

The AI Cold Start Problem Is a Platform Problem

By: [Pete Bahmani](#), Staff Mission Success Manager

Cold Start Affects Both Users and Builders

The empty chat box is the clearest sign that the platform does not yet understand the organization or the work its users need to do.

An AI system can answer in seconds without having the context required to do the work. The AI cold start problem is the setup that follows: users and builders have to recreate information, agents, workflows, and controls that the platform should already carry forward.

When a user starts a task in a new AI application, the system often knows very little about their situation. It may not know who the person is, what information they are allowed to use, where that information lives, which sources the organization trusts, or what the finished product should look like. The user has to explain all of that before the system can provide something useful.

The same problem happens behind the scenes. When a team builds a new AI application, it often reconnects the same data, recreates the same permissions, rebuilds similar agents and workflows, and establishes a new way to test and monitor the system. The organization keeps solving problems it has already solved elsewhere. A dedicated tool may be the faster route for one workflow, but the cost returns with every use case after it.

These are two sides of the same cold start problem. When the platform does not preserve organizational context and reusable capabilities, builders start over when they create an application and users start over when they use it. A mature platform carries that setup forward and combines its data, agents, workflows, and controls into an experience designed for the work at hand. The empty chat box is where the user sees the absence of that foundation.

Chat still has an important role. It works well when someone is exploring an unfamiliar subject, developing an idea, or trying to understand what question to ask. It is less useful as the default interface for work that already follows a known process. If the sources, required format, review steps, and approval rules are already established, the system should bring those elements together for the user instead of asking them to reconstruct the process through a series of prompts.

This is why the speed of the first response can be misleading. A model may answer in seconds, but the user can still spend several rounds correcting its assumptions, providing missing context, and reshaping the output before anyone can review, approve, or act on it. The meaningful measure is not how quickly the system says something; it is how quickly the user receives something they can use.

Consequential Work Amplifies the Cold Start Cost

The more important the work, the less acceptable it is to make the user teach the system how to do it.

An intelligence analyst may need to turn a new set of reporting into an assessment. A general AI tool can produce a response immediately, but it does not automatically know the mission, the reporting period, the intended audience, or which sources the analyst trusts. It may not know the required format, how claims should be cited, or how the organization communicates confidence and uncertainty.

Even then, the first response may only look finished. It might focus on the wrong reporting, leave out a required section, combine conflicting accounts as if they agree, or organize the material in a form the analyst cannot use. The first few exchanges are spent setting up the system. The next few are spent correcting it. Only then does the analyst get to the work that requires judgment.

Defense and intelligence environments make this harder. Relevant information may reside across different systems, networks, classification levels, and formats. No system can assume that every user is allowed to see everything. The finished product must also show where the information came from, distinguish reporting from judgment, and preserve the approvals behind the final decision.

The work does not end when one person's session does. Operations continue across shift changes and personnel rotations. When the context lives only inside a private chat, it leaves with the user. The next person has to reconstruct what happened, which sources mattered, and which decisions were already made.

A platform should keep that context with the work. The incoming user should inherit the current state, supporting information, prior decisions, and outstanding questions instead of beginning again with an empty chat box.

Shadow AI Reveals a Missing Shared Capability

When several people build the same AI workaround, the organization is looking at a missing shared capability.

Shadow AI is the use of AI tools, models, prompts, or agents outside the systems and controls an organization has established. It can include employees sending information to an unauthorized commercial service, but it can also include personal prompts, local templates, and custom agents that no one else can see, evaluate, or reuse.

Shadow AI often appears when the approved platform does not support the work people need to do. Users create workarounds because they are tired of repeating the same setup or because the available tools do not fit their process. Several people may unknowingly build different versions of the same writing assistant, comparison tool, report generator, or document-review workflow.

Those workarounds do not all create the same risk.

If someone sends controlled information to an unauthorized external service, the organization has an immediate security problem. That use should be stopped and the exposure addressed. An unauthorized model running inside the network gets contained the same way. Containment stops the exposure but not the need behind it, so someone should also ask those users what task they were trying to do.

If people are building prompts, templates, or agents inside an approved environment, the response should be different. Those solutions reveal recurring needs that the platform does not yet meet. The organization should collect them, understand the work they support, and look for patterns across teams.

That missing capability might take the form of a reusable writing template, comparison method, review workflow, skill, or mission-specific object. Once it has been tested and governed, other users can configure it for their work instead of rebuilding it independently.

Governed Foundations Make Capabilities Reusable

A platform should make each new experience easier to build because the hard parts are already in place.

When a team needs a new AI experience, it should not have to rebuild everything from the beginning. The platform should already know how to identify the user, protect the organization's information, connect to trusted sources, coordinate models and tools, and monitor whether the system is working as expected.

What the platform provides	What the user does not have to explain	What the builder can reuse
Security and access	Who they are, what they may see, what they may do, and which actions require approval	Login, permissions, audit records, approval rules, and security policies
Organizational knowledge	Which sources are trusted, where relevant information lives, what the organization calls things, and how records relate	Data connections, source tracking, document handling, and common ways to represent people, places, assets, and relationships

Agent coordination	Which model, agent, or tool to use, what order they should run in, and what to do when one fails	Model selection, tool execution, saved state, testing, retries, fallback options, and human escalation
Reusable ways of working	How to turn a recurring task into detailed technical instructions	Skills, templates, workflows, comparison methods, review steps, and common mission concepts
Purpose-built experiences	Which capabilities need to be assembled for the task	Governed building blocks that can be combined around a new customer outcome without creating another AI system

[Security and governance apply across the entire platform](#). The same identity, permissions, approval rules, and audit history should follow the data, agents, workflows, and user experience. A new application should not have to create its own version of those controls.

