

The Experience Layer of AGI

How Real-World Tasks Shape More General AI

Why diverse datasets, custom environments, and expert-designed challenges matter - and where Deaimer fits.

Shahzaib Awan

CEO, Deaimer

12 October 2026

Inside this article

A research perspective on experience, adaptation, and the practical contribution of datasets and environments to AI development.

1. The question beneath the AGI conversation
2. Generality needs a precise meaning
3. From records of work to opportunities to act
4. The anatomy of a useful experience
5. Diversity is a property of decisions
6. Transfer should be designed into the experiment
7. Long workflows expose coordination problems
8. Recovery reveals a different dimension of competence
9. Environments and algorithms develop together
10. Domain expertise makes experience meaningful
11. Where Deaimer fits in this research direction
12. A research program buyers can actually run
13. Questions that remain open
14. Building the experiences that progress depends on

Includes an illustrative capability portfolio, a proposed transfer framework, and an experimental template for model developers and dataset buyers.

Abstract

Artificial general intelligence is often discussed through the capabilities of models: what they know, how well they reason, and which benchmarks they can solve. An equally practical question concerns the experiences through which those capabilities are developed and examined. What situations should an agent encounter? What decisions should it make? How can a researcher determine whether learning survives a change in tools, task structure, or operating conditions? This research perspective examines executable tasks and custom environments as an experience layer for AI development. Drawing on primary research in intelligence measurement, agent training, realistic evaluation, world models, and open-ended learning, it argues for organizing datasets around transferable capabilities rather than superficial task variety. It then connects that argument to Deaimer's public offerings in private coding datasets, custom RL environments, scientific workflows, and ColdStart recovery evaluation. The article proposes a practical framework for designing experience portfolios and studying transfer. It presents no new model-training experiment and makes no claim that a particular dataset establishes AGI.

1. The question beneath the AGI conversation

Imagine an AI agent entering a software system it has never seen. A service is failing. The documentation is incomplete. A recent change looks suspicious, but the visible error could have several causes. The agent can inspect files, run commands, examine logs, and modify the environment. Its objective is clear; the path to that objective is not.

This setting asks for several kinds of competence at once. The agent must interpret evidence, choose useful observations, maintain a working hypothesis, understand dependencies, and revise its plan when an intervention fails. It must also respect the state already present in the system. A fast repair that destroys important data would be a poor outcome.

Now move the same agent into a scientific workflow. It encounters a dataset with unfamiliar conventions, a numerical routine whose assumptions matter, and an output that looks plausible even when the analysis is wrong. Some skills carry over: locating relevant information, testing explanations, managing tools, and checking consequences. Other skills require additional domain knowledge.

The relationship between these two situations is the subject of this article. What can an agent learn from one experience that helps it handle the next? Which parts of its apparent competence depend on familiar templates? How should developers design training experiences to investigate that distinction?

We use **experience layer** as an organizing term for the tasks, executable environments, observations, consequences, and recorded interactions made available to an agent. It is a proposed perspective, rather than an established component of every AI architecture. Its purpose is to make a practical development question visible: alongside decisions about models and training algorithms, developers make decisions about the problems through which learning occurs.

For Deaimer, this question connects directly to dataset and environment construction. The opportunity is to help research teams obtain purposeful experiences that exercise capabilities they want to develop, examine, or improve.

2. Generality needs a precise meaning

AGI has no universally accepted operational definition. Different discussions emphasize human-level performance, broad competence, autonomy, adaptation, or economic usefulness. Treating these as interchangeable makes both research claims and product claims difficult to assess.

Morris and colleagues' *Levels of AGI* offers a useful distinction between the depth of performance and the breadth of capabilities. Their framework also considers autonomy as a related deployment dimension. This helps explain why excellence in one domain, broad competence across domains, and independent action should be examined separately. [1]

Chollet's *On the Measure of Intelligence* develops a complementary argument: demonstrated skill depends on prior knowledge and experience, so task performance alone can conceal differences in generalization. His proposed account emphasizes skill-acquisition efficiency and the difficulty of acquiring capabilities under specified priors and experience. [2]

Our inference from these perspectives is that an AGI-oriented environment program should make its target explicit. A dataset might aim to strengthen repository navigation, improve adaptation to unfamiliar command-line tools, or examine whether diagnosis strategies transfer across system types. These are narrower and more testable goals than claiming to measure general intelligence in its entirety.

