Building MCP-Native Hierarchical AI Scientist Ecosystems: A Perspective on Scaling Multi-Agent Scientific Discovery

Ling Yue , Ching-Yun Ko , Pin-Yu Chen , Shimin Di , Shaowu Pan *

Posted Date: 5 March 2026

doi: 10.20944/preprints202507.1951.v2

Keywords: autonomous scientific discovery; multi-agent systems; hierarchical coordination; Model Context Protocol; tool interoperability; agent ecosystems; organizational design; AI for science

 doi:10.20944/preprints202507.1951.v2

Article

Building MCP-Native Hierarchical AI Scientist Ecosystems: A Perspective on Scaling Multi-Agent Scientific Discovery

Ling Yue [1], Ching-Yun Ko [2], Pin-Yu Chen [2], Shimin Di [3] and Shaowu Pan [1,*]

[1] Rensselaer Polytechnic Institute, Troy, NY, USA
[2] IBM Research, Yorktown Heights, NY, USA
[3] Southeast University, Nanjing, China
* Correspondence: pans2@rpi.edu

Abstract

Large language models (LLMs) are evolving from chatbots with limited tool-using capabilities to agentic AI systems that can perform deep research, assist in proposing hypotheses, help design experiments, automate data analysis, and draft scientific reports. However, there are currently two bottlenecks limiting LLMs' real-world impact on the broader scientific research community beyond academic demonstrations: *lack of interoperability* (repetitive manual tool-integration is required across scenarios) and *the need for scalable coordination* (unstructured communication and memory become brittle as the number of agents grows). In this Perspective, we argue that the next phase of agentic scientific discovery requires the development of an *ecosystem* of protocol-native agents and tools organized through hierarchies inspired by human society, beyond the current paradigm of a single monolithic "AI scientist". We use Model Context Protocol (MCP) as a concrete example of an emerging interoperability layer for scientific tool and context exchange, and we propose three complementary pathways to increase the scaling capabilities of an MCP-native scientific ecosystem by addressing the composability issues: (1) MCP servers for high-value scientific tools maintained by domain experts, (2) automated transformation of existing code repositories into MCP services, and (3) autonomous invention and evolution of new agents and workflows. Finally, we provide a practical roadmap for scaling AI-driven scientific discovery by expanding tool supply and coordination in MCP-native scientific ecosystems.

Keywords: autonomous scientific discovery; multi-agent systems; hierarchical coordination; Model Context Protocol; tool interoperability; agent ecosystems; organizational design; AI for science

1. Introduction

Recent advances in large language models (LLMs) have enabled agents that can interleave reasoning with tool use [1,2] and collaborate via multi-agent conversation and orchestration frameworks [3–5]. Building on these foundations, a growing body of work explores end-to-end "AI scientist" systems that automate parts of the research lifecycle, including literature review, experimentation, and writing [6–12].

These systems suggest potential to automate and accelerate parts of scientific workflows, but they also expose a practical constraint: high-impact science depends on many tools (simulators, databases, lab instruments, visualization pipelines) and many collaborators (specialists who divide labor and integrate results) [13–15]. As the number of agents increases, each agent begins to resemble a member of a research organization: individually limited in context and attention, but collectively effective when structured well [16].

This Perspective argues that scaling from "single-agent automation" to many-agent scientific organizations requires two ingredients. First, the field needs an interoperability substrate so that tools

 doi:10.20944/preprints202507.1951.v2

and capabilities can be composed across models, labs, and domains with minimal glue code [17,18]. Second, as agent populations scale, effective systems will increasingly resemble human research organizations: hierarchical, modular, and grounded in shared artifacts that manage context, accountability, and division of labor [16].

In current systems, both requirements are often unmet [19,20]. The common failure mode of existing "AI scientist" systems is not that they cannot produce impressive closed-loop demonstrations, but that their success is brittle outside curated settings [19]. Tool access is typically rebuilt inside each host stack, execution environments and logging conventions drift across deployments [21–23], and multi-agent collaboration relies on largely unstructured communication and memory [24]. This makes transfer to new labs or domains expensive and makes it hard to audit claims back to computations and data at scale [21,22]. Our perspective synthesizes these bottlenecks and argues for a new angle: protocols make tools composable, but organizational interfaces make teams composable, and scalable scientific discovery requires both.

