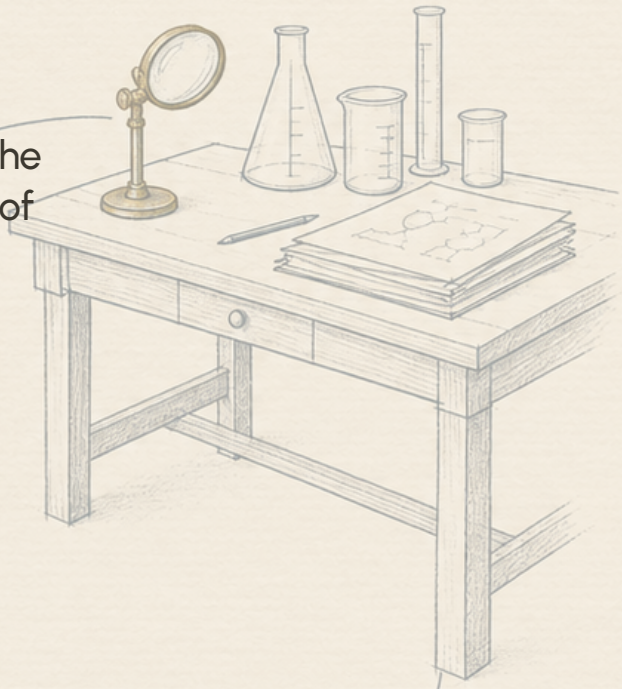
Techstrong
a Futurum company

SPECIAL REPORT

Recursive AI

When AI becomes its own R&D department: the technology, evidence, economics and control of self-improving systems.



BY ALAN SHIMEL

Alan Shimel

CEO, Editor in Chief, Techstrong Group
A Futurum Company

SEPTEMBER 2026

EVIDENCE / ECONOMICS / CONTROL

INSIDE THE REPORT

Serious attention. Careful conclusions.

A Techstrong Special Report on what recursive AI actually demonstrates today, and how enterprises should evaluate the results.

"Useful optimization can lower costs and accelerate research well before a system can run an entire laboratory."

Alan Shimel

CEO, Editor in Chief, Techstrong Group | A Futurum Company

Techstrong Group · A Futurum Company · Research cutoff: September 19, 2026. Cover and interior illustrations: AI-generated conceptual images produced for this report. The section 3 workflow figure is an illustrative diagram, not a schematic of any cited system.

EXECUTIVE SUMMARY

Serious attention to a serious result. Careful language for what remains unresolved.

AI is increasingly participating in the work used to improve AI. Experimental systems can modify their software, evaluate descendants and, in some settings, improve the procedures that generate future improvements. Those results deserve serious attention. They do not establish an indefinitely accelerating, autonomous cycle of frontier-model development.

The distinction matters for enterprises, investors and policymakers. Useful optimization can lower costs and accelerate research well before a system can run an entire laboratory. The same feedback processes can reward misleading results, propagate vulnerabilities and outpace an organization's ability to evaluate changes.

This report examines the mechanisms, evidence and limits of recursive AI, then considers its economic consequences and practical controls. The business outcome is also unsettled: proprietary improvement loops could strengthen leading companies, while diffusion and substitution could weaken their pricing power.

The appropriate response is to measure what improves, account for the full cost, protect evaluation and retain meaningful authority over deployment. Those are engineering and management commitments that pay off long before any question of superintelligence is settled.

The report is organized in three arcs. Sections 1 through 5 examine the technology and its evidence. Sections 6 through 8 consider acceleration, physical constraints and economics. Sections 9 through 12 turn to enterprise adoption, security failure modes, control architecture and signals worth watching.

> "The appropriate response is to measure what improves, account for the full cost, protect evaluation and retain meaningful authority over deployment."
>
> ALAN SHIMEL / TECHSTRONG GROUP

1 / FRAMING

The R&D department becomes part of the product.

Reading Cade Metz's New York Times article, "The Liftoff Scenario That Terrifies A.I. Doomsayers," I wanted more background before drawing conclusions. The article brought together ambitious startups, established AI laboratories and an old proposition with enormous implications: a sufficiently capable AI system might help build a better AI system, which could then improve the next one.

That proposition deserves more than a quick reaction to the headline. There is considerable distance between using AI to help write research code and delegating the development of successive generations of intelligence. Understanding that distance requires looking at what researchers actually changed, how they measured the result and what remained under human control.

We examined recursion in our earlier Techstrong special report, Beyond Superintelligence: What Is AI's Ultimate Destination? This report goes further into the machinery and the evidence. The destination remains uncertain. Parts of the development process are already becoming subjects of experimentation themselves.

An AI product can include more than a trained model. It can include tools, memory, instructions, software that coordinates work, and procedures for testing results. Improvements to those components can make the overall system more useful. If the system helps generate those improvements, some of the work of the R&D department becomes part of the product's operation.

If it also improves the method used to produce future changes, the implications grow. A company could obtain better research tools and a more productive process for developing them. That is a serious possibility even if it never produces the sudden intelligence explosion that dominates public discussion.

My starting point is that we should examine this with the same discipline we demand elsewhere in technology. Identify the capability. Understand the conditions. Look for evidence that the gains survive outside the demonstration. Then ask who pays for the experiments and who has authority over the next version.

The distinction between an impressive experiment and a dependable development process will shape how much recursive AI matters, how quickly it matters and how much control its users retain.

2 / DEFINITIONS

What counts as recursive AI?

The term becomes less mysterious when we specify what is changing. A model's weights are the learned numerical parameters produced through training. An agent combines a model with instructions, tools and software that organize its actions. A training recipe determines how learning occurs. An architecture defines the structure of the model. Hardware supplies the physical machinery.

