

Manual Effort and Maintenance Debt: The Push Toward Autonomy in Software Testing

by [Guy Currier](#), Analyst, Futurum Group

SPONSORED BY



New Techstrong PulseMeter poll reveals how testing organizations are absorbing the downstream pressure of AI-assisted development while much of their own work, ironically, remains manual. The survey also points to what these teams most want next.

Research Insights

Software testing occupies an awkward position in software delivery discussions; it is rarely the subject of the roadmap but is also almost always a constraint on it. A Techstrong PulseMeter poll of 137 practitioners and decision-makers involved in software testing or test automation suggests that testing remains a significant constraint in software change and delivery—and that the reasons have less to do with attitudes toward automation than with the operating economics of the automation these teams already own.

Still doing things by hand

Asked how much of their organization's testing is still performed manually, 25% of respondents said almost all of it, and another 23% said more than half. Roughly seven in ten organizations are running at least half of their testing on human effort.

That distribution deserves a moment. These are not observers speculating about someone else's function. The people closest to the work report that manual effort remains substantial despite existing automation. Thirty-seven percent of respondents actively perform testing or test automation, another 27% combine hands-on work with tooling and strategy decisions, and the remainder either manage QA teams or make automation decisions.

Key Takeaways

- **Manual testing is the baseline, not the exception.** Seven in ten software test engineers say at least half of their organization's testing is still performed by hand, including 48% who put the figure above half.
- **Automation has not eliminated the labor burden; it carries one of its own.** Forty-three percent of test engineers spend more effort maintaining existing automated tests than creating new coverage, and the difficulty or cost of that maintenance is the single most-cited barrier to expanding automation.
- **AI-assisted development is already adding pressure to testing teams.** Seventy-four percent report increased pressure over the past 12 months, and a third of them describe the increase as significant. This is noteworthy at a point when AI-assisted development is still early in most enterprises.
- **Teams prioritize tests that don't break over faster ways to fix them.** "Reducing automated test maintenance" ranks last among improvements test engineers expect will deliver business impact, at 8%. Reducing manual effort (22%) and improving the resilience and adaptability of automated tests (19%) rank first and second.
- **Strategic interest in autonomous testing runs ahead of current activity.** Sixty-two percent say that continuous, largely autonomous testing is a critical or important part of their future delivery strategy; 42% are actively evaluating or adopting AI-powered testing today.

Note that “manual” here is self-assessed by respondents, and unweighted. A team may automate its regression suite thoroughly while exploratory, acceptance, and edge-case testing remain human work, and still describe half its testing as manual. The finding is less a verdict on automation coverage than an indication of where testing capacity is still being spent.



Automation's recurring bill

The second finding helps explain the first. Asked to compare the effort spent maintaining existing automated tests against the effort spent creating new coverage, 43% of respondents said maintenance consumes more, and another 28% said the two are roughly equal. Only 29% report spending more on new coverage than upkeep—and that figure includes the 12% who say their automation requires very little maintenance at all.

Read together, the two insights reveal how automation carries a labor cost of its own. Scripts written to reduce manual effort generate their own class of manual work: diagnosing failures, separating real defects from broken locators, and rewriting tests each time the underlying application changes.

Findings on barriers in the poll support that reading. Asked to name the single biggest obstacle to expanding test automation and adopting agentic testing, respondents put “automated tests are too

difficult or costly to maintain” at the top, at 22%. Lack of time, budget, or staffing and insufficient AI or intelligent automation capabilities follow, tied at 18%, with the requirement for coding or scripting expertise close behind at 16%.

No single barrier dominates. The top four sit within six percentage points of one another—which is, itself, telling. It shows that no single blocker can be productively removed. Respondents are describing cumulative friction: automation that is expensive to keep current, teams without the hours to maintain it, and a skills threshold that limits who can do the work at all.

What teams want fixed—and what that reveals

Here is where the poll findings get especially interesting.

If maintenance is the most-cited obstacle, the reasonable expectation is that reducing maintenance would rank high among the improvements respondents believe would deliver business impact.

Instead, it ranks last.

Asked which single improvement would create the greatest business impact over the next 12 months, only 8% chose reducing automated test maintenance—the lowest-scoring of seven options. The highest-ranked options were reducing manual testing effort (22%) and improving the resilience and adaptability of automated tests (19%). Add the enablement of more non-developers to automate testing, at 12%, and 54% of respondents selected an improvement aimed at making automation easier to sustain or extend.

One interpretation is that respondents are drawing a distinction—prompted by the poll question itself—between the maintenance work and the fragility of the automation that drives it. Asked where the impact lies, teams did not prioritize performing maintenance more efficiently. They prioritized what might generate less of it: tests durable enough to survive the application changes or more people capable of building them.