This precision strengthens commercial relevance. A buyer needs to know which capability a collection exercises, how it differs from existing data, and what evidence would demonstrate usefulness for a particular model. An ambitious research direction becomes actionable when it can be translated into an experiment.

The phrase *toward more general AI* is therefore most useful when attached to observable changes: broader coverage, better transfer, faster adaptation, or more dependable performance under changed conditions. Each requires its own evidence.

3. From records of work to opportunities to act

A static example can show what a good answer looks like. An executable environment can let an agent investigate how to produce one. These formats provide different kinds of experience, and a training program may benefit from both.

Consider a bug-fixing example containing an issue description and a final patch. It can communicate a mapping from problem to solution. Place the issue inside a runnable repository, and additional decisions become available: which file to inspect, which test to run, whether an error supports the current hypothesis, and when to reconsider the proposed fix.

SWE-Gym provides a concrete research example. Its authors introduce software-engineering task instances that combine a natural-language objective, a codebase, an executable runtime, and tests. They report improvements in software-engineering agents trained using that resource. The finding concerns the agents, training methods, and task distributions studied in that work; it does not establish an effect for Deaimer's collections. [3]

AgentGym-RL studies multi-turn reinforcement learning for language-model agents across interactive environments. Its framework and training approach address the practical problem of learning through sequences of decisions, including the relationship between exploration and interaction horizons. [4]

These studies motivate a design question rather than a universal recipe. What information becomes available because the agent acts? A command might reveal a dependency conflict. An experiment might rule out a cause. A tool response might show that an assumed interface is wrong. A failed intervention can alter the next useful action.

An environment is valuable when these consequences connect to meaningful decisions. A long interaction history created by unnecessary steps may provide little additional learning value. Conversely, a compact task can be rich if it demands discriminating evidence and an important revision of strategy.

4. The anatomy of a useful experience

We propose five questions for describing an environment's learning opportunity. They form an authoring framework for this article, not a standardized AGI metric.

4.1 What does the agent initially know?

The starting state determines which uncertainties must be resolved. If the relevant diagnosis is already supplied, the task primarily exercises execution. If several plausible explanations remain, it can exercise investigation. Both can be useful, but they teach and examine different capabilities.

The initial information should match the intended learning goal. A task about unfamiliar tools needs accessible documentation or an informative interface. A task about recovering an existing system needs enough observable evidence to support a defensible repair. Difficulty caused by missing essential information can become arbitrary guessing.

4.2 Which actions change the information available?

Good experiences create reasons to choose among observations. Reading a configuration file, inspecting a log, or running a focused experiment should affect the agent's understanding. This lets researchers examine whether the agent uses tools to reduce uncertainty or merely performs a familiar sequence.

4.3 Which consequences persist?

State connects one action to the next. A modification may change later outputs, consume resources, or create a new failure. Stateful tasks make the agent accountable for maintaining an accurate picture of its own interventions.

4.4 What would a competent revision look like?

A meaningful failure should permit a productive response. The agent might abandon a hypothesis, consult documentation, restore a previous state, or select a different method. Environment design can make these possibilities visible without dictating the solution.

4.5 What remains true when the task changes?

This is the transfer question. The exact file names, tools, and domain may change, while a deeper decision pattern persists: compare expectations with observations, isolate causes, preserve invariants, and test consequences. Designing around that pattern can support an experiment on generalization. Whether it actually does so remains an empirical question.

5. Diversity is a property of decisions

A collection can look diverse while repeatedly asking for the same behavior. Changing the language, directory names, or story surrounding a task may expand surface variety without changing the underlying decisions. That variety can still matter for robustness, but it should be described accurately.

For broader research, task diversity should be considered across several axes. These include what the agent knows at the start, which tools it can use, the structure of dependencies, the consequences of actions, and the expertise needed to interpret results.

The following examples illustrate an authoring approach. They are proposed task families, not a claim about the contents of a delivered Deaimer collection.