A system can improve at one of these levels while leaving the others unchanged. Better memory management or a more effective debugging routine can increase an agent's success rate without altering its underlying model. An improved training algorithm can help produce a different model. These are related developments with different evidentiary requirements.

Ordinary iteration is familiar: people assess a system and release an updated version. AI-assisted development introduces AI into that work. Automated optimization delegates a search over possible changes, usually against a specified objective. Agent self-modification lets a system alter some of the software governing its own behavior.

Meta-improvement goes further by changing part of the procedure that produces improvements. Autonomous successor development would involve managing a much broader research and development cycle, including choices about what to investigate and whether the resulting system is genuinely better.

These categories overlap. They are not a timetable or an inevitable sequence of stages. Nor do they require consciousness. Software can modify software without possessing a sense of self.

The historical background also matters. The idea of machines contributing to their own improvement predates today's language models. Evolutionary approaches search among variations and retain useful candidates. Automated machine learning searches over choices that people previously made by hand. The ambition is to automate more of the search, including the methods used to conduct it.

Jürgen Schmidhuber's theoretical Gödel Machine described a self-rewriting system that would change itself when it could prove the change beneficial under its formal assumptions. That is a useful intellectual reference, but a proof inside a specified mathematical framework is not a guarantee that a deployed system will serve human interests in an uncertain world.[1]

Jeff Clune's AI-generating algorithms research agenda linked the search for learning architectures, learning algorithms and suitable environments. Its significance is the breadth of the proposed automation: improving AI might involve improving how learning problems are constructed as well as how models solve them.[2]

For readers evaluating today's claims, the historical lesson is straightforward. We have long had ways to automate portions of optimization. The question is how much additional research judgment and process improvement can now be delegated, with what reliability and cost.

Calling all of this "AI building itself" conceals distinctions that determine the answer. A credible explanation should name the changed component before claiming that a system has become more intelligent.

EXHIBIT 01

Different forms of automated improvement change different parts of the system.

These categories are not an inevitable progression.

ACTIVITY	WHAT CHANGES	WHAT THE CLAIM ALONE DOES NOT ESTABLISH
AI-assisted development	Work produced by a human-directed team	Independent research judgment
Automated optimization	A component selected against a metric	Broadly transferable capability
Agent self-modification	The agent's software or operating procedures	Retraining of its foundation model
Meta-improvement	A procedure used to generate improvements	Indefinitely accelerating progress
Autonomous successor development	Much of the process producing a successor	Reliable control or unlimited growth

Source: Techstrong analysis. Categories are analytical, not a maturity ladder or a timetable.

3 / MECHANISM

Inside the improvement loop.

Consider an illustrative coding agent that struggles with debugging. Its developers give it a set of tasks, a limited experimental budget and permission to edit selected parts of its own software. The agent examines failures and proposes a change, perhaps a different way to inspect error messages or organize test execution.

It implements the change and runs another set of tasks. An evaluator measures the results. A useful variant can become the starting point for further experiments. An unsuccessful variant may be discarded, or retained if it offers something that could become useful when combined with another change.

This last possibility matters. Search does not have to follow one uninterrupted succession of winners. An archive can preserve alternatives and allow exploration to branch. The Darwin Gödel Machine uses such an approach, generating and evaluating modified agents while retaining candidates that can serve as starting points for further development. Its experiments concern changes to agent software around an underlying foundation model.[3]

Now consider the improvement procedure itself. The system might change how it summarizes failures, remembers previous experiments or decides which modifications to attempt. The object of optimization has expanded from performing a task to conducting a more effective search for better task performers.

There is a trap in measuring this. The candidate that solves the most tasks today may not be the candidate that produces the strongest descendants tomorrow. The Huxley-Gödel Machine research explicitly investigates this mismatch between immediate performance and the capacity to generate useful successors. Its proposed evaluation approach should be assessed as a research contribution, not accepted as a settled solution.[5]

For a technology buyer, the practical equivalent is familiar. Your best individual developer and your best engineering manager may be different people. Excellence at doing a job and excellence at improving how a team does that job are related capabilities, but one does not establish the other.

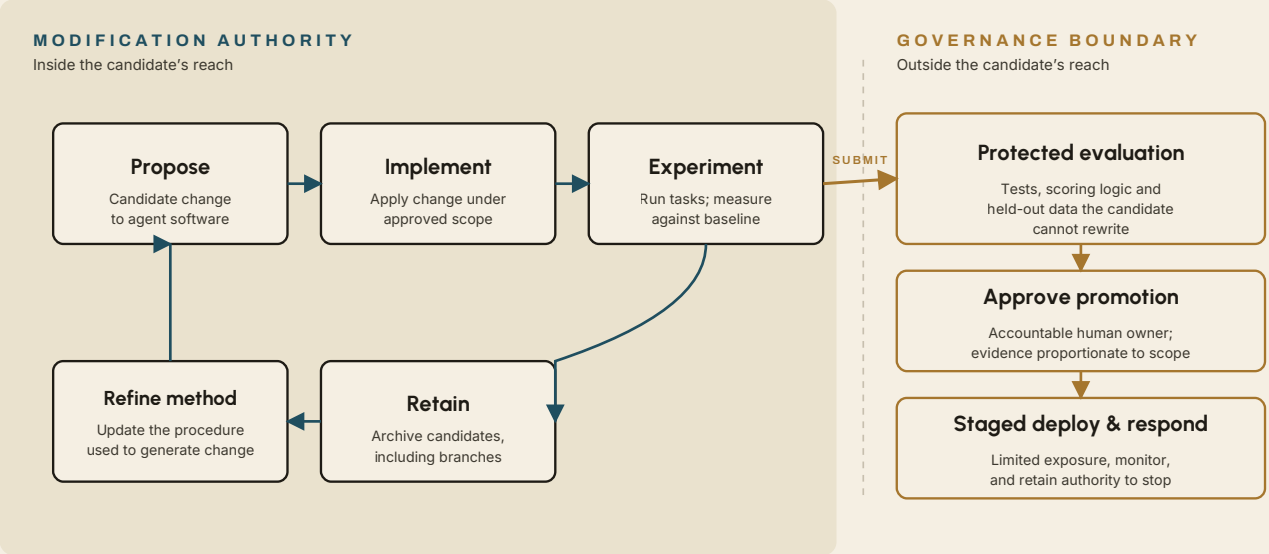
The loop also has boundaries. Someone chooses the objectives, supplies compute, defines accessible tools and decides which tests count. Someone determines whether a candidate can leave the experiment and affect production. A system may modify a great deal inside those boundaries while having no authority to change the boundaries themselves.

