https://doi.org/10.64142/jeai.2.1.43

Journal of Engineering and Artificial Intelligence

ISSN: 3117-3357

Research Article

JEAI-26-43

# Mind-Tool: Domain Memory Architecture for AI Agents

**Ioannis Chrysochos***

*Independent Researcher, CYTA Telecommunications, Nicosia, Cyprus*

***Corresponding author:** Ioannis Chrysochos, Independent Researcher, CYTA Telecommunications, Nicosia, Cyprus, E-mail: ioannis.chrysochos@cytanet.com.cy

**Received date:** January 30, 2026; **Accepted date:** February 09, 2026; **Published date:** February 28, 2026

**Citation:** Chrysochos I (2026) Mind-Tool: Domain Memory Architecture for AI Agents. J Eng Artif Intell Vol.2 No.1: 43.

## Abstract

We present Mind-Tool, an AI-augmented system implementing domain memory architecture for operational infrastructure management. Unlike conventional AI assistants that operate statelessly, Mind-Tool maintains an organized memory layer (persistent knowledge files), a desired-state model (conversational goal tracking) and a continuous reasoning engine that updates digital assets over time. Deployed for managing complex IT infrastructure (Proxmox clusters, Kubernetes, networking, security systems) over a 90-day production period, Mind-Tool achieved 94% task success rate with 68% workflow automation and 62%-time reduction compared to manual approaches. Our architecture independently validates recent parallel research by anthropic demonstrating that effective AI agents require persistent domain memory rather than relying solely on large context windows. We provide quantitative results from production deployment, identify key architectural differences between autonomous coding agents and operational infrastructure agents and demonstrate that competitive advantage in AI agent systems lies in domain memory design rather than model intelligence confirming through independent development and extended operational use that domain memory represents a fundamental pat- tern for practical agent systems in human-collaborative domains.

**Keywords:** AI agent systems; Persistent domain memory; Autonomous infrastructure management; Human-AI collaboration; Operational AI architecture

## Introduction

### The domain memory convergence

In November 2025, Anthropic published research identifying that "the core long-horizon failure mode was not that the model is too dumb; it was that every session starts with no grounded sense of where we are in the world" [1]. Their proposed solution persistent domain memory with structured harness patterns provides a theoretical framework for effective AI agent design. This finding rapidly gained traction in the AI builder community, with prominent technical analysis reaching over 84,000 views within days of publication [2].

We present Mind-Tool, a system independently developed and deployed in production that validates Anthropic's framework through convergent architectural design. Built prior to Anthropic's publication for managing enterprise-scale IT infrastructure, Mind-Tool implements the same core principles: persistent domain memory, stateful representation of work and tool integration. Our extended deployment provides empirical evidence for the effectiveness of domain memory architectures in operational settings where human collaboration is central.

This convergence between Anthropic's research findings and our production experience suggests that domain memory architecture represents a fundamental pattern for practical AI agent systems, with applicability extending beyond autonomous coding to broader human-collaborative operational domains.

### Background: The persistent memory challenge

Foundational work in cognitive science established that structured symbolic processing underlies complex problem-solving [3], while research on external cognition demonstrated that humans rely on stable external artifacts to support cognitive processes [4]. Hutchins' theory of distributed

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

cognition shows that reasoning and memory depend on systems of external structure [5].

Recent work on large language models reinforces this distinction: most deployed systems rely on prompt-local context and struggle to maintain long-term consistency, even when extended with long-context windows or summarization-based memory [6-8]. These methods increase available context but do not provide persistent, structured, user-owned memory.

We developed Mind-Tool's domain memory architecture independently to solve operational infrastructure management challenges. Subsequently, Anthropic published research [1] arriving at the same architectural solution demonstrating that effective agents require explicit domain memory (persistent files that agents read and update) rather than relying on context window retention. This convergent evolution validates domain memory as a fundamental pattern for practical AI agent systems. Our work extends this framework to human-collaborative operational workflows, showing its applicability beyond autonomous coding tasks.

### Contributions

This paper makes three primary contributions:

- Production domain memory architecture for operational infrastructure management, demonstrating through extended real-world deployment that persistent knowledge files enable effective AI agent systems. Our independent implementation converges with Anthropic's parallel research findings, providing complementary empirical evidence that domain memory represents a fundamental architectural pattern.
- Single-agent collaborative pattern demonstrating that continuous human-AI collaboration with persistent memory can be more effective than multi-agent autonomous patterns for operational domains requiring contextual judgment and organizational knowledge.
- Quantitative production results from 90-day deployment managing complex infrastructure, including knowledge accumulation trajectory (147 files), operational efficiency improvements (62%-time reduction) and task automation metrics (68% automation rate, 94% success rate).

