Software testers — your job isn't disappearing. The grind is.

Meet the dark testing factory.

UiPath

Content

4 The old model cannot absorb the new velocity.

5 Testing evolves from something you run to something that runs.

6 The real problem is wasted human judgment.

6 The factory removes the grind, not the tester.

7 From tactical executors to strategic quality leaders.

8 Quality leadership is not a title. It is a responsibility.

9 The more autonomy scales, the more judgment matters.

10 Tacit knowledge is the knowledge that is not in the ticket.

11 AI cannot act on what it cannot access.

11 AI can execute intent. It cannot originate it.

12 A dark testing factory does not build itself.

13 Deep thinking matters. Choosing to think matters more.

14 Effortful thinking is not extra work. It is the work.

15 AI cannot own what humans are accountable for.

16 The factory can run in the dark. Accountability cannot.

17 You do not run every test. You lead the system that does.

19 The promotion is operational, not symbolic.

20 The factory turns testing labor into quality leverage.

21 The darker the factory gets, the more visible testers become.

21 The more the lights dim, the more leadership matters.

23 The future is not fewer testers. It is testers with more leverage.

24 That new operating model is the dark testing factory.

Software testers are used to pressure.

Every new feature, every urgent fix, every integration, every configuration change eventually lands on the tester's desk with the same hard need: **evidence.**

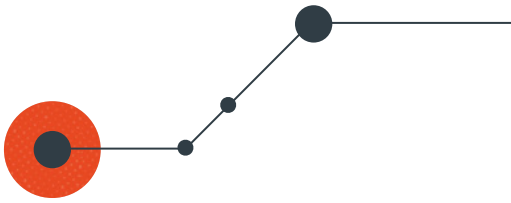
Not a guarantee. Not a release decision. Evidence the business can use to understand what is known about the quality of the software, what is still uncertain, where the risk sits, and whether moving forward is a responsible trade-off. Testers usually do not own the release decision.

The business owns that trade-off. But testers own something critical:

the quality of the evidence,
the clarity of the risk, and
the honesty of the confidence they provide.

That responsibility has never been easy to carry.

The old model cannot absorb the new velocity.



Testing has always needed more time than it gets to collect, interpret, and explain the evidence the business needs. But AI is making the pressure harder to ignore.

Software developers now create more code faster than ever. That code changes more often. More code plus more change means more risk. And that risk does not disappear just because software development got faster. It moves downstream: straight into software testing.

So, if you are a tester and you feel that your job is changing too fast under your feet, you are not imagining it. Faros AI calls this the **acceleration whiplash:** development is accelerating faster than the quality system around it. We are creating risk faster than we can test it.

That means AI is not only changing how software is developed. It is also changing what testing has to absorb. And as agents start operating in areas that used to depend almost entirely on human testers, it is natural for testers to ask, “Well...will an agent replace me?”

But we think that is the wrong question. The better question is: What testing work should agents handle, and what testing work becomes more important for testers to own?

That is where the dark testing factory comes in: A self-operating system for quality engineering, guided by human leadership, to enable organizations to ship value fast with confidence.

Six traits that make a dark factory.

Authorable by coding agents and accelerated by templatized agents to start fast

Humans above the loop

Humans set policy, resolve ambiguity, supply missing knowledge, intervene on exceptions, but don’t push buttons

Always-on, in-flow ops

The factory runs continuously, and is embedded deeply into the broader flow of software delivery

Closed, non-linear system

It’s not a pipeline, but a living loop; a failed test may trigger test repair, data reset, defect filing, human escalation

Risk-driven scheduling

The factory decides what to test next based on change impact, production telemetry, business risk, and policy

Self-improving system

Human decisions become reusable knowledge; the factory improves through every human intervention

Fixed-cost scaling

Repeatable effort becomes reusable knowledge, so deeper coverage does not require proportional headcount

The dark factory turns software testing from labor that repeats into software that scales

The old model cannot absorb the new velocity. (continued)

In a dark testing factory, agents and automations discover, design, automate, execute, analyze, and maintain tests. They respond to change, assess risk, prepare data and environments, run tests, manage defects, and improve the test portfolio over time.

That does not mean humanless testing. It does not mean zero-touch testing. It does not mean testers disappear. Quite the opposite. The dark testing factory aims to free testers from the routine operating loop so they can move into the leadership loop.

For a deeper dive into the dark testing factory itself, this **white paper** is the place to go. Here, we want to focus on what it means for software testers.

Testing evolves from something you run to something that runs.

For years, we have talked about automation as a way to reduce “manual testing”. That is still true, but we should be precise about what we mean by “manual” here.

We do not mean skilled exploration, investigation, experimentation, or all the critical thinking that gives those activities their value. We mean the repetitive and dreary, labor intensive work where humans are effectively asked to behave like machines.

And that is the bigger waste. Not that testers click buttons to confirm known risks. The bigger waste is that testers spend too much time on work that does not deserve or need their judgment and their time.

The real problem is wasted human judgment.

Rerunning failed automated checks. Chasing unstable test environments. Rebuilding test data sets. Updating scripts after UI changes. Connecting requirements, test cases, and test results. Filing defects by hand. Creating status reports. Repeating the same pattern for the tenth time.

That work matters, but it doesn't always require a skilled tester to perform every step. And when testers spend their days buried in administrative busywork, or performing work that machines are better suited to do, the organization loses the very thing testers are best positioned to provide: critical judgment under uncertainty.

The factory removes the grind, not the tester.

Testing has never been just executing mechanical checks repetitively. It has always been skilled investigation, exploration, and experimentation. It is the ability to ask better questions, identify hidden risks, challenge weak assumptions, interpret ambiguous signals, understand business impact, and explain what we know and what we do not know yet.

That type of testing work becomes more important in an AI-accelerated world, not less. Therefore, the dark testing factory is designed to absorb the time-heavy tactical work so testers can focus on the work that shapes quality more strategically.

