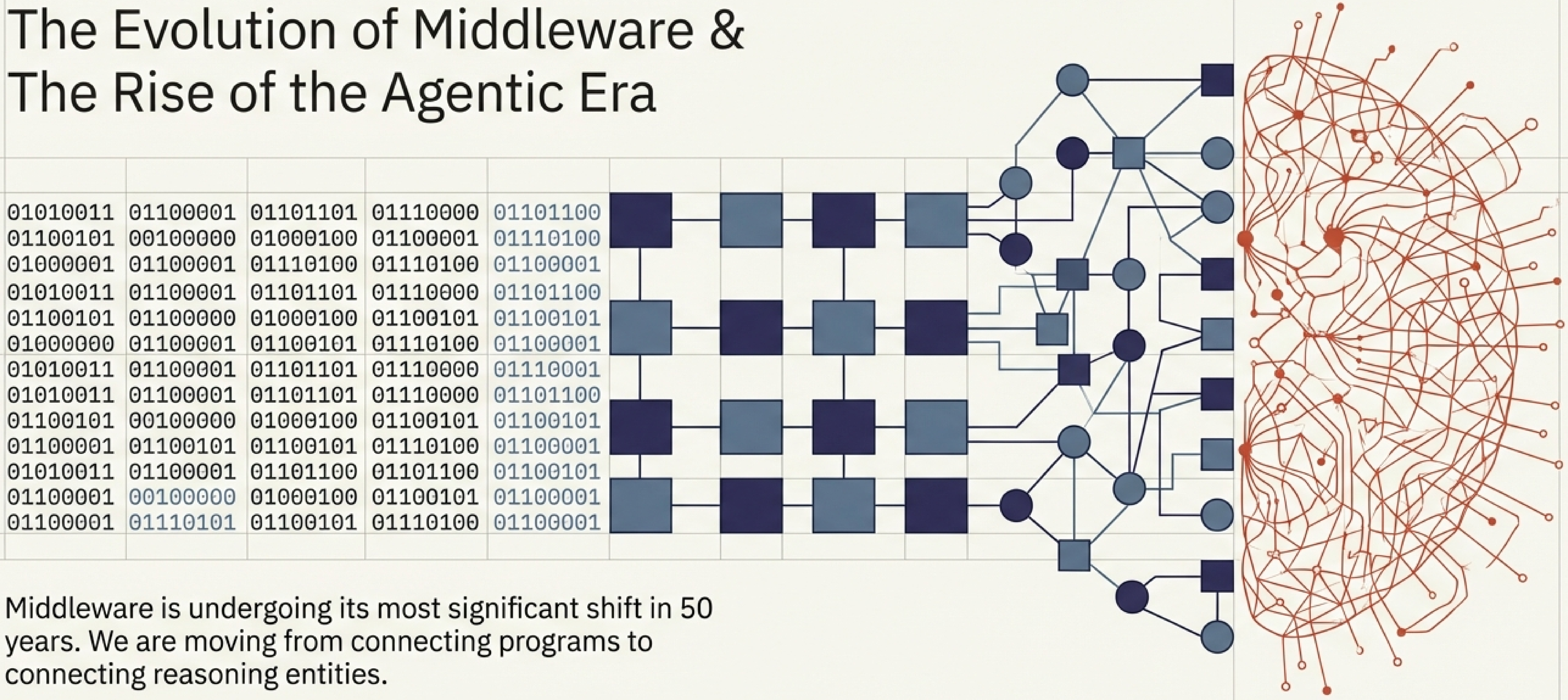
From RPC to Reasoning

The Evolution of Middleware & The Rise of the Agentic Era



Middleware is undergoing its most significant shift in 50 years. We are moving from connecting programs to connecting reasoning entities.

The Unit of Integration

RPC connected functions, messaging connected systems, REST connected resources — MCP and SKILLs now connect agents.

Functions

1970s

Objects & Messages

1990s

Documents & Resources

2000s

Capabilities

2020s

Past: Optimized for latency and developer convenience.

Present: Optimized for autonomy and context exchange.