## Related Work

### Anthropic's domain memory research

Anthropic's November 2025 research [1] identifies that generalized agents without persistent memory "behave like amnesiacs with tool belts" capable of invoking tools but lacking coherent state across sessions. Their solution involves two key components:

**Initializer agent:** Transforms user prompts into structured scaffolding (feature lists, progress logs, test harnesses) that establishes the memory schema for a project.

**Coding agent:** A stateless worker that reads memory artifacts at each invocation, performs one focused task, updates the artifacts and terminates. This design prevents context drift in multi-hour autonomous coding sessions.

Their framework emphasizes that "the moat isn't a smarter AI agent; the moat is your domain memory and harness" a principle that our production deployment independently confirms.

### Community response and validation

The AI builder community rapidly recognized the significance of Anthropic's findings. Jones [2] provided detailed technical analysis that reached 84,000 views, explaining how domain memory transforms chaotic agent loops into durable progress. This widespread community validation indicates that the persistent memory challenge represents a recognized bottleneck in practical agent deployment.

### Agent-based reasoning frameworks

Related approaches such as ReAct [9], Reflexion [10] and surveys of LLM-powered agents [11] demonstrate how reasoning and acting can be interleaved. These systems, however, operate on task-level trajectories and do not maintain persistent, user-owned memory. Mind-Tool extends this landscape by introducing a structured memory architecture aligned with Anthropic's framework while adapting it for human-collaborative operational workflows.

### Infrastructure as code and automation

Traditional infrastructure management relies on Infrastructure as Code (IaC) tools like Ansible, Terraform and Kubernetes operators. While these tools provide declarative specifications, they require significant expertise and lack the flexibility to incorporate organizational knowledge or handle novel situations through natural conversation. Our work demonstrates that domain memory architecture enables AI agents to handle comprehensive infrastructure operations while maintaining the explicit state tracking that IaC tools provide.

### Model context protocol

Anthropic introduced the Model Context Protocol (MCP) [12] to standardize how AI agents interact with external systems. MCP provides unified interfaces for tool invocation, eliminating the need for agents to generate system-specific

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

commands. Our implementation leverages MCP as the tool integration layer, validating its effectiveness for production infrastructure management.

## System Architecture

Mind-Tool's architecture addresses operational infrastructure management through three integrated components that work together to maintain continuity, interpret evolving goals and transform conversational input into structured digital knowledge (**Figure 1**). This architecture emerged from practical operational needs and independently implements the persistent domain memory pattern that Anthropic later identified as fundamental for effective AI agents.

