



# Kosmos AI Scientist: Multi-Agent System with World Model Orchestrator

## Overview of Kosmos and its Multi-Agent Approach

Kosmos is an autonomous “AI Scientist” developed by Edison Scientific that can conduct long, open-ended scientific research sessions by combining multiple specialized agents with a shared memory [1](#) [2](#). Given a natural-language research objective and an initial dataset, Kosmos runs for up to 12 hours, iterating through cycles of **data analysis**, **literature search**, and **hypothesis generation**, before producing a fully cited scientific report [1](#). The key innovation is that Kosmos overcomes the coherence limits of prior AI agents by using a **structured world model** as a long-term memory and coordinator, allowing it to carry out on the order of 200 sequential or parallel agent actions (“rollouts”) without losing the thread of the research [3](#) [4](#). In each run, Kosmos can read ~1,500 scientific papers and execute ~42,000 lines of code – vastly more than previous systems – thanks to this architecture [5](#) [6](#). By the end, it assembles its findings into a report where **every statement is traceable** to either a code execution or a literature source, ensuring transparency and auditability of its conclusions [7](#).

## System Architecture and Agent Roles

Kosmos’s architecture is explicitly modular and multi-agent. At a high level, there are two primary intelligent agents working in tandem, overseen by a central orchestrator (the world model) [8](#) [9](#). The main agents are:

- **Data Analysis Agent** – An agent responsible for computational experiments and data processing. It generates code to analyze the provided dataset (or intermediate data), executes that code in a sandboxed notebook environment, and interprets the results [9](#). For example, it might perform a statistical analysis, train a model, or run a differential analysis on the data as a task (e.g. “*run a differential abundance analysis on a metabolomics dataset*”) [10](#). After execution, it outputs structured results (tables, figures, statistics) and explanatory text.
- **Literature Search Agent** – An agent focused on information retrieval and analysis of scientific literature. It searches external sources (like academic databases or PDFs) for papers relevant to the current hypotheses, reads and summarizes findings, and extracts key evidence or citations [10](#). For example, it might be tasked with “*find any known pathways connecting gene X to disease Y*” and then retrieve relevant papers to summarize those connections [10](#). The agent then writes back its findings, along with references to the source documents, into the shared world model.

In addition, Kosmos incorporates other functional components to complete the research workflow. A **hypothesis generation module** (or planner) proposes new hypotheses or questions to explore based on the current state of knowledge, guiding what tasks the agents tackle in each cycle [9](#). There is also a **synthesis/reporting component** at the end: after many cycles of work, Kosmos traverses the accumulated

world model and composes a final report of discoveries, where each statement is explicitly linked to supporting code results or literature passages <sup>11</sup>. This ensures that the multi-agent process culminates in a coherent output that a human scientist can verify line-by-line. All these components are integrated via the world model, which serves as the “brain” or coordination center of the system.

## Structured World Model as Orchestrator and Memory

At the heart of Kosmos is the **structured world model**, which acts as a shared memory bank and an orchestration hub for the agents <sup>12</sup>. Unlike a standard LLM context or simple vector memory, the world model is implemented as a structured *database* or knowledge graph that persistently stores the evolving state of the research <sup>12</sup>. It contains entries for **entities** (e.g. genes, compounds, variables of interest), their relationships (e.g. correlations, causal links), **experimental results** (data summaries, statistical findings, plots), and **open questions or hypotheses** that have arisen <sup>12</sup>. After every agent action, this world model is updated with the new information gained. Because it is a queryable, structured store rather than a fleeting text buffer, information from early in the run remains accessible and well-organized even after millions of tokens worth of processing <sup>13</sup>. This design lets Kosmos maintain long-horizon coherence; the agents and planning module can always refer back to the world model to recall what’s been learned, what remains unresolved, and what the overarching goals are. In essence, the world model serves as the **single source of truth** and context that all agents consult and contribute to, functioning much like an intelligent “blackboard” in classical AI architectures where specialists write and read knowledge as they work.