The broader protocol landscape has been summarized in [25]. We use the Model Context Protocol (MCP) as a concrete example of an emerging interoperability layer [26]. Complementary efforts such as Agent2Agent (A2A) emphasize agent-to-agent discovery and task-oriented communication across independently deployed agents, and are explicitly positioned as complementing MCP's agent-to-tool and context interfaces [27]. Our core organizational claims (hierarchy, shared artifacts, and provenance) are therefore largely protocol agnostic; MCP serves primarily to ground the discussion of tool and context composability. Concretely, MCP standardizes how LLM applications (hosts) connect to external services (servers) that expose tools and contextual resources through a negotiated interface. This separation helps decouple fast-moving agent/model stacks from slower-moving scientific software and infrastructure. The decoupling is especially important in scientific workflows, where tool execution depends on complex environments, data access policies, and versioned pipelines that are costly to hand-integrate and easy to break [17,28,29]. We do not propose a new protocol; rather, we argue that protocol-native ecosystems—combined with human-inspired organizational design—are necessary to scale from isolated demonstrations to reusable scientific organizations. By "MCP-native", we mean that tools and coordination services (e.g., task boards, lab notebooks, provenance stores) are exposed and composed through the protocol interface, rather than integrated via host-specific wrappers.

At the same time, interoperability alone does not produce scientific throughput. Many-agent discovery introduces predictable coordination failures: duplicated effort, inconsistent intermediate conclusions, fragmented project state, and weak traceability from claims back to computations and data [30,31]. Human research groups mitigate these issues through organizational structure and durable shared artifacts (lab notebooks, experiment logs, issue trackers, papers) [13,24,32]. We argue that MCP-native tool ecosystems and human-inspired organizational design are complementary levers: protocols make tools composable, while organizational interfaces make teams composable.

The remainder of this Perspective summarizes progress and bottlenecks in "AI scientist" systems, motivates protocol ecosystems with MCP as an example, and then presents an organizational layer and a practical roadmap for growing MCP-native scientific agent ecosystems.

2. From Standalone Agents to AI Scientist Systems: Progress and Bottlenecks

Recently, we have witnessed rapid progress in agentic workflows for research [20,33]. Systems such as *The AI Scientist* and subsequent variants show that LLMs can generate hypotheses, write code, run experiments, and draft papers with minimal human intervention [6,7]. Complementary efforts emphasize human-in-the-loop and domain grounding, including systems for research ideation [10], collaborative "co-scientist" paradigms [8], and autonomous multi-agent research loops for broader scientific tasks [9,11].

Despite this momentum, most current systems still operate as small, fixed teams in curated environments: tools are manually wrapped, execution environments are hand-configured, and the end-to-end loop is tuned to a particular benchmark or domain [34]. This makes demonstrations

 doi:10.20944/preprints202507.1951.v2

impressive but difficult to transfer. Real scientific work quickly hits the long tail of specialized repositories, heterogeneous runtimes, and evolving datasets [35]. For example, end-to-end systems such as *The AI Scientist* and its variants show compelling closed-loop automation in relatively controlled ML settings, but extending the same loop to new domains often requires substantial re-engineering of toolchains, environments, and evaluation harnesses [6,7,36–39]. Empirical studies of agents in real-world deployments further suggest that reliability and engineering constraints often dominate practical development, underscoring the need for standardized interfaces, robust logging, and auditable execution traces [19]. In short, scaling from "a working demo" to "a reusable scientific organization" is less about prompting and more about engineering reliable interfaces for tool access, shared state, verification, and accountability.

These observations highlight the following two bottlenecks that increasingly limit real-world scientific impacts. **(1)** The first is tool supply and the $N \times M$ integration problem[1]. Most agent stacks define host-specific tool abstractions and implement per-tool adapters inside the host (schemas, authentication, error handling, and often environment setup) [33]. Even when tools are exposed through generic interfaces such as CLIs or web services, contracts for session state, structured context exchange, version reporting, and audit-friendly execution traces are often implicit and vary across hosts. As a result, if N models/agents need to access M tools, ad-hoc integrations grow as $O(NM)$ and become brittle and costly to maintain [17,25]. Interoperability protocols aim to move this boundary from host-specific glue code to reusable services with explicit, negotiated contracts. MCP is a concrete and timely instance of this direction and serves as the running example in this Perspective: it standardizes how agent hosts connect to external tool/context servers, so that capability discovery, structured context exchange, and auditable traces can be provided as shared infrastructure rather than re-implemented in each stack [26]. In scientific settings, this integration burden is amplified by environment complexity (compiled dependencies, GPU/HPC constraints, container images, random seeds) and data governance (dataset snapshots, access control, provenance), and "running the tool" is often inseparable from reproducing its execution context [22,23,36]. Even when a tool call succeeds, silent behavioral drift across versions can invalidate comparisons and downstream conclusions [40]. **(2)** The second bottleneck is coordination and context management at scale. As the number of agents grows, flat communication patterns become costly and brittle because the number of potential peer-to-peer links grows quadratically [30,41]. Moreover, LLM agents have bounded context windows and imperfect long-horizon memory, making it difficult for any single agent to hold the full state of a complex research program. Scientific work also raises the bar for coordination quality: teams must track experimental conditions, manage uncertainty, and resolve disagreements through replication, ablations, or alternative methods [24]. Without durable shared artifacts and explicit traceability from claims to evidence, multi-agent systems can produce plausible narratives that are difficult to audit, reproduce, or build upon [42].

