

Making AI Work: From Technology to Business Capability

Surveying the AI Landscape

A technical taxonomy for faster business decisions

Lee Farretta and Matthew Farretta • LMM Labs

Prepared in collaboration with ChatGPT • Article 1 of 3

Research cutoff October 1, 2026 • Published October 17, 2026



AI-generated conceptual artwork. An illustration of the landscape, not a technical diagram.

Abstract

An AI application may combine a model, company information, business software, and controls over how work proceeds. This article organizes those capabilities into five working areas and connects them to product choices. It develops the functional model introduced in Making Sense of AI and the Acceleration Opportunity into a practical taxonomy. Figure 1 shows the relationships between capabilities and products; Table 1 provides ten implementation categories, fifty product candidates, and specific selection criteria. A new flowchart shows how to use both: define the business outcome, identify the capabilities needed, shortlist products, check their fit, and test the candidates against common requirements. The discussion explains the technical distinctions that affect those decisions without treating every application as a custom model-development project. As the first article in the Making AI Work trilogy, it prepares readers to define a system blueprint and then compare buying, combining, and building.

Introduction

An AI application often combines a model with company information and business software. The design challenge is deciding which parts are needed, how they connect, and who controls the result. A useful taxonomy classifies components by function so a team can compare products that perform the same job.

Making Sense of AI and the Acceleration Opportunity introduced five working areas for understanding those functions (Farretta & Farretta, 2026). This article connects them to named products and a method for choosing among them. It begins the Making AI Work: From Technology to Business Capability trilogy. Article 2 develops a system blueprint; Article 3 examines buying, combining, and building. The aim is to reduce repeated investigation and move sooner into useful development.

Reading the taxonomy

Figure 1 answers “What does this component do?” Its branches run from the five working areas to capabilities and product examples. A product can serve several functions, so its position identifies a useful role rather than an exclusive category. Managed platforms appear in a shared band because they span several areas.

Table 1 answers “Which products should we consider for that role?” It groups the options into ten implementation categories, each with five candidates and selection criteria. Read across the row that matches a required capability. Figure 2 then shows how to turn that classification into a shortlist and a set of selection checks. There is no need to choose a product from every branch.

Learning and generating

Predictive models classify items, estimate quantities, or forecast future values. Libraries such as scikit-learn provide methods for developing and comparing these models (scikit-learn developers, n.d.). Generative models produce language, code, images, or other content. Model services expose these capabilities through application programming interfaces (APIs), allowing a team to use a model without training its own.