From tactical executors to strategic quality leaders.

In the old model of software testing, testers were often treated as the last line of defense. Work moved through development, and then testing was expected to catch up.

That model has been cracking for years. AI is simply making the cracks impossible to ignore. With AI accelerating development, it becomes unmanageable. The dark testing factory changes that by moving testers from reactive validation to proactive quality leadership.

That sounds good, but it needs to be concrete. Quality leadership is not a vague title upgrade. It means taking responsibility for the decisions that determine how quality is achieved.

From tactical executors to strategic leaders.

What quality leadership means.		Why the dark factory enables it.
Defining quality goals	**Setting risk profiles**	1 Leadership can't be delegated to agentic AI
Handling exceptions	**Designing guardrails**	2 It requires drive, tacit judgment, accountability
Supplying knowledge	**Clarifying ambiguities**	3 The factory frees testers from tactical execution
Deciding trade-offs	**Prioritizing efforts**	4 Their focus shifts to leadership responsibilities
Owning outcomes	**Being accountable**	5 This enables them to shape quality strategically

Testers become quality leaders, leading autonomous agents & shaping quality outcomes instead of being buried in tactical execution

Quality leadership is not a title. It is a responsibility.

It means defining quality goals. What does "good enough" mean for this release, this customer, this process? A payment workflow, a healthcare record, an internal dashboard, and a marketing landing page should not all be judged by the same standard.

It means setting risk profiles. Which parts of the software application carry the highest business risk? Where would failure hurt customers, partners, revenue, compliance, operations? Which changes deserve deep testing, and which can move through lighter checks?

It means designing guardrails. What can the dark testing factory do on its own? When must it ask for approval? Which defects can it file automatically? Which environment or automation fixes can it trigger? Which changes require human review before execution continues?

It means resolving ambiguity. Requirements are rarely complete. Acceptance criteria are often partial. Product intent typically lives across tickets, conversations, design decisions, customer escalations, and production history. Someone must notice when the story is not testable, when the expected behavior is unclear, or when two sources of truth disagree.

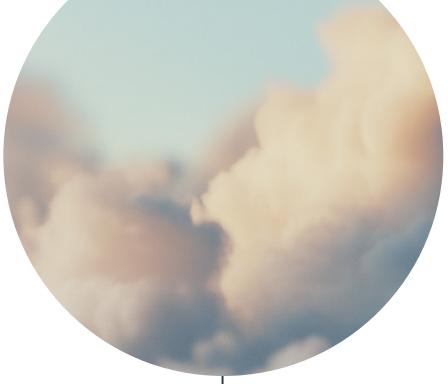
It means supplying missing knowledge. The dark testing factory can only reason over what it can access, trust, and interpret. When testers clarify a business rule, approve an exception, define a better oracle, or explain why a certain failure matters, they are not just helping one test run. They are teaching the factory.

It means deciding trade-offs. We never test everything. We choose. We prioritize. Testing always happens within constraints: the time available, the scope we choose to cover, and the cost we are willing to invest. We accept some risks and reject others. The more autonomous the execution layer becomes, the more important it is that humans define which trade-offs are acceptable, which are not, and when the factory must escalate instead of continue.

Because in the end, testing is not about removing all risk. It never was. In fact, testing does not remove risk on its own. It makes risk visible, understandable, and discussable, so the business can decide how much risk it is willing to leave in the system, where that risk sits, and whether the available evidence is strong enough to move forward.

And quality leadership also means owning outcomes.

A green dashboard is not the same as confidence. A passed test set is not the same as release readiness. Quality leadership means being able to explain, with evidence, what confidence the team has, what uncertainty remains, and what risks the business would need to accept by moving forward.



The more autonomy scales, the more judgment matters.

Those are not execution tasks. Those are leadership responsibilities. It is tempting to think that if agents can design, automate, execute, analyze, and maintain tests, then eventually they can own the totality of software testing. That is where we need to be careful.

AI can do a lot. It can process large volumes of information, find patterns, generate useful test ideas, execute workflows, summarize failures, suggest fixes, and keep repetitive work moving at a scale humans cannot match.

But leadership in testing depends on at least three things AI cannot absorb on its own: judgment grounded in tacit knowledge, accountability for the consequences of decisions, and the human drive to pursue quality when the work is ambiguous, inconvenient, or unfinished.

The darker the factory, the more humans matter.

What AI won't pursue	What AI can't see	What AI can't own
AI can execute intent, not originate it	AI can't act on what it can't access	AI cannot be accountable for us
Human drive	**Tacit judgment**	**Accountability**
The will to start, find out, and improve when "good enough" is not enough	The context behind context: hidden rules, customer realities, & risks only experience reveals	The responsibility to decide which risks the business can accept & which it must reject
What is still not good enough?	What context is missing?	Who is answerable?

Tacit knowledge is the knowledge that is not in the ticket.

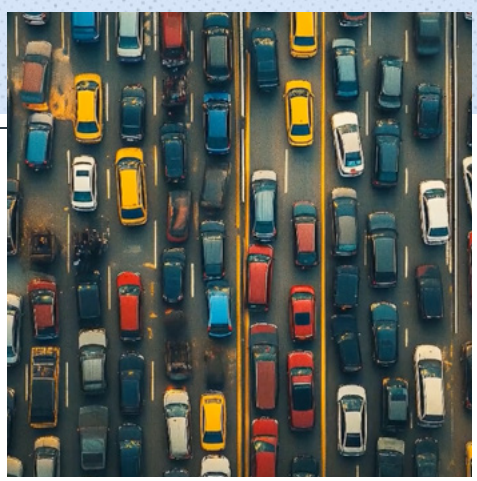
Tacit knowledge is the knowledge we use before we can fully write it down.

It is the context behind the context. It is the rule everyone follows but nobody documented. The workaround support teams know by heart. The customer-specific exception buried in years of operational memory. The production behavior that no longer matches the original design. The good-old “this looks harmless, but it broke billing last quarter” instinct. The compliance nuance that never appears in the acceptance criteria.