These bottlenecks motivate the two design directions developed in the rest of this Perspective: protocol ecosystems (with MCP as a concrete instance) to reduce integration debt and grow tool supply, and a human-inspired organizational layer to shape information flow, externalize memory into shared artifacts, and preserve scientific-grade accountability as agent populations scale.

3. Why Protocol Ecosystems Matter: MCP as an Interoperability Layer

We argue that "agentic science" should be built on open, protocol-based ecosystems rather than bespoke, closed integrations [17,29]. MCP is one prominent example: it standardizes how hosts (LLM applications) connect to servers that provide resources, prompts, and tools, using JSON-RPC 2.0 over stateful connections with capability negotiation. MCP is explicitly inspired by the Language Server Protocol (LSP), which enabled a large ecosystem of reusable language tooling across editors via a common protocol [26,28].

[1] N refers to the number of models/agents, and M refers to the number of tools.

 doi:10.20944/preprints202507.1951.v2

From a scientific discovery perspective, protocol ecosystems offer four benefits. First, they enable composable tool access with clear boundaries. Protocol-defined tools can be versioned, tested, and sandboxed independently of the agents that call them, which decouples fast-moving agent stacks from slower-moving scientific software and infrastructure. Second, they treat context as a first-class interface. Scientific work is not only "calling functions"; it requires passing rich context such as hypotheses, experimental settings, data provenance, and intermediate conclusions. MCP's notion of resources and stateful sessions supports richer, more structured context exchange than prompt-only interfaces [26]. In our view, context management should be treated as an interface design problem, not only a prompting trick. Third, they lower the cost of reproducibility and provenance [43]. When tools are exposed through standardized interfaces, we can more systematically log tool calls, inputs/outputs, and environment versions. Protocols do not guarantee rigor on their own, but they make it easier to build rigorous, auditable pipelines [23]. Finally, a protocol is only as useful as the ecosystem of servers that implement it. We therefore emphasize the "tool supply" problem as a first-class research and engineering challenge [35,44]. For example, Code2MCP [45] provides an automated pathway to populate MCP ecosystems by converting existing GitHub repositories into deployable MCP services with minimal human intervention. Such agentic conversion methods hold great potential to unlock the long tail of domain software required for scalable scientific discovery.

4. Scaling Many-Agent Discovery Needs Human-Inspired Organizational Design

MCP-like interoperability reduces the $N \times M$ integration burden between agents and tools, but it does not by itself make large agent populations effective. As the number of agents grows, the dominant bottleneck shifts to coordination: unstructured communication becomes noisy and expensive, and project state fragments across bounded-context agents [30]. Human research groups mitigate these constraints through organizational structure and shared artifacts [14,24]. We argue that scalable many-agent discovery similarly requires an explicit organizational layer—a set of interfaces for task allocation, shared state, conflict resolution, and accountability—implemented on top of protocol-native tools and resources.

4.1. What Breaks When We Scale Beyond Small Agent Teams

When coordination is left implicit, large agent teams tend to fail in common and repeatable ways. Large agent teams often exhibit redundant work, where multiple agents unknowingly pursue the same subtask and produce inconsistent partial results [30]. Project state can fragment across private chat histories, causing assumptions and intermediate findings to drift over time and eroding a shared "project truth". Communication also scales poorly in flat peer-to-peer structures: the number of possible interactions grows quadratically, diluting high-signal updates with chatter [24]. As tool access expands, inconsistent beliefs and unresolved conflicts accumulate because there is no systematic mechanism to detect, adjudicate, and document disagreements. Finally, without structured provenance, claims become difficult to trace back to specific tool executions, data versions, and experimental settings, weakening accountability and reproducibility. These breakdowns are not model-specific; they are structural consequences of scaling many bounded-context agents [42,46].

