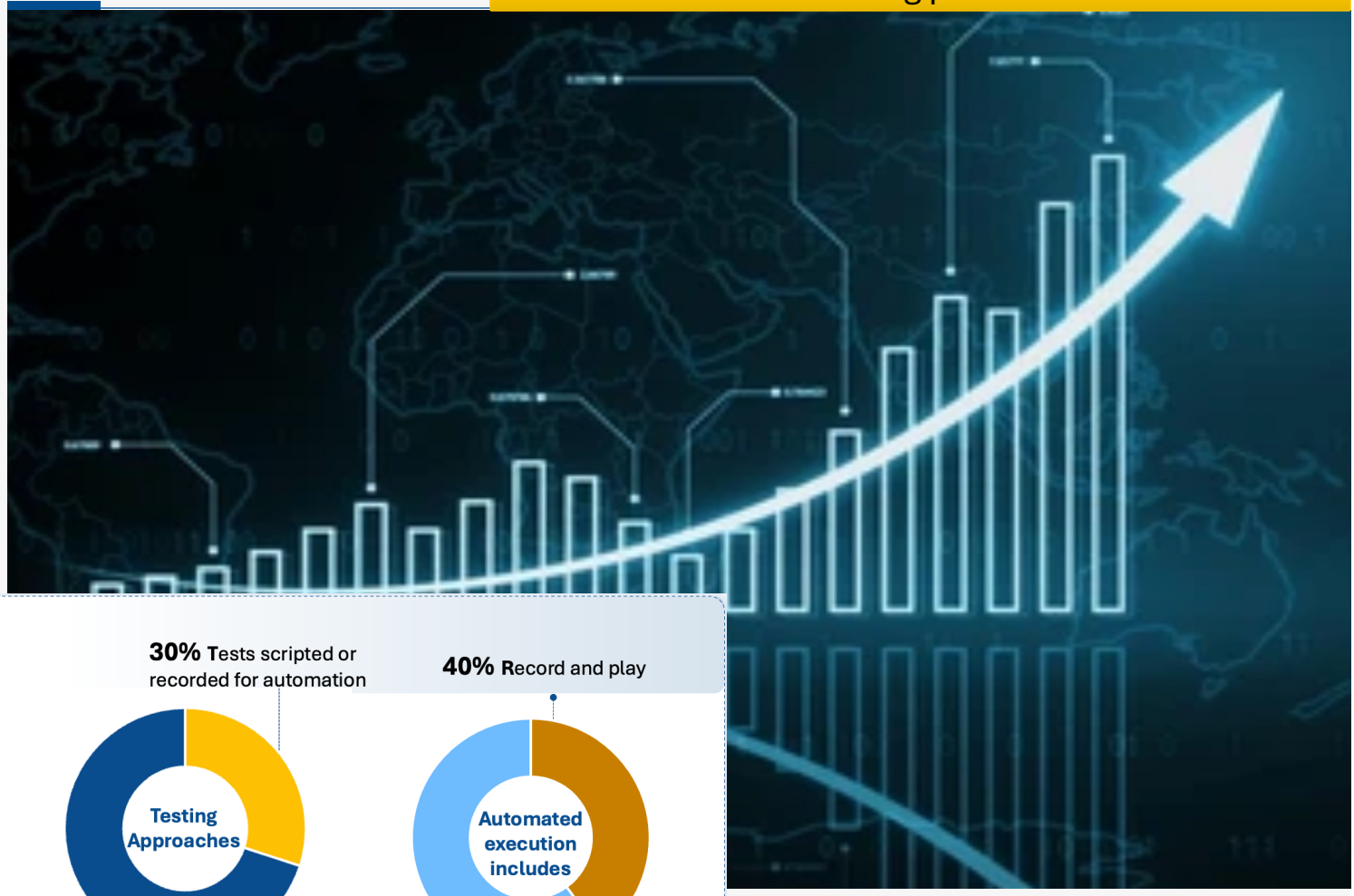
5 Lack of synergy

Even with the most advanced test automation tools, engineers can't leverage automation across similar workflows without significant modifications.



Scaling Test Automation

Hint: Assess which testing phase can be automated



1. Start evaluating what phases of the test cycle can be further automated

Today, test automation is limited to test execution and, to some extent, reporting. Automating Requirement Analysis, Test Design, and Test Development can provide the needed automation scale for overall testing.

