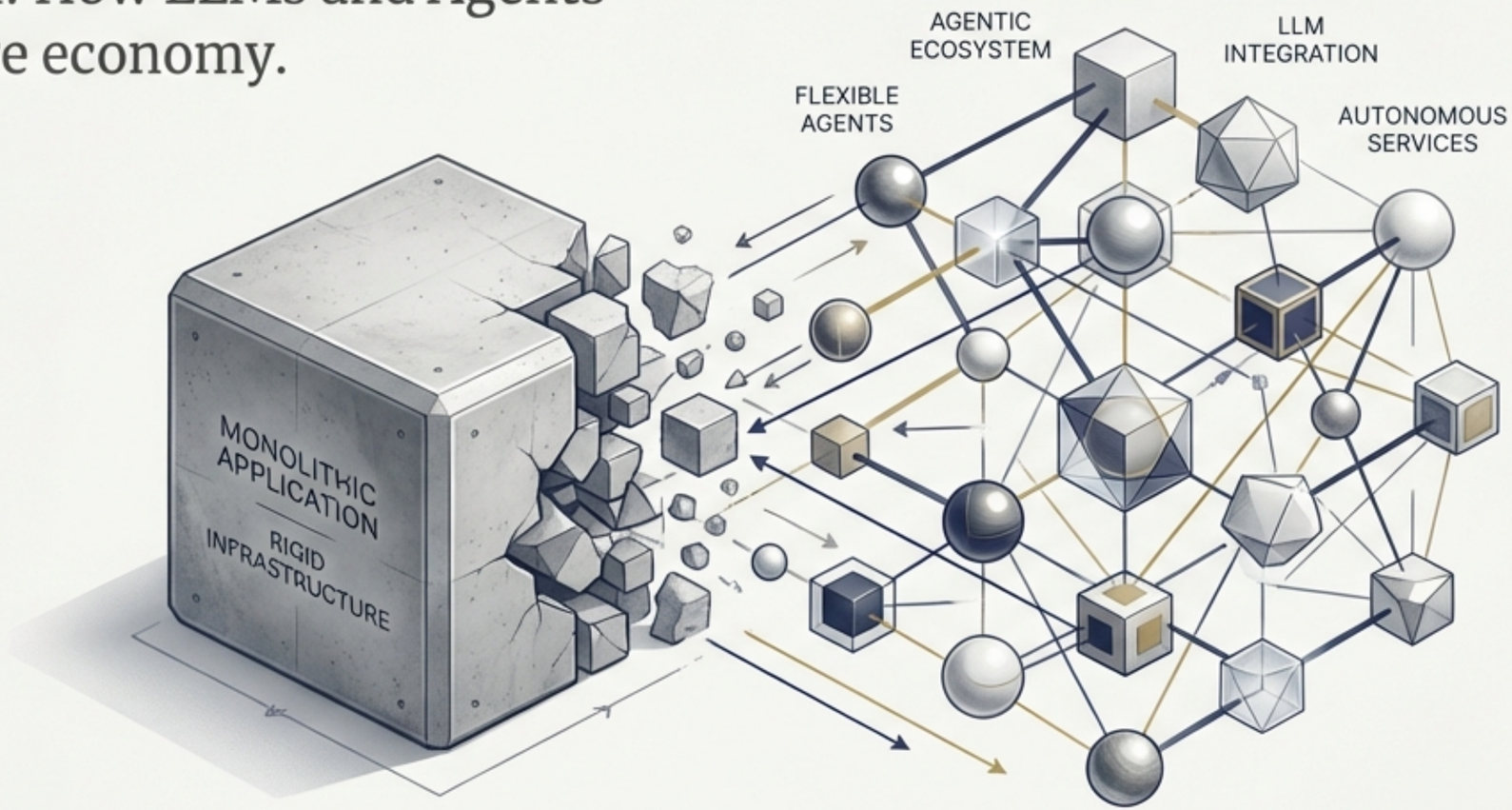
The Trillion-Dollar Shift to Agentic Software Architecture

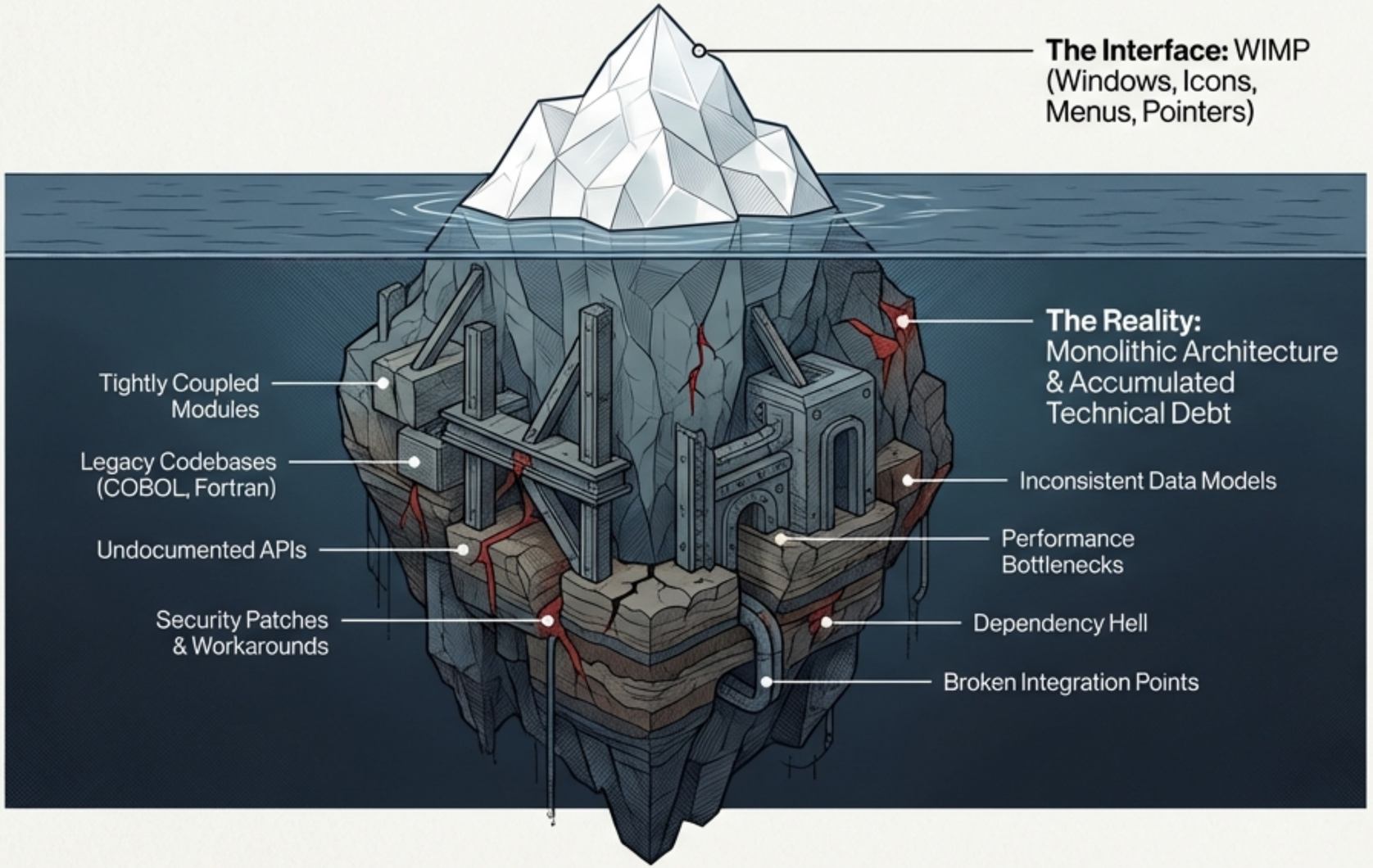
Moving beyond the Monolith: How LLMs and Agents are restructuring the software economy.