1970s–80s: The Ambition of Network Transparency

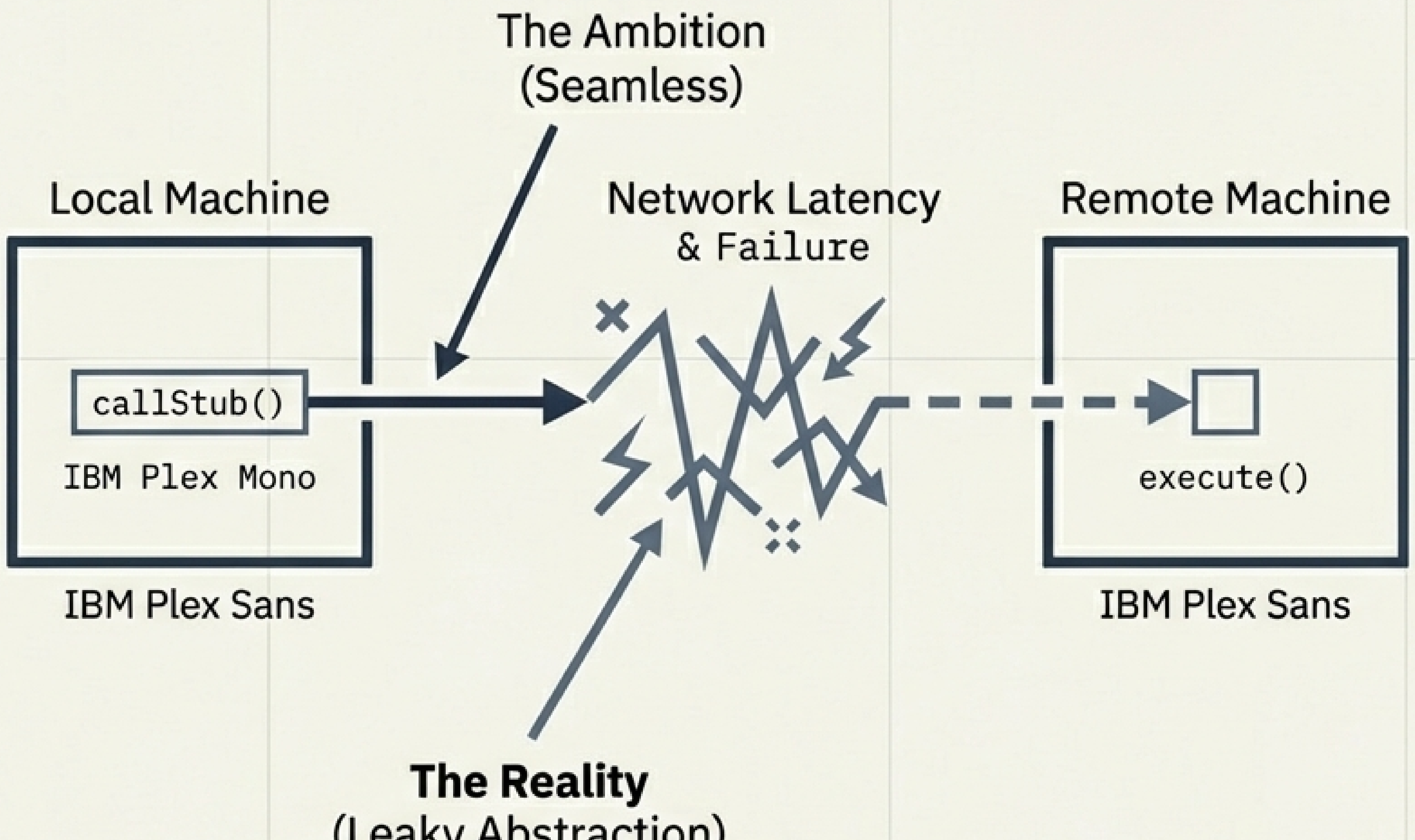
UNIT OF INTEGRATION: FUNCTIONS

The Goal: "Make remote look local." The aim was to hide distribution behind standard procedure calls.

The Tech: Birrell & Nelson RPC, Sun RPC, NFS.

Legacy: Established the baseline for request/response patterns, but proved that hiding the network completely is dangerous for system resilience.

