SAMPLE

Building the Agentic AI of Tomorrow

A Practical Guide to Designing, Building, and Operating Autonomous AI Systems

Aiman JoeHanz

Free sample: Introduction & Chapter 1

putrabooks.github.io

Buy the full book:

Carousell carousell.sg/p/building-the-agentic-ai-of-tomorrow-ebook-pdf-1465657883

Gumroad

Ko-fi

ElevenReader

Payhip

Contents

Introduction

Chapter 1: The Current State of Agentic AI

Chapter 2: Where Today's Agentic AI Practices Fall Short

Chapter 3: Common Mistakes in Agentic AI Programs

Chapter 4: The Bedrock Principles of Agentic AI

Chapter 5: Core Agentic AI Architectures and Models

Chapter 6: Implementation Strategies for Production Agentic AI Systems

Chapter 7: Technology Deep Dive: From Tool Use to Multi-Agent Orchestration

Chapter 8: Case Studies: Agentic AI in the Field

Chapter 9: Scaling and Future-Proofing Agentic AI Programs

Chapter 10: The Future of Agentic AI

Chapter 11: An Action Plan for Building Agentic AI Capability

Chapter 12: Advanced Ground: Governance, Compliance, and Emerging Risks

Chapter 13: Trust, Ethics, and the Responsible Agentic AI Journey

Frequently Asked Questions

Glossary

This free sample contains the Introduction and Chapter 1. The complete book contains the full contents above.

Introduction

Agentic AI, artificial intelligence systems that can plan, make decisions, and take actions with varying degrees of autonomy, represents a fundamental shift from the conversational AI that most organizations have deployed. Where a chatbot answers questions, an agent solves problems. Where a chatbot waits for each instruction, an agent decomposes a goal into steps, selects tools, executes actions, observes results, and adapts its plan based on what it finds. This shift from response to action, from single-turn to multi-step, from passive to proactive, is what makes agentic AI both transformative and challenging.

The excitement around agentic AI is justified by genuine capability advances.

Large language models can now reason about multi-step problems, use external tools through function calling, maintain state across interactions, and collaborate with other agents. Frameworks for building agents, from open-source libraries to commercial platforms, have matured significantly. Real organizations are deploying agents for customer support automation, software development assistance, research and analysis, data processing pipelines, and operational tasks that previously required human judgment.

The challenge is that agentic AI is genuinely harder than conversational AI. Agents make decisions, and wrong decisions have consequences. Agents take actions, and actions can be irreversible. Agents operate over multiple steps, and errors compound across steps. Agents use tools, and tool misuse can cause real damage. The patterns that work for chatbot deployment, where the worst case is a bad response, do not work for agent deployment, where the worst case is an action that cannot be undone.

This book is for the engineer building agentic systems, the technical leader responsible for agent deployment, and the executive evaluating whether and how to adopt agentic AI. It assumes familiarity with large language models and basic AI application development but does not require prior experience building agents.

Across thirteen chapters, you will learn the current state of agentic AI practice, the specific failure modes that agent deployments encounter, the architectural patterns for building reliable agents, the implementation practices that ensure agents operate safely and effectively, the common mistakes that undermine agent projects, and the governance practices that agentic AI requires. We will examine real deployment case studies, walk through building agentic capability from first principles, and close with the trust, ethics, and responsibility considerations that autonomous AI systems uniquely raise.

Each chapter includes a PRACTICAL INSIGHT callout distilling the single most actionable lesson, and a KEY TAKEAWAYS summary for quick reference. This book does not promise that building agents is easy; agentic AI is a frontier technology with genuine complexity and risk. It promises a clear-eyed, practitioner-grounded understanding of what works, what fails, and how to build agentic systems that deliver value while operating safely within appropriate boundaries.

By the final page, you should be able to evaluate which use cases are appropriate for agentic AI, design agent architectures that match the requirements of each use case, implement safety and control mechanisms that prevent harmful actions, and deploy agents that deliver sustained value without requiring constant human intervention. Let's begin with where agentic

AI practice actually stands today.

Chapter 1: The Current State of Agentic AI

Agentic AI has moved from research demonstrations to production deployments, but the gap between what is theoretically possible and what is reliably achievable in production remains significant. Understanding this current state is essential for any organization considering agentic AI adoption.