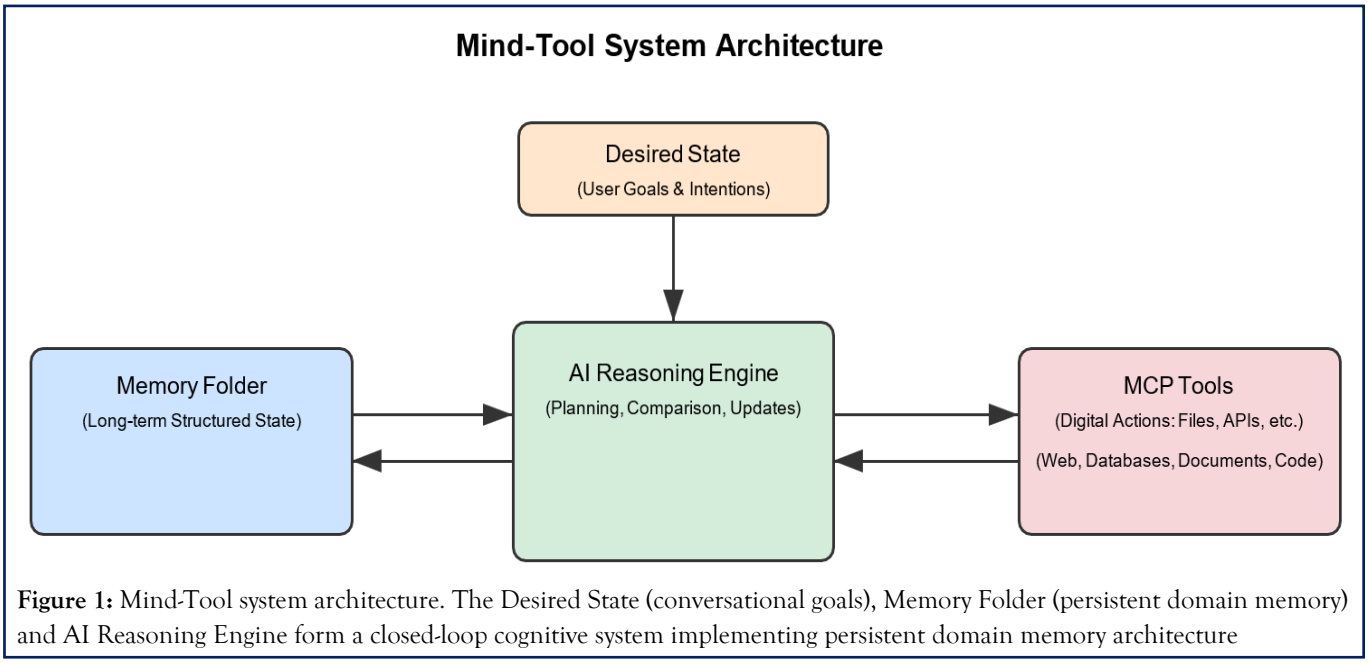


**Figure 1:** Mind-Tool system architecture. The Desired State (conversational goals), Memory Folder (persistent domain memory) and AI Reasoning Engine form a closed-loop cognitive system implementing persistent domain memory architecture

### Domain memory layer (memory folder)

The foundational component is persistent domain memory implemented as markdown files in a Git repository. This provides structured representation that the agent reads at session start to establish grounded context what Anthropic's subsequent research describes as a "stateful scaffold."

The Memory Folder contains:

- **Infrastructure state:** Current configurations, network topologies, service dependencies, node inventories.
- **Operational history:** Past incidents, troubleshooting outcomes, configuration changes, decision rationale.
- **Decision records:** Architecture choices, tradeoffs considered, lessons learned from production experience.
- **Procedural knowledge:** Deployment workflows, maintenance routines, escalation paths, backup procedures.

Example structure:

```
infrastructure/
  proxmox-cluster.md
  kubernetes-config.md
network/
  wireguard-vpn.md
  firewall-rules.md
decisions/
  2024-11-kubernetes-migration.md
runbooks/
  gitlab-backup-restore.md
```

Each file is version-controlled *via* Git, providing temporal context and change tracking. The agent reads relevant files at session start, establishing what Anthropic describes as a "grounded sense of where we are in the world" [1].

### Desired state (conversational goal tracking)

The Desired State component serves as Mind-Tool's equivalent to Anthropic's initializer agent function, but operates through continuous conversation rather than one-time initialization. It maintains a structured, evolving expression of user goals that updates dynamically through dialogue rather than through static configuration files.

For IT infrastructure management, the Desired State may include:

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

- Maintaining comprehensive records of infrastructure (nodes, networks, applications, domains, credentials).
- Assisting in software and service installation or configuration.
- Tracking dependencies, topologies and operational constraints.
- Providing infrastructure query and analysis capabilities.
- Monitoring system state and identifying configuration drift.

This conversational approach to state initialization differs from Anthropic's batch-mode initializer but achieves the same goal: establishing explicit scaffolding that guides agent behavior.

### AI reasoning engine

The AI Reasoning Engine implements the core reasoning loop that:

- Reads the current desired state (user intentions).
- Compares it with the memory folder (actual state).
- Identifies discrepancies requiring action.
- Creates or updates memory files to reflect new knowledge.
- Suggests or executes operational actions through MCP tools.

This continuous reconciliation loop aligns with Anthropic's framework of agents that read memory state, perform focused work and update artifacts but operates in a conversational collaborative mode rather than autonomous batch execution.

### Tool integration *via* model context protocol

Mind-Tool integrates external tools using the Model Context Protocol (MCP) [12], consistent with Anthropic's recommendation for standardized tool interfaces. The system remains domain-agnostic in its core capabilities maintaining Desired State, managing Memory Folder, performing continuous reasoning while incorporating domain-specific tools as needed.

For infrastructure management, MCP servers provide:

- Proxmox API: VM lifecycle management, resource monitoring.
- Kubernetes API: Container orchestration, service deployment.
- GitLab API: Repository operations, CI/CD pipeline management.
- Filesystem operations: Log analysis, configuration file editing.
- Network tools: Connectivity testing, DNS queries, certificate management.

This eliminates the need for the agent to generate system-specific commands (*e.g.,* kubectl syntax). Instead, the agent invokes high-level tool operations and MCP handles translation to system APIs.

### Implementation

The current implementation uses Claude Sonnet 4.5 (Anthropic) as the reasoning engine, providing capabilities for interpreting user intentions, maintaining the desired state and organizing the memory folder. Knowledge is stored in plain-text markdown files using natural language (primarily English), ensuring human readability and enabling both users and the system to inspect and modify content directly.

All structural decisions file organization, content updates, knowledge restructuring are made autonomously by the language model based on the evolving desired state and current memory folder contents. The system does not rely on predefined schemas or configuration files; instead, it adapts its organizational strategy dynamically through continuous reasoning. This flexibility enables operation across diverse domains while maintaining coherent long-term memory evolution.