Experience family	Decisions the agent practices	Useful transfer question
Repository repair	Locate the responsible component, test a hypothesis, update dependent code	Does diagnosis transfer to an unseen repository structure?
Terminal operations	Interpret tool output, manage dependencies, produce an artifact	Can the agent adapt when a familiar utility is replaced?
Scientific analysis	Interpret data meaning, choose methods, examine assumptions	Does method selection survive a change in data conventions?
Stateful recovery	Diagnose a fault, preserve existing state, assess repair durability	Can the agent repair a new incident without avoidable damage?
Combined workflow	Carry an objective across several tools and representations	Can previously learned skills be composed in a new sequence?

A useful portfolio may contain both specialized tasks and tasks that combine capabilities. Specialization provides depth; combinations create opportunities to study coordination. The balance depends on the model's starting competence and the development objective.

For example, tool adaptation can be studied in an accessible domain before it is combined with demanding scientific content. Otherwise, a failure may be difficult to interpret: was the agent unable to understand the tool, the science, or their interaction?

This leads to a practical procurement principle. Buyers should ask what new decisions a dataset adds to their existing experience distribution. A task count answers how many examples exist. A capability map helps explain what those examples might contribute.

6. Transfer should be designed into the experiment

Transfer is often used as a broad promise. For environment research, it becomes more informative when the proposed change is specified.

We distinguish four forms of transfer for practical discussion. **Interface transfer** changes the tool or representation. **Structural transfer** changes the dependency arrangement or problem organization. **Compositional transfer** combines familiar skills in a new workflow. **Domain transfer** moves a decision pattern into a setting with different knowledge requirements. These categories are our analytic framework and can overlap.

An interface experiment might train on one command-line tool and evaluate on another with a different syntax but similar purpose. Documentation would remain available. The objective would be to examine adaptation, rather than recall of a known command.

A structural experiment might preserve the general repair goal while changing where the fault sits in a dependency chain. A compositional experiment might require an agent to inspect a repository, transform input data, and repair an integration failure in one task, after learning those skills separately.

Domain transfer warrants particular care. A debugging strategy can carry over to scientific computing, but selecting a scientifically valid method may require expertise absent from software repair data. Broad transfer should be demonstrated under conditions that reveal which knowledge is supplied, learned, or assumed.

The separation between training and evaluation must occur at the level relevant to the claim. If the claim concerns unfamiliar repositories, splitting near-identical issues from the same repository may be insufficient. If it concerns new fault mechanisms, renamed variants of a familiar fault are weak evidence.

A stronger design reserves entire task families, repositories, tools, or combinations, as appropriate. The required separation is a research choice with costs: harder holdouts can reduce measured success and require more authoring work. Their value is that they make the meaning of a gain easier to interpret.

7. Long workflows expose coordination problems

Long-horizon work asks an agent to maintain coherence across changing observations. The final outcome may depend on choices made much earlier: a dependency selected, a data transformation applied, or a configuration assumption accepted without examination.

WebArena was designed as a realistic, reproducible environment for language-guided agents performing web tasks across functional websites. Its contribution illustrates why studying complete interactive workflows can reveal challenges that simpler task settings omit. [5]

GAIA takes a different approach, presenting questions for general AI assistants that require combinations of reasoning, multimodal handling, browsing, and tool use. It offers another example of evaluating capabilities together in practical tasks. [6]

Neither benchmark is a complete definition of AGI. For this article, they motivate the importance of composition: useful work often demands that several abilities remain coordinated through an entire attempt.

As an illustrative calculation, suppose a workflow contains ten indispensable stages and each independently succeeds with probability 0.95. The probability that all ten succeed is approximately 0.60. This is a mathematical example, not a measured agent result; real workflows contain dependencies, recovery opportunities, and unequal stage difficulties. Its purpose is to show why local competence can coexist with weak end-to-end reliability.

The design response is to examine what happens between stages. Does the agent preserve its objective? Does it notice that the input format changed? Does it retain the evidence behind a decision? Can it recover from an intermediate failure?

Tasks should be long because the work contains meaningful dependencies. Expanding an instruction merely to create more text or adding repetitive commands can obscure the actual capability being studied.