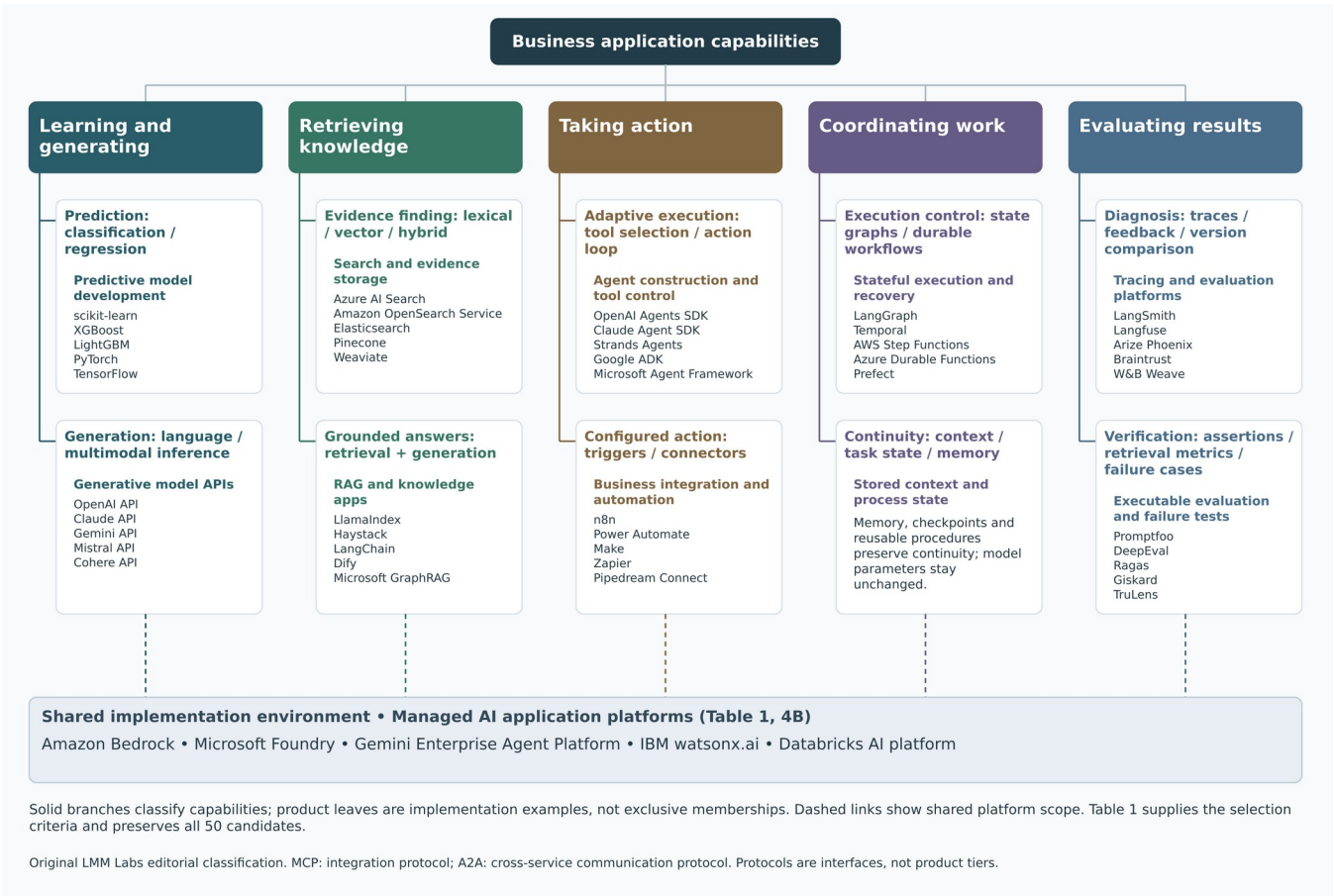
Training and fine-tuning change a model’s learned parameters. Instructions and retrieved documents influence a response without, by themselves, changing those parameters. This matters when company information changes: updating access to documents is different from adapting the model itself. Table 1 separates predictive development from generative APIs so the shortlist begins with the kind of task the application must perform.

Retrieving knowledge

Retrieval finds information outside the model. Keyword search is useful for exact terms and identifiers. Vector search compares numerical representations, called embeddings, to find related meaning. Hybrid search combines the two (Microsoft, 2026). The application must also check permissions and document revisions; similarity alone does not establish that information is authorized or applicable.

Retrieval-augmented generation (RAG) gives retrieved material to a model as evidence for its response (Lewis et al., 2020). Table 1 separates search and storage products from tools for building RAG applications. The latter help connect document processing, retrieval, and generation, but the design still needs source attribution and answer checks. GraphRAG adds relationships and summaries for questions spanning a collection (Edge et al., 2025). Consider that extra work when the questions require it.

Figure 1
Functional taxonomy from working areas to product examples



Note. Original LMM Labs taxonomy. Follow each branch from a working area to its capabilities and product examples. Dashed links show the shared scope of managed platforms. Product placement identifies a role, not exclusive membership; Table 1 provides the complete shortlist and selection criteria.

Taking action

Tools connect AI to software operations, such as reading a database or submitting a business record. The model can propose an operation and its arguments; application code validates and executes the request. The system must enforce permissions and any required human approval.

A workflow follows predefined steps and branches. An agent selects its next actions from intermediate results, a pattern explored in ReAct (Yao et al., 2023). A workflow can contain a bounded agent stage where adaptation is useful. Compare that flexibility with its added cost and delay (Anthropic, 2024). Table 1 distinguishes agent construction from business integration. Model Context Protocol (MCP) standardizes connections to tools and context; it does not set the application’s authority or approval policy (Model Context Protocol, n.d.).

Coordinating work

Coordination manages steps, waits, and recovery. Context is the information supplied to a model call. Task state records progress and pending decisions. Retained memory makes selected information available to later work. Saving any of these does not, by itself, change the model’s learned parameters.