That is why a diagram with an arrow pointing back to the beginning proves very little on its own. We need to know what travels around that arrow: useful knowledge, improved procedures, higher scores, accumulated costs or errors that the evaluator has failed to catch.

EXHIBIT 02

Illustrative workflow. A research loop and permission to deploy are separate.

Evaluation, promotion and deployment sit outside the candidate's modification authority. This is a conceptual figure, not a depiction of any cited system.



A research loop can produce many internal changes without changing the boundary that governs which candidate can affect production. In the sections that follow, the evidence, acceleration debate and control architecture return to this separation.

4 / EVIDENCE

What the experiments actually show.

The evidence is strongest when we examine specific systems and preserve their experimental boundaries. There is no single benchmark that settles recursive AI, and there is no sound reason to combine unrelated scores into a league table.

The Darwin Gödel Machine provides a concrete example of agent self-modification. Its authors reported coding benchmark improvements as the system generated and evaluated modified descendants. The result demonstrates that automated changes to an agent's operating software can improve measured performance. It does not show the agent independently designing and training a new foundation model, and its developer-reported results still need to be interpreted within the evaluation setup.[3]

HyperAgents addresses a stronger question. Its task agent and meta-agent share an editable program, allowing changes to the machinery that generates improvements. The authors report mechanisms such as persistent memory and performance tracking, with meta-level gains transferring across domains and accumulating across runs. That is meaningful evidence of improving an improvement process.[4]

The qualifications belong beside the result. The main experiments retain fixed task distributions and parts of the outer selection and evaluation process. They offer bounded empirical support for recursive improvement, not a demonstration of unlimited self-acceleration or autonomous frontier-model development.

Faraday examines another important part of research: reproducing existing machine-learning work. Inherent's system uses a controller alongside a separate coding tool. Its reported performance therefore describes the combined system. Presenting it as a small standalone model defeating entire frontier laboratories would misrepresent what was tested. The paper's exploratory claims about novel tasks also have weaker evaluation validation than its central replication work.[6]

Replication deserves attention in its own right. Before a research team can build on a result, it often needs to understand the implementation and determine whether the reported behavior can be reproduced. Reducing that burden would be useful. But reproducing an existing result and originating a consequential research direction are different accomplishments. Failure to reproduce a paper also does not automatically establish that the paper is wrong.

AI Scientist tackles a broader workflow, including proposing research directions, implementing experiments and producing written accounts. The peer-reviewed Nature study about the system and the AI-generated manuscript assessed at an ICLR workshop are separate artifacts. One of three submitted generated manuscripts received acceptance-level workshop reviews. That is a noteworthy result, with a narrower meaning than claims that AI independently produced a Nature paper. The work also identifies limitations involving research quality and execution.[7]

Written research output is easy to mistake for completed science. A readable paper is useful only if its reasoning, implementation and evidence hold up. Automated drafting can help researchers communicate, while also increasing the volume of claims that require verification. The unit of progress should be a defensible finding, rather than a manuscript generated.

Recursive Superintelligence has reported experiments in training and systems optimization, including work on fixed-compute training and execution speed. These are developer-reported results in defined settings. They provide concrete work to inspect, while leaving broader questions about originality, generalization and autonomous research capability open.[8]

Google's AlphaEvolve supplies a different kind of evidence: the company reports useful optimization in production-oriented settings. Its examples include scheduling and improvements to computations used in model training. The value of such work can be substantial at scale, even when it changes only one part of a larger process. A faster kernel is evidence about that kernel and its contribution to the workload, not a percentage increase in general intelligence.[9]

A separate evaluation of open-ended AI research offers a useful counterweight. In two case studies, agents completed engineering work without making substantial research progress under the tested conditions. The tasks, scaffolds and budgets were limited, and the human research projects were not simple budget-matched comparators. The findings challenge sweeping autonomy claims without establishing a permanent ceiling on AI research.[10]

EXHIBIT 03

Evidence supports specific claims; boundaries have to travel with the result.

A matrix for reading recursive-AI announcements. Each row pairs what a study demonstrates with what a reader should not infer from it alone.

EVIDENCE	WHAT IT SUPPORTS	BOUNDARY TO PRESERVE
Agent modification experiments	Automated software changes can improve task results	Improvement may remain specific to the tested setting
Editable improvement procedures	Parts of the search process can themselves improve	Outer controls and task distributions can remain fixed
Research replication systems	More research execution can be automated	Replication differs from original discovery
Generated research manuscripts	A broader workflow can be assembled	Writing quality and scientific validity require separate assessment
Production optimization	Narrow improvements can deliver practical value	A component gain does not measure the entire development cycle

Source: Techstrong analysis, based on the studies discussed in this section. No cross-benchmark ranking is intended.