It is knowing the difference between what a software application technically permits and what the business can safely accept. A software application may allow a 100% discount, a payout without second approval, or a transaction with a missing compliance check. But just because the software application allows it does not mean the business can tolerate it.

A tester’s value often lives exactly there.

Not because testers magically know everything, but because professional testers are trained to look for what is missing. They notice gaps between the requirement and the real-world use. They recognize when a “simple change” touches a sensitive process. They question whether the oracle is good enough. They ask who could be harmed, what could be misunderstood, what could fail silently, and what evidence would actually build confidence.



AI cannot act on what it cannot access.

AI can infer. It can suggest. It can ask. It can help structure and reuse what humans provide. But it cannot be expected to know the unwritten rules of your business, your systems, your politics, your release history, and your risk appetite unless humans surface that knowledge and turn it into something the factory can use.

That is why the human role does not shrink as the factory becomes darker. It changes shape.

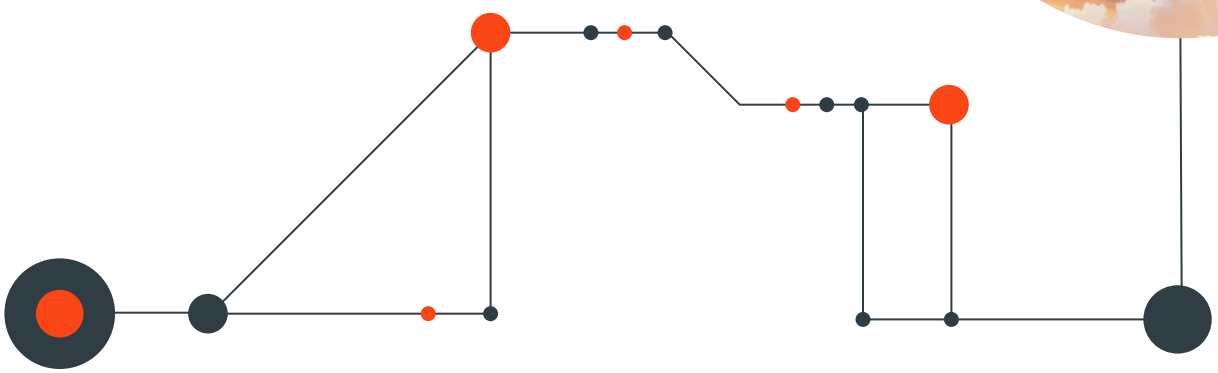
Every time a tester clarifies an ambiguous requirement, improves a test oracle, labels a false positive, explains why a defect matters, or defines an escalation rule, they turn tacit knowledge into explicit knowledge. They turn their understanding into reusable factory knowledge.

That is how the factory gets smarter and darker over time. Not because humans disappear, but because human judgment becomes part of the system.

AI can execute intent. It cannot originate it.

There is another human capability we should not overlook: **drive.**

Not drive in the motivational-poster sense. Not generic passion. We mean the will to find out. The willingness to pursue quality before the work has been fully defined. The willingness to ask the awkward question, follow the weak signal, challenge the comfortable answer or assumption, and keep investigating when the dashboard is green.



A dark testing factory does not build itself.

The dark testing factory comes with default guardrails, reusable quality objectives, specialized agents, and risk profiles. Those things matter. They give the factory structure. But defaults are not strategy. They have to be grounded in context before they become meaningful.

Someone has to decide what quality means here: For this product. For this customer. For this process. For this release. For this regulatory environment. For this business.

That grounding does not happen automatically.

Someone has to decide that the current testing approach is not good enough. Someone has to care enough to turn scattered knowledge into usable knowledge. Someone has to set the first guardrails, review the first outputs, correct the first wrong assumptions, and keep improving the dark factory when the easy thing would be to accept whatever the automation produced.

AI can help enormously once that intent exists. It can generate test ideas, execute workflows, summarize results, detect failure patterns, propose repairs, and keep the factory moving.

But it does not wake up with the desire to test. It does not decide that a vague user story is unacceptable. It does not feel the tension between release pressure and unresolved risk. It does not insist that “probably fine” is not good enough when the impact is high.

That drive still has to come from people.

And there is something important hidden inside that drive. Intelligence in software testing is not only the ability to think well. It is the willingness to think hard, often, especially when thinking hard is inconvenient. That distinction matters.

Deep thinking matters. Choosing to think matters more.

Many people can reason carefully when the problem is clear, the question is neatly framed, and someone has asked them to solve it. But testing rarely works like that. In testing, the signal is often weak. The requirement is often vague. The deadline is loud. The dashboard may already be green. The easy thing is to accept the surface answer and move on.

The tester's advantage often comes from choosing not to.

Think about running. A strong runner gets around faster than someone else does partly because they can run faster. But the bigger difference is that they choose to run in moments where another person would probably walk. Their advantage is not only running speed. It is the habit of using that speed.

Testing works in a similar way.

The strongest testers are not valuable only because they are capable of deep thinking. They are valuable because they choose to use that capability more often. They choose to slow down a comfortable conclusion. They choose to question the evidence. They choose to follow the weak signal. They choose to think when it would be easier to move on.

And the more often a tester thinks deeply, the sharper that thinking becomes. The better they get at it, the more useful it feels. And the more useful it feels, the more likely they are to do it again.

Over time, effortful thinking stops being an occasional heroic act and becomes a professional habit.

Effortful thinking is not extra work. It is the work.

That is hopeful. Because while raw cognitive capability may differ from person to person, the willingness to think hard more often is something testers can practice. You can build the habit of not accepting the first answer. You can learn to enjoy the awkward question. You can train yourself to follow uncertainty instead of avoiding it.

The dark testing factory can execute, analyze, summarize, and recommend. But it will not choose to care more than the goal it was given. It will not decide on its own that a vague requirement deserves another question, that a weak signal deserves another look, or that a comfortable answer is not yet good enough. That is why drive matters.