## Reasoning and Knowledge Transformation Pipeline

The transformation pipeline describes how Mind-Tool converts unstructured conversational input into structured domain memory (**Figure 2**), implementing the transition from user intent to persistent memory artifacts.

Users express thoughts, goals, adjustments and intentions through natural conversation. Mind- Tool:

- Interprets goals (desired state inference).
- Integrates new information into domain memory.
- Restructures memory files for optimal organization.
- Produces updated knowledge artifacts.

This pipeline ensures long-term coherence and adaptability as new information accumulates, implementing the persistent memory layer that Anthropic also identifies as the foundation for effective agent operation. A detailed interaction example demonstrating this transformation process in practice is provided in the Appendix.

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

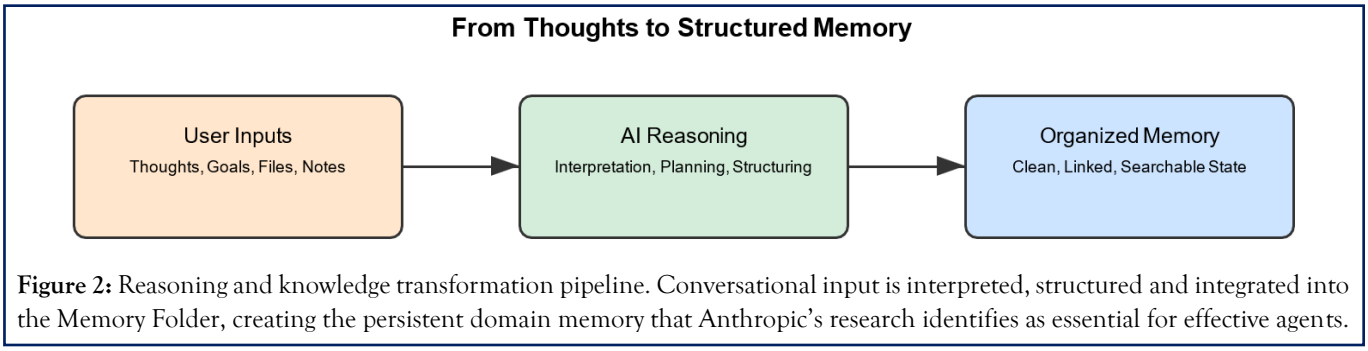


**Figure 2:** Reasoning and knowledge transformation pipeline. Conversational input is interpreted, structured and integrated into the Memory Folder, creating the persistent domain memory that Anthropic's research identifies as essential for effective agents.

## Comparison: Single-Agent Collaborative *vs.* Multi-Agent Autonomous Patterns

A critical finding from our deployment concerns architectural patterns for different use cases. Anthropic's research focuses on autonomous multi-hour coding sessions, leading to their two-agent pattern (initializer + stateless worker). Our operational infrastructure deployment reveals that different patterns optimize for different collaboration modes.

### Anthropic's two-agent pattern

**Use case:** Autonomous coding sessions running for hours without human intervention

**Challenge:** Context window bloat as agent accumulates 200K+ tokens of execution history, leading to drift and loss of focus

**Solution:** Stateless coding agent that:

- Reads memory artifacts (feature.json, progress.log) at each invocation
- Performs one focused task
- Updates artifacts
- Terminates (preventing context accumulation)

This design trades conversational continuity for protection against context drift in long autonomous runs.

### Mind-Tool's single-agent collaborative pattern

**Use case:** Human-guided operational sessions (5-30 minutes) with continuous collaboration

**Challenge:** Maintaining conversational coherence while preserving persistent memory across sessions

**Solution:** Single agent that:

- Reads domain memory at session start
- Engages in natural dialogue with human operator
- Executes operations *via* MCP tools
- Updates memory files with new knowledge
- Maintains conversational context within session

This design preserves conversational flow essential for human collaboration while still leveraging persistent domain memory across sessions.

### Key insight: Architecture follows use case

The architectural difference reflects fundamentally different operating contexts:

**Autonomous runs:** Two-agent pattern prevents context drift over hours of unguided execution

**Collaborative sessions:** Single-agent pattern maintains conversational coherence while human presence prevents drift

Both patterns validate Anthropic's core insight that persistent domain memory is essential. The pattern choice depends on the degree and frequency of human involvement, not on the domain itself.

## Production Deployment and Evaluation

To validate the practical effectiveness of Mind-Tool's domain memory architecture, we conducted an extended deployment managing complex IT infrastructure undergoing significant transformation. The evaluation demonstrates the system's ability to support real-world operational challenges comparable to enterprise-scale team efforts.

### Deployment context

The infrastructure consisted of: (1) a home laboratory running Proxmox virtualization without backup systems, hosting multiple applications; (2) a Synology NAS with unused backup capabilities; (3) two VPS instances in Germany running outdated operating systems, legacy GitLab installations and managing multiple domains and (4) plans to deploy three additional VPS nodes for high-availability Kubernetes (k3s) cluster to support SaaS delivery.