That distinction changes how a testing investment should be evaluated. Coverage counts and test-generation speed measure how much automation a team can produce. Resilience measures how much of it survives contact with a moving application. The poll suggests software testers are asking the second question.

Pressure from a source still ramping up

Meanwhile, the amount of change hitting the testing function continues to rise. In the least surprising result of any polls we've run lately, 74% of respondents report that AI-assisted software development has increased pressure on their testing teams over the past 12 months; nearly half describe this as "significantly increased."

The timing of this finding is notable. Roughly a quarter of respondents are not reporting increased pressure on testing due to AI-assisted development, suggesting that it has not fully reached every organization. If the strain is already this tangible while the accelerant is only beginning to be applied, the trajectory is easy to extrapolate: even more pressure is coming for software testing.

From interest to operating model

Given the conditions, appetite for AI-powered software testing is unsurprising. Forty-two percent of respondents say their organizations are actively evaluating or adopting AI-powered testing; another 22% are interested but still assessing practical value and trust. The remaining third is split: 14% have not meaningfully explored it, 12% are focused on improving existing automation first, and 10% consider their current approaches sufficient.

That third is worth analyzing. "Fix the automation we have before adding AI to it" is a different position from "we don't need this," and together those two responses account for nearly a quarter of respondents—a constituency likely to evaluate AI-powered testing on whether it addresses the durability problem rather than on whether it is AI.

This leads directly to the rising end state for automation: autonomy. In our poll, strategic interest far outpaces current activity. Sixty-two percent describe continuous, largely autonomous testing—"lights-out" or "dark factory" testing—as either a critical strategic priority or important but early-stage. The gap between that 62% and the 42% acting on AI-powered testing today is where it will be decided whether testing becomes a lasting restraint on software delivery velocity.



The intent-execution space is also where the harder questions live. Autonomous testing is not primarily a tooling change. It implies decisions about who owns the process when tests generate, repair, and execute themselves; how exceptions are escalated; what evidence a release gate accepts when the underlying analysis was machine-produced; and what testers do with the capacity returned to them. The poll makes the strategic imperative clear. It also highlights the need to focus on operating models—and whether they're ready to absorb what autonomy demands.

Analyst View

The poll findings describe a testing function under substantial operating pressure—and experiencing it before the source of greatest concern, AI-driven coding, has fully arrived.

Testing organizations are not resisting automation; far from it. Most already use it, and most want more. They report that automation carries an ongoing cost that competes directly with the coverage they still need to achieve. Nearly half of teams spend more effort maintaining tests than creating them. Maintenance difficulty is the leading barrier to expanding automation—and therefore also to autonomous testing. No wonder seven in ten organizations still run at least half their testing manually.

If automation is difficult to sustain, progress in retiring manual work is clearly limited.

Layer AI-assisted development onto that condition, and the arithmetic worsens: more code, changing more often, produced by tools that are themselves still being adopted and therefore exhibiting unpredictable error rates. Most software test engineers already feel it. Script-based automation was designed for applications that changed on a release cadence rather than a daily one, and maintenance load scales with the rate of change. Testing is yet another area that must modernize with AI because of AI.

The most useful signal in the poll data may be the one about the reliability of automated testing. Offered a choice of improvements, test engineers pass over reducing maintenance in favor of reducing manual effort and improving the adaptability of automated tests. That is a preference for automation that can absorb change over automation that requires regular repair—and it sets a demanding bar for autonomous (AI-powered and agentic) approaches. Generating tests faster addresses the volume problem but not the durability problem, and durability is what testing teams identify as their primary automation issue.

The evaluation criteria for the next phase follow from that, even as the adoption path remains naturally unclear in a rapidly shifting testing landscape. Can an approach keep working as the way that applications change, changes? Can teams see and audit what an autonomous system decided and why? Can non-specialists use autonomous testing without introducing a new skills bottleneck? Do these tools integrate with the lifecycle management systems, pipelines, and packaged applications that testing organizations actually run against? Forty-two percent of respondents are already evaluating or adopting AI-powered testing, and 62% consider largely autonomous testing strategically important. Interest is not the variable to watch. Whether the technology answers the durability question at enterprise scale is.