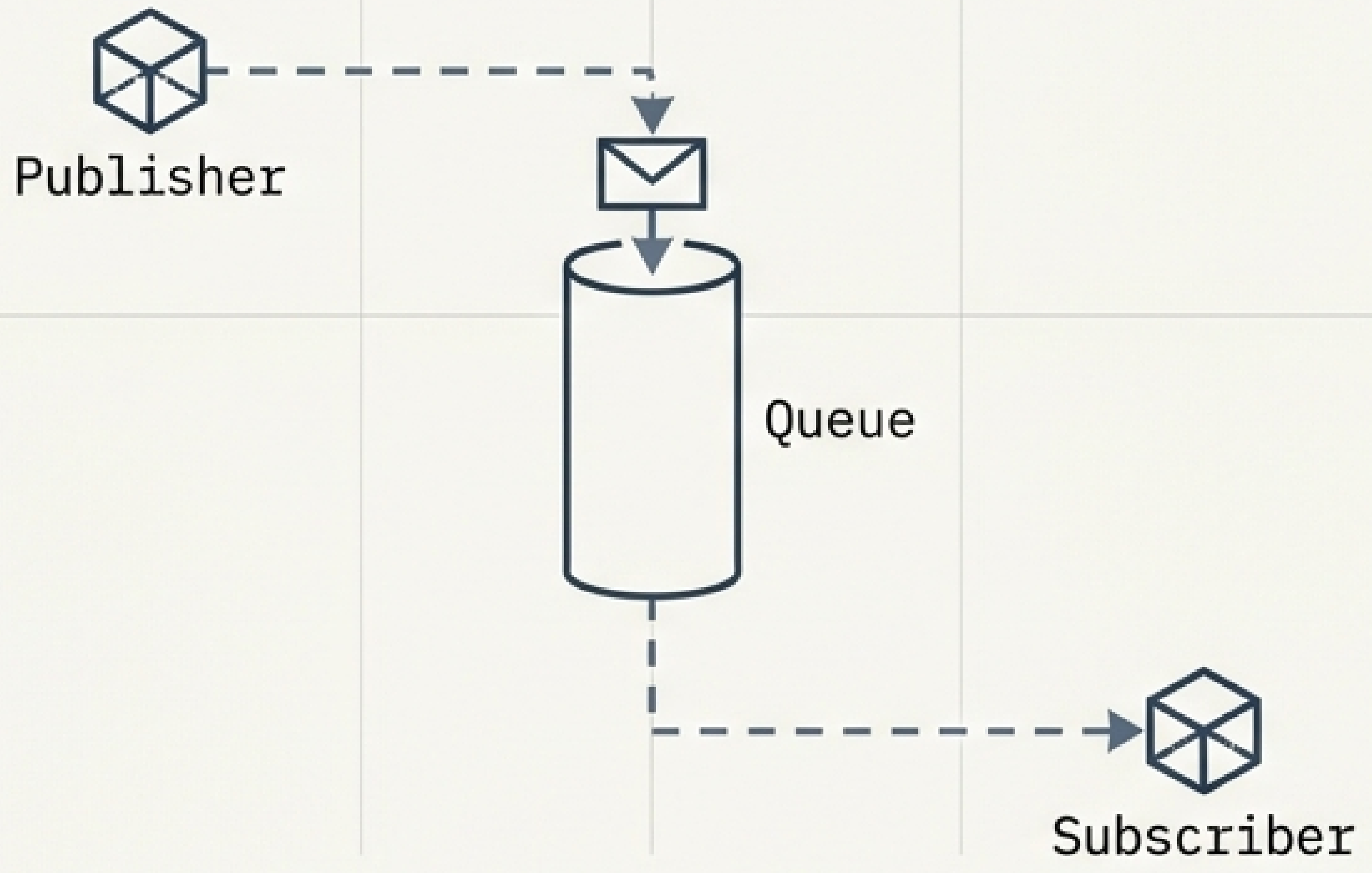
The Fallacy of Distributed Computing



1990s: The Divergence of Objects and Messages

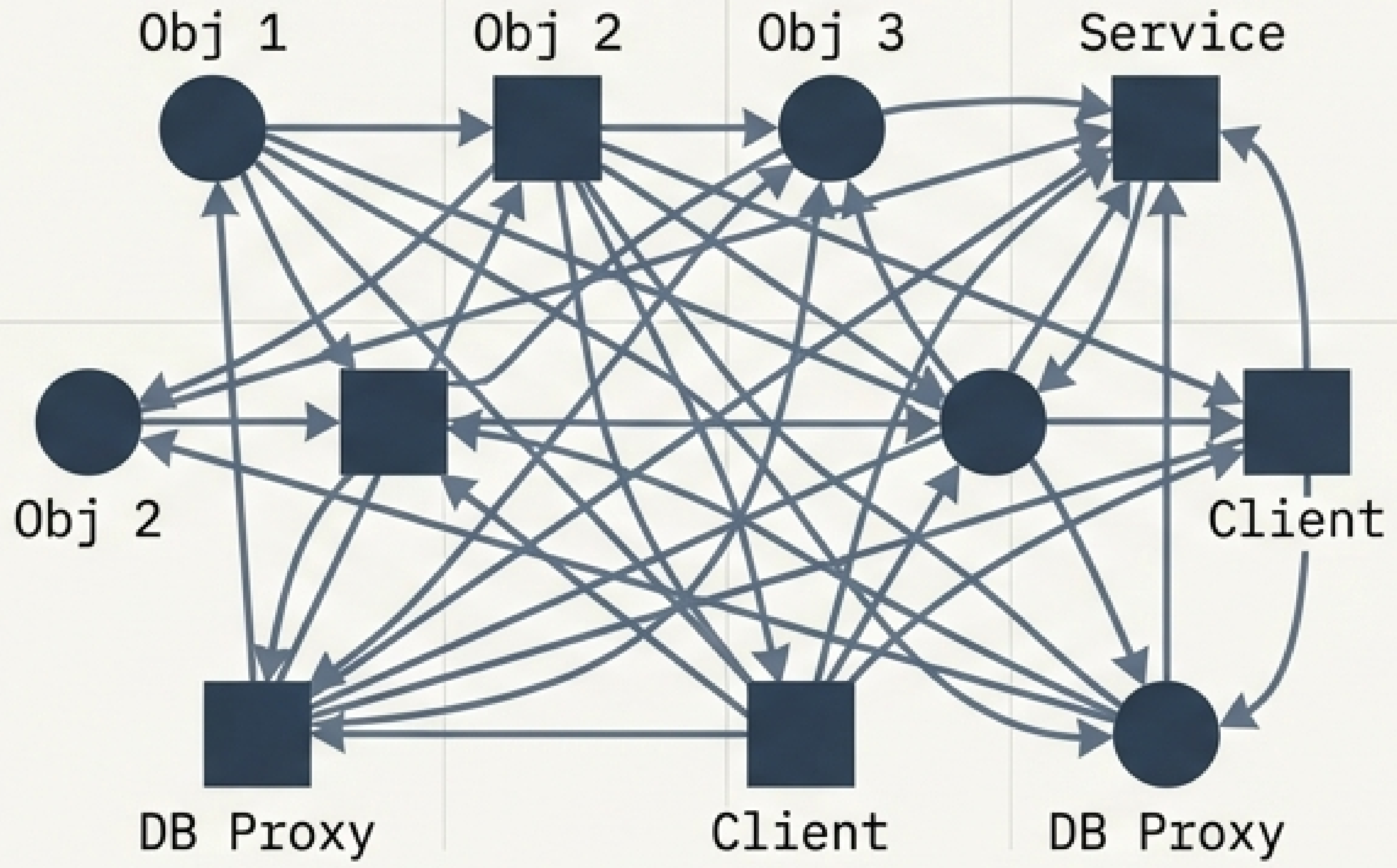
UNIT OF INTEGRATION: OBJECTS VS. MESSAGES

Path A: Message-Oriented Middleware (MOM)



- **Tech:** IBM MQ, TIBCO, JMS
- **Philosophy:** Asynchronous queues and Pub/Sub.
- **Outcome:** Loose coupling and reliability.

Path B: Distributed Object Middleware

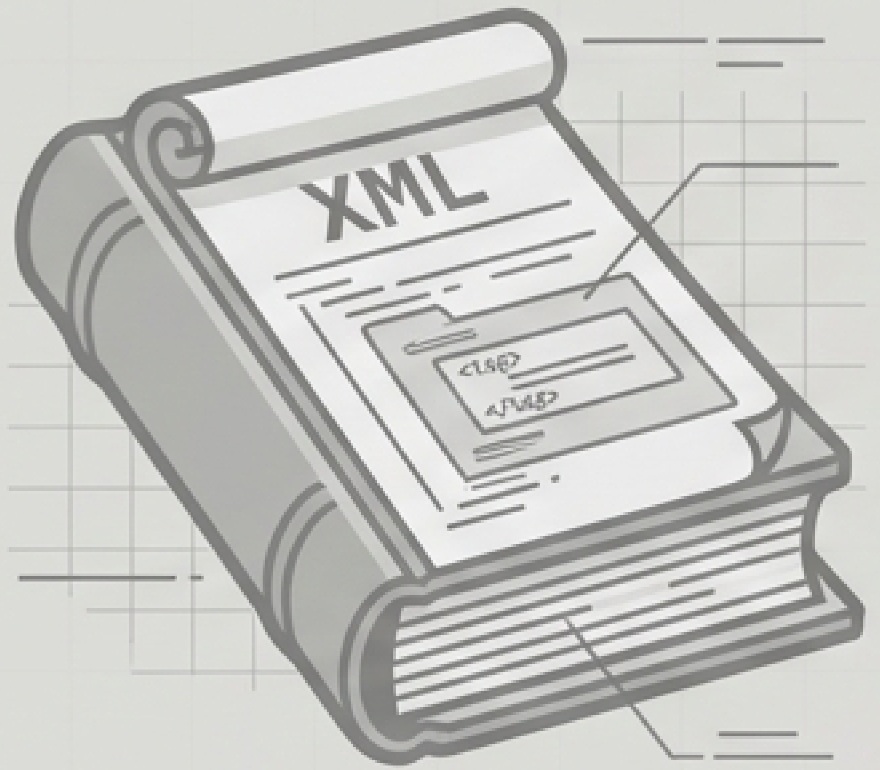


- **Tech:** CORBA, DCOM, Java RMI
- **Philosophy:** Mapping OOP directly to the network.
- **Outcome:** Complexity explosion and brittle systems.

2000s: The Document Wars and the Triumph of REST

UNIT OF INTEGRATION: DOCUMENTS & RESOURCES

The Challenger: SOAP



Enterprise rigor, WSDL, ESBs.

Focus: Formal contracts.

The Victor: REST

Web simplicity, HTTP + JSON.

Focus: Usability and scalability.

Why REST Won: It utilized existing web primitives rather than inventing new complex specifications. The industry traded the strict contracts of SOAP for the developer usability of REST.