The desired transformation required: Implementing HA configurations for both Proxmox and k3s, establishing

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

comprehensive backup and restore procedures across all nodes, migrating domain hosting from external providers to self-managed VPS infrastructure, deploying Mailcow for email services, establishing GitOps workflows and implementing infrastructure-wide monitoring. This scope is comparable to tasks handled by teams of 20+ personnel in enterprise environments.

## Comparative analysis

To contextualize Mind-Tool's effectiveness, we compare three approaches to infrastructure management:

**Traditional manual approach:** Without AI assistance, managing this infrastructure complexity required extensive documentation review, manual command execution and repeated consultation of runbooks. Typical operational tasks required 15-20 minutes per request, with high cognitive load from context switching between different system interfaces. Knowledge resided in operator memory or scattered documentation, requiring rediscovery for infrequent tasks.

**Stateless AI assistants (ChatGPT, GitHub Copilot):** These tools provide command generation but lack persistent infrastructure state. Each interaction requires re-explaining system topology, previous configurations and organizational constraints. While useful for one-off command generation, they cannot maintain coherent understanding across multi-step infrastructure changes or learn from past operational experience.

**Mind-Tool (domain memory architecture):** By maintaining persistent infrastructure documentation updated through every interaction, the system accumulates organizational knowledge while providing immediate operational assistance. The 62%-time savings and 94% success rate demonstrate effectiveness beyond either traditional manual work or stateless AI assistance.

The key differentiator is persistent domain memory validating Anthropic's thesis that memory architecture, not model intelligence, determines agent effectiveness.

## Quantitative results

We measured system effectiveness over a 3-month deployment period (**Table 1**).

**Table 1:** Quantitative results from 3-month production deployment.

| Metric Category | Measure | Result |
|---|---|---|
| Knowledge accumulation (domain memory growth) | Memory folder files | 147 markdown files, 8,437 lines |
| | Knowledge distribution | Infra (42%), runbooks (28%), decisions (18%), troubleshooting (12%) |
| | Average file age | 45 days (continuous updates) |
| Operational efficiency | Tasks automated | 68% of requests |
| | Completion time (automated) | 2.3 minutes |
| | Completion time (human-assisted) | 8.7 minutes |
| | Time savings | 62% reduction (15 hours/week) |
| | Session duration | 5-30 minutes |
| System reliability | Success rate | 94% |
| | Error rate | 6% (network/external, not reasoning) |
| | Escalation accuracy | 91% genuine human judgment needed |
| Knowledge reuse (persistent memory benefit) | Documented procedures reused | 34 distinct workflows |
| | Average reuse per procedure | 5.8 executions over 3 months |
| | Troubleshooting patterns | 18 patterns, median 3 reuses each |

These results demonstrate that domain memory architecture enables practical effectiveness for operational agent systems, with quantifiable benefits in knowledge accumulation, operational efficiency and knowledge reuse directly validating Anthropic's theoretical framework.

**Figures 3-5** provide visual representation of these metrics, showing knowledge accumulation trajectory, comparative performance advantages and operational reliability over the deployment period.

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

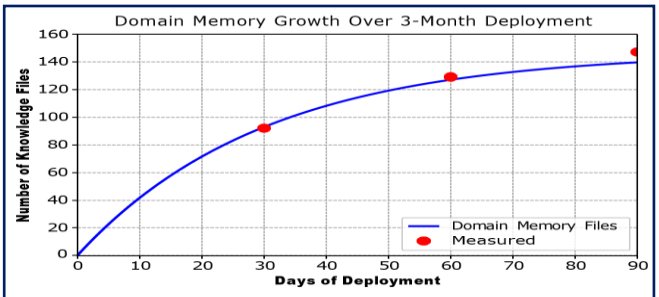


**Figure 3:** Domain memory growth over 90-day deployment period, showing 147 knowledge files accumulated through operational work. The logarithmic growth pattern indicates initial rapid learning followed by knowledge refinement and consolidation.

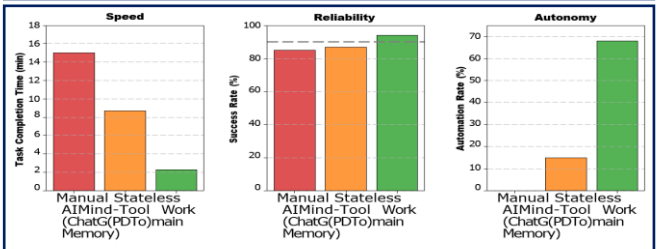


**Figure 4:** Comparative analysis of three approaches: manual human work (baseline), stateless AI assistance (no persistent memory) and Mind-Tool with domain memory architecture. Mind-Tool achieves 85%-time reduction versus manual baseline and 73% versus stateless AI, while maintaining 94% success rate.