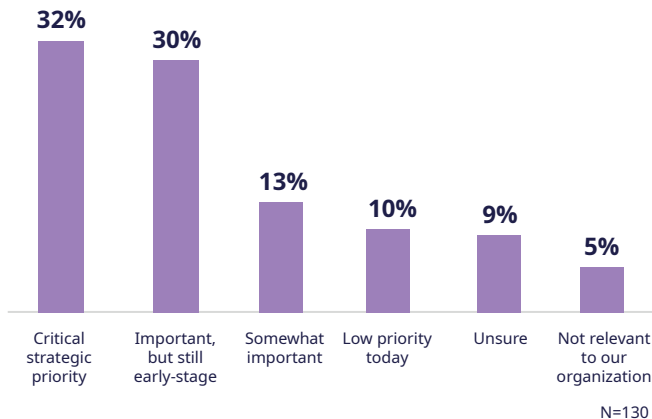
Recommendations

- **Establish a baseline for manual effort and maintenance load.** Measure how much testing remains manual, how much team capacity goes to maintaining existing tests, and which applications or environments generate the most upkeep. Without that baseline, modernization decisions rest on impressions rather than evidence, and improvements are difficult to attribute afterward.
- **Evaluate automation on durability, not volume.** Test counts and generation speed measure output. Resilience measures what survives an application change. Ask how a potential approach behaves when locators shift, workflows are reordered, or an underlying package is upgraded—and what share of failures it can resolve without human intervention.
- **Assess the degree to which testing can absorb faster development.** AI-assisted development is increasing demand for testing, even as adoption is still early. Identify where coverage, environments, test data, and review processes will bind first as change volume rises and treat those as capacity planning questions rather than tooling ones.
- **Introduce AI-assisted and agentic testing via defined use cases.** Start where manual effort or maintenance burden is highest and where outcomes can be measured cleanly. Controlled pilots reveal what the broader case cannot: reliability under real application change, how much human review is actually required, integration effort, and whether the value holds outside the demonstration.
- **Define governance and observability before autonomy expands.** Establish how AI-generated tests and machine-produced failure analysis will be reviewed, monitored, and audited. Teams need visibility into what was generated, what was executed, what was repaired, and on what basis—so that increased autonomy does not quietly reduce accountability at the release gate.
- **Treat autonomous testing as an operating-model change.** “Lights-out” testing requires more than new tooling. Process ownership, exception handling, escalation paths, integration across the delivery lifecycle, and the redefined role of testers all shift when systems can generate, maintain, and execute tests with limited intervention. Organizations that focus only on the tooling will, with certainty, discover the rest later.
- **Connect testing modernization with delivery outcomes.** Track release readiness, defect escape rates, coverage growth, maintenance effort, and cycle time. Clear outcome measures distinguish meaningful progress from experimentation and give teams a defensible basis for scaling—or stopping.

Research Results

How important is the ability to execute continuous, largely autonomous software testing with minimal human intervention (“lights-out” or “dark factory” testing) to your organization’s future software delivery strategy? (Select one)

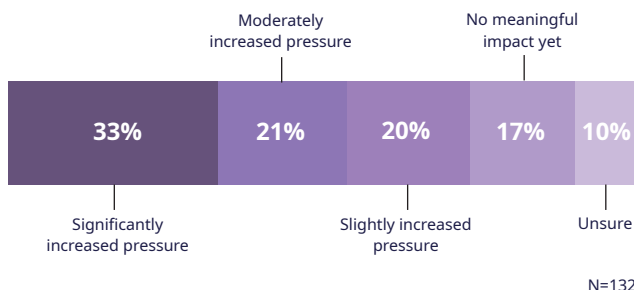
Strategic Importance of Autonomous Testing



Continuous, largely autonomous testing is emerging as a recognized component of future delivery strategy. Sixty-two percent of respondents consider it either a critical strategic priority (32%) or important but still early-stage (30%), and 75% regard it as at least somewhat important. Only 5% call it not relevant. The near-even split between “critical” and “important but early-stage” is the useful detail: strategic interest is broad, but a substantial share of that interest is directed at a capability respondents do not yet consider mature. Read alongside the finding on current AI-powered testing posture, this suggests that broad strategic interest in greater testing autonomy is running ahead of active engagement with AI-powered approaches that would enable it.