The software industry is undergoing a structural redesign comparable to the shift to Cloud or Mobile. We are moving from rigid, monolithic applications to flexible, agent-driven ecosystems.

This document outlines the architectural imperative for technical decision-makers and investors.

The GUI is the tip of the iceberg; beneath lies a mountain of technical debt



The Situation Analysis

WIMP interfaces dramatically improved usability, concentrating value in platforms like Windows and macOS. However, they obscured the architectural realities beneath.

The Hidden Cost

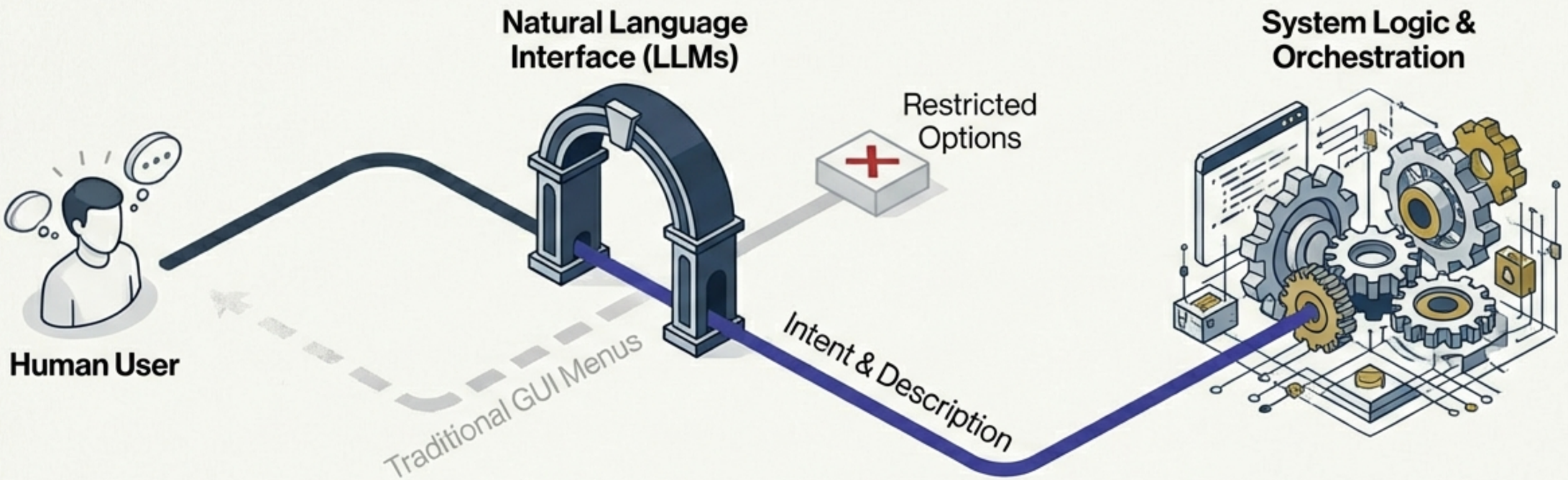
While Unix/Linux backends relied on command lines, user-facing software became tightly coupled monoliths to support the GUI.

Open standards promised modularity, but API inconsistencies led developers to make pragmatic compromises.

The Result

Hard-to-maintain systems where easier usability led to harder maintenance.

Large Language Models break the GUI monopoly



Multimodal & Multilingual

LLMs introduce natural language as a new UI/UX layer, accessible to everyone.

The Paradigm Shift

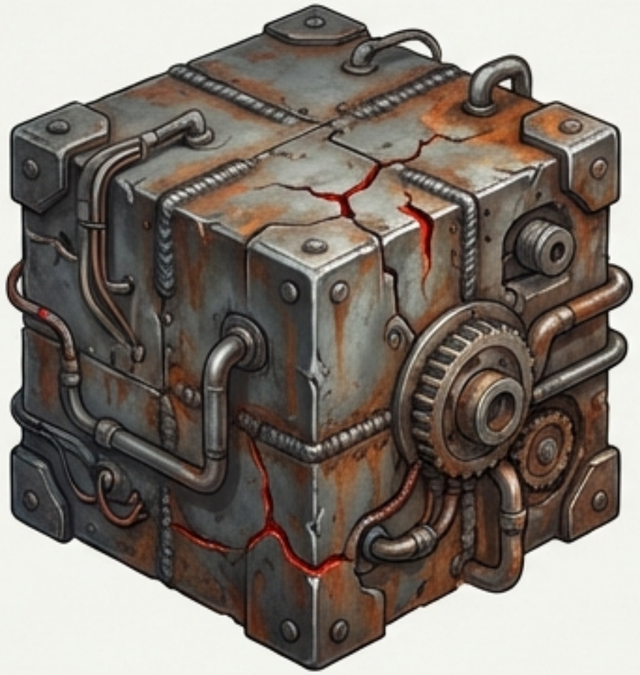
Natural language is no longer just for *using* software; it is for *building and orchestrating* it.

