Strategic Analysis of Agentic AI Frameworks (2025)

Rajendra Singh Rawat
`@airawatraj`

December 9, 2025

Abstract

This report provides a strategic comparative analysis of leading Agentic AI frameworks and enabling technologies, including LangChain, CrewAI, AutoGen, Semantic Kernel (SK), Vertex AI, and Pydantic AI. It evaluates them based on core capabilities, language support, market adoption, and critically, the risk of vendor lock-in. The findings suggest that while open-source frameworks offer flexibility, the choice for enterprise deployment often balances this flexibility against the production readiness and governance provided by platform-based solutions, emphasizing the role of robust data validation (like Pydantic AI) for agent reliability.

1 Introduction to Agentic AI Frameworks

Agentic AI frameworks transform Large Language Models (LLMs) from reactive prompt-response systems into proactive, goal-driven digital workers. They provide the necessary components—planning, tool use, memory, and orchestration—to manage complex workflows. The selection of a framework is a critical architectural decision, directly impacting development agility, deployment costs, and long-term strategic independence.

2 Framework Comparison and Analysis

The table below summarizes the key attributes, including the languages supported and the critical assessment of vendor lock-in risk. The Lock-in Risk is categorized based on the reliance on proprietary platforms and the difficulty of switching LLM providers or underlying cloud services.

Table 1: Agentic AI Frameworks and Enabling Technologies: Core Attributes and Lock-in Risk

Framework	**Primary Role**	**Key Language(s)**	**Lock-in Ri**
LangChain / LangGraph	General Orchestration / State Graphs	Python	Low (Code
CrewAI	Role-based Collaboration	Python	Low (Code
AutoGen	Conversational Systems	Python	Low (Code
Pydantic AI	Structured Output / Data Validation	Python	Very Low (
Semantic Kernel (SK)	Enterprise Integration / Skills	C#, Python	High (Ecos
Vertex AI (Managed)	Managed Agent Platform	Python, Node.js, Java, Go	Moderate
Google ADK	Open-Source Tooling/Abstractions	Python, Go, Java	Low (Code

2.1 Visual Data and Capabilities

For an interactive guide and visualization of the comparative performance and market position of these frameworks, including the Capability Radar Chart and the Adoption and Community Growth Chart, please refer to the following external resource:

https://rawatlabs.github.io/research/agentic-frameworks.html

2.2 Key Findings from Capability Analysis

Based on the capability metrics, the frameworks show distinct strengths:

- LangChain/LangGraph: Maintains the highest overall adoption, offering superior State Management and RAG (Retrieval-Augmented Generation) Power due to its extensive ecosystem and modular design. LangGraph introduces graph-based control, significantly boosting orchestration capability.
- Pydantic AI (Enabling Technology): While not an orchestrator, Pydantic AI is fundamental to modern Agentic AI. It provides the necessary infrastructure for Structured Output and Reliable Tool Use by validating the JSON response schemas from LLMs, which is critical for making agents robust and deterministic in production environments.

- AutoGen & CrewAI: These frameworks excel in Multi-Agent Systems. AutoGen prioritizes asynchronous, free-form conversation, while CrewAI provides a structured, role-based delegation model, leading to higher Ease of Start for collaborative tasks.
- Vertex AI (Managed): Offers the highest Observability and Production Readiness due to its nature as a managed platform solution, providing integrated monitoring, security, and scalability features expected in an enterprise cloud environment. The platform offers a complete solution for deploying, monitoring, and governing agents at scale.
- Google ADK: Provides open-source Python, Go, and Java SDKs to create agent components, offering a more flexible, code-centric alternative to the fully managed Vertex AI environment.
- Semantic Kernel: Positioned strongly for integration into existing `.NET` and C# enterprise applications, focusing on creating reusable "skills" rather than complex graph orchestrations.

3 Analysis of Vendor Lock-in Risk

The risk of vendor lock-in is a primary strategic consideration, especially for mission-critical AI applications. In the context of Agentic AI, lock-in occurs at three levels: Codebase, Model, and Platform/Ecosystem.