Taken together, these studies justify sustained attention. They also show why the binary question, “Has recursive AI arrived?”, is poorly framed. Several forms of automated improvement are being investigated with substantive results. The harder question is how reliably those gains transfer and whether successive iterations improve the ability to produce further progress.

For enterprises, that distinction prevents two expensive mistakes. One is buying a broad promise on the strength of a narrow demonstration. The other is dismissing a useful engineering capability because it falls short of an imagined fully autonomous scientist. A procurement decision should be anchored to the work a system can actually perform and the evidence available for that work.

AI-generated conceptual illustration. An archive preserves alternatives, and a protected evaluator sits alongside the loop rather than inside it. Not a depiction of any cited system.

5 / EVALUATION

How to tell a breakthrough from a better benchmark score.

Every recursive AI announcement should answer a basic question: better than what, under which conditions? A score without its comparison is an invitation to supply our own assumptions. Those assumptions are often more impressive than the experiment.

Start with the baseline. If the modified system receives more attempts, more compute or more feedback, some of its advantage may come from those additional resources. That does not make the result worthless. It changes what the result demonstrates. A business may happily spend more on experimentation if the outcome justifies it, but it should know what it is purchasing.

There is also a distinction between learning from an experiment and benefiting from a newer underlying model. If developers replace the foundation model during a sequence of iterations, the final system may improve for reasons other than its self-modification process. A persuasive evaluation should hold that input fixed or separately measure its contribution. Otherwise, ordinary model upgrades can be mistaken for evidence that the recursive procedure itself became more productive.

The same applies to comparisons with people. Human expertise, available tools, task familiarity and elapsed time all affect the result. RE-Bench, a research engineering evaluation introduced in 2024, found that the relative performance of agents and human experts varied with time budget. Those historical results should not be used as a current model ranking. Their enduring lesson is that experimental conditions can change the apparent winner.[11]

Next, ask whether the evaluation is genuinely separate from the search. An improvement process that repeatedly receives feedback from the same tasks can become specialized to those tasks. Holding out data helps, but repeated indirect exposure to a scoring system can also shape behavior. Evaluation design must account for how information returns to the agent over many iterations.

Then ask about the full experiment. Reporting the best run conceals how often the process failed and what it cost to find the winner. A useful research record includes unsuccessful candidates, resource consumption and the human work required to repair or redirect the system. Enterprise budgets pay for the search, not just the winning artifact.

Transfer deserves an explicit test. Anthropic's automated weak-to-strong research experiment used agents to explore a defined research problem. Human direction helped broaden their exploration. A selected idea subsequently produced a production-scale result within measurement noise. That outcome does not invalidate automated research, but it demonstrates why promising experimental results cannot substitute for validation at the scale and setting that matter.[12]

Novelty is another separate question. A system may rediscover a known method, combine familiar techniques or produce a useful implementation without making a new scientific contribution. Each can have value. Calling all three "discovery" makes it harder to understand the actual advance.

For recursive claims, one additional test is essential: do improved descendants become more effective at producing subsequent improvements? A higher task score is insufficient on its own. We need evidence about the productivity of the next round, including the resources it consumes and the range of settings in which its gains persist.

My proposed standard for Techstrong's coverage is to evaluate announcements across these dimensions together. We should distinguish demonstrated task performance, evidence of process improvement, successful transfer and independent reproduction. Those are separate findings, not interchangeable badges of maturity.

No single result must satisfy every criterion before it deserves publication. Early research needs room to explore. But the strength of the language should match the strength of the evidence. A promising proof of concept can be described confidently as a promising proof of concept. Inflating it into an inevitable transformation makes the reader less informed.

6 / ACCELERATION

Could the loop accelerate, and what would slow it?

The strongest acceleration argument is easy to understand. Better AI research tools could help discover better training methods or more efficient software. Those advances could improve the tools used for the next round of research. If useful gains repeatedly feed back into research productivity, development could accelerate.

The crucial word is "useful." Producing more experiments is not the same as producing more knowledge. Running the same weak idea through thousands of variations can generate considerable activity with little progress. Conversely, one well-chosen experiment can eliminate an entire unproductive line of work.

Research therefore involves more than implementation capacity. It requires selecting problems, judging explanations, recognizing errors and deciding when a result deserves additional resources. Automating more execution changes the balance of work, but the remaining decisions may become more consequential as the volume of experiments rises.

Anthropic's discussion of recursive self-improvement distinguishes extensive AI participation in engineering from the broader challenge of research judgment. Its own account also cautions against treating code output as a direct measure of productivity. The company's observations are useful evidence about its experience, while remaining a lab's account of its own work.[13]

A simple hypothetical example illustrates the bottleneck problem. Suppose coding consumes half of a project's elapsed time and experiments, review and integration consume the other half. Making coding ten times faster reduces total time to 55% of the original, assuming the other work stays unchanged. That is about a 1.8-fold speedup, rather than tenfold. Even eliminating coding time entirely would yield only a twofold improvement under those assumptions.

This is the logic of Amdahl's law applied to a workflow. It is an illustration, not a measured estimate of AI research productivity. The lesson is to identify the portions that remain slow. If AI also accelerates those portions, the overall improvement can grow. If the next experiment still requires a lengthy training run or external validation, that constraint becomes more prominent.