Direct Access

This allows interaction with logic directly, bypassing rigid interface constraints and unlocking the ability to describe intent rather than navigate menus.

We are returning to the ideal of **loosely coupled software**—this time, it works

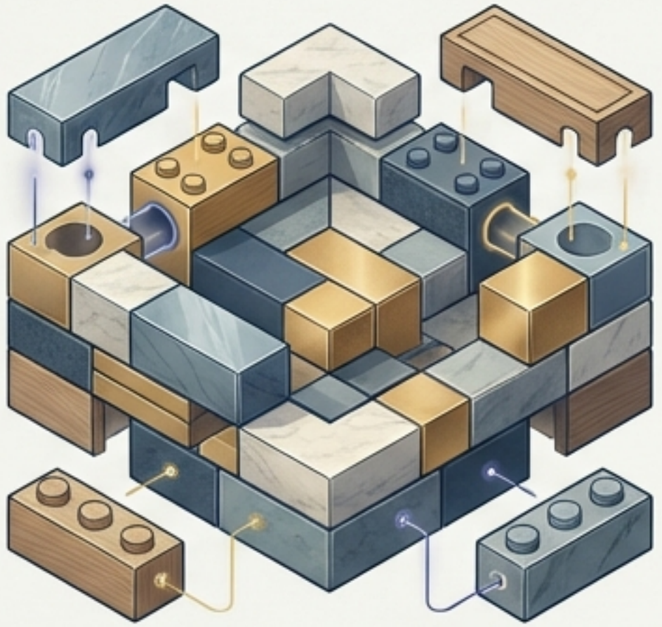
Tightly Integrated Monoliths



Rigid structures where changes require massive refactoring.

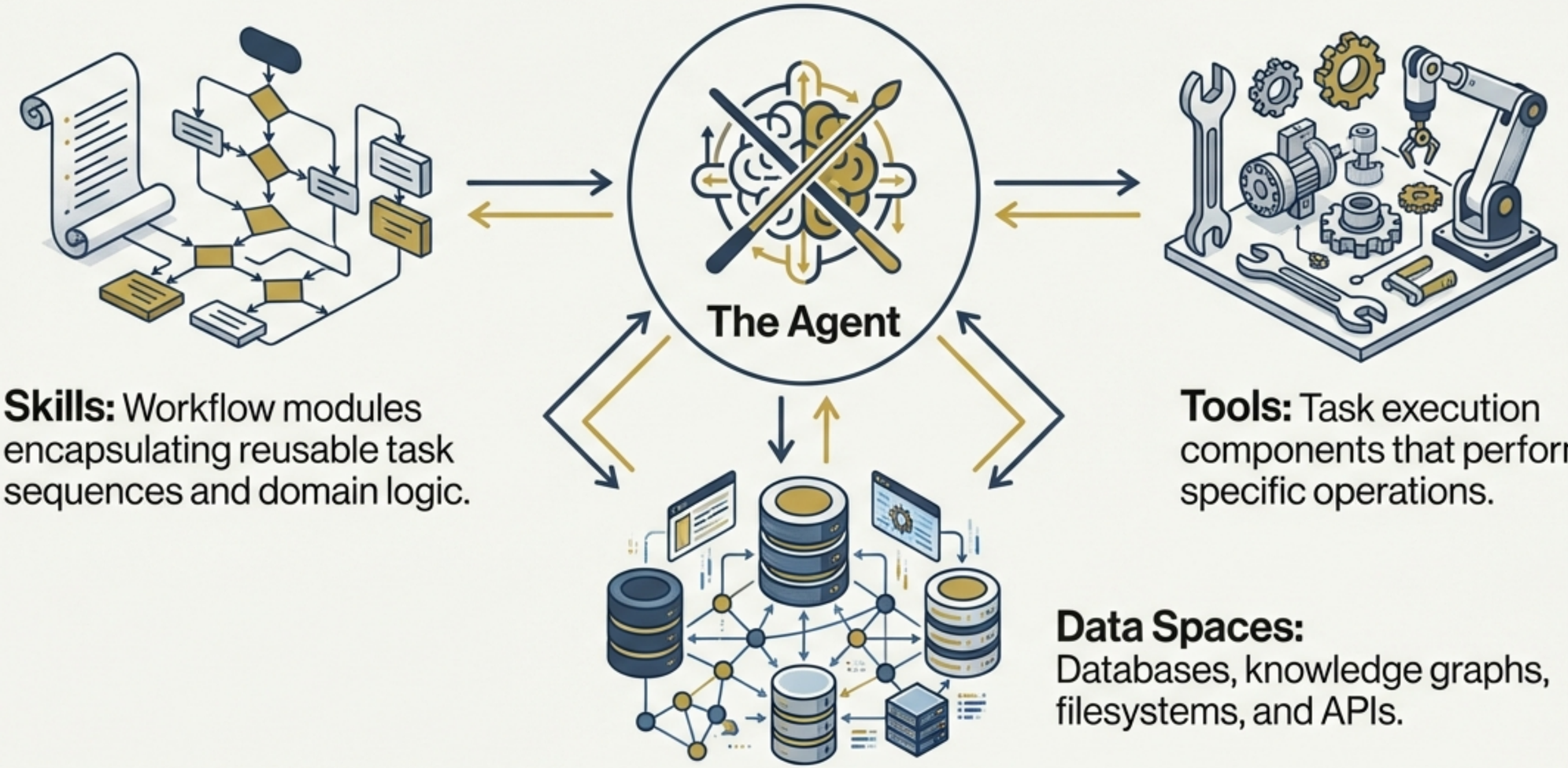
Dynamic Composition

LLMs allow components to connect dynamically based on context.



The shift to LLMs revives the architectural ideal of software assembled as loosely coupled components connected via open standards. Benefits include reduced technical debt, improved adaptability, and systems that evolve rather than stagnate.