How much has AI-assisted software development increased the pressure on your testing teams over the past 12 months? (Select one)

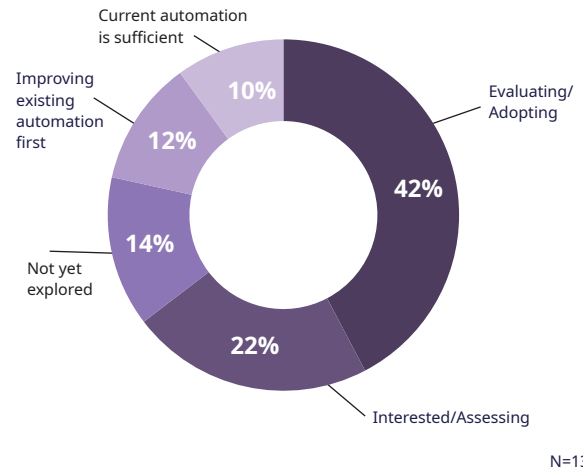
Impact of AI-Assisted Development on Testing Pressure



AI-assisted software development is increasing pressure on testing organizations. Nearly three-quarters of respondents (73%) report at least some increase, and 54% describe it as moderate or significant. Roughly a quarter report no meaningful impact yet or are unsure, indicating that the effect is real but not yet evenly distributed. The findings suggest testing processes may struggle to keep pace with faster software change—generating, updating, and executing coverage at the rate the code now demands.

Which statement best reflects your organization’s current perspective on AI-powered or agentic software testing? (Select one)

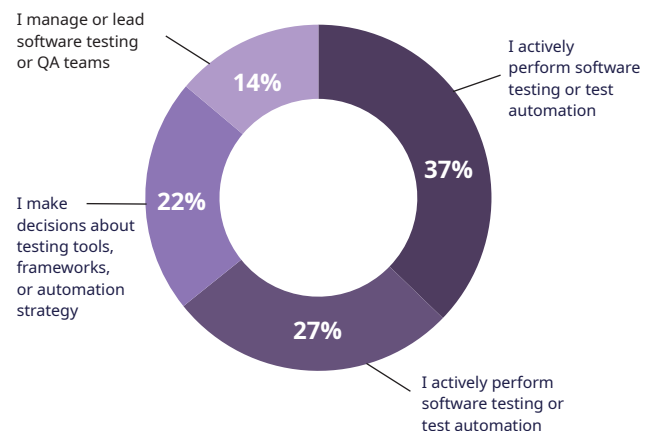
Current AI-Powered Testing Posture



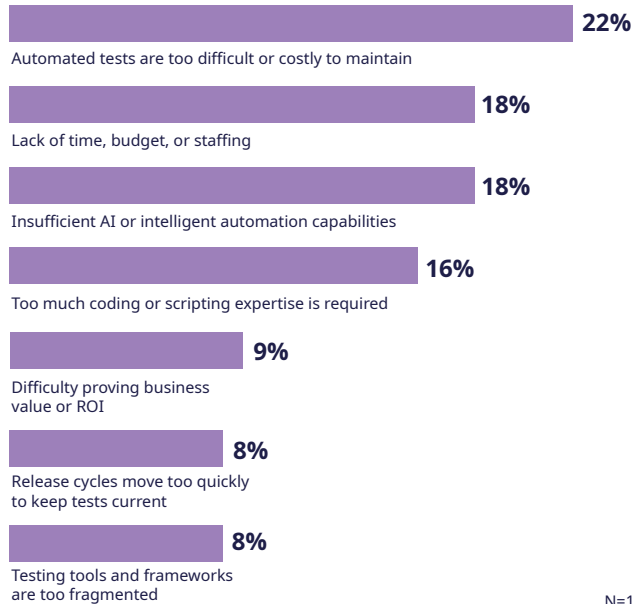
Engagement with AI-powered testing is substantial, though maturity varies widely. Forty-two percent of respondents are actively evaluating or adopting AI-powered testing approaches, and another 22% are interested but still assessing practical value and trust—together, nearly two-thirds are engaged with the opportunity. The remaining third divides into meaningfully different positions: 14% have not explored it, 12% are prioritizing improvements to existing automation first, and 10% consider current approaches sufficient. The middle of those three is less a rejection than a sequencing decision, and it points to a segment that will judge AI-powered testing on whether it resolves existing automation problems.

Which of the following best describes your involvement in software testing or test automation at your organization?