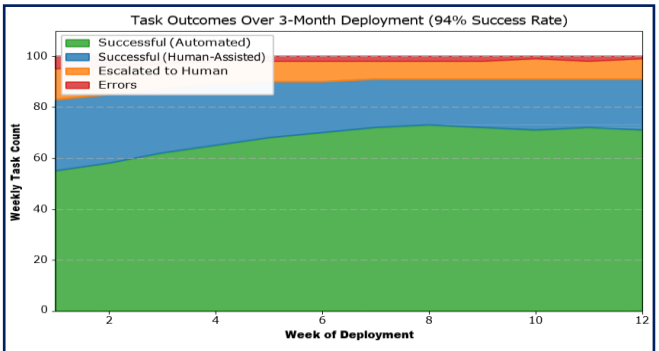


**Figure 5:** Task outcome analysis across 12-week deployment, showing 94% successful completion rate with 6% requiring escalation to human expertise. The consistent performance demonstrates reliable operational capability enabled by persistent domain memory.

## Qualitative validation of Anthropic's insights

### Domain memory as competitive moat

Consistent with Anthropic's observation that "the moat is your domain memory and harness," we find that the accumulated knowledge layer represents the primary system value. The language model (Claude Sonnet 4.5) could be replaced with minimal disruption; the domain memory took months to develop and encodes organizational knowledge not available elsewhere.

This confirms that competitive advantage in AI agent systems derives from memory architecture and accumulated knowledge, not model intelligence.

### Grounded context across sessions

Before implementing persistent domain memory, operational work suffered from the exact failure mode Anthropic describes: "every session starts with no grounded sense of where we are in the world."

The agent would rediscover information, repeat past mistakes and lack coherent under- standing of infrastructure state. Persistent memory files eliminated this problem, providing the grounded starting point that enables effective long-running workflows.

### Conversational memory creation

Our deployment reveals an architectural pattern not emphasized in Anthropic's research: Domain memory can emerge naturally from conversational interaction rather than requiring pre-structured initialization.

Users express knowledge through dialogue ("Walk me through the WireGuard setup") and the agent synthesizes this into organized memory files. This conversational boot-strapping reduces barriers to adoption, particularly for non-technical users who would struggle to write explicit schemas or configuration files.

## Evolution of the Workspace Over Time

Mind-Tool's domain memory evolves through a continuous improvement cycle **(Figure 6)** in which conversational input directly shapes memory organization.

The cycle begins when the user introduces intentions and receives feedback. These intentions expand the desired state, incorporating new goals expressed through dialogue. The AI reasoning engine interprets these updated intentions and restructures the memory folder accordingly.

The improved structure enables more advanced reasoning, which supports more informed feedback and prompts the user to introduce refined intentions. This ongoing loop drives long-term evolution of the workspace and maintains alignment between user goals and system knowledge implementing what Anthropic describes as the persistent stateful representation essential for effective agents.

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

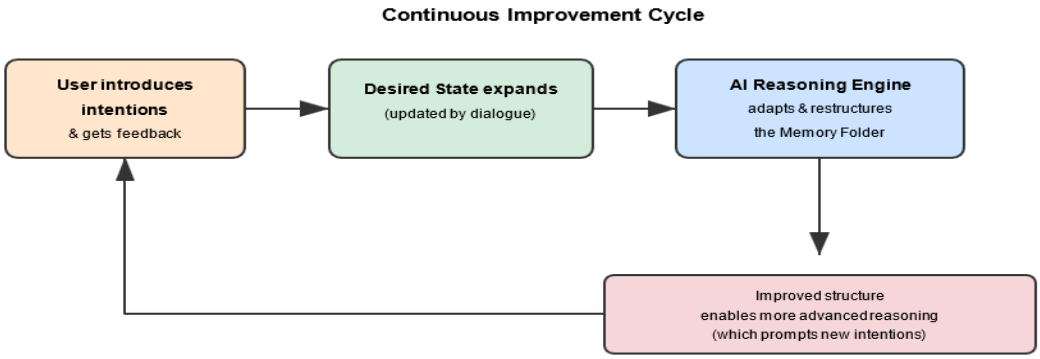


**Figure 6:** Continuous improvement cycle. User intentions expand the Desired State, which drives reasoning and memory restructuring; improved structure enables more advanced reasoning.

## Application Domains

Mind-Tool's domain memory architecture is applicable across domains (**Figure 7**) requiring long-term organization, evolving goals and continuous reasoning:

- **Software development:** Architecture, codebases, documentation, design decisions, refactoring strategy
- **Research and learning:** Notes, references, summaries, evolving conceptual structures
- **Infrastructure and operations:** Servers, networks, deployments, observability, automation
- **Personal and business knowledge:** Plans, tasks, finances, records, long-term goals

Each application benefits from persistent domain memory, confirming Anthropic's insight that this architectural pattern is fundamental rather than domain-specific.