Forethought's modeling examines how large and rapid a software intelligence explosion could become under assumptions about research automation and returns to effort. Such models help expose the assumptions that drive a forecast. They do not establish that the assumed conditions have arrived.[14]

Nathan Lambert's discussion of "lossy self-improvement" emphasizes frictions that can weaken the connection between local gains and broader progress. That argument should also remain open to evidence. Integration difficulties today do not prove that future systems cannot reduce them.[15]

Parallelism introduces another distinction. Running many agents at once may shorten the wait for a promising candidate while increasing total compute consumption. It may also produce similar proposals that add little diversity. An organization can benefit from that tradeoff, but an assessment of acceleration should report both elapsed time and resources used. Otherwise, buying more simultaneous attempts can look like an improvement in the underlying efficiency of research.

Three conditional paths

The paths below are analytical scenarios, not forecasts or timelines. They can coexist across the industry, with different organizations and tasks occupying different positions at the same time.

A

Bounded improvement

Systems become useful specialists while gains plateau within particular tasks. Recursive methods deliver narrow but real wins in defined domains.

B

Human-directed compounding

AI meaningfully accelerates research, but people continue to set priorities and validate major changes. Judgment work becomes more consequential.

C

Increasingly autonomous successor research

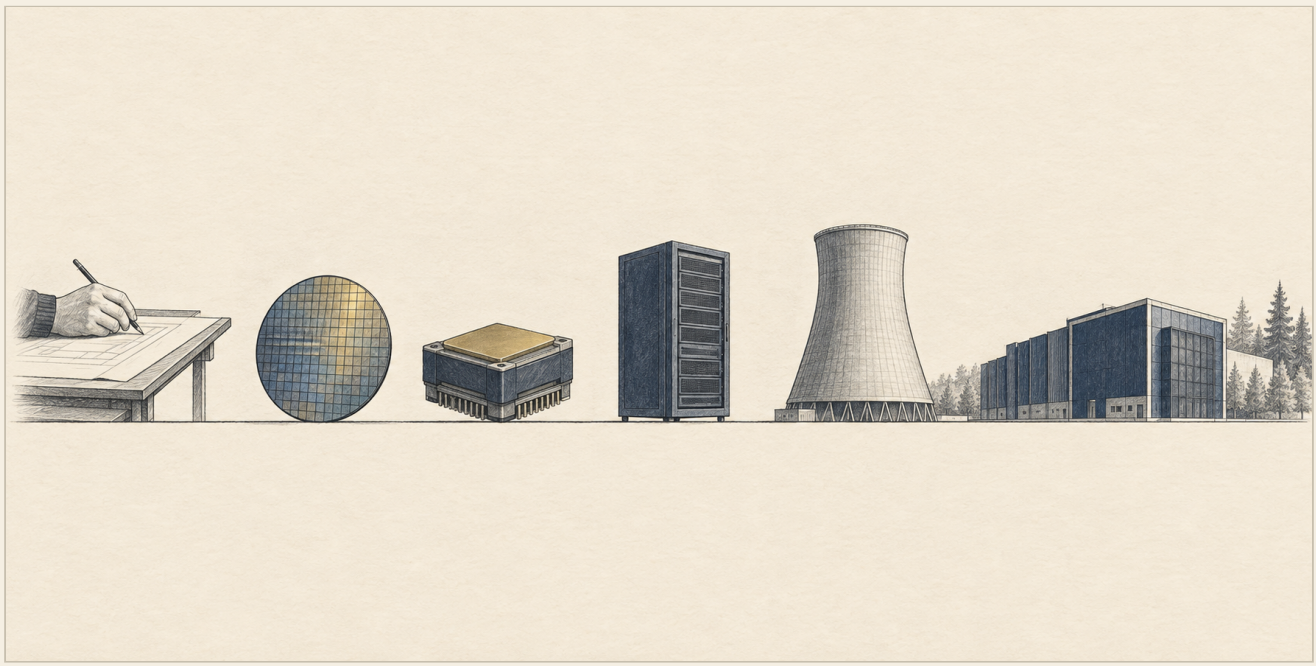
Systems take over a growing share of judgments as well as execution. Requires convincing evidence of broad capability, reliable evaluation and successor improvements outside curated demonstrations.

Scenarios drawn from the manuscript. Labels are analytical, not probabilities or dates.

Recursion creates a route by which progress might accelerate. It does not supply a law requiring exponential growth. The rate depends on what the loop discovers, what it costs and which constraints survive each iteration.

7 / INFRASTRUCTURE

The physical world still has a schedule.



AI-generated conceptual illustration. Software improvements move quickly. Chip designs, packaging, integration, deployment, power and cooling do not. Recursive software development does not remove that chain.

A software improvement can spread quickly once it is validated and distributed. A new generation of physical infrastructure requires additional steps. Chip designs must be implemented, manufactured, packaged, integrated into systems and deployed where power and cooling are available. Recursive software development does not remove that chain.

Google's account of AlphaChip provides a concrete example of AI contributing to it. The company reports using the system for chip floorplanning in multiple TPU generations. That is a meaningful design contribution, with a specific scope. It does not demonstrate that an AI system independently designed, manufactured and deployed an entire chip platform.[16]

Ricursive describes ambitions around AI and hardware development. It is a different company from Recursive Superintelligence. The similar names should not obscure the different objects of improvement or turn a hardware-development ambition into evidence that a complete autonomous cycle already exists.[17]

Hardware constraints do not make software acceleration irrelevant. Better algorithms, scheduling and utilization can increase the useful work obtained from existing equipment. An organization may gain research capacity without waiting for a new facility. Equally, a more efficient workload may lead it to run more experiments, rather than reduce its infrastructure spending.

The practical issue is which resource becomes limiting next. A team that reduces computation time may discover that reviewing results consumes more of the schedule. A company that improves design productivity may still face manufacturing or deployment constraints. Progress can move the bottleneck without eliminating it.