The distinction between conversational AI and agentic AI is not a binary but a spectrum. At one end, a simple chatbot that answers questions has no agency: it responds to input without planning, decision-making, or action. At the other end, a fully autonomous agent that receives a goal, decomposes it into steps, selects and uses tools, observes results, and adapts its plan has significant agency. Most practical deployments fall somewhere in between, with agents that have bounded autonomy: they can plan and act within defined constraints, with human oversight at key decision points.

The key capabilities that make AI agentic are planning, the ability to decompose a goal into actionable steps; tool use, the ability to select and invoke external tools; observation, the ability to interpret the results of actions; adaptation, the ability to revise plans based on observations; and memory, the ability to maintain state across steps and sessions. An AI system that has some but not all of these capabilities has partial agency, and the degree of agency determines the complexity of implementation and the level of risk.

The evolution from conversational AI to agentic AI has been driven by several capability advances. Function calling, which allows models to invoke external tools, gave AI the ability to take actions beyond generating text. Extended context windows, which allow models to process large amounts of information, gave AI the ability to maintain state across multi-step workflows. Improved reasoning, which allows models to decompose complex problems, gave AI the ability to plan multi-step approaches. Together, these advances transformed AI from a response generator to a potential action taker.

This evolution is not complete. The reasoning capabilities that enable planning are still imperfect, leading to plans that fail or produce unexpected results. Tool use is still error-prone, leading to actions that do not match the intended outcome. State management is still fragile, leading to agents that lose context or become confused over long workflows. These limitations, discussed in detail in Chapter 2, are what make agentic AI challenging despite the genuine advances that have been achieved.

Several architectural patterns for agentic AI are in common use. Single-agent architectures, where one agent handles the entire task, are the simplest and

most common. The agent receives a goal, plans its approach, uses tools as needed, and produces a result. Multi-agent architectures, where multiple agents collaborate or specialize, are more complex but can handle more sophisticated tasks by dividing work among agents with different capabilities.

Within these broad patterns, specific approaches vary. ReAct (Reasoning and Acting), where the agent alternates between reasoning about its next step and taking that step, is a common pattern for single agents. Plan-and-execute, where the agent plans the entire approach upfront and then executes the plan, is an alternative that provides more structured execution. Reflection, where the agent evaluates its own outputs and revises them, adds self-correction to either pattern. The choice of architecture depends on the task complexity, the need for adaptability, and the acceptable level of risk.

A growing ecosystem of tools and frameworks supports agentic AI development. Open-source frameworks like LangChain, LangGraph, and CrewAI provide building blocks for agent construction. Commercial platforms offer managed agent infrastructure with built-in safety, monitoring, and deployment capabilities. Model providers increasingly offer agent-specific features, from function calling to code execution environments.

This ecosystem makes agent development more accessible but also requires careful evaluation. Frameworks that work well for prototyping may not be suitable for production, and the quality of results varies significantly across tools and use cases. Organizations should evaluate agent frameworks based on their maturity, community support, production readiness, and alignment with their specific requirements.

Enterprise adoption of agentic AI follows a maturity progression. Organizations typically begin with internal, low-risk use cases like research assistance, data analysis, or content generation, where errors are caught before they have external impact. They then progress to semi-automated use cases where the agent proposes actions that a human reviews and approves. Finally, they adopt fully autonomous agents for well-bounded, well-tested use cases where the risk of autonomous action is acceptable.

This progression reflects appropriate caution: agents that act without human review should only be deployed after extensive testing and with robust safety mechanisms. Organizations that skip this progression, deploying autonomous agents without first validating their reliability in supervised settings, consistently encounter failures that could have been caught through progressive deployment.

Current agentic AI capabilities are real but bounded. Agents can successfully

handle tasks that have clear goals, well-defined tools, and bounded scope. They struggle with tasks that require long-range planning, ambiguous goals, complex tool interactions, or understanding of context that goes beyond what the model was trained on. Understanding these capabilities and limitations is essential for selecting appropriate use cases and setting realistic expectations.

The most capable current agents excel at tasks like information retrieval and synthesis, where the goal is clear and the tools are well-defined; code generation and modification, where the agent can test its output; and structured data processing, where the task can be decomposed into well-defined steps. They struggle with tasks that require creative problem-solving, nuanced judgment, or understanding of complex real-world constraints that are not explicitly represented in the agent's context.