Software is no longer hard-coded; it is orchestrated



Skills: Workflow modules encapsulating reusable task sequences and domain logic.

Tools: Task execution components that perform specific operations.

Data Spaces:
Databases, knowledge graphs, filesystems, and APIs.

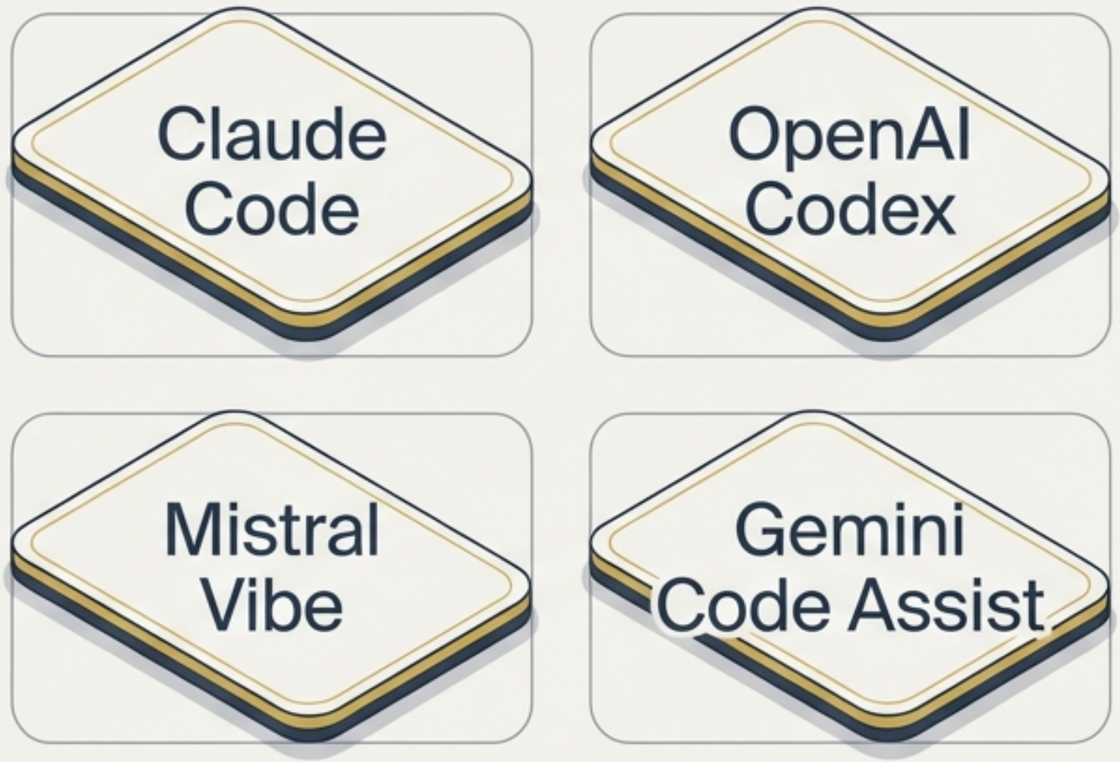
Agents act as the glue (orchestrators), dynamically coordinating these three elements to deliver outcomes aligned with user intent, rather than executing rigid application logic.

From writing lines of code to describing intent

Development is shifting from writing code line-by-line to describing intent and assembling capabilities.

The New Workflow: Users, domain experts, and developers collaborate with coding agents.

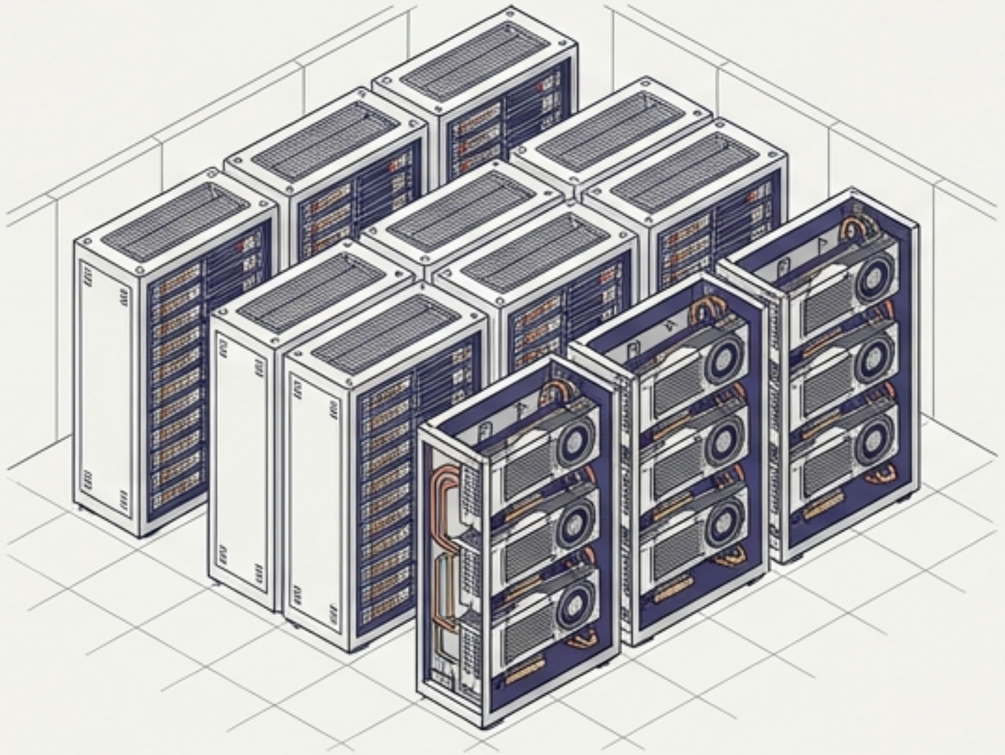
The Outcome: Software becomes a “composite system” where humans collaborate with agents to assemble capabilities.



Tools that began in the command line are now integrating into graphical environments, democratising access.