4.2. Organizational Primitives as First-Class Interfaces

To counter these failures, we advocate an artifact-centered organizational layer built from a small set of primitives that can be standardized and reused across domains:

- **A task interface (assignment, ownership, and budgets).** Every subtask should have an explicit owner, inputs, success criteria, and resource limits. This supports de-duplication, progress tracking, and controlled exploration [30,31].
- **A shared artifact interface (externalized state).** Hypotheses, datasets, experiment configurations, results, and summaries should live as versioned, inspectable artifacts rather than private chat context. These artifacts become the durable "memory" of the organization [16,42].

 doi:10.20944/preprints202507.1951.v2

- **A decision and disagreement interface (escalation and repair).** When results conflict, the system should create structured disagreement records and trigger repair actions (replication, ablations, alternative methods, or human review). This reduces silent divergence [32,47].
- **A provenance interface (traceability).** Claims should link to evidence artifacts and execution traces (tool calls, environment versions, data lineage), supporting audit and reproducibility [23, 48].

In a MCP-native ecosystem, these primitives can be implemented as protocol-accessible servers and resources (e.g., task boards, lab notebooks, provenance stores), making coordination logic modular and composable across labs and agent frameworks.

4.3. A Minimal Hierarchy is an Information Contract, Not Just a Role Chart

Figure 1 illustrates a pragmatic three-layer hierarchy that functions primarily as an information contract rather than a static role chart. At the task level, specialist workers execute concrete actions via MCP tools and write back result artifacts together with execution traces. At the project level, domain managers allocate tasks, maintain shared project context through curated artifacts, reconcile conflicts, and distill evidence into structured summaries and decision records. At the program level, meta-orchestrators define long-horizon objectives, allocate budgets across projects, and integrate project-level summaries into a portfolio view of progress and uncertainty.

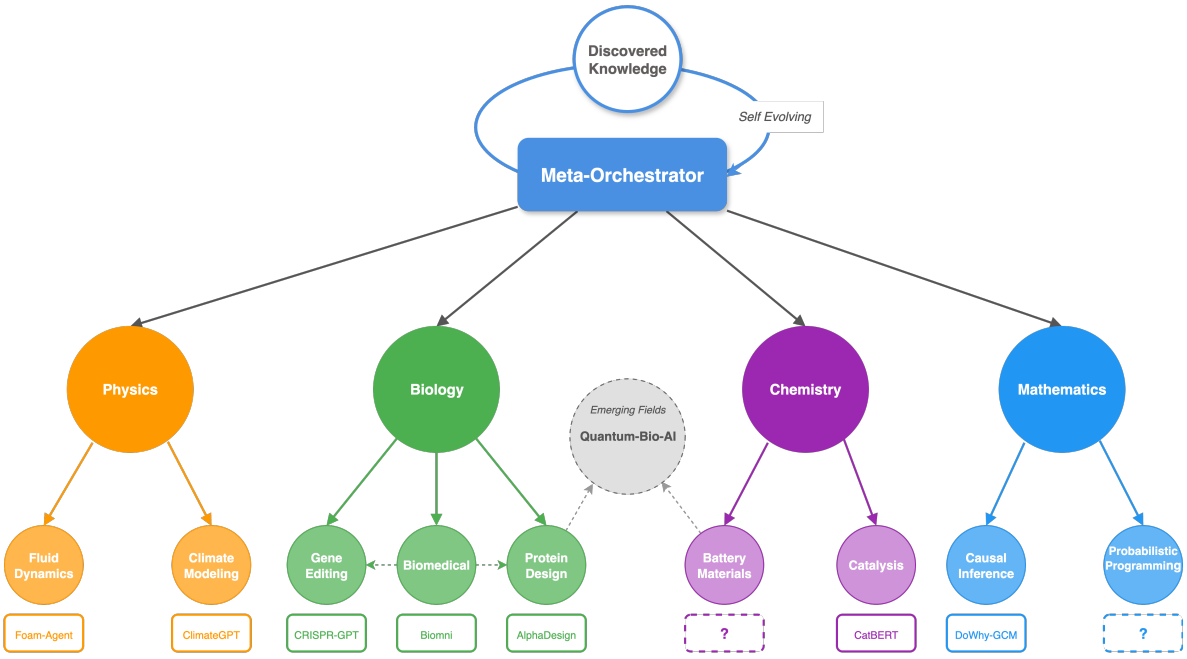

Figure 1. An MCP-native hierarchical AI scientist ecosystem. A human-inspired hierarchy organizes many agents into meta-orchestrators that manage long-horizon research programs, domain managers that coordinate projects and maintain mid-level context, and specialist workers that execute concrete tasks via MCP tools.

The hierarchy constrains communication by design [16]. Most information exchange remains local between workers and managers, while only compressed, decision-relevant updates propagate upward. Long-horizon state is concentrated at higher levels and externalized through shared artifacts, reducing dependence on any single agent's context window. In this sense, hierarchy scales not by increasing conversation volume but by shaping information flow that mirrors how human laboratories grow through specialization combined with explicit interfaces for tasks, artifacts, and accountability [41].