The economic case for agentic AI is based on automation of tasks that previously required human judgment. Where conversational AI can automate information retrieval and response generation, agentic AI can automate multi-step processes that require planning, decision-making, and tool use. This deeper automation can deliver greater cost savings and efficiency gains than conversational AI, but it also carries greater risk, because the actions that agents take can have real consequences.

Organizations evaluating agentic AI should assess both the potential savings and the potential risk. Tasks that are expensive for humans to perform, that have clear goals and well-defined tools, and where errors are detectable and correctable, are good candidates for agentic automation. Tasks where errors are costly or difficult to detect, where goals are ambiguous, or where the tools are unreliable, are poor candidates regardless of the potential savings.

A common misconception about agentic AI is that agents must be fully autonomous to be useful. In practice, the most successful deployments use partial autonomy, where the agent handles routine steps autonomously and defers to humans for complex or high-stakes decisions. Full autonomy, where the agent handles everything without human input, is appropriate for only a narrow range of well-bounded tasks, and it is not a prerequisite for agentic AI value.

Organizations should evaluate the appropriate level of autonomy for each use case rather than aiming for full autonomy as a default. Partial autonomy can deliver most of the value of agentic AI while maintaining the safety of human oversight for consequential decisions.

Foundation models, large language models trained on broad data, are the reasoning engine that powers agentic AI. The quality of the foundation model

directly affects the agent's planning, tool use, and reasoning capabilities. As foundation models improve, agents built on them become more capable without changes to the agent architecture.

However, foundation models are not agent-specific: they are general-purpose models that have been adapted for agentic use through prompting and framework design. This means that agent capability depends not just on the model but on how the model is used, and a well-designed agent on a less capable model can outperform a poorly designed agent on a more capable model.

It is important to distinguish agentic AI from traditional workflow automation. Traditional workflows follow predefined rules: if this condition, then that action. Agents make dynamic decisions: based on what they observe, they choose what to do next. This distinction matters because the two approaches have different capabilities, different risks, and different appropriate use cases.

Workflows are appropriate for tasks with well-defined, stable processes where the steps and conditions are known in advance. Agents are appropriate for tasks that require judgment, adaptation, and flexibility, where the steps and conditions cannot be fully determined in advance. Many practical systems combine both: workflows for the predictable parts and agents for the parts that require judgment.

Reasoning, the model's ability to think through problems step by step, is the core capability that enables agentic AI. Without reasoning, the model could not plan multi-step approaches, decide which tools to use, or interpret complex tool outputs. The quality of the model's reasoning directly determines the quality of the agent's behavior.

Current model reasoning is imperfect, which is why agents make planning errors, misinterpret tool outputs, and fail to recognize when they are on the wrong track. Understanding the limitations of model reasoning is essential for setting appropriate expectations for agent performance and for designing safety mechanisms that account for reasoning errors.

Agent failures have economic impacts beyond the wasted computation: failed agent actions may require human remediation, may cause damage that needs to be repaired, and may erode user trust that reduces the value of the agent system. Understanding the full economic impact of failure is essential for making informed decisions about agent autonomy and safety investment.

The economic impact of failure should be factored into the agent's business case, alongside the savings from automation. An agent that saves 80 percent of task cost but has a 10 percent failure rate with costly remediation may

deliver less net value than an agent that saves 50 percent but has a 1 percent failure rate, and the full economic picture should inform agent design and deployment decisions.

Agent design directly affects user experience: the agent's speed, accuracy, and communication style all contribute to the user's perception of the system. An agent that is fast but inaccurate creates a frustrating experience, while an agent that is accurate but slow may not meet the user's expectations for responsiveness.

User experience should be a design consideration from the start, not an afterthought. The agent's interaction design, including how it communicates its actions and results, how it handles errors, and how it responds to user feedback, should be designed with the user's experience in mind, not just the agent's task completion.

Benchmarks — uniform task suites for comparing different agent approaches — play an important role in agent evaluation. They provide a way to compare different architectures, models, and configurations on standardized tasks, enabling organizations to make informed decisions about which approaches work best for their use cases.

However, benchmarks have limitations: they may not cover the full range of real-world conditions, they may not capture all quality dimensions, and they may not reflect the specific requirements of each use case. Organizations should use benchmarks as one input to agent evaluation, alongside use-case-specific testing that captures the requirements that generic benchmarks may not cover.