LangGraph supports flows combining models and code; Temporal supports durable workflows across failures and long waits (LangChain, n.d.-a; Temporal Technologies, n.d.). Recovery must also account for external actions. Repeating a request can duplicate a completed write if the receiving system lacks suitable protection. Agent2Agent (A2A) supports exchanges of tasks and results between agent services (A2A Project, n.d.). Use these components where the process needs them, rather than adding coordination layers by default.

Evaluating results

Evaluation checks whether the system meets its requirements. Observability records what happened so a result can be investigated. A trace links the sources, model calls, tool results, and outcome of an execution (LangChain, n.d.-b). Table 1 separates platforms for these records and feedback from tools that run checks, while recognizing that their features overlap.

The checks must match the failure. A formatted answer can still use the wrong evidence. RAGAs examines retrieval and answer quality; τ -bench examines tool use, final state, and consistency across repeated trials (Es et al., 2024; Yao et al., 2024). These methods inform testing, but an application needs evidence from its own data and integrations. Include correction, human review, elapsed time, and total cost when judging whether it accelerates accepted work.

Using the product map

Use the categories to compare products that address the same requirement. Table 1 identifies differences worth testing, including exact-identifier search, permission filters, approval waits, recovery, and records that explain a failed case. A managed platform may cover several roles, but its services must still fit the application's data, deployment, and integration needs.

Check product lifecycle as well as features. Pipedream Connect, for example, is distinct from the Workflows product scheduled to close in March 2027 (Pipedream, n.d.). Keep the shortlist tied to the required capability and the work that remains. Figure 2 turns these comparisons into a selection process.

Applying the taxonomy

Figure 2 shows how to use the taxonomy and product table together. Begin with a specific business outcome and decide what would count as success. Use Figure 1 to identify the capabilities needed, then Table 1 to shortlist products for those roles.

Check whether the candidates can work together within the data, permission, deployment, and cost limits. Test them with the same representative inputs and acceptance criteria. If a candidate fails, return to the shortlist. If none fits, make any scope change an explicit business decision. Record the selected components and remaining integration work for the system blueprint and make-or-buy analysis.

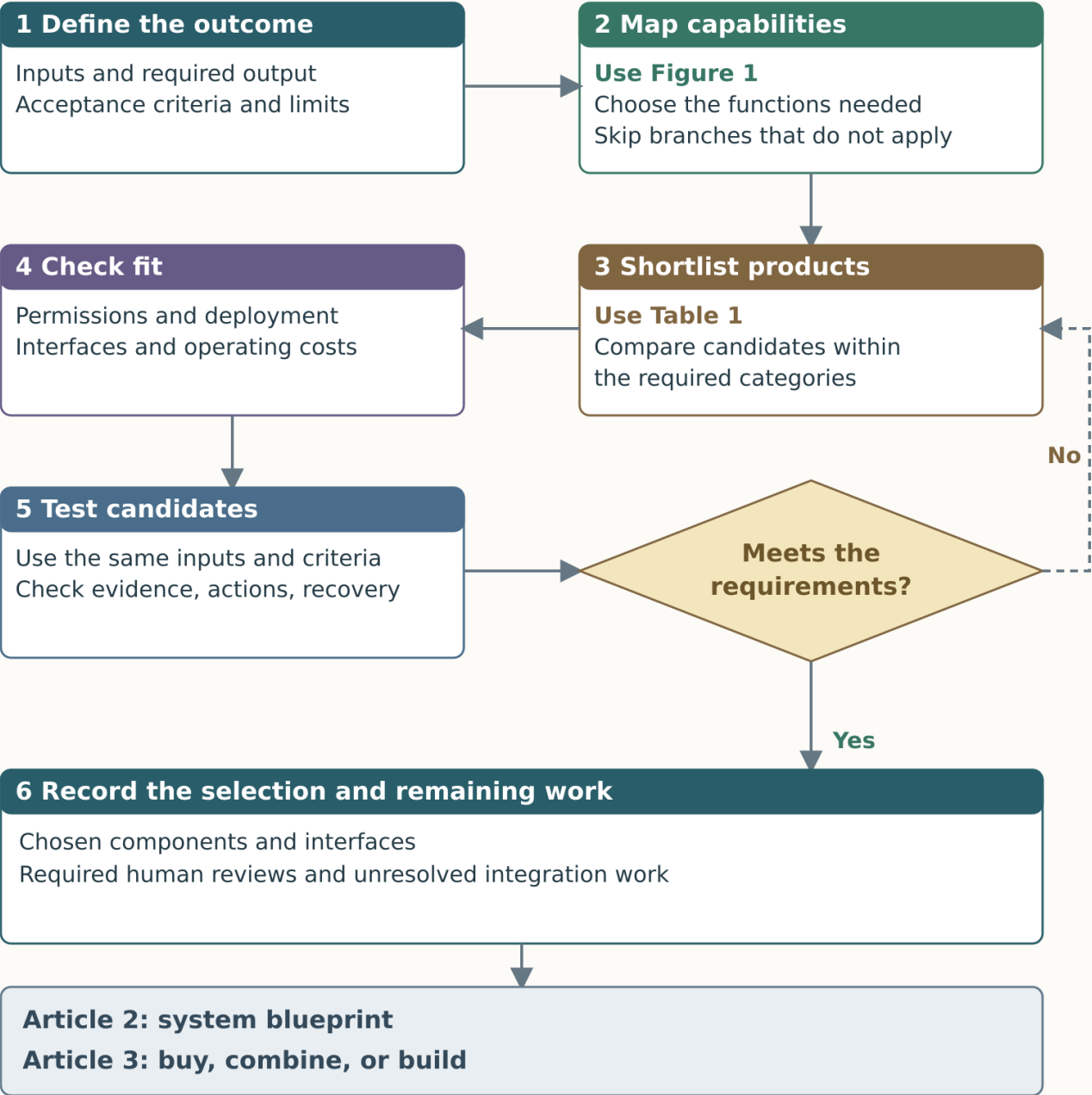
Table 1*Five working areas and ten implementation categories*