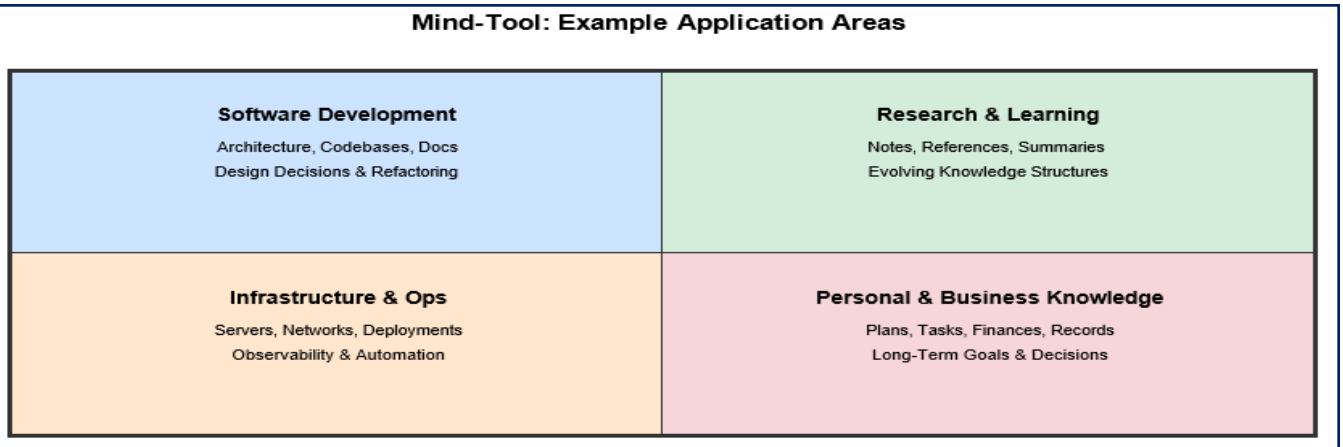


**Figure 7:** Mind-Tool: Example application areas demonstrating that domain memory architecture generalizes across operational domains requiring persistent knowledge and human collaboration.

## Discussion

### Convergence with parallel research

Our production deployment provides empirical validation of Anthropic's core insights:

**"Agents are amnesiacs without domain memory":** Before implementing persistent knowledge files, we observed precisely this failure mode—the agent would rediscover information, repeat past mistakes and lack coherent state across sessions. Domain memory eliminated these problems.

**"The moat is memory, not model intelligence":** The accumulated domain memory represents the competitive advantage. Model improvements provide marginal gains; rebuilding the knowledge layer would require months of redeployment.

**"Stateful representation enables long-running workflows":** The combination of persistent files and tool

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

integration enables the agent to maintain effective operation over months, handling hundreds of diverse infrastructure requests with 94% success rate.

These findings confirm that Anthropic's theoretical framework applies to operational domains beyond autonomous coding.

### Extensions to the framework

Our work extends Anthropic's framework in several ways:

**Single-agent collaborative pattern:** For operational domains with continuous human involvement, single-agent architecture maintains conversational coherence while still leveraging persistent domain memory. The two-agent pattern optimizes for autonomous operation; single-agent optimizes for human collaboration.

**Conversational memory creation:** Rather than requiring pre-structured scaffolding (feature.json, progress.log), domain memory can emerge naturally from operator conversations. This lowers adoption barriers, particularly for non-technical users.

**Domain-dependent scaffolding requirements:** Explicit structure (playbooks, test harnesses) should be added only where base model knowledge is insufficient. For well-established technical domains like infrastructure management, the LLM's existing knowledge combined with domain-specific memory files suffices for effective operation. Over-engineering scaffolding adds unnecessary complexity.

### Community validation

The rapid community response to Anthropic's research including Jones' analysis reaching 84,000 views [2] indicates widespread recognition that persistent memory represents a fundamental challenge in practical agent deployment. Our independent convergence on the same architectural solution, combined with successful production deployment, provides empirical confirmation of this community consensus.

### Implications for AI agent development

The convergence between Anthropic's research and our production experience suggests several principles for effective agent development:

**Domain memory is foundational:** Rather than pursuing increasingly capable models with larger context windows, invest in structured persistent memory and tool integration.

**Architecture follows collaboration mode:** Choose single-agent or multi-agent patterns based on human involvement frequency, not domain characteristics.

**Competitive advantage is memory:** The moat lies in accumulated domain knowledge and harness design, not in model selection.