5. Roadmap: Three Pathways to Grow an MCP-Native Scientific Agent Ecosystem

To make this vision actionable, we outline three complementary pathways for bootstrapping and scaling an MCP-native ecosystem (Figure 2).

 doi:10.20944/preprints202507.1951.v2

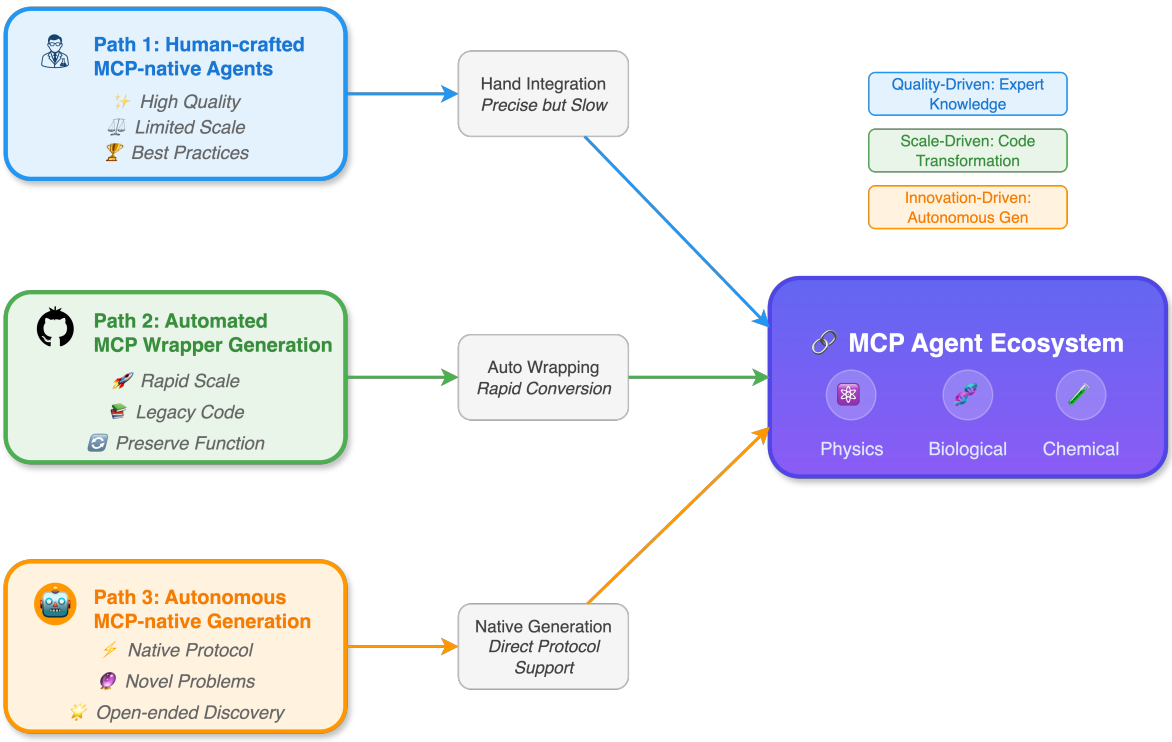

Figure 2. Three pathways to build an MCP agent ecosystem. Human-crafted MCP-native agents (Pathway 1) provide high-quality foundations, automated MCP wrapper generation (Pathway 2) enables rapid scaling from legacy code, and autonomous MCP-native generation (Pathway 3) creates novel agents for unprecedented problems. All three paths converge into a unified MCP agent ecosystem spanning diverse scientific domains.

Pathway 1: expert-crafted MCP servers for high-value scientific tools. In the short term, the highest-leverage approach is to build MCP servers for widely used scientific tools (e.g., simulation packages, lab automation APIs, scientific databases). Human experts can encode domain constraints, safety checks, and best practices. These "gold standard" servers can also serve as reference implementations for evaluating automated conversion methods [36,38,39].

Pathway 2: automated code-to-MCP transformation to unlock the long tail. Most scientific capabilities live in open-source repositories that were never designed as agent-ready services [36]. Automated frameworks like Code2MCP propose converting arbitrary repositories into MCP services through multi-agent workflows that analyze code, reproduce environments, design tool schemas, and iteratively debug via self-correction loops [45]. We view this pathway as essential for scaling tool supply and enabling scientific agents to access specialized methods without extensive hand engineering [40].