Simulation, testing agents in simulated environments that approximate real-world conditions, is a valuable testing approach for agents that interact with external systems. Simulation allows testing of agent behavior in controlled environments where failures do not have real-world consequences, and it allows testing of scenarios that would be difficult or dangerous to test in production.

Simulation is particularly valuable for agents that interact with expensive or dangerous systems, where testing in production is too costly or risky. However, simulation environments may not perfectly reflect real-world conditions, and agents that perform well in simulation may still fail in production due to differences between the simulation and the real environment.

User feedback — explicit (ratings, comments) or implicit (usage patterns, acceptance rates) — is among the most valuable sources of information for agent improvement. Users encounter situations that testing may not cover,

and their feedback reveals failure modes and improvement opportunities that internal analysis may miss.

Organizations should implement user feedback mechanisms for all agent deployments, making it easy for users to report issues, rate agent performance, and suggest improvements. This feedback should be systematically analyzed and used to inform agent improvements, closing the loop between user experience and agent development.

Agent performance depends significantly on domain knowledge: the model's understanding of the specific domain in which the agent operates. A general-purpose model may perform adequately on common tasks but poorly on specialized tasks that require domain-specific knowledge. Domain knowledge can be provided through system prompts, tool outputs, retrieval from knowledge bases, or fine-tuning.

Organizations should assess the domain knowledge requirements of their agent use cases and ensure that the agent has access to the knowledge it needs. For domains where the model's built-in knowledge is insufficient, retrieval-augmented generation or domain-specific tools can provide the missing knowledge.

PRACTICAL INSIGHT Before building an agent, clearly define what the agent is allowed to do and what it is not allowed to do. This definition of boundaries, sometimes called a "action perimeter," is the single most important design decision for agentic AI. Agents without clear boundaries will inevitably attempt actions that are inappropriate or dangerous, and the boundaries should be defined before development begins, not discovered during deployment.

KEY TAKEAWAYS - Agentic AI exists on a spectrum from conversational to fully autonomous, with most practical deployments using bounded autonomy - The evolution from chatbots to agents was driven by function calling, extended context, and improved reasoning - Common architectures include single-agent, multi-agent, ReAct, plan-and-execute, and reflection patterns - A growing tool ecosystem makes agent development accessible but requires production-readiness evaluation - Enterprise adoption follows a maturity progression from supervised to semi-automated to autonomous - Current agents excel at bounded tasks with clear goals and well-defined tools but struggle with ambiguity

The agentic AI technology stack has matured rapidly, with layers of infrastructure now available for building production agents. The stack includes: the model layer (the LLM that provides reasoning, planning, and tool use — models like GPT-4, Claude, and Gemini are being specifically trained and

optimized for agentic tasks, with better function calling, longer context windows, and improved reasoning), the framework layer (the orchestration framework that manages the agent's lifecycle — LangGraph, CrewAI, OpenAI Assistants API — providing planning, tool execution, memory, and human-in-the-loop), the tool layer (the tools the agent uses — APIs, databases, code execution environments — and the infrastructure for registering, executing, and monitoring them), and the observability layer (the monitoring and evaluation infrastructure — LangSmith, Langfuse, custom dashboards — that provides visibility into agent behavior and performance).

The technology stack is evolving rapidly in all layers. The model layer is seeing new models specifically designed for agentic tasks — with extended thinking, better tool use, and longer context. The framework layer is consolidating around a few production-ready options — LangGraph for complex multi-agent systems, CrewAI for simpler single-agent deployments, OpenAI Assistants for managed deployments. The tool layer is seeing the emergence of tool marketplaces — pre-built tools that can be plugged into any agent — reducing the effort to equip agents with capabilities. The observability layer is maturing — with standard metrics, evaluation frameworks, and production monitoring becoming available. The technology stack maturity means that building production agentic AI is becoming faster and more reliable — but the rapid evolution means the team must continuously evaluate new tools and frameworks.

End of free sample. The full book is available at putrabooks.github.io

ABOUT THE AUTHOR

Aiman JoeHanz writes about the technologies that are quietly reshaping the world. His Building of Tomorrow series examines the infrastructure behind artificial intelligence -- from blockchain and cloud computing to agentic systems and token optimization -- and asks how these technologies will evolve over the coming decade.

His books are built on technical research and engineering documentation, written for readers who want to understand systems rather than trends. He covers what is being built today and what it will take to build what comes next.

He believes that understanding infrastructure is the difference between watching the future happen and helping to create it.