The platform should also carry forward what it has learned about the organization. Different teams may use different sources, but they should not need separate methods for connecting data, [tracking where information came from](#), applying permissions, or understanding common people, places, assets, and relationships.

Agent orchestration is the part of the platform that coordinates the AI components behind the experience. It chooses the appropriate models and tools, keeps track of the work, [checks whether the expected result was produced](#), and determines what happens when something fails. That response might be to try again, use another approved model, stop the action, or return the decision to a person.

A reusable abstraction is a common capability, such as structured writing, source comparison, or approval, that the platform packages so more than one application can use it. These capabilities sit above the technical foundation and let builders combine proven ways of working instead of recreating the process each time.

The user should not have to see or manage the platform layers. An analyst should see an assessment to complete. A researcher should see evidence to compare. A maintenance team should see an asset, its history, and the work that needs authorization.

Reusable Abstractions Turn Capability Into Experience

Each new AI experience should reuse what the organization has already paid to build.

Many program offices fund each AI use case as a separate project. A team building an intelligence tool, maintenance application, or procurement workflow may reconnect the same data, recreate similar review steps, and build another version of capabilities that already exist elsewhere.

Some capabilities apply to almost every kind of work. The platform may already know how to bring in source material, populate a template, compare evidence, identify missing information, track citations, coordinate review, and route a product for approval.

Other capabilities give that work meaning within a particular mission. An intelligence application may work with requirements, reporting, hypotheses, and confidence. A maintenance application may work with facilities, assets, faults, parts, and work orders. A procurement application may work with requirements, vendors, proposals, and evaluations.

The platform becomes more valuable when improvements flow back into this shared set of capabilities. If a user creates a better template, comparison method, prompt sequence, or review workflow, the organization should be able to test it, place it under the appropriate controls, and make it available to others.

This changes where the user starts. The analyst from the earlier example no longer opens an empty chat box. The system already knows the analyst's permissions, connects to the approved sources, provides the required product format, and includes the appropriate review steps.

The analyst still decides the question, reporting period, and judgment to make. Those decisions belong to the analyst. The platform handles the repeatable setup so the analyst can begin with the work that requires expertise.

Three Deployments Test Different Parts of the Argument

Field results help an evaluation only when they say which claim they support.

The available field evidence tests different parts of this argument. It shows the benefit of assembling context and capabilities before the user begins, and it shows that new mission software can be composed quickly on an existing foundation. It does not yet isolate the cost reduction created by reusing one abstraction across several distinct deployments.

At Scarlet Dragon 26-01 in December 2025, the XVIII Airborne Corps G2 evaluated Centurion by Legion Intelligence on its tactical SIPR network (IL6), fully disconnected with no cloud reach-back. The system used task-specific agents built around real staff workflows such as intelligence summary drafting. Analysts determined which authoritative reporting the system could use.

Intelligence summary production that had required roughly four hours of analyst effort was drafted in approximately five minutes, then reviewed and refined by analysts before

dissemination: [48x faster](#). That result shows what happens when the agent, the sources, and the product requirements are assembled before the analyst sits down. It does not separate how much of the gain came from assembled context and how much came from drafting speed.

At [Project Convergence Capstone 6](#), a planning need surfaced while Legion engineers were embedded with a Future Operations planning cell. A working application was built on tactical SIPRNet in under 24 hours, inside the environment the cell was already using, inheriting its existing data connections, permissions, and approval gates. That result demonstrates rapid composition on an existing governed foundation, although it does not by itself measure how much prior abstraction reuse reduced the cost of the build.

At a U.S. Department of Energy national laboratory, Legion built a template-based drafting tool for [Work Planning and Control documents](#). It combines source collections, templated writing, gap analysis, supporting images, role-based review, and staged approval. Once approved, a plan returns to the knowledge base and becomes source material for the next one.

This demonstrates capability accumulation within one experience. Facility maintenance planning, scientific work, and procurement packages can draw on many of the same capabilities, but the deployment does not yet measure the reduction in cost or delivery time across those adjacent experiences. The tool is in final testing with more than 40 users, and cycle-time and gap-accuracy metrics have not yet been gathered.

The Next Use Case Should Start Further Ahead

What users have already built shows what the platform is missing.

Saved prompts, local templates, and personal agents show which tasks users repeat, which sources they trust, and what their finished work should include. They are not complete requirements, but they reveal capabilities the platform does not yet provide.

Ask what a new experience inherits when it is added. It should reuse existing permissions, data connections, tools, workflows, testing, and approval controls. If the team must rebuild those elements, the organization is funding another point solution and paying the cold start cost again.

See how [Legion Packs](#) arrive with the context and controls a role needs already assembled.

Continue the series

Follow Legion Intelligence on LinkedIn: [linkedin.com/company/legion-intel](https://www.linkedin.com/company/legion-intel)

Explore all Command Papers: legionintel.com/command-papers

Ready to talk: legionintel.com/contact