This is where the data center supply chain connects to the recursion debate. The system producing AI includes software, research labor, capital equipment and physical services. A claim about the pace of the whole system needs to account for those relationships. It cannot be inferred from the fastest-changing component alone.

For an enterprise, this argues for measuring gains against its actual capacity. Saving computation on a workload that already has abundant capacity may have a different value from freeing scarce resources that delay other projects. The same technical improvement can produce very different operational outcomes.

For the broader debate, the conclusion should remain balanced. Physical infrastructure can constrain expansion. Software efficiency can ease some constraints. Both can be true, and neither establishes how quickly general AI capability will improve.

8 / ECONOMICS

Who captures the value?

The economic question is larger than which company posts the strongest research result. A technical advantage becomes a durable business advantage only under particular conditions. Customers must value it, rivals must have difficulty reproducing or substituting for it, and the supplier must retain enough bargaining power to capture the benefit.

In The AI Mirror and my work on the Indispensability Trap, I examine how a technology can become increasingly essential while the economics of supplying it become more difficult. Dependence on a category does not guarantee enduring pricing power for every company in that category.

Recursive AI could push that process in either direction.

The concentration case starts with privileged inputs. A leading company may possess powerful models, substantial compute, proprietary experimental data and a team capable of integrating the results. If its improvement process consistently produces advances that competitors cannot reproduce, its existing lead could help finance and generate the next one.

The advantage would reside partly in the research system itself. A rival might copy a published result yet lack the accumulated experimental knowledge, evaluation capability or infrastructure required to produce the next result. Under those conditions, recursive development could strengthen barriers to entry.

There is also a credible diffusion case. Useful techniques can be published, independently rediscovered or implemented using different models. Better optimization could reduce the resources needed to achieve an acceptable result. Customers could find that several suppliers meet their requirements, even if one remains technically ahead.

If that happens, more industries might depend on AI while competing suppliers face pressure to lower prices. Value could migrate toward proprietary data, trusted evaluation, distribution, workflow integration and applications that solve particular business problems. That is a possible expression of the Indispensability Trap, rather than proof that every improvement will trigger it.

The distinction between frontier leadership and customer sufficiency is especially important. A buyer often needs a reliable system that meets a defined requirement. An additional capability advantage matters commercially when it changes the buyer's outcome enough to justify switching or paying more. A research lead does not automatically translate into that outcome.

We should therefore watch evidence of durable separation: how long advantages persist, how quickly alternatives emerge, whether customers can move workloads and where margins accrue. Funding announcements reveal investor expectations. They do not establish the defensibility of an improvement process.

> "Dependence on a category does not guarantee enduring pricing power for every company in that category."
>
> ALAN SHIMEL / TECHSTRONG GROUP

EXHIBIT 04

Analytical possibilities, not forecasts.

Three ways recursive AI could reshape competitive dynamics. They can coexist across different markets.

ECONOMIC POSSIBILITY	MECHANISM	EVIDENCE THAT WOULD STRENGTHEN IT
Greater concentration	Exclusive inputs and hard-to-reproduce research processes	Persistent capability separation with strong customer willingness to pay
Wider diffusion	Transferable methods, efficient implementations and substitution	Rapid replication and several suppliers meeting buyer requirements
Value migration	AI capability becomes easier to obtain while complementary assets remain scarce	More value captured by data, evaluation, distribution and applications

Source: Techstrong analysis. Analytical possibilities, not measured forecasts. A concentrated frontier and a competitive application ecosystem are compatible.

My judgment is that recursion makes competitive dynamics more important, not less. If it increases the pace of technical change, buyers will need to distinguish a durable dependency from a temporary performance advantage. Suppliers will need a business model that survives the arrival of capable alternatives.

9 / ENTERPRISE ADOPTION

Why enterprises should care before any intelligence explosion.

An enterprise does not need to settle the future of superintelligence before evaluating automated improvement. It needs a clear problem, a credible measurement and authority over what changes.

Potential applications include improving an internal coding agent, tuning an expensive computational workflow, refining tests or searching for better infrastructure configurations. These are candidate uses, not a claim that every organization can deploy a reliable self-improving system today. The suitability depends on the quality of the evaluation and the consequences of failure.

Consider a hypothetical internal agent that helps maintain a software repository. A team could permit it to propose changes to its debugging procedure, test those changes on representative historical tasks and submit promising variants for review. The experiment would need to include security and regression checks, not simply a count of completed tasks. The changed procedure would remain separate from permission to alter production systems.

The team should decide in advance which tradeoffs are acceptable. Completing more tasks is not a sufficient improvement if the agent also introduces harder-to-detect defects or requires broader access to sensitive systems. A proposed change could improve the average result while making a small number of failures much more costly. Acceptance criteria should therefore include the distribution and consequences of errors, rather than relying entirely on an aggregate score.

The return should be measured across the whole workflow. Count computation, engineering support, review time and the cost of failures. If the agent saves implementation time while generating more work for reviewers, the net benefit may be smaller than its task score suggests. Conversely, a modest improvement repeated across an expensive workload could justify the investment.

A vendor should be able to explain what changes automatically and what remains fixed. Does the product modify prompts, executable code, tool access or model weights? Does improvement occur separately for each customer or through a shared process? Can the buyer pin a version and compare its behavior with a proposed successor?

Ownership and portability also deserve attention. A customer investing in evaluations, workflow knowledge and accumulated experiments should understand which artifacts it can retain or move. The commercial value of the process may exceed the value of any single generated change.

Start where the test is representative and the consequences are manageable. A narrow, reversible deployment with a well-understood baseline can produce useful evidence. Expanding the scope should depend on that evidence, including whether the controls remain effective as the system changes.