Working area	Implementation category A	Implementation category B
Learning and generating	1A Predictive model development scikit-learn · XGBoost · LightGBM · PyTorch · TensorFlow Selection criteria. Start with scikit-learn baselines and compare XGBoost/LightGBM on tabular data. Use PyTorch/TensorFlow when a neural model is justified. Test leakage, calibration and costly errors—not accuracy alone.	1B Generative model APIs OpenAI API · Claude API · Gemini API · Mistral API · Cohere API Selection criteria. Compare the same document, extraction and tool-call cases across APIs. Require valid output schemas, adequate modality support, acceptable response time and cost, and suitable data-retention terms for the chosen model.
Retrieving knowledge	2A Search and evidence storage Azure AI Search · Amazon OpenSearch Service · Elasticsearch · Pinecone · Weaviate Selection criteria. Test exact equipment IDs alongside semantic queries. Require revision/date filters and document access controls before evidence reaches the model. Compare indexing freshness, ranking quality and the burden of operating the search service.	2B RAG and knowledge apps LlamaIndex · Haystack · LangChain · Dify · Microsoft GraphRAG Selection criteria. Use conventional RAG for passage-level facts; consider GraphRAG for relationships and collection-wide synthesis. Compare coded pipelines with visual assembly. Check parsing, source provenance and whether added indexing work improves the required answers.
Taking action	3A Agent construction and tool control OpenAI Agents SDK · Claude Agent SDK · Strands Agents · Google ADK · Microsoft Agent Framework Selection criteria. Test approval before writes, saved-state resume, tool-argument validation and failure handling. Compare built-in execution tools with custom tools. Choose an SDK whose process, sandbox and model-provider requirements fit the application.	3B Business integration and automation n8n · Power Automate · Make · Zapier · Pipedream Connect Selection criteria. Verify connector operations, authentication scopes and trigger delivery. Test retries and duplicate events before writes. Compare visual configuration with coded API steps. Use Pipedream Connect for developer integrations; Pipedream Workflows retires in March 2027.
Coordinating work	4A Stateful execution and recovery LangGraph · Temporal · AWS Step Functions · Azure Durable Functions · Prefect Selection criteria. LangGraph controls mixed model/code paths; Temporal supports durable business processes; AWS/Azure runtimes fit their cloud estates. Test recovery during an approval wait and after a write. Repeated delivery must not repeat the business action.	4B Managed AI application platforms Amazon Bedrock · Microsoft Foundry · Gemini Enterprise Agent Platform · IBM watsonx.ai · Databricks AI platform Selection criteria. Compare model availability, identity/data integration, deployment boundaries and required feature maturity. Count custom integration work and operating responsibility. Existing cloud alignment helps only if the required services and controls are actually available.
Evaluating results	5A Tracing and evaluation platforms LangSmith · Langfuse · Arize Phoenix · Braintrust · W&B Weave Selection criteria. Require one trace linking retrieved sources, model calls, tool results and the final outcome. Compare feedback-to-dataset workflows, version comparisons, sensitive-data handling and export. Trace visibility must support diagnosing a specific failed case.	5B Executable evaluation and failure tests Promptfoo · DeepEval · Ragas · Giskard · TruLens Selection criteria. Match the test to the failure: Ragas for retrieval/answer metrics, Promptfoo for output assertions, DeepEval for agent trajectories. Require known failure cases and executable outcome checks; calibrate model judges against expert decisions.