Pathway 3: autonomous invention and evolution of new agents and workflows. In the longer term, the ecosystem should not only wrap existing tools but also invent new ones. Recent work on automated agent design and self-improving systems that include Automated Design of Agentic Systems (ADAS), AFlow, and the Darwin Gödel Machine, suggests that agents can search over workflow code and iteratively improve themselves using execution feedback [49–51]. In an MCP-native setting, new agents can be materialized as new MCP servers (tools) or as new orchestration policies (managers), expanding the ecosystem over time [52].

6. Discussion

Our perspective reframes "AI scientist" progress as an ecosystem problem. Once protocol layers like MCP reduce the $N \times M$ integration burden between agents and tools, the main constraints become whether tool supply can scale with scientific-grade reliability and whether large agent populations can coordinate through organizational structure and shared artifacts.

 doi:10.20944/preprints202507.1951.v2

This emphasis is specific to scientific discovery because science imposes stronger requirements than generic automation [53]. The cost of error is highly asymmetric: a false positive can waste days of compute or weeks of lab time, and a silently wrong result can contaminate downstream hypotheses [54]. Scientific workflows are also deeply environment-dependent: conclusions may hinge on software versions, random seeds, hardware, dataset snapshots, preprocessing pipelines, and experimental settings [22,23]. As a result, "doing the task" is inseparable from "being able to rerun and audit the task". This is why we emphasize artifact-centered coordination primitives (tasks, shared artifacts, disagreement/repair, and provenance), and why we view tool interfaces as contracts that should make evidence traceable rather than merely callable.

In the near term, Pathway 2 is strategically important but still brittle in practice [45]. Conversion pipelines can struggle to reproduce environments, infer correct tool schemas, and detect silent wrong outputs. This direction arguably receives less attention than its leverage warrants. Advancing it likely requires treating conversion as a testing-and-provenance problem, with standardized harnesses for environment capture, regression checks, and versioned tool contracts that make "wrap" synonymous with "trust and rerun" [40]. In scientific settings, automated wrapping should also default to conservative behaviors such as explicit version reporting, structured logging, and verification hooks for replication or sanity checks.

Over longer term, Pathway 3 raises a deeper question: what should automatically generated scientific workflows look like in a multi-agent ecosystem? One risk is that self-improving systems optimize for short-horizon success signals while accumulating hidden technical debt in tool usage and provenance. For scientific discovery, useful self-evolution must therefore be coupled with auditability and rollback: evolving orchestration policies, external memory (e.g., file- or repo-backed artifacts and workflow libraries), and possibly even model components, while preserving reproducibility and the ability to attribute claims to specific evidence.

Progress will likely depend on ecosystem-level evaluations that go beyond task success rates [55]. Practical metrics include tool coverage (how much of a domain toolchain is accessible), integration cost (time and maintenance burden per tool), coordination cost (redundant work and conflict rates as teams scale), and reproducibility under continuous evolution (the fraction of results that remain rerunnable as tools, environments, and agents change). These evaluation directions align with the core thesis of this Perspective: protocols make tools composable, but organizational design makes teams composable, and both are required for scalable scientific discovery.

Author Contributions: LY conceptualized the perspective and drafted the manuscript. CYK contributed to conceptual framing, literature positioning, and manuscript editing. PYC contributed to conceptual framing and manuscript editing. SD contributed to manuscript editing. SP contributed to conceptual framing and manuscript editing. All authors approved the final version.

Funding: This work received funding support by IBM-Rensselaer Future of Computing Research Collaboration (2026) and the U.S. Department of Energy with ID DE-SC0025425. Shaowu Pan is supported by the Google Research Scholar Program. Computing resources are supported by the Lambda research grant program, NSF-ACCESS-MCH260003 and by the National Energy Research Scientific Computing Center under award NERSC DDR-ERCAP0030714.

Data Availability Statement: No new datasets were generated or analyzed in this study.

Acknowledgments: We thank the broader open-source community building MCP servers, scientific software, and agentic tooling.