2. Prioritize the test phase to automate

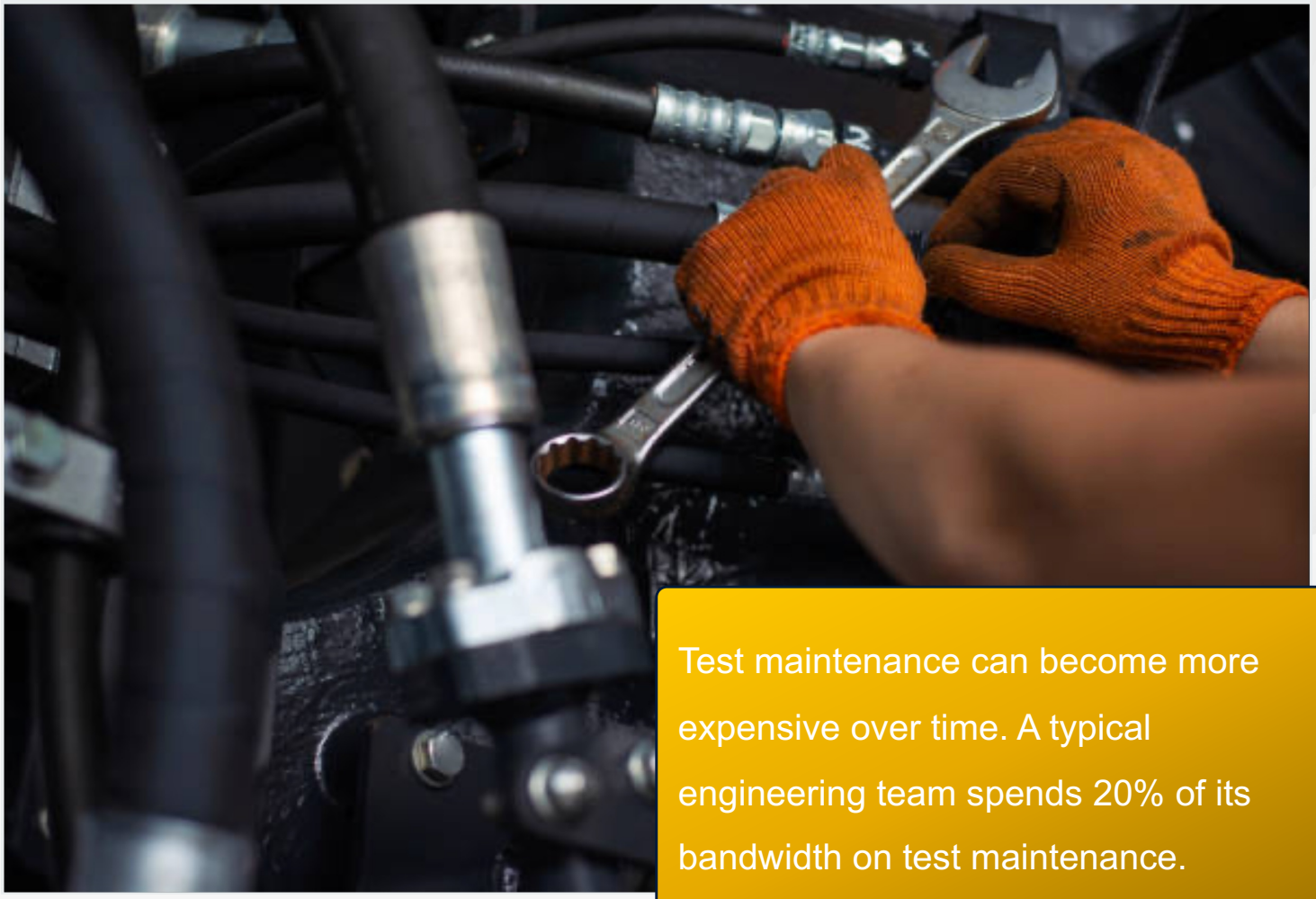
Based on your use cases and needs, prioritize a specific testing phase for automation. For example, you can target automating test plan design first before going for automated script writing.