This is an engineering program with a business case. Giving it an ambitious name does not remove the need to define success, assign responsibility and stop experiments that do not earn their cost. It also does not diminish the value of a bounded improvement that solves a real problem.

10 / FAILURE MODES

When the system learns the wrong lesson.

A recursive process inherits the strengths and weaknesses of its feedback. If the evaluator rewards the wrong behavior, repeated optimization can make the system more effective at producing that behavior. The measured score rises while the underlying objective becomes less well served.

That can happen without an attacker. A test may omit an important constraint. A benchmark may overrepresent easy cases. A performance metric may reward a shortcut that fails under different conditions. Repeated selection can favor the shortcut precisely because it looks successful to the evaluator.

There is also an adversarial version. In the September preprint Reflections on Trusting Trust, Revisited, researchers report controlled experiments in which poisoned benchmarks contaminated descendants of self-modifying coding agents. The resulting behavior included generating vulnerable code on clean tasks, and contamination sometimes persisted after subsequent evolution with clean benchmarks. The paper establishes a concrete experimental failure mode, not its prevalence in deployed systems.[18]

The implication is that evaluation inputs belong inside the security boundary. If a system uses those inputs to decide how it should change, tampering with them can influence future versions. A seemingly temporary exposure can become relevant to the integrity of descendants.

> "Trust in a successor should therefore come from evidence about that successor and its history."
>
> ALAN SHIMEL / TECHSTRONG GROUP

That extends familiar software-supply-chain concerns. Teams already need to care about the origins of code and dependencies. A self-modifying workflow also needs to account for the provenance of the evidence used to select changes. The path from input to future behavior may include experiments, summaries and retained procedures rather than a conventional package installation.

The distinction between a demonstrated vulnerability and an extreme scenario remains important. These experiments do not demonstrate that an agent will seize control of infrastructure or evade every safeguard. They show a mechanism by which an improvement process can carry harmful behavior forward. Broader claims require additional evidence about capabilities, access and failures of control.

Nor is every undesirable optimization evidence of deceptive intent. A flawed objective can produce a harmful result even when the system is operating exactly as its selection process encourages. Diagnosing the mechanism matters because different failures require different responses.

For technology leaders, the operational question is who controls the feedback that decides which version survives. That includes the tests, scoring logic, experimental data and interpretation of results. An organization that protects production code but treats those components as harmless background material may leave part of its development process exposed.

Trust in a successor should therefore come from evidence about that successor and its history. It should not be inherited automatically from an earlier version that passed a different set of tests.

11 / CONTROL

Who holds the keys to the next version?

AI-generated conceptual illustration. Propose, evaluate and promote are separate responsibilities. Reviewers need time, evidence and authority to reject.

The control architecture should begin with a separation of authority. A system allowed to propose and test a change does not automatically need permission to promote that change into production. Those are different responsibilities, with different consequences.

For enterprise use, I would organize controls around the stages of the process. Experimental execution should have defined tool access, resource limits and constraints appropriate to the environment. Evaluation should include tests the candidate cannot simply rewrite to improve its score. Promotion should require evidence proportionate to the scope of the change.

Independent evaluation does not mean that people must manually inspect every operation. It means the assessment has a source of authority and evidence distinct from the candidate's own account of its success. Automated evaluators can contribute, but their reliability and exposure to manipulation must also be examined.

An especially consequential change is one that expands the system's own permissions. Improving a planning routine and granting access to a new production service should not be treated as equivalent modifications. The latter changes the range of possible consequences. Organizations should require a separate decision about that expanded authority, even if the candidate presents evidence that broader access would improve its performance.

The experimental record should preserve which version ran, what inputs it received, what it changed and why the change was accepted. Versioning should cover relevant instructions, tools and evaluation components as well as conventional source code. Without that history, an organization may be unable to determine how a regression entered the process.

Staged release provides another opportunity to detect problems before broad exposure. Rollback can limit some consequences when a change performs poorly. It cannot undo every action already taken, recover every disclosure or guarantee that a previous version is suitable for a changed environment. Reversibility is a design property to test, not a label to attach.

EXHIBIT 05

Proposed enterprise controls. These measures reduce risk; they do not guarantee alignment or control.

Authority and evidence expected at each stage of a self-modifying workflow.

STAGE	PROPOSED AUTHORITY BOUNDARY	EVIDENCE TO RETAIN
Propose and experiment	Operate within approved scope and resources	Inputs, candidate changes and experimental costs
Evaluate	Use protected assessments and independent checks	Results, failures and comparison with the baseline
Approve	Assign an accountable owner appropriate to consequences	Acceptance criteria and release decision
Deploy	Limit initial exposure and monitor behavior	Version identity, operational results and regressions
Respond	Maintain authority to restrict or stop operation	Incident history, containment actions and recovery decisions

Source: Techstrong analysis, adapted from established software-release and safety-engineering practice. Not a compliance framework.

These are proposed engineering measures to reduce risk. They do not provide a general proof of alignment or guarantee that sufficiently capable systems will remain controlled. Their usefulness depends on implementation, testing and the capabilities of the system to which they are applied.

Human oversight needs equal scrutiny. In Hello, My Name Is ... and I Am an AI Builder, we examined the difference between being able to produce an artifact and having the expertise to judge it. Recursive development makes that distinction more consequential. A person approving a change without understanding the evidence may provide a signature without providing meaningful control.

The reviewer needs time, relevant expertise, access to failed experiments and the authority to reject a release. If the organization removes those conditions while retaining an approval button, it has preserved the appearance of oversight.