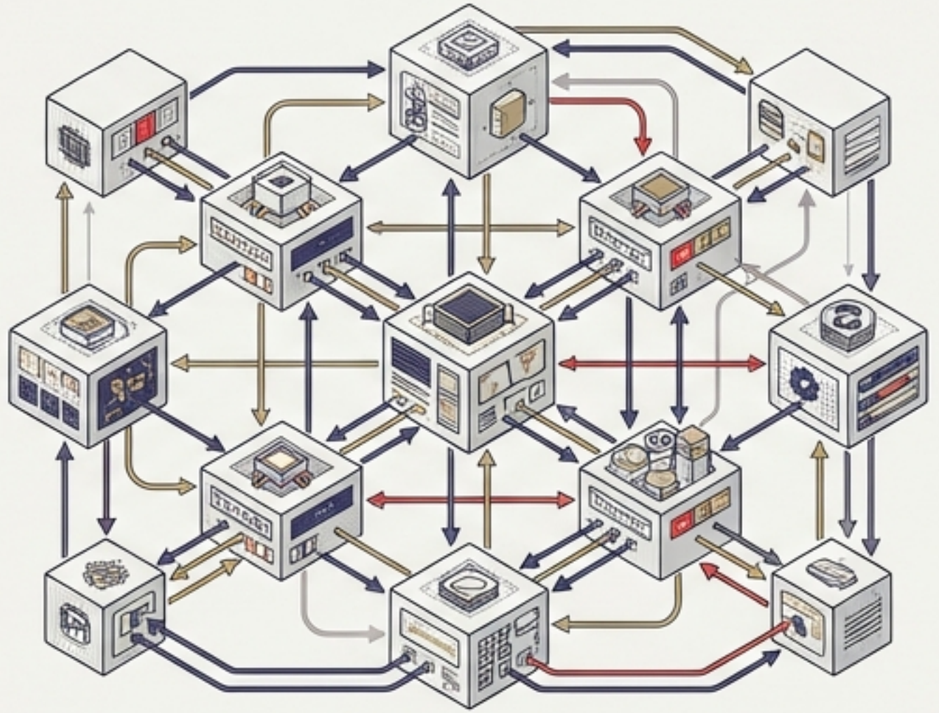
Agentic infrastructure is to software what GPUs are to AI

Infrastructure Hardware



Data Centres: Redesigned for Parallel Processing

Infrastructure Software



Architecture: Redesigned for Agentic Orchestration

The build-out of GPU infrastructure forced a global redesign of data centres. A similar shift is now unfolding in software architecture. This is an infrastructural imperative: existing monoliths cannot compete with the speed and adaptability of agent-driven systems.

Broadening participation and crushing technical debt

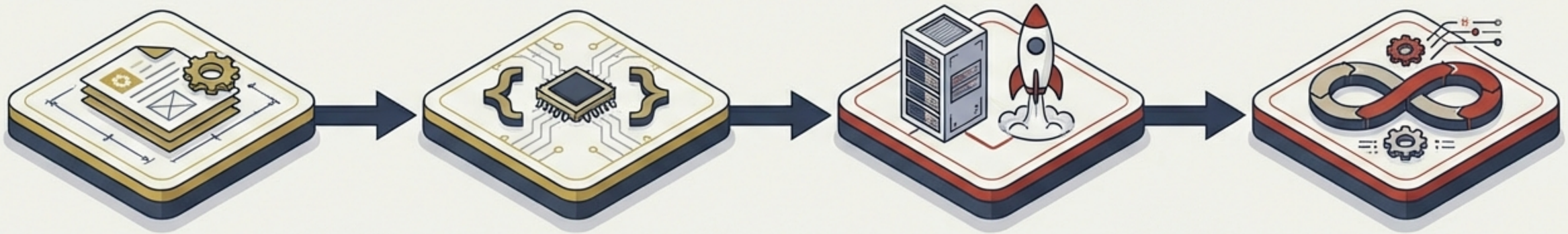
Unified Process

Product Specification

Programming

Deployment

Evolution



Democratisation

Markdown and natural language act as the new assembly language, allowing domain experts to join the development process.

Efficiency

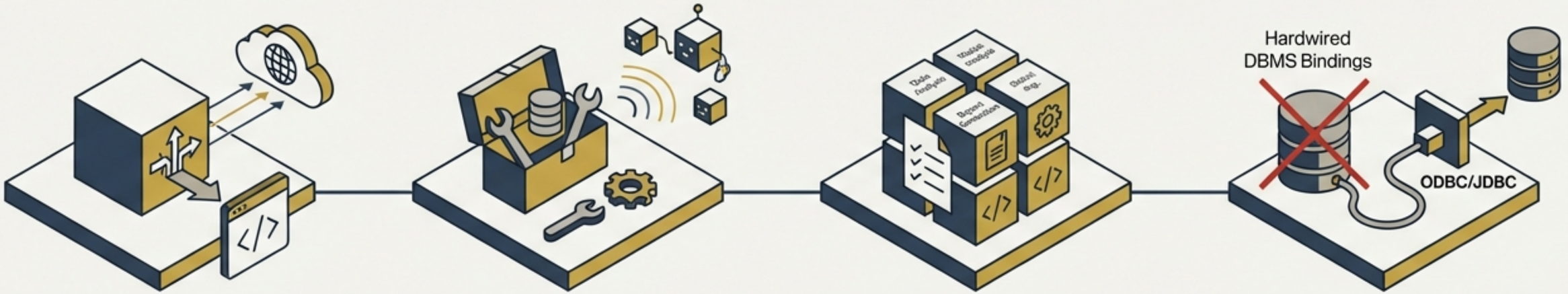
Multiple stages of development are compressed into a unified descriptive process. Traditional hand-offs and siloed expertise are eliminated.

Economic ROI

Reliance on monoliths diminishes, reducing the accumulation of value-eroding technical debt.

Horizontally focused ISVs must evolve into ‘Agent-Ready’ service layers

Transition Strategy



Step 1: Expose Core Functions

Use **OpenAPI** to make functionality accessible to external systems.

Step 2: Publish Tools

Use the **Model Context Protocol (MCP)** so agents can discover services dynamically.