In testing, the advantage does not come only from being able to think. It comes from being willing to think when thinking is the work.

For example, no one told us to write this article. No agent woke up concerned that the role of testers in the dark testing factory might be misunderstood. We did. AI helped us draft, challenge, refine, and improve the argument. But the concern that this distinction matters did not come from AI. It came from a human sense that something important needed to be said.

Testing works the same way since testing is not only a set of mechanical tasks. It is a pursuit.

It's a pursuit of quality-related information. The dark testing factory can accelerate that pursuit, but it needs humans to aim it. Without human drive, the factory may become efficient at doing whatever it was told to do. With human drive, it becomes a system that learns what matters, improves how it tests, and helps the organization make better decisions over time.

The factory can run in the dark. The will to test cannot.

AI cannot own what humans are accountable for.

There is another reason quality leadership cannot be handed over to AI: **accountability.**

- Not because human testers never make mistakes. They do.
- And not because an agent cannot explain what it did. It can.

The point is: when something goes wrong, the business cannot stop at “the agent decided”. The business still needs to know who owned the risk, who had the authority to accept it, whether the evidence was strong enough, whether the right guardrails were in place, and what must change so the same failure does not happen again.

That is why accountability matters. It turns decisions into something the business can govern. Without accountability, a bad release becomes a technical accident. With accountability, it becomes a decision the organization can inspect, learn from, and improve. So, accountability matters because the business needs decisions it can govern, not just actions it can explain.

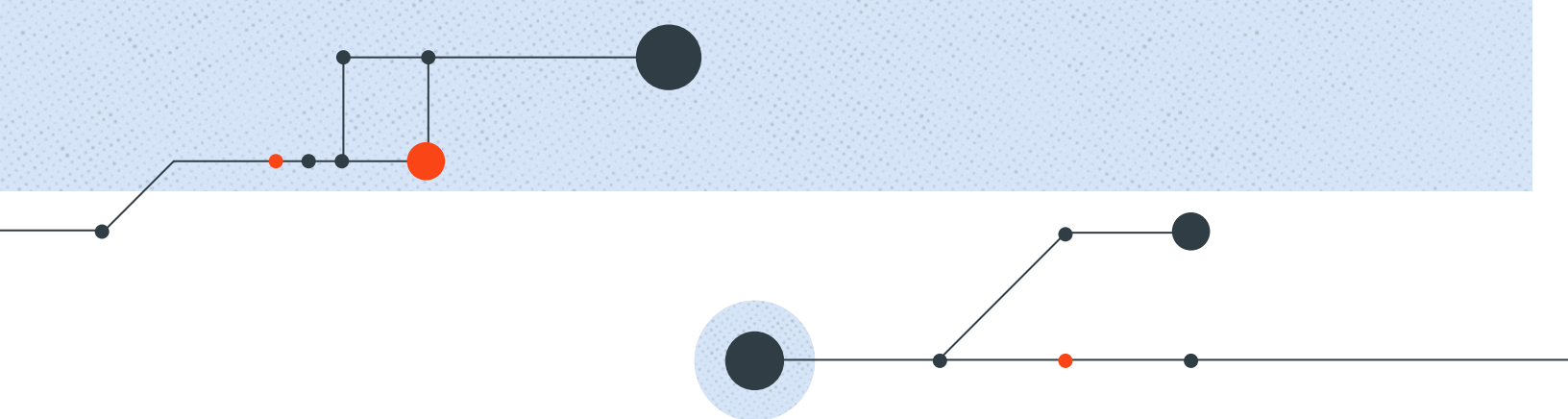
An agent can support that decision. It can collect evidence, summarize risk, and recommend a path forward. But it cannot be answerable for the consequences. It cannot own the trade-off between moving fast and protecting customers, revenue, compliance, and trust. Humans do.

And this brings us right back to the pressure testers have always carried. Testers usually do not own the release decision. The business owns that trade-off. But testers own the evidence the decision rests on. They are accountable for whether that evidence is strong enough, whether the risks are clear enough, and whether the confidence being communicated is honest enough.

In an AI-accelerated world, that accountability becomes more important, not less.