Note. Managed AI platforms (4B) span generation, retrieval, action, and evaluation as well as coordination. Primary placement is a navigation aid, not exclusive membership. The ten categories preserve all 50 product entries and their full original selection criteria. Product information retains the October 1, 2026 research cutoff; the companion source register supplies the official links. No performance ranking is implied.

Figure 2
Using Figure 1 and Table 1 to select components

Applying the AI taxonomy



Compare candidates against the same acceptance criteria.
Change scope through an explicit business decision.

Note. Original LMM Labs selection guide. These are component-selection checks, not proof that a complete application is ready for use. A failed candidate returns to the shortlist; a scope change requires an explicit business decision.

Conclusion

The taxonomy is useful when it narrows a real decision. It should leave a team with a component shortlist, reasons for each choice, and specific questions to resolve in design. Figure 2 provides that path without requiring every technology in the map. Human goals, architecture, and review remain central to LMM Labs' development methodology (LMM Labs, n.d.). Carry the selection record into the next article's system blueprint, then compare implementation paths in Article 3.

Research and preparation note

Technical explanatory synthesis with an original editorial taxonomy and selection flowchart. Research cutoff and product information: October 1, 2026. Editorial revision: October 17, 2026. Sources and product roles remain documented in the companion register. The flowchart is an editorial guide, not a validated selection model. No comparative experiment, new performance measurement, or market-share ranking is reported. Prepared in collaboration with ChatGPT; Lee Farretta is executive editor. © 2026 LMM Labs. Third-party rights are retained.

References

- A2A Project. (n.d.). *Core concepts*. <https://a2a-protocol.org/latest/topics/key-concepts/>
- Anthropic. (2024, December 19). *Building effective agents*. <https://www.anthropic.com/engineering/building-effective-agents>
- Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2025). *From local to global: A GraphRAG approach to query-focused summarization* [Preprint]. arXiv. <https://arxiv.org/abs/2404.16130v2>
- Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In N. Aletras & O. De Clercq (Eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (pp. 150–158). Association for Computational Linguistics. <https://aclanthology.org/2024.eacl-demo.16/>
- Farretta, L., & Farretta, M. (2026, September 30). *Making sense of AI and the acceleration opportunity*. LMM Labs. <https://lmmllabs.com/articles/making-sense-of-ai-and-the-acceleration-opportunity/>
- LangChain. (n.d.-a). *LangGraph overview*. <https://docs.langchain.com/oss/python/langgraph/overview>
- LangChain. (n.d.-b). *LangSmith observability*. <https://docs.langchain.com/langsmith/observability>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474. <https://papers.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html>
- LMM Labs. (n.d.). *Development approach—Methodology*. <https://lmmllabs.com/development-methodology/>
- Microsoft. (2026, August 31). *Hybrid search using vectors and full-text search in Azure AI Search*. <https://learn.microsoft.com/en-us/azure/search/hybrid-search-overview>
- Model Context Protocol. (n.d.). *Architecture overview*. <https://modelcontextprotocol.io/docs/2026-07-28/learn/architecture>
- Pipedream. (n.d.). *Workflows and String are shutting down on March 31, 2027*. <https://pipedream.com/docs/workflows>
- scikit-learn developers. (n.d.). *User guide*. https://scikit-learn.org/stable/user_guide.html
- Temporal Technologies. (n.d.). *Temporal Workflow Execution overview*. <https://docs.temporal.io/workflow-execution>
- Yao, S., Shinn, N., Razavi, P., & Narasimhan, K. (2024). *τ-bench: A benchmark for tool-agent-user interaction in real-world domains* [Preprint]. arXiv. <https://arxiv.org/abs/2406.12045v1>
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*. <https://arxiv.org/abs/2210.03629v3>