3.1 Low Lock-in Risk (LangChain, CrewAI, AutoGen, Pydantic AI, Google ADK)

These open-source, community-driven frameworks and toolkits are fundamentally LLM-agnostic.

- Mechanism: They utilize abstract interfaces to connect to various LLM providers (e.g., OpenAI, Gemini, Anthropic) and vector databases. Pydantic AI facilitates reliable data contract creation, and the Google ADK provides abstraction layers for common agent functions.
- Benefit: This design allows enterprises to switch models or providers based on performance, cost, or regulatory requirements with minimal code refactoring, limiting strategic dependence on any single vendor's roadmap or pricing structure.

3.2 Moderate Lock-in Risk (Vertex AI Managed Platform)

Platform-based solutions like Vertex AI Managed Platform carry a moderate risk.

- Mechanism: While the underlying agents can be built using low-lock-in tools like the Google ADK, core functionalities (like high-scale deployment, integrated data connectors, and built-in security governance) are deeply coupled with the Google Cloud ecosystem.
- Impact: Migrating a fully managed, production-scale Vertex AI agent to a different cloud or environment would require a significant effort to re-implement the surrounding MLOps infrastructure and security controls.

3.3 High Lock-in Risk (Semantic Kernel)

Semantic Kernel (SK), primarily driven by Microsoft, presents the highest potential for ecosystem lock-in, particularly for C# implementations.

- Mechanism: SK's design is heavily optimized for integration with Microsoft's enterprise stack (Azure services, Microsoft 365, `.NET` environments).
- Impact: While SK has Python support, organizations using its C# core and deeply integrating it with Azure AI services face high switching costs. The convenience of deep integration with the existing enterprise architecture becomes a technical and financial barrier to adopting non-Microsoft solutions in the future.

4 Conclusion and Strategic Recommendation

The Agentic AI landscape offers a dichotomy between flexible, open-source orchestration tools (LangChain, CrewAI, AutoGen, Google ADK) and vendor-specific enterprise platforms (Semantic Kernel, Vertex AI Managed Platform). The choice is further complicated by the need for reliable data processing, where tools like Pydantic AI are essential.

- For Maximum Flexibility: Organizations prioritizing model and LLM-provider independence should select open-source Python frameworks (e.g., LangChain for general tasks, CrewAI/AutoGen for complex multi-agent flows) and always integrate a robust data validation library like Pydantic AI for structured outputs. Using the Google ADK further enhances flexibility for building components in Python, Go, and Java.
- For Enterprise Deployment: Organizations with existing dependencies on a cloud provider's ecosystem should consider their respective platforms (e.g., Vertex AI Managed Platform for Google Cloud, Semantic Kernel for Microsoft/Azure). However, they must adopt middleware and abstraction layers (e.g., using a common API gateway for all LLM calls) to mitigate the long-term strategic risk of vendor lock-in.

The critical takeaway is that an architecture designed for interoperability, swappability, and data reliability is the most effective defense against the rapid evolution and vendor churn inherent in the Agentic AI space.

Legal Disclaimer

References

[1] Google, ”Google ADK Documentation,” 2025. Available: https://google.github.io/adk-docs/

[2] Pydantic, ”Pydantic AI Documentation: Reliable structured output for AI,” 2025. Available: https://ai.pydantic.dev/

[3] LangChain, ”LangGraph Documentation: State-based Orchestration,” 2025. Available: https://langchain.com/langgraph

[4] Microsoft, ”Semantic Kernel Developer Guide,” 2025. Available: https://devblogs.microsoft.com/semantic-kernel/

[5] AutoGen Contributors, ”AutoGen: Enabling Next-Generation LLM Applications,” *ArXiv*, vol. 2308, no. 08155, 2024.

[6] CrewAI, ”CrewAI Documentation: Role-based Collaborative AI,” 2025. Available: https://www.crewai.com/

[7] Google Cloud, ”Vertex AI Agent Builder and Orchestration Services,” 2025. Available: https://cloud.google.com/vertex-ai

[8] Google Cloud, ”Vertex AI SDK for Python Documentation,” *Google Cloud Documentation*, 2025. Available: https://docs.cloud.google.com/vertex-ai/docs/python-sdk/use-vertex-ai-sdk