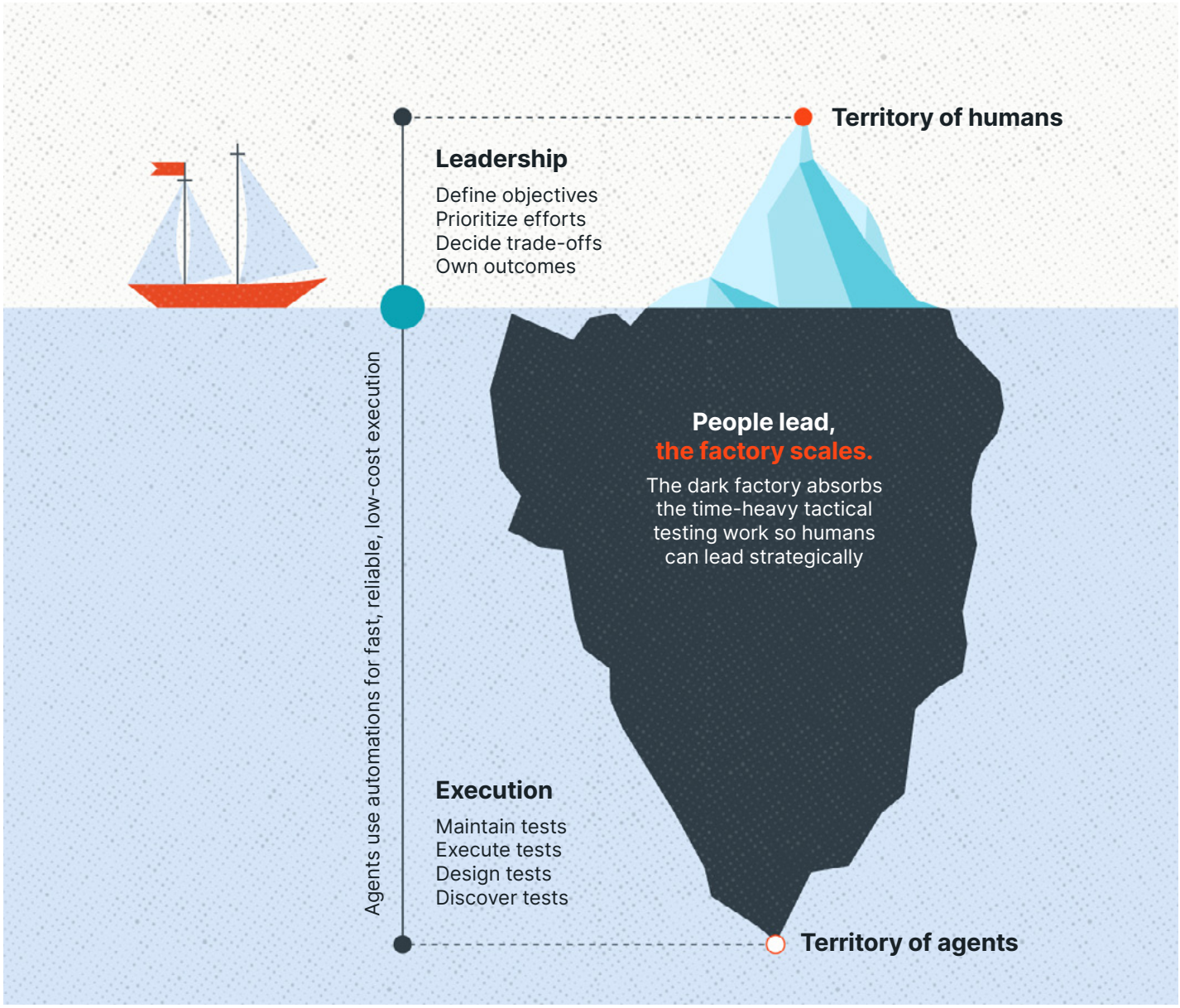
The more autonomous the factory becomes, the more important this accountability becomes.

Why? Because autonomous systems scale outcomes. That is their power. But they can also scale the wrong outcomes if the goals, guardrails, risk models, or assumptions are wrong.



The factory can run in the dark. Accountability cannot.

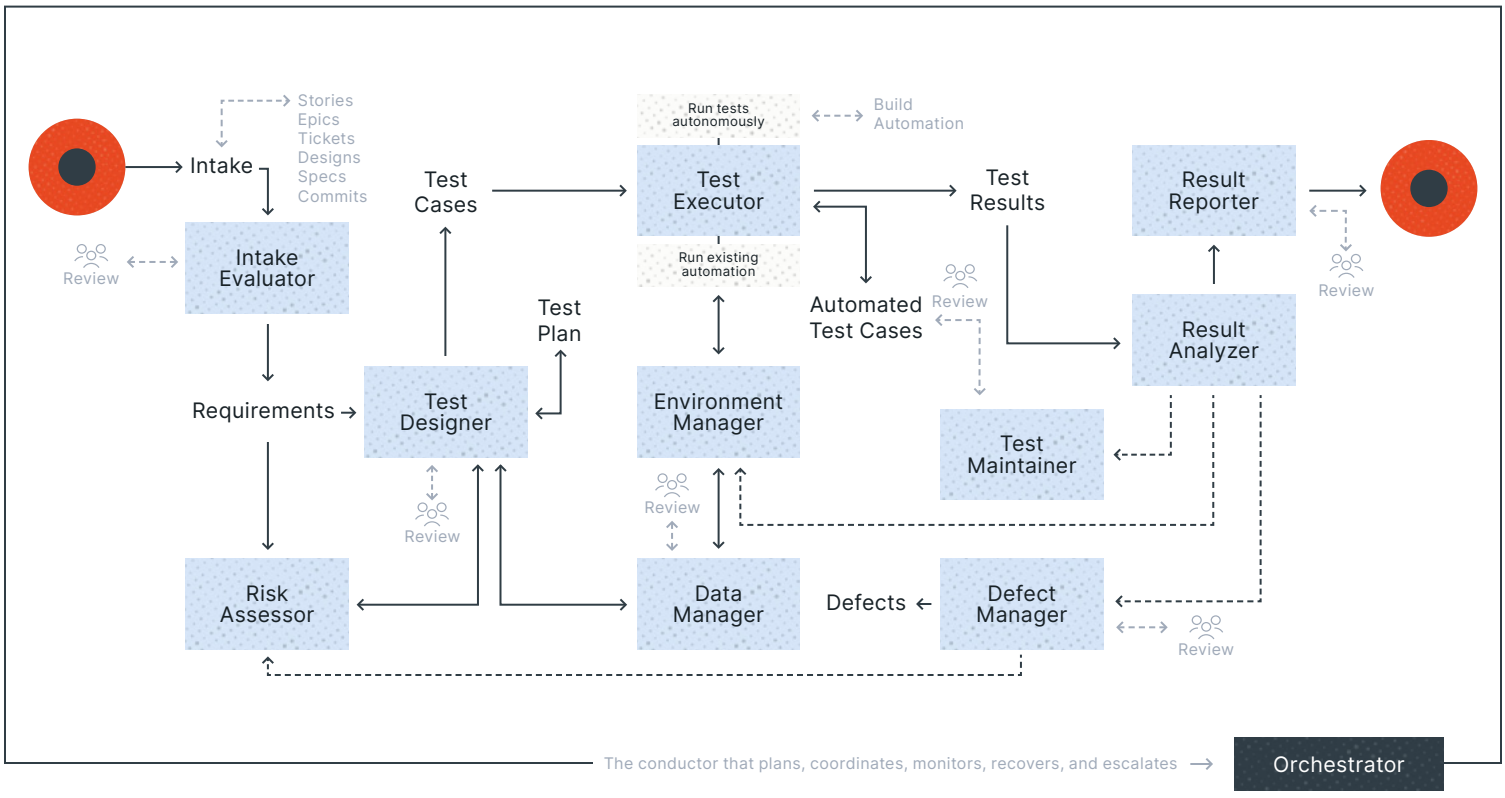
A dark testing factory that runs fast without any human leadership does not create confidence. It creates higher velocity but not safer delivery. Therefore, the dark testing factory is not about replacing human judgment. It is about giving human judgment more leverage, so autonomous systems can move faster without turning risk into something nobody truly owns.