Crucially, the world model doesn’t just passively store facts – it actively **orchestrates the agents’ activities**. At the start of each cycle, Kosmos (via the orchestrator logic) queries the world model to identify promising next steps: for instance, unresolved hypotheses or new leads that emerged from the latest findings <sup>9</sup>. Based on this state, the system generates a set of concrete tasks (up to about 10 per cycle) for the agents to execute <sup>9</sup>. These tasks are essentially the **plans or actions** for that iteration, each tied to a specific agent. For example, if the world model indicates an interesting gene candidate with unknown function, the orchestrator may queue a task for the literature agent to “Search for studies linking [Gene] to [Condition]”, and simultaneously queue a task for the data agent to “Test [Gene]’s association with outcomes in the dataset”. By embedding the current world model state into the agents’ prompts or environment, Kosmos ensures each agent is context-aware – they know the current hypotheses and focus when carrying out their task. Once the agents complete their tasks (e.g. code finished running or papers summarized), they feed the results (new data insights, new citations, etc.) back into the world model in a structured format <sup>9</sup>. This cycle of **plan → execute → update** repeats, with the world model coordinating and remembering each step. Through this mechanism, the world model effectively manages the evolving set of hypotheses (tracking which have evidence, which need more investigation), the list of actions taken or planned, and the overall research plan over time.

## Sandbox Environment and Integration of Code Execution

A distinguishing feature of Kosmos is its ability to not only *read* and reason about text (papers) but also to *write and run code* for data analysis. To enable this safely and efficiently, Kosmos integrates a **sandboxed execution environment** (often a Jupyter-like notebook or container) where the data analysis agent’s code is executed <sup>10</sup>. This sandbox is isolated from external side effects, ensuring that any code (which could include statistical analysis, machine learning training, plotting, etc.) runs in a controlled environment for security and reproducibility. When the world model (or orchestrator) assigns a task to perform an

experiment, the data analysis agent writes the necessary code and runs it in the sandbox, acting almost like a virtual scientist writing scripts to test a hypothesis. The outputs – whether they are numerical results, figures, or tables – are captured. The agent then interprets these outputs in a textual form (e.g. “*Gene X showed a 2-fold increase under condition Y ( $p<0.01$ )*”) and stores both the interpretation and a reference to the raw results (such as a notebook cell or figure number) back into the world model <sup>10</sup>.

This design means the sandbox is tightly coupled to the world model: the **interface** between them is managed by the orchestrator. The orchestrator triggers code execution in the sandbox via the data agent and ensures the results flow into the world model. In turn, the world model’s contents can inform what code to run next. For example, if a certain analysis result in the world model suggests an anomaly, the next cycle might include a sandbox task to drill deeper into that anomaly with a new script. By containing code execution in a sandbox, Kosmos ensures that even complex computational experiments (42,000 lines of code on average per run) can be conducted without losing track of their outcomes or compromising system stability <sup>5</sup>. Each executed action has a recorded outcome in the world model, linking code and data to scientific insight. This combination of an AI agent with coding abilities and a sandbox to execute code is relatively novel in multi-agent systems, effectively giving Kosmos the tools to **autonomously perform “wet-lab” style analyses in silico**, guided by the world model’s orchestrated plan.

## Inter-Agent Communication and Modularity

Kosmos’s agents are designed to be modular and specialized, and they communicate **indirectly** through the world model rather than by direct messaging. This is analogous to a team of experts using a shared lab notebook or knowledge base to coordinate. The literature agent and the data analysis agent do not chat with each other; instead, each posts its findings and reads what the other has posted via the central world model. This decoupling via a shared memory ensures consistency and avoids the need for complex dialog protocols between agents. It also means agents can operate in parallel during a cycle – for example, one agent scouring papers while another crunches data – since they will later merge their results in the world model for all to see <sup>9</sup>. The world model effectively serializes their contributions into a unified knowledge state.