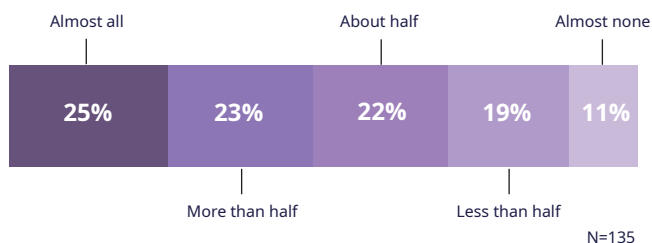
Respondent Testing Involvement



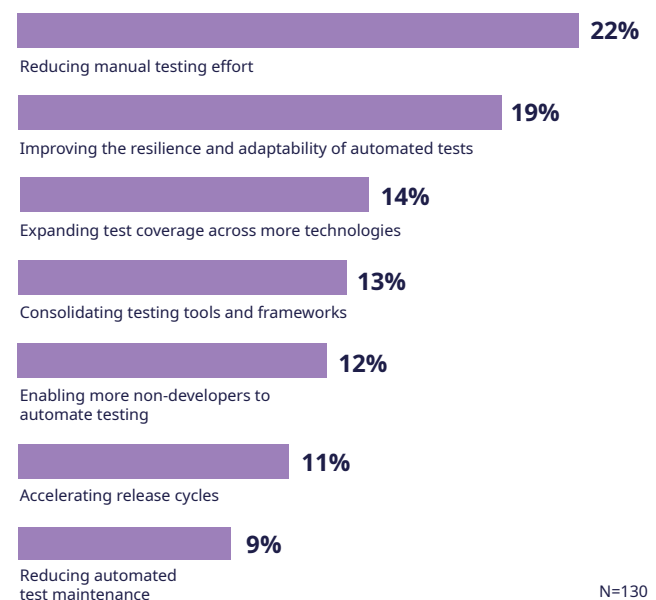
Among the 137 qualified respondents, 37% actively perform software testing or test automation, and another 27% combine hands-on testing with responsibility for tooling and strategy decisions. The remainder either make testing technology and automation decisions (22%) or manage testing and QA teams (14%). Nearly two-thirds of the sample therefore do the work directly, which is worth keeping in view when reading the findings that follow: these are first-hand accounts of testing conditions rather than second-hand estimates.

What is the biggest obstacle preventing your organization from expanding software test automation and adopting agentic testing? (Select one)**Primary Barrier to Expanding Test Automation**

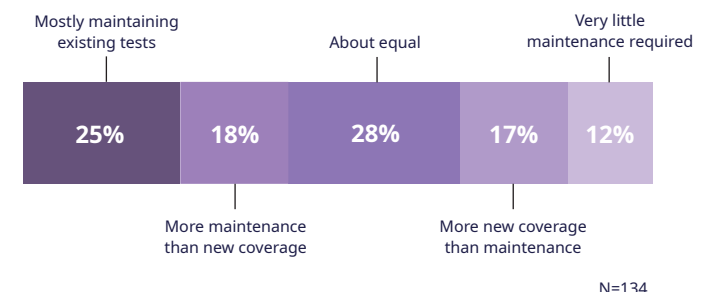
Maintenance difficulty and cost is the most frequently cited obstacle to expanding test automation and adopting agentic testing, selected by 22% of respondents. Resource constraints and insufficient AI or intelligent automation capabilities follow at 18% each, with the coding and scripting expertise requirement at 16%. Two features of this distribution stand out. First, the top four responses fall within six points of one another, indicating cumulative friction rather than a single dominant blocker. Second, operational constraints—maintenance burden plus limited time, budget, or staffing—account for 40% of responses, making them at least as consequential as technology readiness.

How much of your organization's software testing is still performed manually today? (Select one)**Extent of Manual Testing**

Manual effort remains a substantial share of enterprise software testing. Nearly seven in ten respondents (70%) report that at least half of their organization's testing is still performed manually, including 48% who say more than half remains manual. Only 11% describe their testing as almost entirely automated. The distribution is notably flat—no single response captures more than a quarter of the sample—suggesting a wide variation in how organizations balance manual and automated testing rather than a common norm.

Which improvement would create the greatest business impact for your organization over the next 12 months? (Select one)**Top Testing Improvement for Business Impact**

Respondents expect the greatest near-term business impact from reducing manual testing effort (22%) and improving the resilience and adaptability of automated tests (19%)—42% between them. Reducing automated test maintenance ranks last, at 9%, despite maintenance difficulty being the most-cited barrier in Q4. The apparent contradiction is instructive: respondents appear to distinguish between the maintenance work and the fragility that generates it, favoring automation that withstands change over faster ways to repair automation that doesn't. Read alongside the 12% who selected enabling more non-developers to automate testing, 54% chose an improvement oriented toward making automation easier to sustain or extend.

How much effort does your team spend maintaining existing automated tests compared to creating new test coverage? (Select one)**Automated Test Maintenance Burden**

Maintaining existing automated tests consumes significant team capacity. Forty-three percent of respondents spend more effort on maintenance than on creating new coverage, compared with 29% who devote more effort to expanding coverage—a figure that includes the 12% reporting very little maintenance at all. Another 28% describe the effort as roughly equal. Taken together, 71% spend at least as much effort maintaining automation as extending it, which may constrain teams' ability to reduce the manual baseline.



This Techstrong PulseMeter is sponsored by UiPath.
To learn more about UiPath, visit uipath.com.