Conflicts of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

 doi:10.20944/preprints202507.1951.v2

References

1. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.R.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the The eleventh international conference on learning representations, 2022.
2. Schick, T.; Dwivedi-Yu, J.; Dessì, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems* **2023**, *36*, 68539–68551.
3. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155* **2023**.
4. Li, G.; Hammoud, H.; Itani, H.; Khizbullin, D.; Ghanem, B. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems* **2023**, *36*, 51991–52008.
5. Hong, S.; Zheng, X.; Chen, J.; Cheng, Y.; Wang, J.; Zhang, C.; Wang, Z.; Yau, S.K.S.; Lin, Z.; Zhou, L.; et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* **2023**, *3*, 6.
6. Lu, C.; Lu, C.; Lange, R.T.; Foerster, J.; Clune, J.; Ha, D. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292* **2024**.
7. Yamada, Y.; Lange, R.T.; Lu, C.; Hu, S.; Lu, C.; Foerster, J.; Clune, J.; Ha, D. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066* **2025**.
8. Gottweis, J.; Weng, W.H.; Daryin, A.; Tu, T.; Palepu, A.; Sirkovic, P.; Myaskovsky, A.; Weissenberger, F.; Rong, K.; Tanno, R.; et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864* **2025**.
9. Schmidgall, S.; Su, Y.; Wang, Z.; Sun, X.; Wu, J.; Yu, X.; Liu, J.; Moor, M.; Liu, Z.; Barsoum, E. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025* **2025**, pp. 5977–6043.
10. Baek, J.; Jauhar, S.K.; Cucerzan, S.; Hwang, S.J. Researchagent: Iterative research idea generation over scientific literature with large language models. In Proceedings of the Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2025, pp. 6709–6738.
11. Team, N.; Zhang, B.; Feng, S.; Yan, X.; Yuan, J.; Yu, Z.; He, X.; Huang, S.; Hou, S.; Nie, Z.; et al. NovelSeek: When Agent Becomes the Scientist–Building Closed-Loop System from Hypothesis to Verification. *arXiv preprint arXiv:2505.16938* **2025**.
12. King, R.D.; Rowland, J.; Aubrey, W.; Liakata, M.; Markham, M.; Soldatova, L.N.; Whelan, K.E.; Clare, A.; Young, M.; Sparkes, A.; et al. The robot scientist Adam. *Computer* **2009**, *42*, 46–54.
13. Stokols, D.; Hall, K.L.; Taylor, B.K.; Moser, R.P. The science of team science: overview of the field and introduction to the supplement. *American journal of preventive medicine* **2008**, *35*, S77–S89.
14. Council, N.R.; et al. Enhancing the effectiveness of team science. *National Research Council. 2015. Enhancing the Effectiveness of Team Science. Washington* **2015**, p. 19007.
15. Wilson, G.; Aruliah, D.A.; Brown, C.T.; Chue Hong, N.P.; Davis, M.; Guy, R.T.; Haddock, S.H.; Huff, K.D.; Mitchell, I.M.; Plumbley, M.D.; et al. Best practices for scientific computing. *PLoS biology* **2014**, *12*, e1001745.
16. Simon, H.A. The architecture of complexity. In *The Roots of Logistics*; Springer, 2012; pp. 335–361.
17. Parnas, D.L. On the criteria to be used in decomposing systems into modules. *Communications of the ACM* **1972**, *15*, 1053–1058.
18. Baldwin, C.Y.; Clark, K.B. *Design rules, Volume 1: The power of modularity*; MIT press, 2000.
19. Pan, M.Z.; Arabzadeh, N.; Cogo, R.; Zhu, Y.; Xiong, A.; Agrawal, L.A.; Mao, H.; Shen, E.; Pallerla, S.; Patel, L.; et al. Measuring agents in production. *arXiv preprint arXiv:2512.04123* **2025**.
20. Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science* **2024**, *18*, 186345.
21. Peng, R.D. Reproducible research in computational science. *Science* **2011**, *334*, 1226–1227.
22. Sandve, G.K.; Nekrutenko, A.; Taylor, J.; Hovig, E. Ten simple rules for reproducible computational research. *PLoS computational biology* **2013**, *9*, e1003285.
23. Wilkinson, M.D.; Dumontier, M.; Aalbersberg, I.J.; Appleton, G.; Axton, M.; Baak, A.; Blomberg, N.; Boiten, J.W.; da Silva Santos, L.B.; Bourne, P.E.; et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific data* **2016**, *3*, 1–9.
24. Horling, B.; Lesser, V. A survey of multi-agent organizational paradigms. *The Knowledge engineering review* **2004**, *19*, 281–316.