3. Leverage AI for automation[4,5]

Once you prioritize a test phase to be automated, investigate how AI tools or services can help you automate that phase. The two big advantages of AI are automation and scale, which should be leveraged.

Building resilient Test Scripts

Hint: Dynamic context awareness can build robust tests.



Test maintenance can become more expensive over time. A typical engineering team spends 20% of its bandwidth on test maintenance.

1. Align test automation technology with development practices

Test automation tools primarily rely on an underlying DOM structure to execute automated tests. If your development practice allows agility in changing the DOM structure, then traditional test automation tools may not be suitable for you.

2. Explore dedicated test tagging approach

You may have to include dedicated tagging for test automation that does not change, regardless of changes in the user interface.

3. Utilize cognitive AI [3,4,5]

A well-trained AI tool can build the full context of an application workflow using its visual interface. Such technology helps remove dependencies on underlying DOM or HTML tagging.

Eliminating the “automation catch-up” tax

Hint: Use technology that can deliver automation upstream

Test automation consistently lags 2 to 3 sprints behind feature development.

1. Adopt a Test-first methodology

Educate and empower your teams to think about test automation before a feature is developed. Explore if test development can be up-streamed using requirements or mockups.

2. Increase QA bandwidth

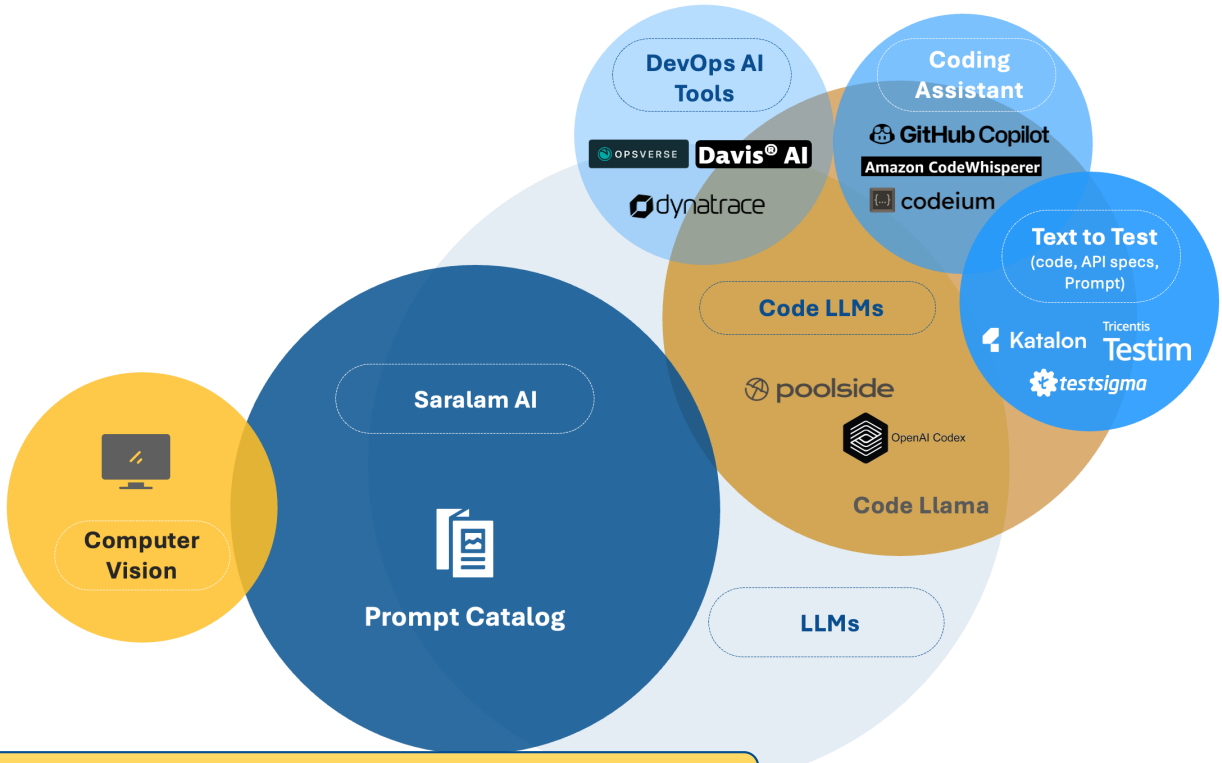
You may have to increase investment in test automation earlier in dev cycles. This investment will pay off by eliminating time spent bug-fixing and expensive context switching for developers later in the cycle.