Step 3: Package Skills

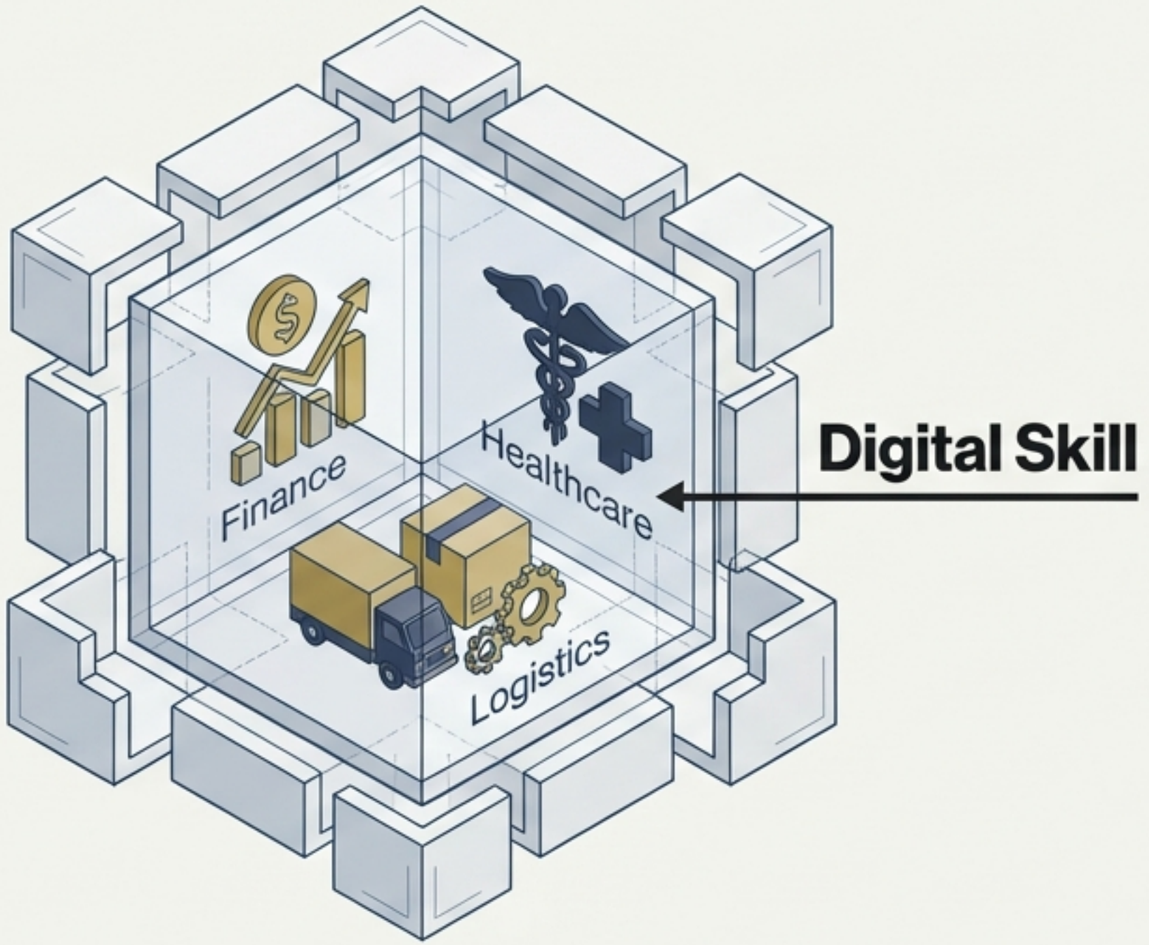
Deploy capabilities via the emerging **SKILLs.md** standard.

Step 4: Modernise Data Access

Move away from hardwired DBMS bindings to portable access via **ODBC/JDBC**.

Takeaway: Applications must become reusable toolkits for agent orchestration rather than isolated products.

VARs move from System Integrators to Orchestrators of Domain Intelligence



The opportunity lies in packaging domain expertise into agent-enabled solutions.

Strategic Blueprint:

1. Create APIs over industry systems of record.
2. Publish vertical tools via MCP.
3. Package domain workflows as Skills for reuse.
4. Decouple solutions from specific databases.

OpenLink provides a vertically integrated portfolio for the Agentic era

OpenLink AI Layer (OPAL)

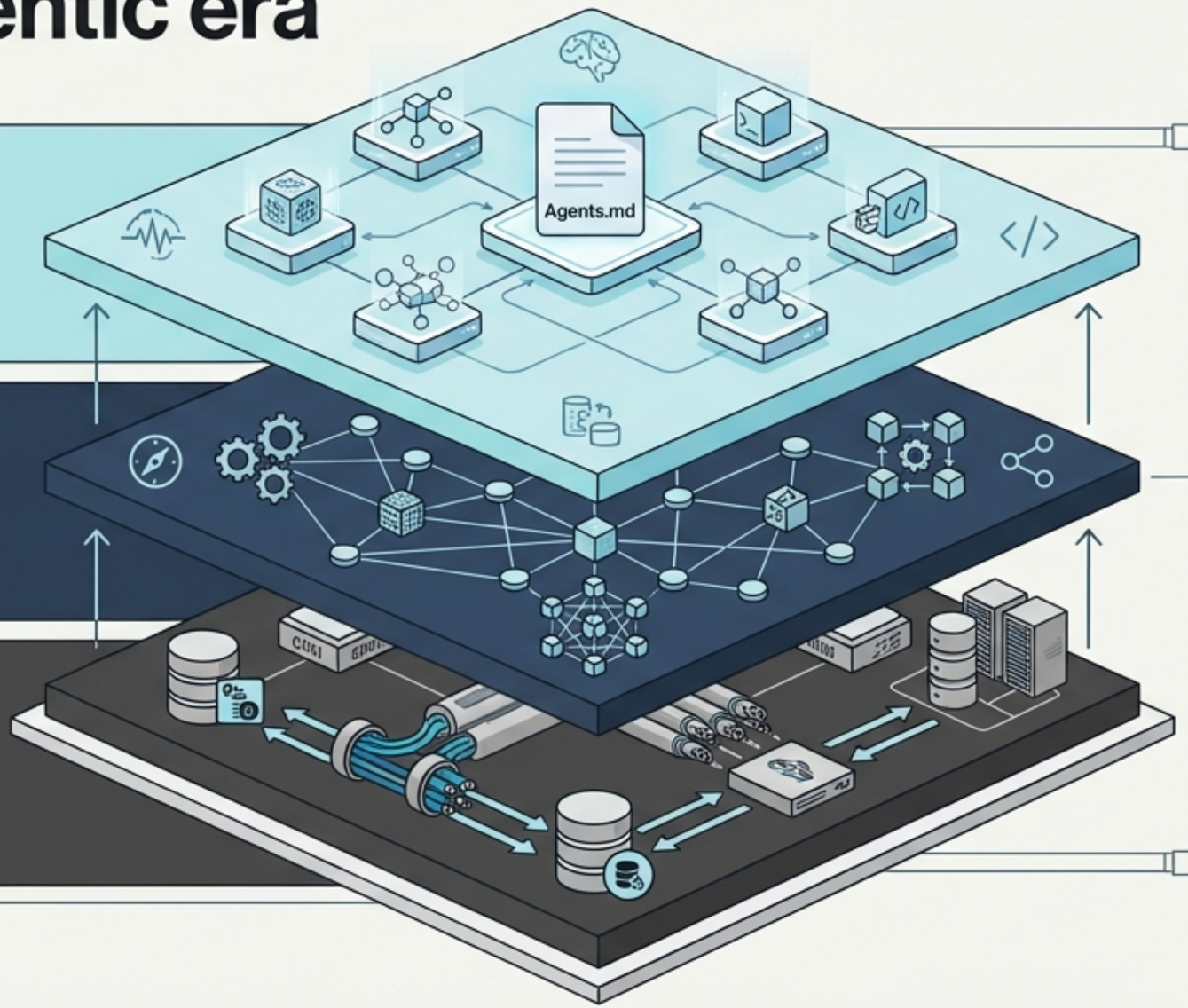
Tooling for creating agents using Markdown (Agents.md).