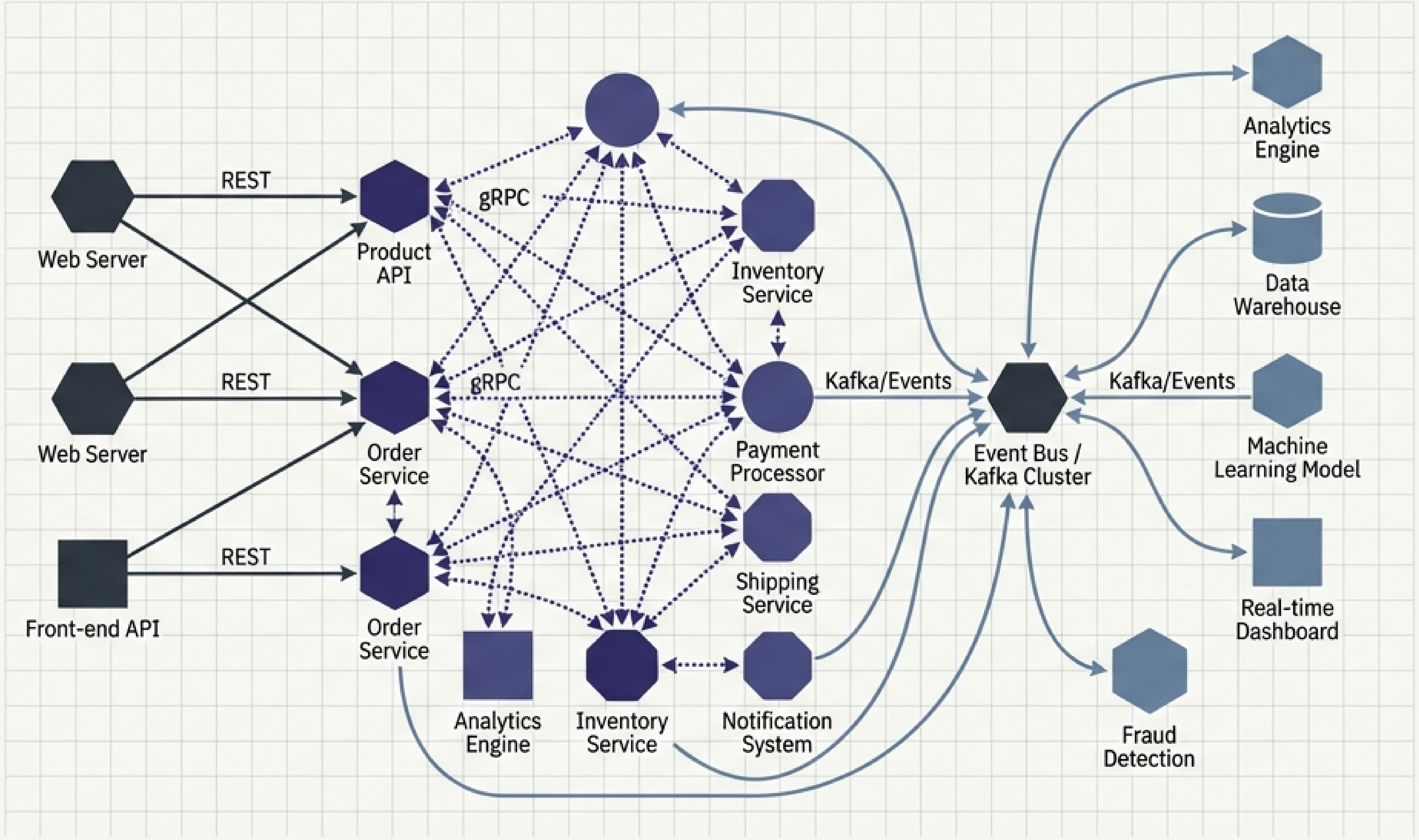
NotebookLM

2010s: The Fragmentation of Microservices

UNIT OF INTEGRATION: SERVICES & EVENTS

The Landscape:
A mix of REST (external), gRPC (internal performance), and Kafka (real-time data). A heterogeneous ecosystem where different protocols serve distinct purposes.

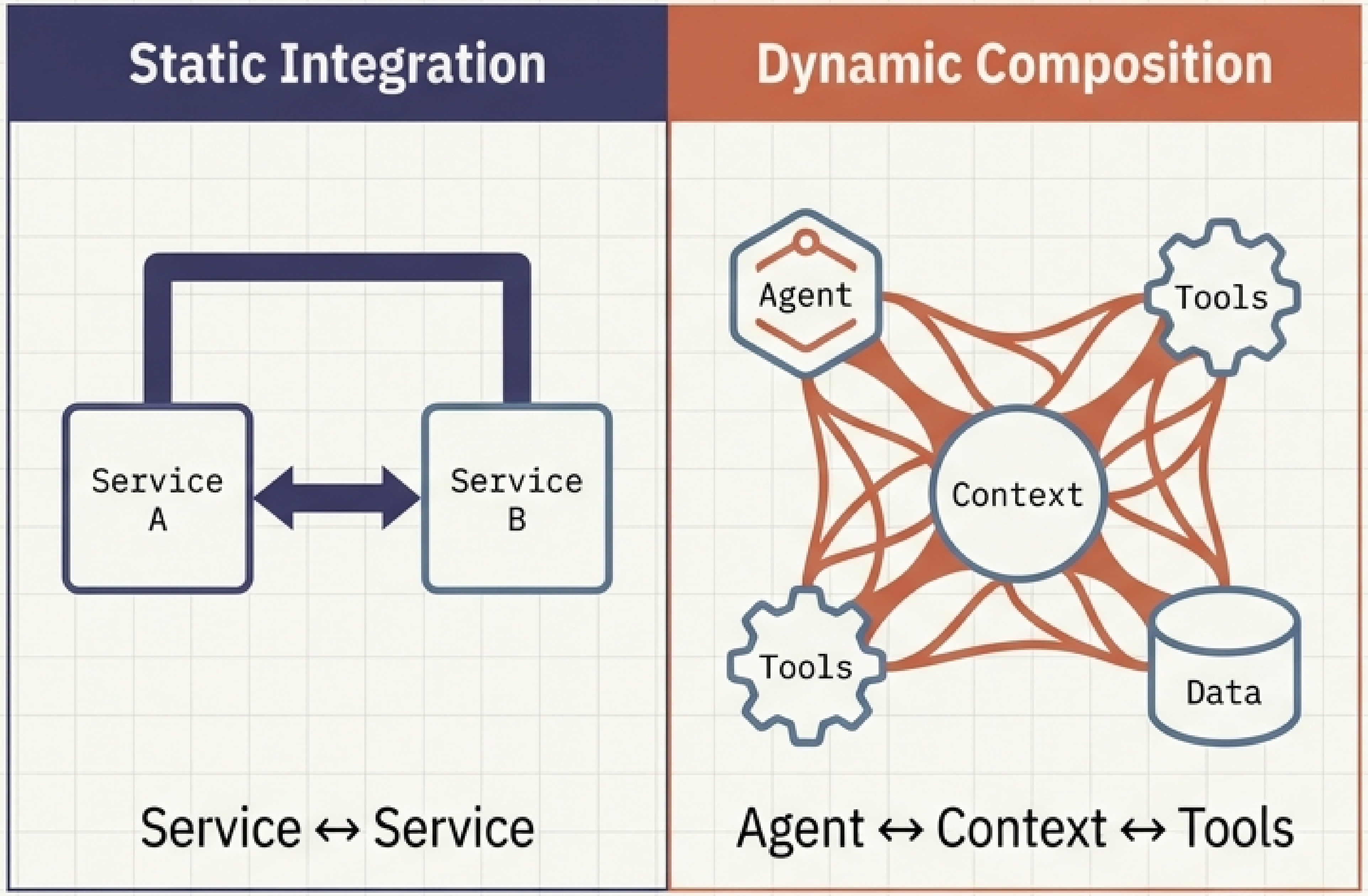
Takeaway: There is no single winner. Modern systems are hybrid composites. This heterogeneity makes manual integration difficult, setting the stage for autonomous agents.