You do not run every test. You lead the system that does.

In a dark testing factory, testers do not lead by manually touching every single test. They lead by shaping the system that performs the tactical testing work.

That means software testers become the people who guide a fleet of specialized agents: intake evaluators, risk assessors, test designers, test data managers, test environment managers, test executors, test result analyzers, defect managers, test maintainers, and reporters.



You do not run every test. You lead the system that does. (continued)

The tester's job is no longer to personally perform every action in the loop. The tester's job is to make sure the loop is worth trusting. That includes asking questions like:

- What signals should actually trigger testing?
- What risks matter most for this product area?
- What evidence do we need for release decisions?
- Where is the factory allowed to act independently?
- Where do we need human review and approval?
- What knowledge is missing from the system?
- Which failures should trigger repair, rerun, or defect filing?
- Which test results are useful, and which are noise?
- Which tests are outdated, redundant, or misleading?
- Which business rules need to become explicit?
- Which decisions should become reusable knowledge?

This is leadership work. It is also deeply testing work.

The tester is still investigating risks. Still questioning assumptions. Still evaluating evidence. Still helping the organization make better-informed business decisions such as shipping and/or fixing decisions. But now the tester is doing it with leverage.

The promotion is operational, not symbolic.

We should be careful with the word “elevate” because it can sound like marketing gloss. In the dark testing factory, the elevation of the tester’s role is not symbolic. It is operational.

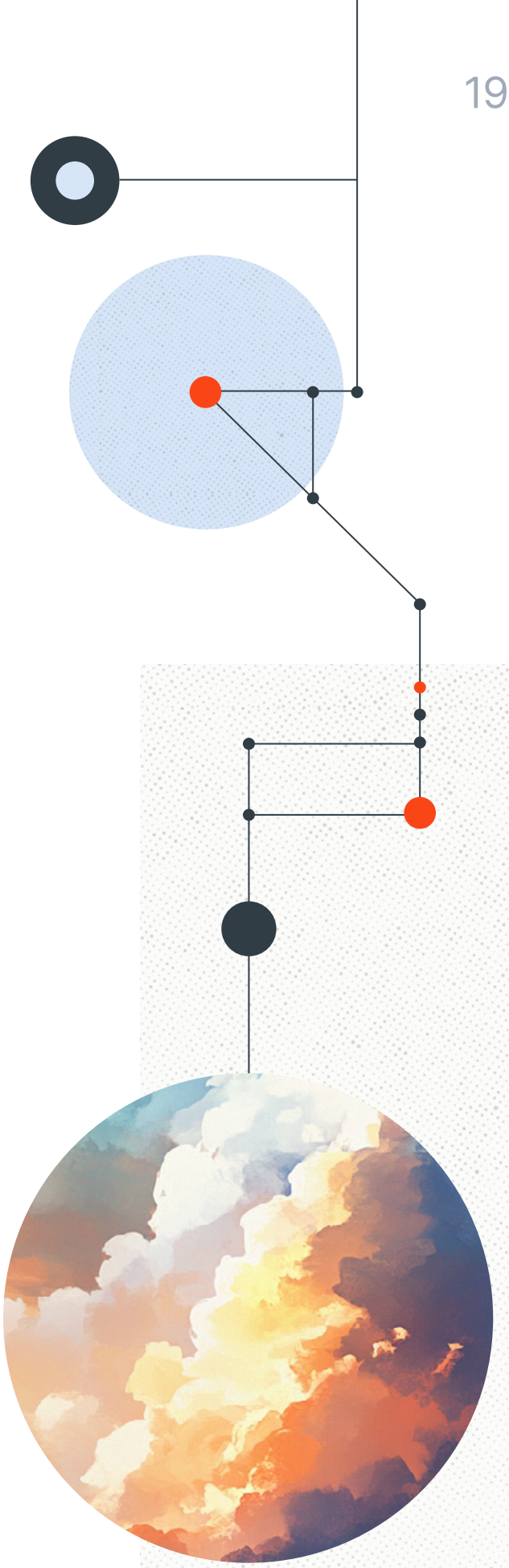
Today, too many testers are still measured by only how much work they can do, such as: How many tests performed. How many defects found. How many checks automated.

In a dark testing factory, this measure starts to shift. The value of a tester becomes visible in the quality of the system they help create. Better risk models. Better test oracles. Better guardrails. Better escalation rules. More coverage where it matters. Less wasted execution time. Fewer repeated mistakes. Faster evidence. More trustworthy release decisions.

That is a real promotion. Not necessarily a change in job title, but a change in influence.

The tester moves from being perceived as the bottleneck of the delivery process to becoming the person who helps the business move fast without flying blind.