This architecture confers **modularity**: each agent focuses on its domain (coding vs. reading) and can be improved or replaced independently as long as it adheres to the world model interface format. New agents could be added as well (e.g. a specialized hypothesis-generator agent or a visualization agent) by granting them access to read/write appropriate parts of the world model. The orchestrator (world model + planning logic) is the only component that needs a global view. In Kosmos’s current design, the orchestrator uses the world model to decide on tasks, but the heavy lifting of each task is offloaded to the specialized agent best suited for it <sup>14</sup>. This separation of concerns makes the system more scalable and maintainable, akin to how large human research teams divide tasks among members and compile their contributions in shared documents. Moreover, because all information exchange is through a structured store, issues like context overflow or forgetting are mitigated – an agent can always query the world model if it “forgets” something, rather than relying on a single prompt history. This approach is reminiscent of the classic **blackboard system** in AI, where independent expert modules write to and read from a common blackboard to solve a problem collaboratively. Kosmos revitalizes this idea using modern LLM agents and ensures that each agent’s work remains **aligned** with a single source of truth (the world model). The result is coherent teamwork: for instance, the data agent might flag an unexpected pattern in results, which the literature agent then sees in the world model and uses to guide a new literature query. In summary, inter-agent

communication in Kosmos is achieved through the orchestrated updates to the world model, enabling asynchronous yet coordinated collaboration among highly modular agents <sup>2</sup>.

## Iterative Research Workflow

The end-to-end workflow of Kosmos proceeds in iterative **cycles** that gradually expand and refine the knowledge in the world model, much like the scientific method in looped miniature. Here's how a typical task execution cycle works in Kosmos:

1. **Task Planning:** At the start of a cycle, the orchestrator (using the world model) examines the current objective and the state of knowledge/hypotheses. It generates a list of new tasks or questions to pursue next <sup>9</sup>. These tasks are concrete and aimed at reducing uncertainty or exploring leads. For example, the system might propose: "*Compute the correlation between gene A and metabolite B in the dataset*" (for the data agent), and "*Find if any literature reports link metabolite B to hypothermia*" (for the literature agent). Up to about 10 such tasks might be proposed in parallel, covering different facets of the research goal <sup>9</sup>. The planning takes into account what's already known (to avoid redundancy) and what is most promising or urgent to investigate (perhaps guided by which hypotheses are still unverified).
2. **Agent Execution:** The respective agents undertake the tasks. The data analysis agent translates its assigned task into executable code (or uses predefined analysis templates) and runs it in the sandbox environment. The literature agent formulates search queries or uses APIs to find relevant papers, then reads the content (often using LLM capabilities to summarize or extract key points) <sup>10</sup>. During this phase, each agent works independently on its task, possibly using large language model reasoning internally to decide how to carry it out. Notably, because tasks are well-specified, multiple tasks (especially of different types) can be executed concurrently, utilizing parallelism to speed up the cycle. This parallelism is a design choice that sets Kosmos apart from simpler sequential agents.
3. **World Model Update:** Once an agent completes a task, it structures the results and updates the world model. For a code experiment, the agent might store the quantitative results (e.g. "correlation = 0.8, p=0.01"), any generated figures or tables (as references), and a brief interpretation of what that means for the hypothesis. For a literature finding, the agent would add the key insight (e.g. "Paper X reports that metabolite B rises during hypothermia") along with a citation to the source <sup>10</sup>. Each entry in the world model is tagged to the originating task and agent, building a traceable log of actions. The world model also keeps track of which hypotheses have gained supporting evidence, which ones were refuted, and what new questions emerged from the findings.
4. **Review and Next Iteration:** With the world model now richer, the orchestrator (or a supervisory logic) can evaluate progress. If the objective is not yet met or there remain unanswered questions, the cycle repeats. The new state may reveal further hypotheses or might require drilling down deeper into a particular thread. Kosmos will initiate another round of task planning, now informed by everything that came before. In practice, Kosmos might run through ~20 such cycles in a full session, which users found equivalent to about six months of manual research work in total <sup>15</sup> <sup>16</sup>. Throughout these iterations, the system remains focused on the original objective by virtue of the world model's memory: even after dozens of steps, it can recall the high-level goal and all intermediate conclusions, preventing it from drifting off-topic. This careful orchestration yields a **coherent research trajectory**: early cycles gather broad information, and later cycles zero in on

detailed analysis or edge cases, much like a human scientist would refine their experiments over time.