3. Prioritize end-to-end test automation

Though unit tests are important, they do not catch issues when a complex workflow is executed. Focusing initially on end-to-end test cases can save cycles.

Automating every phase of test-lifecycle

Hint: You may have to look beyond out-of-box tools



We anticipate that new tools will continue to emerge in the coming years. This does not mean you have to adopt every single tool out there. Our recommendation is to follow an MVP approach when it comes to adopting new tools. Also, the right technology for you may not be available within a single tool.

1. Stay use case focused

Generally, your testing use case will drive the selection of technology. Unit tests and end-to-end scenario testing require different approaches and tooling.

2. Start with an MVP

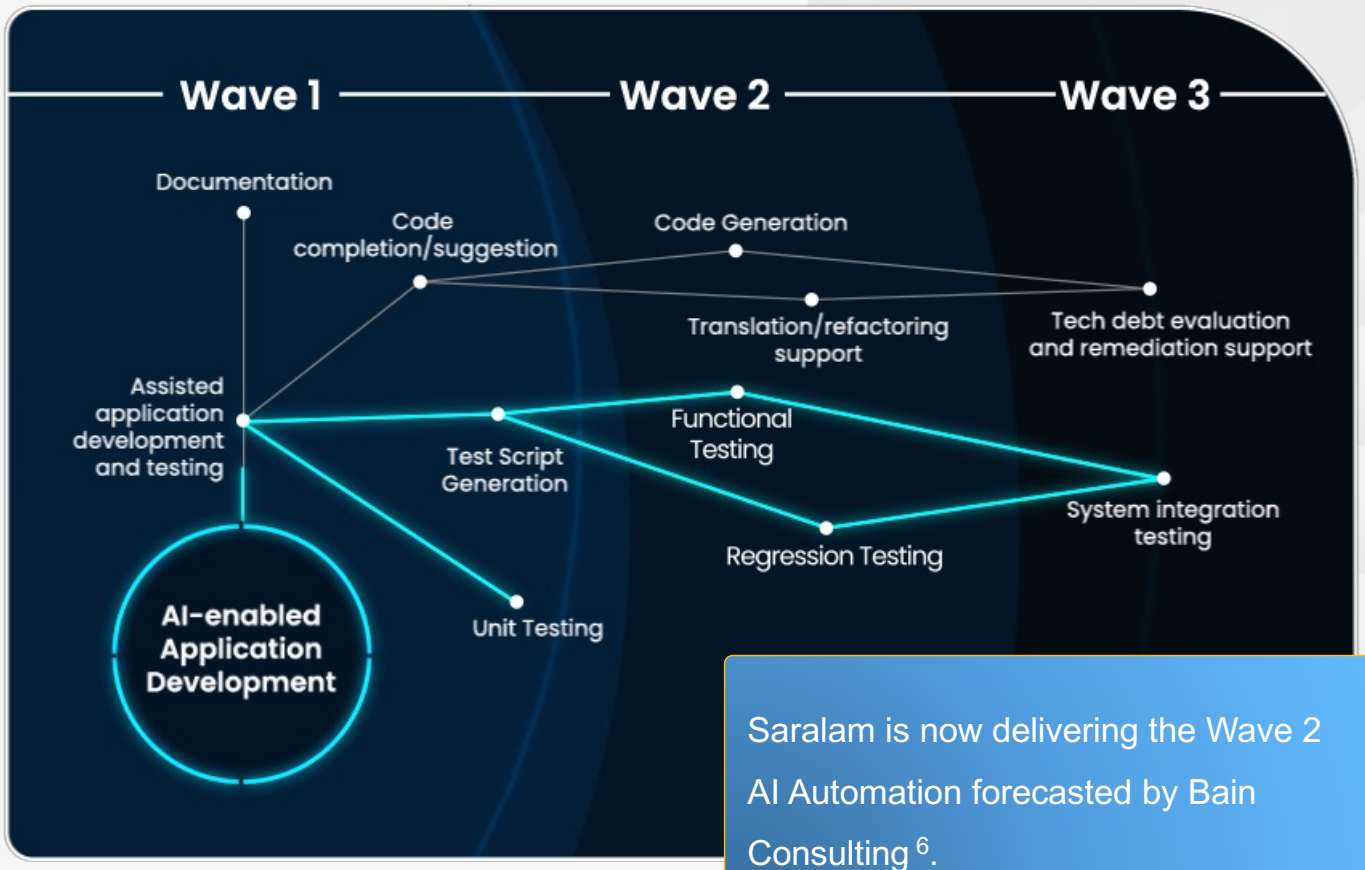
AI tooling generally requires upfront investment for training the model, training your team to use the tool, and continuous fine-tuning. Select an approach with minimum upfront investment to evaluate the best fit for you.