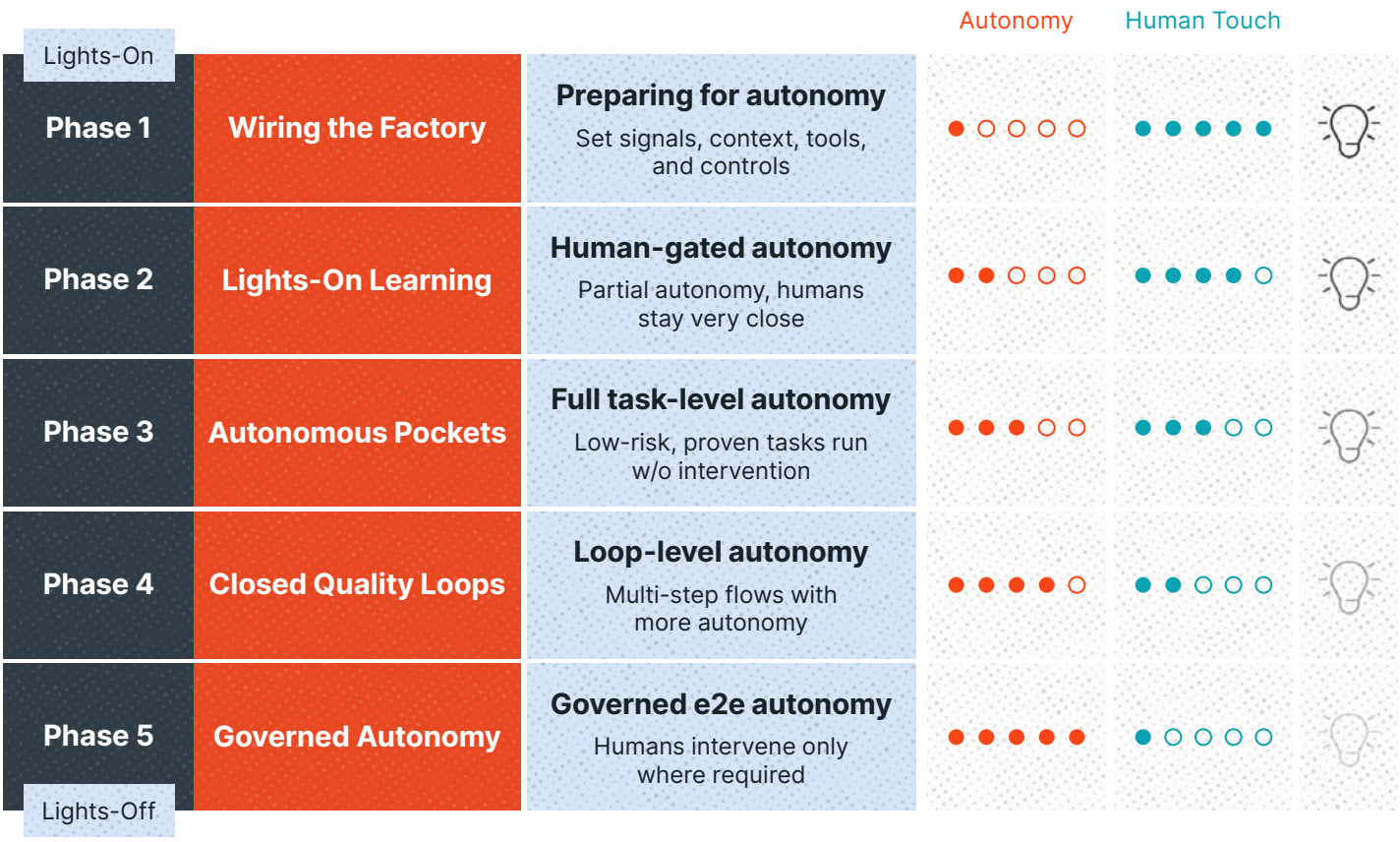
That matters, especially now. In an AI-accelerated delivery model, organizations will not slow down just because software testing is overloaded. The pressure will continue. The cost of poor software quality signals will rise. The demand for confidence, in terms of warranted belief under uncertainty, will increase.



The factory turns testing labor into quality leverage.

Therefore, testers need more than another tool that generates more tests. They need a system that gives them leverage, control, and visibility across all the testing work.

That is the promise of the dark testing factory. But that shift won't happen overnight. Nobody flips a switch and suddenly has a dark testing factory. It is earned step by step.



At first, humans stay very close. They review, approve, correct, and redirect. That is not a failure of autonomy. That is how autonomy learns safely. A dark testing factory is not built in one step. It is grown through trust, evidence, feedback, and gradually expanding levels of autonomy.

Over time, routine decisions become reusable knowledge. Low-risk tasks start to run without intervention. Multi-step flows become more autonomous. The factory learns when to proceed, when to repair, when to rerun, when to escalate, and when to ask for help.

The darker the factory gets, the more visible testers become.

As a tester, this means your day-to-day work starts to change. You spend less time executing the same steps manually and more time deciding which work should be delegated to AI.

You spend less time creating test reports from scratch and more time interpreting what the evidence means. You spend less time maintaining brittle automated tests and more time improving the knowledge that keeps the factory reliable.

You spend less time reacting at the end of the testing cycle and more time shaping confidence continuously. That does not make testing less skilled. It makes skilled testers more visible.

The more the lights dim, the more leadership matters.