**5. Synthesis and Reporting:** After the allotted time or cycles, Kosmos enters a final phase of synthesizing the accumulated knowledge. The synthesis component traverses the world model's contents, organizing the confirmed findings, evidence, and reasoning into a narrative structure <sup>11</sup>. It essentially writes a mini-review or research report addressing the original objective. Because every item in the world model is linked to a provenance (code cell ID or literature reference), Kosmos is able to produce a report where **each claim is cited** <sup>11</sup>. For instance, a sentence in the report might state a conclusion and then reference the specific experiment (notebook cell) that supports it, as well as any paper that corroborates it. This results in an output that is not only insightful but also verifiable. Human scientists can inspect the raw code or data behind any result using the references, making the workflow transparent rather than a "black box" <sup>11</sup>. The final report marks the completion of the autonomous research run, ready for human review or follow-up.

This iterative workflow, guided by the world model, ensures that Kosmos systematically **generates hypotheses, tests them, integrates new knowledge, and repeats** – closely mimicking the iterative nature of real scientific discovery. By automating this loop, Kosmos can explore far more avenues in a short time than a human could, yet maintain coherence and direction throughout.

## Comparison with Other Multi-Agent and World-Model Systems

Kosmos's design builds upon concepts from prior AI agent systems while introducing notable advances in how long-term state and coordination are handled. Traditional single-agent autonomous systems (like early versions of AutoGPT or BabyAGI) often struggled with long-horizon tasks because they relied on a single LLM's context window or ephemeral memory; as a result, they would lose track of objectives or repeat themselves after a limited number of steps <sup>3</sup> <sup>17</sup>. Some of those systems attempted to use vector databases or summaries as memory, but those are unstructured and require the agent to explicitly query them, making it hard to maintain a complex, shared understanding of a problem. In contrast, Kosmos introduces a persistent, structured world model that all agents share, which is more akin to a continuously updated knowledge graph or "dynamic database" of the task <sup>12</sup>. This allows for a far greater number of coherent actions (hundreds of agent rollouts) without forgetting earlier context <sup>18</sup> <sup>4</sup>. In essence, Kosmos revives the old **blackboard model** of AI (from expert systems) but at a new scale: multiple AI agents post and read intermediate results on a common board (the world model), enabling collaboration and sustained reasoning. Recent research has indeed started exploring blackboard-style multi-agent frameworks for LLMs, but Kosmos is one of the first to apply it successfully to complex scientific workflows, whereas many other multi-agent demos have been limited to toy problems or shorter tasks.

Compared to other multi-agent orchestration frameworks, Kosmos's approach is distinctive. For example, **HuggingGPT** and similar systems use one agent (or LLM) as a *manager* that delegates tasks to various specialist models, but those frameworks typically focus on one-shot or short sequences of tool use, and they don't maintain an explicit structured memory of all past tool outputs. Kosmos, by contrast, involves an ongoing interplay where the "manager" role is partly embedded in the world model itself (plus planning logic) and every tool/agent output is logged in a persistent store <sup>12</sup> <sup>9</sup>. Another line of comparison is with **Generative Agents** (such as the famous AI town simulation), which gave agents long-term memory via semantic vector stores and summary mechanisms. Those agents could remember facts about their world and had a notion of planning, which is somewhat analogous to Kosmos's memory. However, Generative

Agents were individual characters operating in a simulated environment and communicating via natural language, whereas Kosmos's agents are *specialized modules* cooperating on a single complex goal, and their shared memory is structured and formal (not just conversational history). Additionally, generative agent simulations didn't involve executing code or interfacing with external knowledge sources like academic literature in real time – capabilities that Kosmos integrates via its sandbox and search components.

In the scientific domain, there have been earlier tool-using AI systems (for instance, **ChemCrow** for chemistry or other domain-specific assistants) that chain LLM reasoning with coding and data analysis. Those often rely on a sequential chain-of-thought prompting where the LLM decides the next tool use based on intermediate results. Kosmos generalizes this idea into a concurrent multi-agent paradigm with a central world model, which could lead to more robustness. For example, instead of a single monolithic agent alternately reading papers and coding (which could overwhelm one context), Kosmos splits the duties and has a repository (world model) that both can update in parallel – this likely contributes to its efficiency and depth of analysis <sup>16</sup>.