Virtuoso

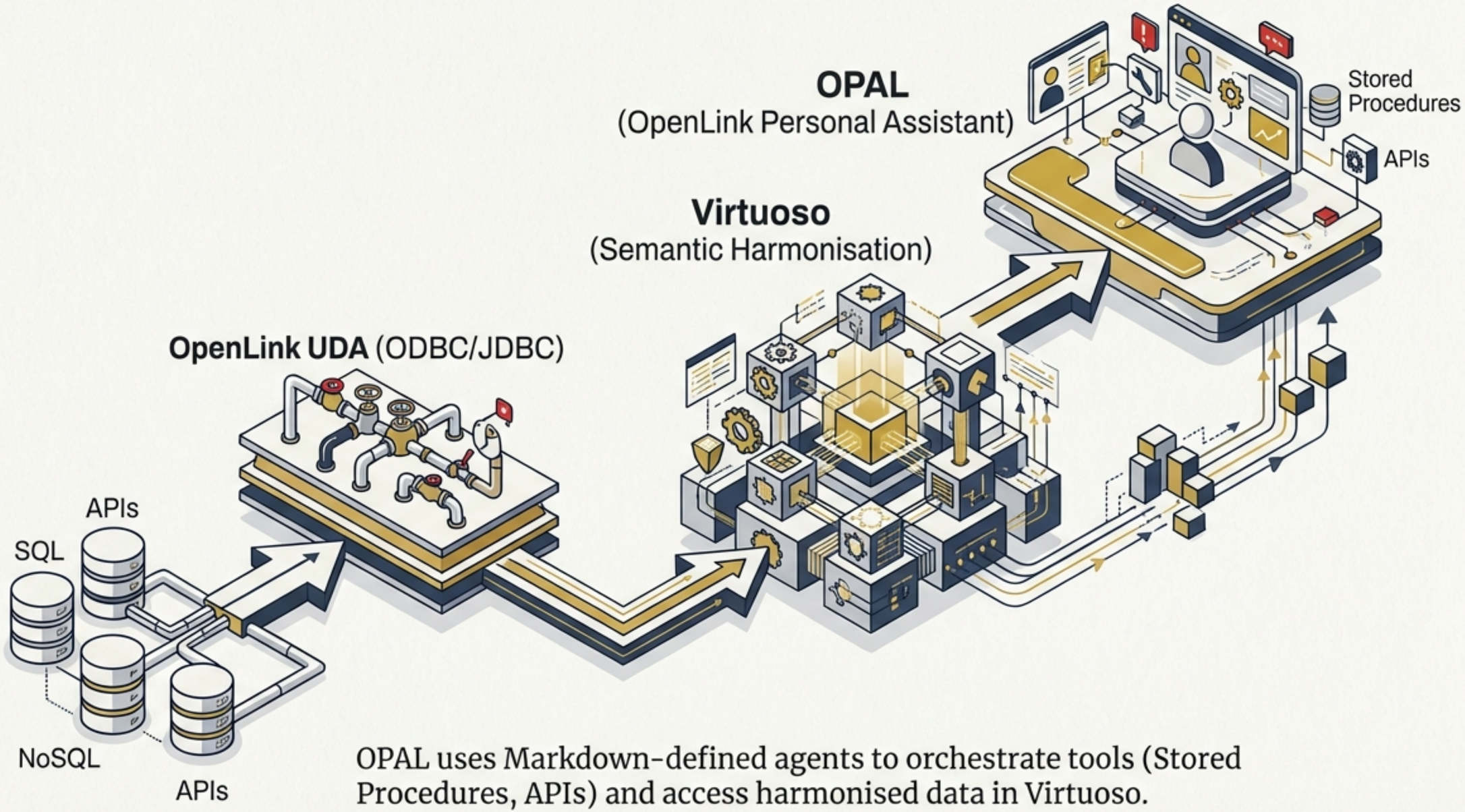
Semantic Harmonisation layer loosely coupled with data spaces.

ODBC/JDBC Drivers

High-performance connectivity to systems of record and data lakes.



Connecting the dots between Data, Semantics, and Agents



OPAL uses Markdown-defined agents to orchestrate tools (Stored Procedures, APIs) and access harmonised data in Virtuoso.