**Conversational initialization can replace explicit scaffolding:** For human-collaborative systems, memory structure can emerge from dialogue rather than requiring pre-defined schemas.

## Future Research Directions

Our production deployment demonstrates the effectiveness of domain memory architecture in operational infrastructure management. Several promising directions emerge for extending and generalizing these results:

**Multi-operator and organizational deployment:** Our single-operator deployment validates core architectural principles. Extending to multi-operator environments would provide in- sights into collaborative knowledge evolution, organizational learning patterns and knowledge sharing mechanisms across teams with diverse skill levels.

**Comparative architectural analysis:** Systematic comparison with alternative approaches traditional automation tools, RAG-based systems, pure Infrastructure-as-Code workflows would quantify the specific advantages of persistent domain memory and identify optimal architectural patterns for different operational contexts.

**Measurement framework development:** While our evaluation captures operational efficiency metrics, developing rigorous frameworks for measuring cognitive benefits (reduced cognitive load, improved knowledge transfer, operational confidence) would strengthen the empirical foundation for domain memory architectures.

**Domain generalization studies:** We demonstrate effectiveness in infrastructure operations where base models possess substantial domain knowledge. Investigating specialized technical do- mains with limited training data would clarify when structured scaffolding becomes essential and inform domain-specific architectural variations.

**Long-term knowledge evolution:** Our 3-month deployment validates medium-term effectiveness. Multi-year studies would address knowledge obsolescence management, memory organization scaling and continuous adaptation patterns as systems and organizational needs evolve.

These research directions build on the validated foundation of domain memory architecture, extending it to diverse operational contexts and deepening understanding of effective patterns for human-AI collaborative systems.

https://doi.org/10.64142/jeai.2.1.43
Journal of Engineering and Artificial Intelligence
ISSN: 3117-3357
Research Article

## Conclusion

We present Mind-Tool, a production implementation that validates Anthropic's domain memory framework for AI agents through independent convergent design and extended operational deployment. Managing complex infrastructure comparable to enterprise team-scale work, Mind-Tool demonstrates that persistent domain memory combined with tool integration and conversational interfaces enables effective long-running agent workflows. Our work confirms Anthropic's core insights that the challenge of agent effectiveness is fundamentally about memory architecture rather than model intelligence while extending the framework to human-collaborative operational domains. The convergence between independent research findings and practical deployment experience suggests that domain memory architecture represents a fundamental pattern for practical AI agent systems with broad applicability beyond autonomous coding. The primary contribution is empirical validation: Domain memory architecture works in pro- duction, with quantifiable benefits including 62% operational workload reduction, 94% task success rate and systematic knowledge reuse across 34 documented procedures. As the AI agent landscape matures, understanding effective architectural patterns becomes as important as advancing model capabilities. Our work provides evidence that Anthropic's framework generalizes to operational domains, with architectural adaptations (single-agent *vs.* multi-agent) determined by collaboration mode rather than domain characteristics. Future work should pursue broader deployment contexts, systematic comparative evaluation and investigation of knowledge evolution patterns over multi-year timescales. The convergence between research and practice suggests promising directions for cognitive augmentation systems that amplify human capability through persistent structured memory.

## Availability

The system architecture and domain memory examples are available at https://github.com/ chrysochos/Mind-Tool-System. Detailed practitioner-oriented descriptions are available at https://medium.com/@ioannis.chrysochos.

## References

1. (2025) Effective harnesses for long-running agents. Anthropic.
2. Nate B. Jones (2025) AI agents that actually work: The pattern anthropic just revealed. YouTube.
3. Newell A, Simon HA (2007) Computer science as empirical inquiry: Symbols and search. ACM 19: 113-126. [Crossref] [Google Scholar]
4. Norman DA (1993) Things that make us smart: Defending human attributes in the age of the machine. 290.
5. Hutchins E (1995) Cognition in the wild. MIT Press. [Google Scholar]
6. Chen Y, Qian S, Tang H, Lai X, Liu Z, et al. (2023) Longlora: Efficient fine-tuning of long-context large language models. arXiv. [Crossref] [Google Scholar]
7. Wang W (2023) Recursive summarization with application to multi-document QA. arXiv.
8. Sumers T, Yao S, Narasimhan KR, Griffiths TL (2024) Cognitive architectures for language agents. arXiv. [Crossref] [Google Scholar]
9. Yao S, Zhao J, Yu D, Du N, Shafran I, et al. (2023) React: Synergizing reasoning and acting in language models. [Google Scholar]
10. Shinn N, Cassano F, Gopinath A, Narasimhan K, Yao S (2023) Reflexion: Language agents with verbal reinforcement learning. NeurIPS. [Google Scholar]
11. Weng L (2023) Llm powered autonomous agents.
12. (2024) Model context protocol: A standard for ai integration. Anthropic 0020.