NotebookLM

2020s: The Agentic Shift

We stop integrating services, and start integrating agents.



New Middleware Requirements:

- **Autonomy:** Actors operate independently.
- **Tool Use:** Agents must discover and execute external functions.
- **Context Exchange:** State is semantic context, not just DB rows.
- **Delegation:** Capability to hand off tasks.

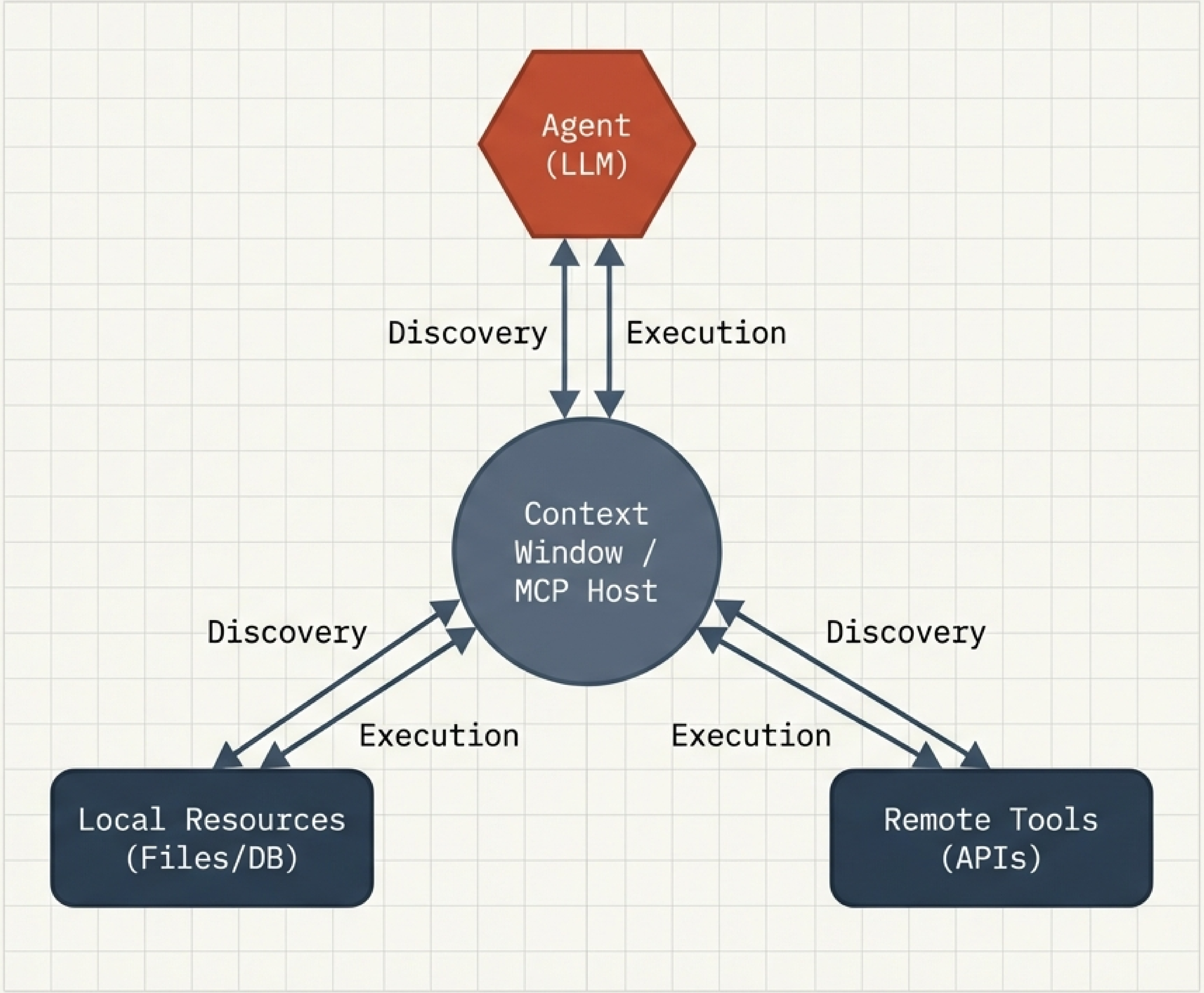
The Model Context Protocol (MCP)