Another relevant concept is the notion of **world models** in reinforcement learning (e.g. Ha & Schmidhuber's World Models, or planning agents that learn environment models). Those "world models" typically refer to learned predictive models of an environment's dynamics. Kosmos's world model is different – it's not a learned neural network of an environment, but rather a structured knowledge store curated by the AI through extraction. It serves more as a **knowledge graph and plan state** than a predictive simulator. Nevertheless, both uses of the term share the goal of enabling longer-term planning: in RL, a world model helps anticipate future states; in Kosmos, the world model helps keep track of complex interdependencies in scientific data and literature so that the AI can plan many steps ahead coherently <sup>3</sup>.

In summary, Kosmos's closest analogies are systems based on a **blackboard architecture or long-term memory** for multi-agent collaboration. Recent academic proposals (e.g. LLM-based blackboard systems <sup>19</sup>) echo this pattern, but Kosmos distinguishes itself by demonstrating it on real scientific research tasks with tangible results. Its use of a structured world model to coordinate a literature agent and a data agent is **unique in scope and scale** – enabling the AI to act as a researcher that can both experiment (with code) and read (papers) over prolonged sessions <sup>20</sup>. The result is a system that achieves greater depth of reasoning and problem-solving than prior multi-agent setups that were constrained by context or simplistic memory schemes.

## Unique Innovations and Impact of Kosmos

Kosmos introduces several innovative elements that set it apart from previous AI research assistants and multi-agent systems:

- **Long-Horizon Coordination via World Model:** The structured world model is Kosmos's core innovation, allowing it to aggregate and organize knowledge over tens of thousands of tokens and hundreds of actions <sup>18</sup>. This leads to unprecedented coherence in long tasks – where earlier agents would "lose the plot" after a few dozen steps, Kosmos remains focused on the research objective throughout a 12-hour run <sup>3</sup> <sup>4</sup>. This is a breakthrough in enabling depth of reasoning, as evidenced by Kosmos achieving what experts estimated as *months* of research work in one run while maintaining logical continuity <sup>21</sup> <sup>16</sup>.

- **Integrated Multi-Modal Abilities (Coding + Reading):** Kosmos seamlessly blends language-based reasoning with code-based data analysis. The ability to autonomously run extensive code (42k lines per run on average) in a sandbox and then immediately incorporate those results into its reasoning is a major step beyond agents that only use API calls for trivial calculations <sup>5</sup>. This means Kosmos cannot just talk about data – it can *directly analyze* data. Coupled with an agent that voraciously consumes literature (1,500 papers per run) <sup>5</sup>, Kosmos effectively combines the functions of a data scientist and a research librarian, orchestrated in one framework. Few if any prior systems have combined these capabilities at such scale.
- **Explicit Traceability and Scientific Rigor:** A unique aspect of Kosmos is its emphasis on traceable outputs. Every claim in its final reports is backed by either a snippet of code execution (with results) or a citation to primary literature <sup>11</sup> <sup>7</sup>. This goes beyond typical LLM outputs or even tools like Bing Chat that cite web sources. Kosmos's citations are interwoven with its multi-agent process – it is effectively generating a provenance graph as it works. This focus on provenance addresses a common critique of AI in science (the “black box” problem) by providing an audit trail for each conclusion <sup>22</sup> <sup>11</sup>. It’s an innovation especially crucial for scientific adoption: the system not only accelerates discovery but also instills confidence that its findings can be verified.
- **Parallel and Scalable Agent Execution:** By running specialized agents in parallel (when tasks allow) and not strictly sequentially, Kosmos improves efficiency. This design showcases an innovation in agent orchestration – many earlier agent frameworks were strictly sequential or single-threaded in how they processed steps. Kosmos’s parallelization (e.g. reading papers while simultaneously processing data) more closely mirrors how a real research team might operate and maximizes utilization of the available time. The world model acts as a synchronization point when needed (agents write results when done), allowing concurrency without chaos. This parallelism is an innovative twist that contributes to its high throughput of actions in a limited wall-clock time.
- **Domain-Agnostic Scientific Reasoning:** While some AI systems have been built for specific domains (chemistry, programming, etc.), Kosmos was designed to be broadly applicable across scientific fields – its seven showcased discoveries span metabolomics, materials science, neuroscience, genetics, and more <sup>23</sup> <sup>24</sup>. The modular agent approach, combined with a domain-independent world model schema (entities, relations, results), means Kosmos can plug in new tools or domain-specific databases when needed but the overall framework remains the same. This generality is innovative because it hints at a template for **AI-driven research in any domain**: one needs a way to parse literature, a way to analyze data, and a knowledge model to tie them together. Kosmos provides a working example of this template, whereas past efforts often built one-off systems per domain.