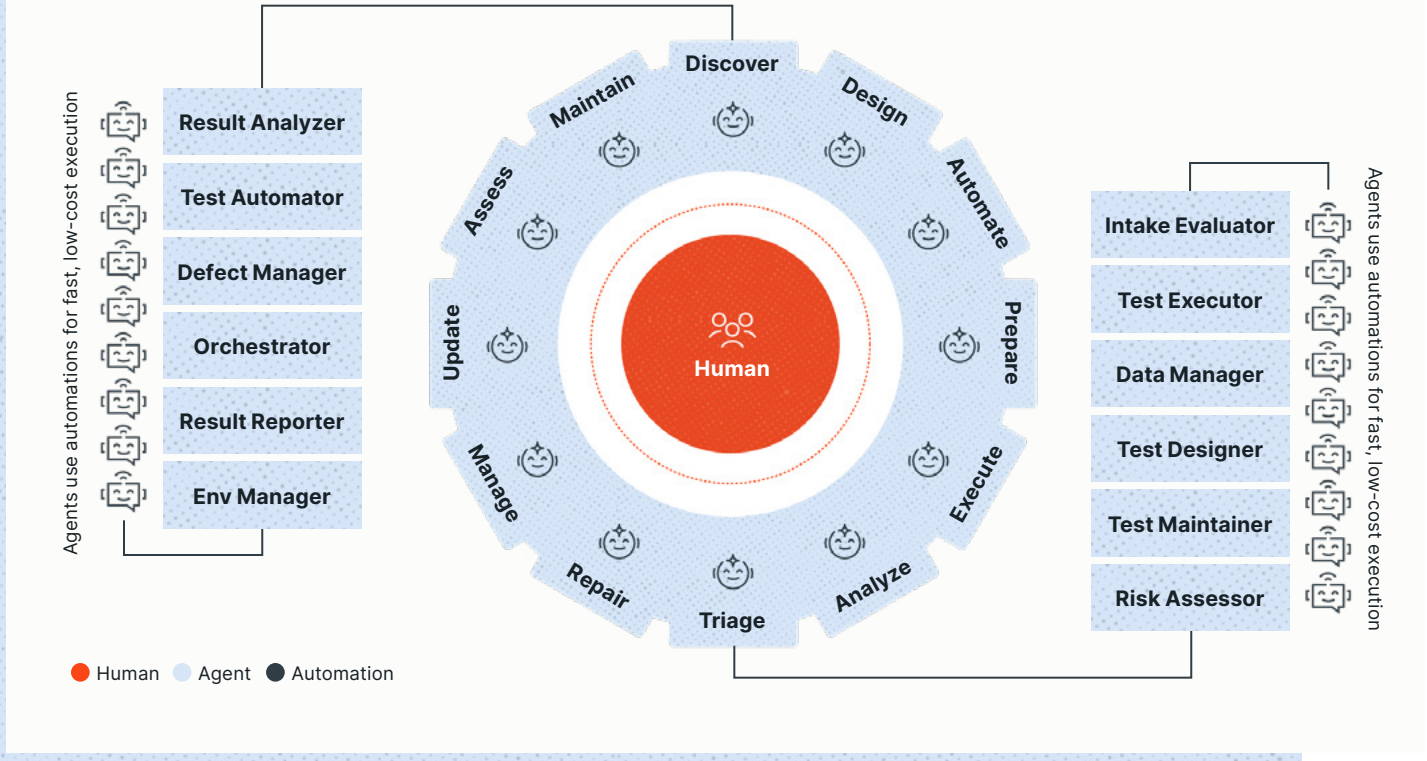
The best way to prepare for the dark testing factory is not to become less human. Quite the opposite. It is to become more deliberate about the unique human parts of testing.

Start by learning how to make tacit knowledge explicit. Capture the business rules that usually live in conversations. Document the exceptions. Name the risks. Improve the oracles. Write down why certain failures matter. Explain what makes a release safe or unsafe in your context.

Then learn to think in policies and guardrails. Where should autonomy be allowed, and where not? Where should it be constrained? Which autonomous decisions are reversible? Which are high-risk? Which approvals are necessary, and which are just habit?

Also, we would suggest getting comfortable leading agents. That does not only mean prompting them. It means reviewing their work critically, correcting their assumptions, feeding them better context, and deciding when their output is good enough to become part of the quality system.

The more the lights dim, the more leadership matters. (continued)



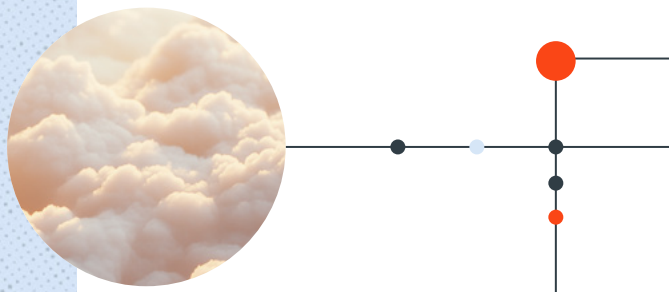
Finally, strengthen your ability to communicate the testing outcome. In the end, testing serves a decision. The organization needs to know whether it is ready to ship, or not. The dark factory gives you more evidence faster, but humans still have to make that evidence meaningful.

The future is not fewer testers. It is testers with more leverage.

The dark testing factory is not the story of software testers being automated away. It is the story of testers finally getting out from under the tactical grind that has consumed too much of their time for too long. The factory handles execution. Testers lead quality. That is the shift.

And that shift matters because the AI-accelerated world will not wait for traditional operating models in testing to catch up. Software delivery is moving faster. Risk is growing faster. The old operating model cannot absorb the new velocity.

That is why testing needs a new operating model: one that is always on, embedded in delivery, driven by risk, guided by human leadership, and able to improve with every cycle.



That new operating model is the dark testing factory.

But testers need to be central to that. Not as button-pushers. Not as last-minute gatekeepers. Not as people buried under endless regression cycles. As quality leaders.

As the people who bring judgment where AI alone cannot. As the people who understand ambiguity, risk, customer impact, and release readiness. As the people who teach the factory what matters, govern how it acts, and own the confidence it produces.

So, the darker the testing factory gets, the brighter the testing future can become. Not because humans disappear but because software testers finally get to lead.

About UiPath

UiPath Test Cloud is an AI-powered software testing solution that helps teams deliver higher-quality software faster and with greater confidence. Powered by the UiPath Platform, Test Cloud combines enterprise-ready test automation, test management, orchestration, and AI agents to accelerate every phase of the testing lifecycle, from test design and automation to execution, analysis, and reporting.

With support across web, mobile, desktop, API, SAP, and enterprise applications, Test Cloud enables organizations to modernize software testing at scale while improving coverage, reducing manual effort, and increasing release confidence. Built for the era of agentic software testing, Test Cloud brings together people, robots, automations, and AI agents in a governed, production-grade platform for continuous quality.

Learn more

UiPath