Modernising Model-View-Controller for the AI Age

Diagram A: Classic MVC

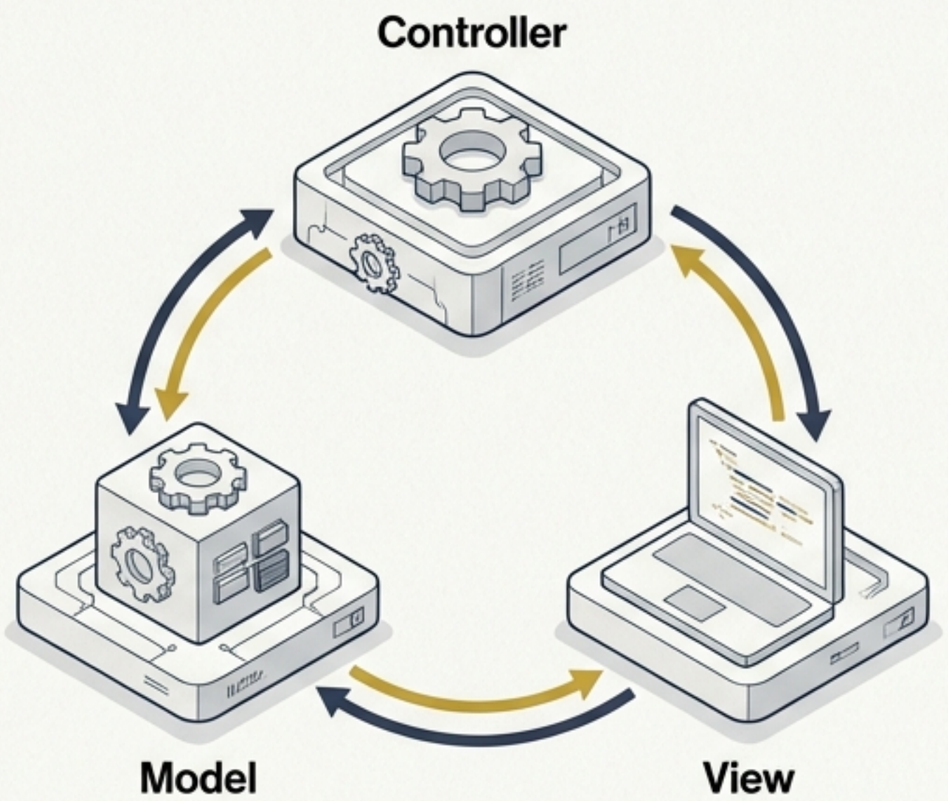
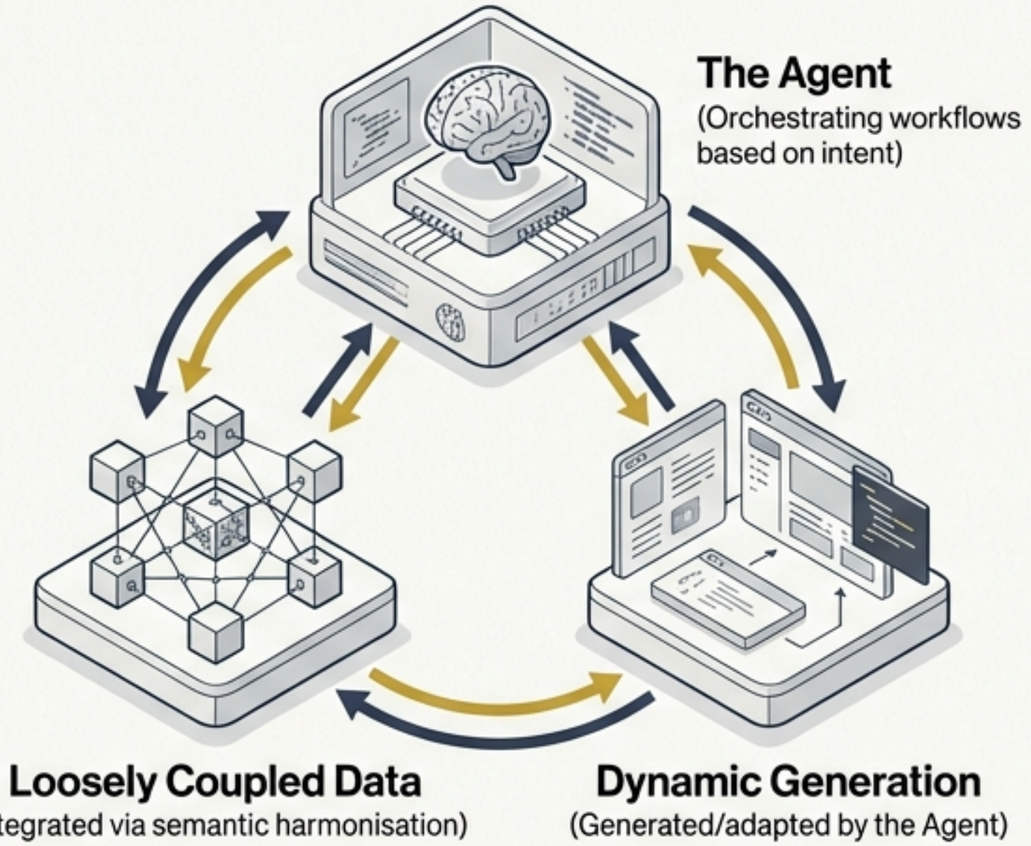


Diagram B: Agentic MVC



In the agentic era, the Controller becomes an autonomous Agent, the View becomes dynamic, and the Model becomes a semantic data space.

The Trillion-Dollar Opportunity

We are witnessing the end of the monolithic era and the rise of the agentic ecosystem. AI agents are the new orchestration layer that binds modular components into adaptive systems.

Takeaway: Those who adopt open standards (OpenAPI, MCP) and agent-ready architectures will capture the value. Those clinging to the 'Iceberg' of monolithic debt will struggle to evolve.

Deep Dive Resources

- General Reading
  - Software Componentization in the Age of AI: How LLMs Are Reshaping Development
  - LLMs as Generic RDF Clients: Understanding the semantic connection
- Tooling Documentation
  - About Claude Code (Anthropic)
  - About OpenAI Codex
  - Mistral Vibe
  - Gemini Code Assist
- OpenLink Technology
  - OPAL (opal.openlinksw.com)
  - Virtuoso Universal Server
  - UDA Drivers