 doi:10.20944/preprints202507.1951.v2

25. Ehtesham, A.; Singh, A.; Gupta, G.K.; Kumar, S. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279* **2025**.
26. Anthropic. *Model Context Protocol Specification (Version 2025-11-25)*, 2025.
27. The Linux Foundation (A2A Protocol Project). *Agent2Agent (A2A) Protocol Documentation and Specification*, 2026. Official documentation site. Accessed 2026-02-26.
28. Microsoft. *Language Server Protocol*, 2021. Accessed: 2026-02-11.
29. Fielding, R.T. *Architectural styles and the design of network-based software architectures*; University of California, Irvine, 2000.
30. Malone, T.W.; Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR)* **1994**, *26*, 87–119.
31. Smith, R.G. The contract net protocol: High-level communication and control in a distributed problem solver. In *Readings in distributed artificial intelligence*; Elsevier, 1988; pp. 357–366.
32. Grosz, B.J.; Kraus, S. Collaborative plans for complex group action. *Artificial Intelligence* **1996**, *86*, 269–357.
33. Xu, W.; Huang, C.; Gao, S.; Shang, S. LLM-Based Agents for Tool Learning: A Survey: W. Xu et al. *Data Science and Engineering* **2025**, pp. 1–31.
34. Shah, C.; White, R.W. Agents are not enough. *Computer* **2025**, *58*, 87–92.
35. Wilson, G.; Bryan, J.; Cranston, K.; Kitzes, J.; Nederbragt, L.; Teal, T.K. Good enough practices in scientific computing. *PLoS computational biology* **2017**, *13*, e1005510.
36. Kurtzer, G.M.; Sochat, V.; Bauer, M.W. Singularity: Scientific containers for mobility of compute. *PloS one* **2017**, *12*, e0177459.
37. Köster, J.; Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* **2012**, *28*, 2520–2522.
38. Di Tommaso, P.; Chatzou, M.; Floden, E.W.; Barja, P.P.; Palumbo, E.; Notredame, C. Nextflow enables reproducible computational workflows. *Nature biotechnology* **2017**, *35*, 316–319.
39. Crusoe, M.R.; Abeln, S.; Iosup, A.; Amstutz, P.; Chilton, J.; Tijanić, N.; Ménager, H.; Soiland-Reyes, S.; Gavrilović, B.; Goble, C.; et al. Methods included: standardizing computational reuse and portability with the common workflow language. *Communications of the ACM* **2022**, *65*, 54–63.
40. Guo, Z.; Cheng, S.; Wang, H.; Liang, S.; Qin, Y.; Li, P.; Liu, Z.; Sun, M.; Liu, Y. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024, 2024, pp. 11143–11156.
41. Conway, M.E. How do committees invent. *Datamation* **1968**, *14*, 28–31.
42. Park, J.S.; O'Brien, J.; Cai, C.J.; Morris, M.R.; Liang, P.; Bernstein, M.S. Generative agents: Interactive simulacra of human behavior. In Proceedings of the Proceedings of the 36th annual acm symposium on user interface software and technology, 2023, pp. 1–22.
43. Belhajjame, K.; B'Far, R.; Cheney, J.; Coppens, S.; Cresswell, S.; Gil, Y.; Groth, P.; Klyne, G.; Lebo, T.; McCusker, J.; et al. Prov-dm: The prov data model. *W3C Recommendation* **2013**, *14*, 15–16.
44. Huang, D.; Lapp, H. Software engineering as instrumentation for the long tail of scientific software. *arXiv preprint arXiv:1309.1806* **2013**.
45. Ouyang, C.; Yue, L.; Di, S.; Zheng, L.; Yue, L.; Pan, S.; Yin, J.; Zhang, M.L. Code2MCP: Transforming Code Repositories into MCP Services. *arXiv preprint arXiv:2509.05941* **2025**.
46. Wang, G.; Xie, Y.; Jiang, Y.; Mandlekar, A.; Xiao, C.; Zhu, Y.; Fan, L.; Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291* **2023**.
47. Tambe, M. Towards flexible teamwork. *Journal of artificial intelligence research* **1997**, *7*, 83–124.
48. Groth, P.; Moreau, L. PROV-overview. An overview of the PROV family of documents **2013**.
49. Hu, S.; Lu, C.; Clune, J. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435* **2024**.
50. Zhang, J.; Xiang, J.; Yu, Z.; Teng, F.; Chen, X.; Chen, J.; Zhuge, M.; Cheng, X.; Hong, S.; Wang, J.; et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762* **2024**.
51. Zhang, J.; Hu, S.; Lu, C.; Lange, R.; Clune, J. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents. *arXiv preprint arXiv:2505.22954* **2025**.
52. Shinn, N.; Cassano, F.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems* **2023**, *36*, 8634–8652.
53. Ioannidis, J.P. Why most published research findings are false. *PLoS medicine* **2005**, *2*, e124.

 doi:10.20944/preprints202507.1951.v2

54. Munafò, M.R.; Nosek, B.A.; Bishop, D.V.; Button, K.S.; Chambers, C.D.; Percie du Sert, N.; Simonsohn, U.; Wagenmakers, E.J.; Ware, J.J.; Ioannidis, J.P. A manifesto for reproducible science. *Nature human behaviour* **2017**, *1*, 0021.
55. Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; Men, K.; Yang, K.; et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* **2023**.