In conclusion, Kosmos’s multi-agent architecture with a world model orchestrator represents a significant step forward in autonomous systems for complex tasks. It blends ideas from blackboard systems, tool-using agents, and long-context language models into a cohesive platform tailored for scientific discovery. Early evaluations have shown it can reproduce unpublished findings and generate new hypotheses, effectively acting as a junior researcher that never tires <sup>25</sup> <sup>26</sup>. Importantly, rather than replacing human scientists, Kosmos augments them: it handles the heavy lifting of reading and experimentation at superhuman speeds, while providing outputs that humans can trust and verify. This symbiotic design – AI agents working with a structured world model to produce human-auditable science – is an **innovative paradigm** introduced by Kosmos, and it may pave the way for more “AI scientists” in the future. The system

stands out among multi-agent AI frameworks for its sustained reasoning, robust knowledge integration, and alignment with the needs of real-world scientific research <sup>27</sup>.

### Sources:

- Mitchener et al., "Kosmos: An AI Scientist for Autonomous Discovery," arXiv preprint 2511.02824 (2025)  
<sup>1</sup> <sup>28</sup>.
  - Edison Scientific Blog, "Kosmos: An AI Scientist for Autonomous Discovery" (Nov 2025) <sup>3</sup> <sup>5</sup>.
  - MarkTechPost, "Meet Kosmos: An AI Scientist that Automates Data-Driven Discovery" (Nov 9, 2025) <sup>12</sup>  
<sup>10</sup>.
  - Sutter, *Science & Tech News: Edison Scientific Kosmos – Accelerating Six Months of Science in a Day* (2025)  
<sup>2</sup> <sup>11</sup>.
- 

<sup>1</sup> <sup>7</sup> <sup>8</sup> <sup>15</sup> <sup>17</sup> <sup>20</sup> <sup>28</sup> [2511.02824] Kosmos: An AI Scientist for Autonomous Discovery

<https://arxiv.org/abs/2511.02824>

<sup>2</sup> <sup>4</sup> <sup>6</sup> <sup>9</sup> <sup>10</sup> <sup>11</sup> <sup>12</sup> <sup>13</sup> <sup>16</sup> <sup>24</sup> <sup>26</sup> <sup>27</sup> Meet Kosmos: An AI Scientist that Automates Data-Driven Discovery - MarkTechPost

<https://www.marktechpost.com/2025/11/09/meet-kosmos-an-ai-scientist-that-automates-data-driven-discovery/>

<sup>3</sup> <sup>5</sup> <sup>18</sup> <sup>21</sup> <sup>22</sup> <sup>23</sup> <sup>25</sup> Kosmos: An AI Scientist for Autonomous Discovery

<https://edisonscientific.com/articles/announcing-kosmos>

<sup>14</sup> GitHub - jimmc414/Kosmos: Kosmos: An AI Scientist for Autonomous Discovery - An implementation and adaptation to be driven by Claude Code or API - Based on the Kosmos AI Paper - <https://arxiv.org/abs/2511.02824>

<https://github.com/jimmc414/Kosmos>

<sup>19</sup> Exploring Advanced LLM Multi-Agent Systems Based on Blackboard ...

<https://www.alphaxiv.org/overview/2507.01701v1>