UNIT OF INTEGRATION: CONTEXT & TOOLS

Definition: Middleware specifically designed for agents. It replaces rigid API calls with dynamic discovery and context injection.

Mechanism: Enables Runtime Tool Usage. Agents understand *what* a tool does, not just *how* to call it.

Analogy: MCP = RPC + Messaging + Discovery (Agent-Centric).



Engineering Editorial

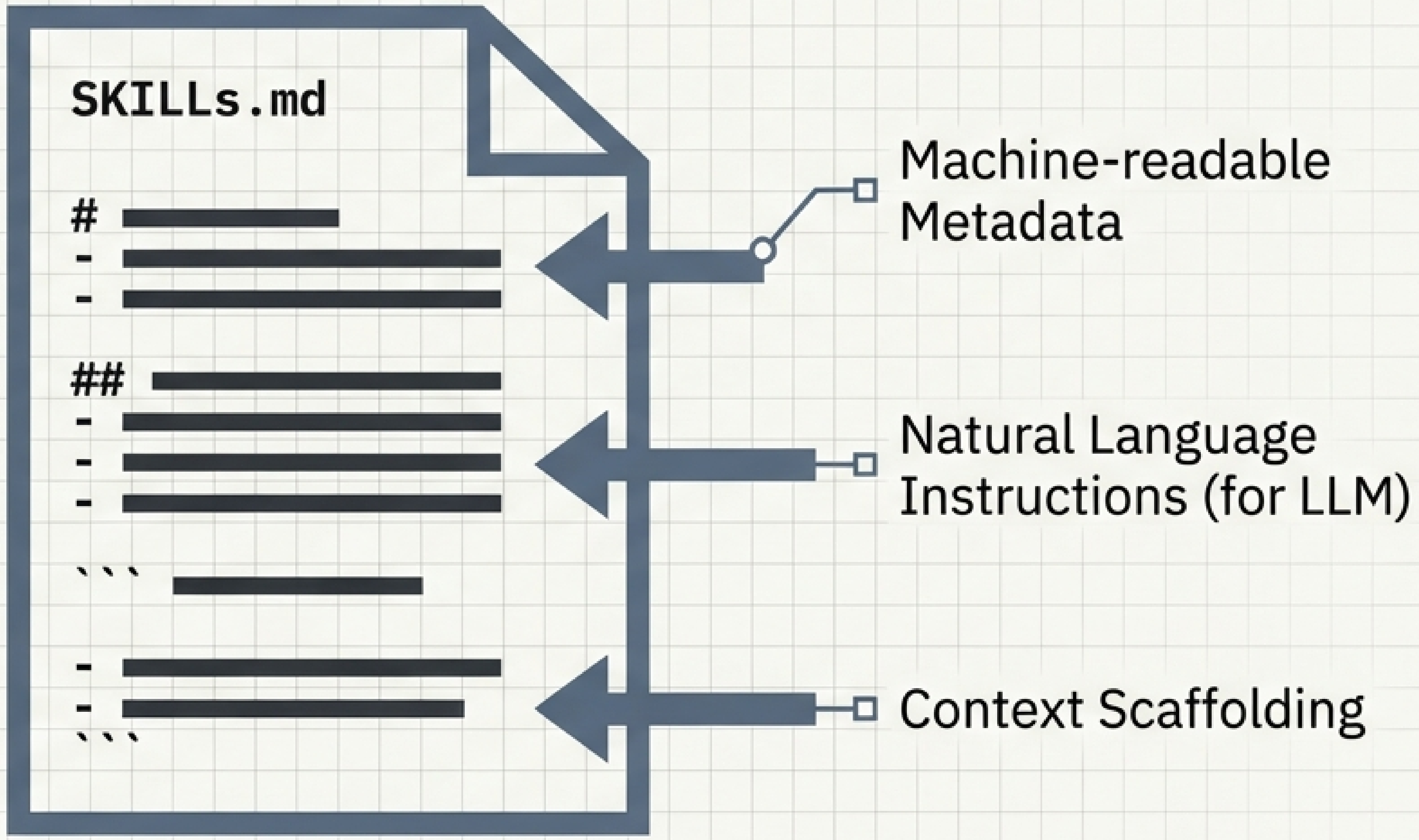
SKILLs.md: Capability as Middleware

Metadata is the New Interface

The Shift: Moving from static "openapi.json" to semantic "SKILLs.md".