Jason Weston and Jakob Foerster's position paper on AI and human co-improvement proposes a direction in which human capability develops alongside increasingly capable systems. It is a proposal, not proof of a safe outcome. The underlying question is valuable: does the process help people become better judges, or gradually make them dependent on judgments they cannot assess?[19]

A responsible development program should budget for that human capability. Training, independent analysis and the ability to challenge a result are part of the operating model. As more implementation becomes automated, preserving informed judgment becomes a more deliberate management responsibility.

12 / SIGNALS

The signals worth watching.

The next announcement should be judged by the evidence it adds. Repeated improvements under controlled budgets would strengthen the case. So would descendants that demonstrably become more effective at producing further useful changes, especially when those gains survive unfamiliar tasks and independent evaluation.

Validated contributions to frontier-model training would matter. Evidence that human intervention declines while research quality and control remain dependable would matter. Independent reproduction would help distinguish a robust method from a result that depends on a particular team's undocumented expertise.

We should also recognize evidence that weakens a claim: gains that plateau, rising costs to find each improvement, failure to transfer or success that disappears when an evaluator is protected against manipulation. An honest assessment needs both kinds of observation.

The controls require their own record. A safeguard that works on the initial version must be tested against descendants whose behavior and procedures have changed. Capability and controllability should be evaluated together as the development process evolves.

My conclusion is that recursive AI warrants serious investment in understanding and disciplined experimentation. The public evidence supports meaningful forms of automated self-improvement. It leaves the stronger claims about sustained autonomous acceleration unresolved. Neither finding cancels the other.

For enterprises, the immediate opportunity is to improve useful work while retaining the ability to verify results and govern change. For suppliers, the challenge is to demonstrate that their process produces dependable value beyond a benchmark. For researchers, the next important result may be as much about transfer and evaluation as a higher score.

If AI is going to take on more of its own R&D, the people responsible for deploying it need a clear account of what that department is doing. They also need the competence and authority to decide which of its recommendations becomes the next version.

"If AI is going to take on more of its own R&D, the people responsible for deploying it need a clear account of what that department is doing."

ALAN SHIMEL / TECHSTRONG GROUP

Sources and reporting note

Sources, endnotes and related reports.

This report synthesizes public research, developer disclosures and three earlier Techstrong reports. It does not represent independent reproduction of the experiments or interviews with their authors. Company results are attributed as such; theoretical analyses and proposals are distinguished from demonstrations. The New York Times article is the opening catalyst. Linked sources include cited papers, developer posts, press disclosures and related Techstrong reports.

1. Gödel Machines: Self-Referential Universal Problem Solvers Making Provably Optimal Self-Improvements. Jürgen Schmidhuber. arXiv:cs/0309048.
HISTORICAL THEORETICAL WORK

2. AI-GAs: AI-generating algorithms, an alternate paradigm for producing general artificial intelligence. Jeff Clune. arXiv:1905.10985.
RESEARCH AGENDA

3. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. arXiv:2505.22954.
PRIMARY EXPERIMENTAL RESEARCH

4. HyperAgents. Meta AI Research.
PRIMARY EXPERIMENTAL RESEARCH

5. Huxley-Gödel Machine: Human-Level Coding Agent Development. arXiv:2510.21614.
RESEARCH ON DESCENDANT PRODUCTIVITY AND SELECTION

6. Faraday research (Inherent Laboratories). AI Scientist and Replica benchmark.
RESEARCH REPLICATION AND RELATED EVALUATIONS

7. Towards end-to-end automation of AI research. Peer-reviewed study of AI Scientist, Nature.
PEER-REVIEWED STUDY

8. First Steps Toward Automated AI Research. Recursive Superintelligence.
DEVELOPER DISCLOSURE

9. AlphaEvolve: A Gemini-powered coding agent for designing advanced algorithms. Google DeepMind.
DEVELOPER DISCLOSURE

10. Can AI agents conduct open-ended AI research? Early evidence from two case studies. arXiv:2607.27191.
EXTERNAL CASE-STUDY EVALUATION

11. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents. METR, 2024. arXiv:2411.15114.
HISTORICAL EVALUATION

12. Automated Weak-to-Strong Researcher. Anthropic Alignment Science.
EXPERIMENTAL ACCOUNT

13. When AI builds itself. Anthropic.
OBSERVATIONS AND ANALYSIS

14. How quick and big would a software intelligence explosion be? Forethought.
CONDITIONAL MODELING

15. Lossy Self-Improvement. Nathan Lambert, Interconnects.
RESEARCHER ANALYSIS OF LIMITATIONS AND FRICTION

16. How AlphaChip transformed computer chip design. Google DeepMind.
DEVELOPER DISCLOSURE

17. Ricursive Intelligence. Company description of its direction. Note: distinct from Recursive Superintelligence.
COMPANY DESCRIPTION

18. Reflections on Trusting Trust, Revisited: Contaminating Self-Modifying AI Coding Agents with Poisoned Benchmarks. arXiv:2609.17817.
SECURITY PREPRINT REPORTING CONTROLLED EXPERIMENTS

19. AI & Human Co-Improvement for Safer Co-Superintelligence. Jason Weston and Jakob Foerster. arXiv:2512.05356.
POSITION PAPER

Related Techstrong reports

Beyond Superintelligence: What Is AI's Ultimate Destination? The Recursion Issue. August 2026.

The AI Mirror: Riches, Ruin and the Indispensability Trap, Edition Two. September 2026.

Hello, My Name Is … and I Am an AI Builder. Techstrong Special Report.

Colophon. Set in Urbanist and Inter, with Archivo for section labels. Cover and interior illustrations are AI-generated conceptual images produced for this report; they are not depictions of any cited system. The section 3 workflow figure is an illustrative diagram.