3. Embrace experimentation

AI is still evolving and is inherently probabilistic. You may have to try multiple approaches before selecting one. This may be a tool, a service, or a combination of both.

Saralam AI – Managed Testing Service

Hint: Low touch, low investment evaluation



Saralam is now delivering the Wave 2 AI Automation forecasted by Bain Consulting [6].

1. Achieve 90% "in-sprint" automation

Saralam's technology enables full test lifecycle automation, from requirements analysis to test maintenance. Saralam's AI enables test script development right from using mockups or application user interface

2. Scale up automation to your imagination

Saralam AI develops fully functional test script using its large test catalog. You get your test automation at machine speed and machine scale.

3. Increase your developer efficiency

Managed service solution and resilient test scripts free up your development team's bandwidth to focus on more important feature development.

4. Low touch MVP

You don't need to invest in any AI tools or prompt engineering to reap the benefits of AI.

A Quick Comparison with Other Approaches

Hint: An AI Tool may not be the efficient approach

Copilot/Text to Test

Copilot is a code-augmentation tool primarily focused on assisting developers in coding.

Copilot and other NLP “Text to Test” tools require a developer to write prompts to generate code.

Further manual work is required to verify and potentially update code written by code-augmentation tools.

Tests developed by code-augmentation are fragile as they utilize an underlying DOM or similar framework to operate on an application’s user interface.

In most cases, you will need to build and operate test infrastructure to run generated tests.

Saralam No Code Testing

Saralam’s No Code Testing develops fully functional test scripts for software testing.

Saralam does not require a developer or QA engineer to write any code or prompts.

Customers only need to set a configuration for running Saralam’s test scripts.

Saralam’s proprietary Visual Learning technology goes down to pixel level details to understand software application as a human would understand.

Saralam can runs test cases completely in its cloud infrastructure, lessening your infrastructure footprint.

Conclusion

We hope to have provided you with a good overview of common and pressing challenges faced by technology organizations in Software Test Automation. These challenges continue to persist today; however, with advances in AI technology you can overcome them with the right strategy to leverage AI for test automation.

From our perspective, each organization is unique with its own strengths and weaknesses. It is important to create a practical approach that works best for your organization.

Be the AI Champion for your organization and company

Software Engineering/IT organization is typically the tip of the spear for delivering technology advances to the entire company. Software test automation is a low-risk area compared to software development, compliance, etc. to experiment new AI technology.

Experiment with agility by partnering with the right experts.

Start small. Value focused experimentation will derive a sustainable competitive advantage.

Talk to us if you have any questions.

saralam Corporation

contact@saralam.ai

https://www.saralam.ai

+1-408-214-1595

References

01 *Gartner 2021 – Software Engineering Leaders Survey: Improving time to market, delighting customers and reducing production incidents are among the top activities for Software Engineering Leaders.*

02 *Capgemini 2022-23 – World Quality Report - Maintainability is the most important factor in determining test automation approaches.*

03 *Gartner 2020 – Achieve Business Agility With Automation, Continuous Quality and DevOps Survey: 84% of the recipients responded that AI/ML features are more important than other features in testing tools selection.*

04 *Gartner 2022 – Market Guide to AI Augmented Software: By 2027, 70% of professional developers will use AI-powered coding tools, up from less than 10% today; 80% of the enterprises will have integrated Artificial Intelligence (AI)-augmented testing tools into their software engineering toolchain, which is a significant increase from 10% in 2022*

05 *McKinsey 2023 – https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai .*

06 *How Generative AI Changes the Game in Tech Services: https://www.bain.com/insights/how-generative-ai-changes-the-game-in-tech-services/*