8. Recovery reveals a different dimension of competence

Success in a clean environment is useful evidence. Recovery from an unexpected condition can reveal how an agent reasons when its initial plan no longer fits the world.

Consider an illustrative service incident. A worker stops processing jobs after a configuration change. Restarting it clears the immediate error, but the queue still contains pending work and a second restart reproduces the problem. A capable attempt would investigate the recurring cause, preserve the queue, and assess whether the repair persists.

The relevant learning opportunity is the relationship between uncertainty, intervention, and consequences. The agent must decide which state to inspect before changing it, distinguish a temporary symptom from a cause, and use the result of an intervention to update its explanation.

ColdStart's public description focuses on recovery of unfamiliar, production-like broken systems. Its stated dimensions include functional recovery, state safety, durability, integrity, evidence quality, and minimal-change behavior. It also identifies the model-agent configuration as the unit of evaluation. [7]

This connects recovery evaluation to the article's broader thesis. General capability in operational settings includes acting responsibly within an existing state, even when the environment is unfamiliar. ColdStart provides a public example of how Deaimer frames that problem.

A benchmark supports measurement. Separate training environments can be developed around related capability goals while protecting reserved evaluation material. The relationship should be designed deliberately: train on a relevant class of reasoning problems, then examine performance on independently held-out incidents.

The scientific value lies in what can be learned about behavior. An agent that identifies the right cause but applies a fragile repair presents a different development problem from one that repeatedly gathers irrelevant evidence. A single outcome score can be supplemented by analysis of these decision patterns without turning the article into a discussion of internal validation methods.

9. Environments and algorithms develop together

Environment design cannot be separated completely from the learning method. The same task may produce useful demonstrations for one training approach and sparse or uninformative experience for another.

Dreamer research offers an important example from control. Hafner and colleagues describe an agent that learns a world model and improves behavior through imagined trajectories, demonstrating an approach across diverse control tasks. This is evidence about a particular learning architecture and experimental program, rather than proof that diversity alone causes general intelligence. [8]

POET explores another direction: generating environmental challenges while optimizing agents to solve them, with solutions transferring between environments. Its results motivate research into expanding curricula and the possibility that a task can serve as a stepping stone for another challenge. [9]

Our inference is that an environment portfolio can be treated as a research object that evolves with the agent. Some experiences provide foundations. Others exercise new combinations. Additional tasks can be commissioned when failures expose a gap in existing coverage.

This does not imply that harder tasks are always better. If a model cannot perform prerequisite actions, a very demanding environment may produce repetitive failure with little useful variation. A more accessible task can help isolate a skill before it is tested under greater uncertainty or a longer workflow.

Curriculum design should also preserve useful breadth. A progression focused on a single failure family may improve that family while leaving other capabilities unchanged. Monitoring several reserved capability groups can help researchers see whether specialization is accompanied by transfer or by tradeoffs.

For environment providers, the practical task is to support this iteration. Researchers need enough control over scope, difficulty, interfaces, and domain assumptions to connect new experiences to a learning question.

10. Domain expertise makes experience meaningful

A task environment represents a view of what competent work requires. Its author chooses which constraints matter, which evidence is informative, and which decisions distinguish expertise from routine execution.

In software engineering, that distinction might concern compatibility, maintainability, or the interaction of several modules. In scientific computing, it might concern units, numerical stability, experimental assumptions, or the meaning of an output. In recovery work, it might concern preserving existing state while restoring service.

These choices cannot be replaced by surface complexity. A task with many files and tools may still permit a shallow solution. A smaller task can demand substantial expertise if its essential decision requires understanding a consequential assumption.

Expert input is especially useful when creating alternatives that appear plausible. Suppose two analysis methods both produce a clean plot. The learning opportunity arises when the data conditions justify one method and undermine the other. The environment must make those conditions observable and the goal sufficiently clear.

For a buyer, this makes domain expertise part of dataset value. The question becomes whether the collection reflects the decisions their model will face. A generic corpus may provide broad exposure, while a custom environment program can focus on a specific class of work or an observed capability gap.