Impact: Replaces integration with Composition.

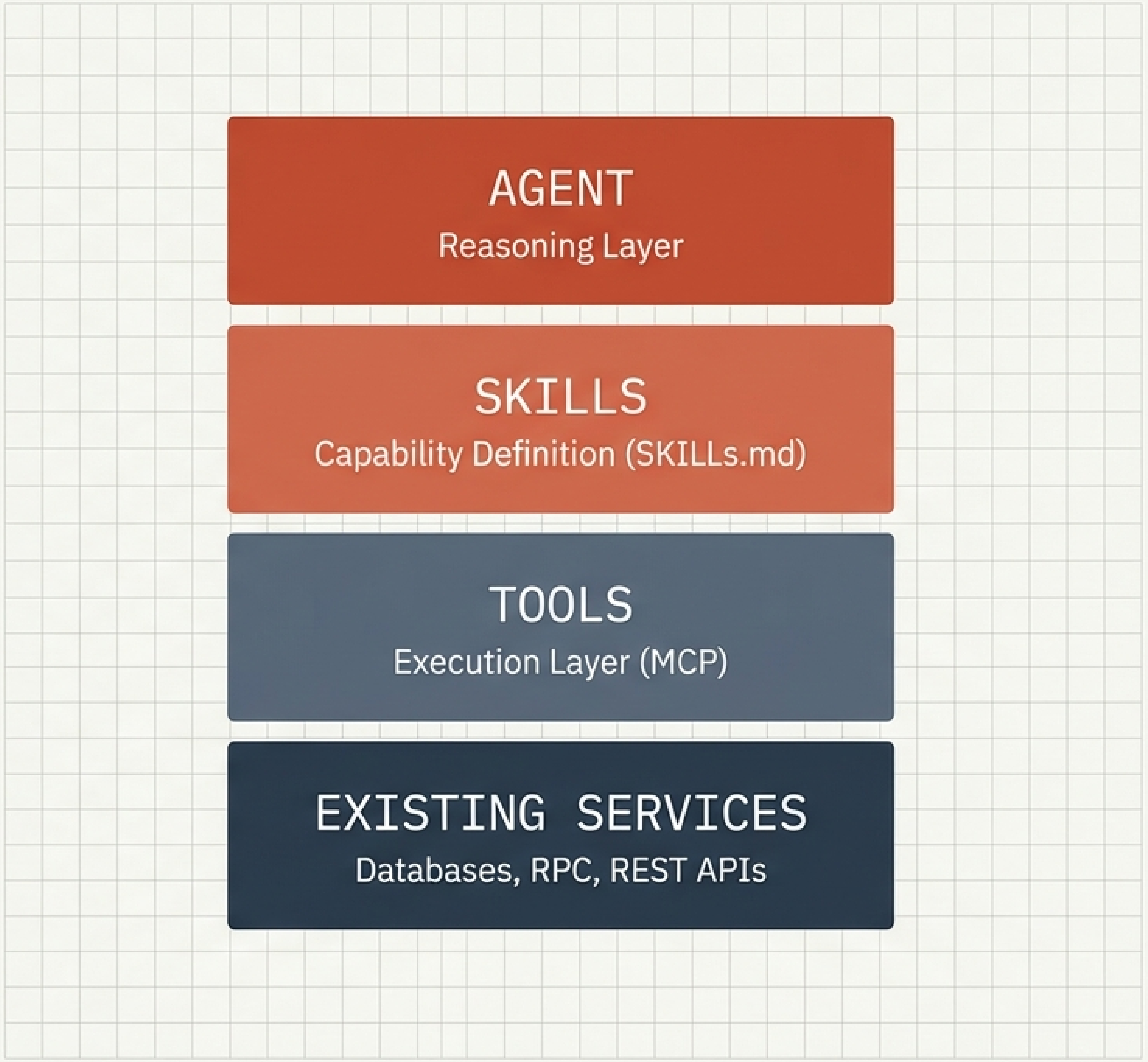
Agents read the "manual" at runtime **to figure out how to work together.**


SKILLs.md
#
-
-
##
-
-
-
```
-
-
```
Machine-readable Metadata
Natural Language Instructions (for LLM)
Context Scaffolding

Engineering Editorial

The New Anatomy of Application Architecture

Existing services do not disappear; they become “Tools” wrapped in “Skills” for “Agents”. This preserves legacy investment while enabling the new era.



History Rhymes: The New Architectural Trade-offs

Old Question	New Question
RPC vs. Messaging?	API vs. Agent Interaction?
Sync vs. Async?	Tool Use vs. Autonomy?
Service Orchestration?	Agent Collaboration?

We are entering a new cycle of architectural decision-making where we must balance control with autonomous behavior.

The Future of Connecting Reasoning Entities

Era	Unit of Integration
RPC Era	Functions
Object Era	Objects
Messaging Era	Messages
Web Era	Documents/Resources
Agentic Era	**Capabilities & Context**

The evolution from rigid procedures to fluid context is complete. As we build the next generation of systems, we must optimize for context exchange as rigorously as we once optimized for latency.