The human contribution is also interpretive. Experts can help distinguish a sensible alternative solution from an accidental shortcut, or identify when failure stems from an unrealistic scenario. Here the emphasis is on meaningful task design and research relevance; detailed internal review procedures are outside this article's scope.

11. Where Deaimer fits in this research direction

Deaimer's public offerings include private terminal-style tasks, scientific-computing collections, repository-level software-engineering tasks, and custom RL environments. The company describes work spanning terminal operations, debugging, DevOps, system administration, security, scientific computing, and longer workflows. Package contents and evidence availability depend on the collection and agreed scope. [10, 11]

Termetheus is publicly described as Deaimer's offering for custom terminal-style datasets, combining objectives with prepared software environments and supporting assets. [12] This provides a practical connection between a research question and an executable task collection.

The proposed role for Deaimer in progress toward more general AI is to supply deliberately designed experience: environments in which developers can investigate tool adaptation, diagnosis, workflow composition, scientific reasoning, and recovery.

For example, a research team could commission a portfolio around adapting to unfamiliar developer tools. An initial collection might isolate documentation use and interface discovery. A subsequent collection could combine those demands with repository changes. A reserved evaluation set could examine the same capabilities using different tools and structures. This is an illustrative engagement design, not a description of a completed customer experiment.

Another team could focus on stateful recovery, with training experiences covering varied failure mechanisms and separate evaluation incidents that test transfer. A scientific-agent developer could request tasks that combine software execution with discipline-specific assumptions.

The contribution becomes measurable when the buyer evaluates the resulting model under an agreed experimental design. Deaimer can help define and construct the experiences; evidence of learning gains requires model runs and appropriate comparisons.

This is a commercially useful position because it connects the deliverable to a concrete development problem. Dataset value can arise from relevant coverage, purposeful variation, practical usability, and the ability to support a research question that existing resources do not adequately address.

12. A research program buyers can actually run

The following is a proposed experimental template for assessing a custom experience portfolio. It is included to make the article's argument actionable, rather than to report results.

First, define a target capability in observable terms. For example: adapting to unfamiliar command-line tools using documentation, or diagnosing a service failure while preserving existing state. Specify the scope of the claim before selecting tasks.

Second, establish the baseline system. Record the model version, tool interfaces, context policy, available documentation, and attempt budget. An agent is a configured system, and changes to its surrounding software can affect the outcome.

Third, separate data by the intended transfer boundary. A tool-adaptation study needs genuinely reserved tools or interfaces. A repository-transfer study needs reserved repositories or suitably distinct structures. A composition study needs reserved combinations of skills.

Fourth, compare the custom portfolio with a relevant alternative under matched conditions. Depending on the question, this might be the existing training mixture, a size-matched task collection, or a portfolio with less structural variation. Match training and evaluation resources as closely as practical and disclose meaningful differences.

Fifth, examine several outcomes. These could include completion rate, reliability across attempts, time or cost conditional on success, adaptation effort, and a small number of behavior categories relevant to the target capability. Avoid interpreting a gain in one dimension as a gain in all of them.

Finally, study where the effect survives. Does improvement remain on unseen task families? Does it persist when documentation is reorganized? Does the agent perform better because it identifies informative evidence, or because it has learned a repeated solution pattern?

The strongest product claim that such a program can support is specific: under stated conditions, exposure to this portfolio improved a defined capability on a defined held-out distribution. That claim can be valuable to a model developer without requiring a conclusion about AGI as a whole.

13. Questions that remain open

Executable environments make experimentation possible, but several research questions remain unresolved.

How much task variety is needed before additional surface variation has diminishing value? Which forms of structural diversity contribute to transfer? When do specialized scientific experiences improve more general investigation skills? How should a curriculum balance foundational tasks with challenging combinations?

There are also questions about agent memory and adaptation. An agent may use information within a single attempt, across a series of tasks, or through changes to its model parameters. These are different learning mechanisms. A study should specify which mechanism is available before claiming that the agent learned from experience.

Simulation adds another boundary. An environment can be reproducible and useful while omitting aspects of deployment: organizational context, shifting requirements, real user preferences, or unpredictable external services. The closer the intended claim is to production performance, the more important it becomes to examine that boundary.

Digital work also represents only part of the broader AGI question. Terminal and repository tasks can exercise valuable capabilities, yet evidence from them does not establish broad competence in embodied activity, social judgment, or every form of reasoning.

These open questions create an agenda for collaboration between model developers, domain specialists, and environment builders. More purposeful experiences can help researchers ask sharper questions. The results must then determine which design choices are actually useful.

14. Building the experiences that progress depends on

Progress toward more general AI involves decisions about what systems encounter, how they act, and what researchers learn from the consequences. The experience layer gives these decisions a place in the development conversation.

A meaningful task can invite investigation. A thoughtfully varied portfolio can support a study of transfer. A stateful environment can reveal whether an agent understands the consequences of its own actions. A domain expert can ensure that the challenge reflects the work it is intended to represent.

Deaimer's opportunity is to turn those needs into practical datasets and custom environments. Its coding, terminal, scientific, and recovery work provides several entry points for research teams seeking more targeted experience distributions.

The next useful question for a model developer is concrete: **What should our agent be able to handle that its current experiences do not teach or adequately examine?**

Answering that question can guide a dataset brief, a custom environment program, and an experiment with interpretable results.

AGI remains a broad research ambition. Building experiences that help agents adapt to unfamiliar work is a practical contribution that can be investigated today.

Work with Deaimer

If your team is developing coding agents, scientific agents, or systems that operate across tools and environments, Deaimer can discuss a private dataset collection or a custom environment scope around your target capabilities. Start with the workflow, the capability gap, and the transfer conditions you want to study. Explore [Deaimer datasets](#), [custom RL environments](#), and [ColdStart](#).

References

1. Morris, M. R., et al. (2024). Position: Levels of AGI for Operationalizing Progress on the Path to AGI. Proceedings of the 41st International Conference on Machine Learning, PMLR 235, 36308-36321.
<https://proceedings.mlr.press/v235/morris24b.html>
 2. Chollet, F. (2019). On the Measure of Intelligence. arXiv:1911.01547.
<https://arxiv.org/abs/1911.01547>
 3. Pan, J., et al. (2024). Training Software Engineering Agents and Verifiers with SWE-Gym. arXiv:2412.21139.
<https://arxiv.org/abs/2412.21139>
 4. Xi, Z., et al. (2025). AgentGym-RL: Training LLM Agents for Long-Horizon Decision Making through Multi-Turn Reinforcement Learning. arXiv:2509.08755.
<https://arxiv.org/abs/2509.08755>
 5. Zhou, S., et al. (2023; revised 2024). WebArena: A Realistic Web Environment for Building Autonomous Agents. arXiv:2307.13854.
<https://arxiv.org/abs/2307.13854>
 6. Mialon, G., et al. (2023). GAIA: a benchmark for General AI Assistants. arXiv:2311.12983.
<https://arxiv.org/abs/2311.12983>
 7. Deaimer. ColdStart - Audited AI Agent Recovery Benchmark. Public benchmark description. Accessed 12 October 2026.
<https://www.deaimer.com/benchmarks/coldstart>
 8. Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. (2025). Mastering diverse control tasks through world models. Nature 640, 647-653.
<https://doi.org/10.1038/s41586-025-08744-2>
 9. Wang, R., Lehman, J., Clune, J., and Stanley, K. O. (2019). Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions. arXiv:1901.01753.
<https://arxiv.org/abs/1901.01753>
 10. Deaimer. Private Coding-Agent Datasets for Licensing. Public offering description. Accessed 12 October 2026.
<https://www.deaimer.com/datasets>
 11. Deaimer. RL Environments & Coding-Agent Evaluations. Public offering description. Accessed 12 October 2026.
<https://www.deaimer.com/rl-environments>
 12. Deaimer. Termetheus and the Foundations of Environment Based AI Learning. Public research article. Accessed 12 October 2026.
<https://www.deaimer.com/research/termetheus-and-the-foundations-of-environment-based-ai-learn-wwqsr>
- Research perspective: cited studies provide context for the broader field. Illustrative tasks, the experience-layer framework, transfer categories, and the experimental template are this article's proposals. No customer results or private task content are disclosed.