Preface

Welcome to "Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems." As we look across the landscape of modern artificial intelligence, we see a clear evolution from simple, reactive programs to sophisticated, autonomous entities capable of understanding context, making decisions, and interacting dynamically with their environment and other systems. These are the intelligent agents and the agentic systems they comprise.

The advent of powerful large language models (LLMs) has provided unprecedented capabilities for understanding and generating human-like content such as text and media, serving as the cognitive engine for many of these agents. However, orchestrating these capabilities into systems that can reliably achieve complex goals requires more than just a powerful model. It requires structure, design, and a thoughtful approach to how the agent perceives, plans, acts, and interacts.

Think of building intelligent systems as creating a complex work of art or engineering on a canvas. This canvas isn't a blank visual space, but rather the underlying infrastructure and frameworks that provide the environment and tools for your agents to exist and operate. It's the foundation upon which you'll build your intelligent application, managing state, communication, tool access, and the flow of logic.

Building effectively on this agentic canvas demands more than just throwing components together. It requires understanding proven techniques – **patterns** – that address common challenges in designing and implementing agent behavior. Just as architectural patterns guide the construction of a building, or design patterns structure software, agentic design patterns provide reusable solutions for the recurring problems you'll face when bringing intelligent agents to life on your chosen canvas.

What are Agentic Systems?

At its core, an agentic system is a computational entity designed to perceive its environment (both digital and potentially physical), make informed decisions based on those perceptions and a set of predefined or learned goals, and execute actions to achieve those goals autonomously. Unlike traditional software, which follows rigid, step-by-step instructions, agents exhibit a degree of flexibility and initiative.

Imagine you need a system to manage customer inquiries. A traditional system might follow a fixed script. An agentic system, however, could perceive the nuances of a customer's query, access knowledge bases, interact with other internal systems (like

order management), potentially ask clarifying questions, and proactively resolve the issue, perhaps even anticipating future needs. These agents operate on the canvas of your application's infrastructure, utilizing the services and data available to them.

Agentic systems are often characterized by features like **autonomy**, allowing them to act without constant human oversight; **proactiveness**, initiating actions towards their goals; and **reactiveness**, responding effectively to changes in their environment. They are fundamentally **goal-oriented**, constantly working towards objectives. A critical capability is **tool use**, enabling them to interact with external APIs, databases, or services – effectively reaching out beyond their immediate canvas. They possess **memory**, retain information across interactions, and can engage in **communication** with users, other systems, or even other agents operating on the same or connected canvases.

Effectively realizing these characteristics introduces significant complexity. How does the agent maintain state across multiple steps on its canvas? How does it decide *when* and *how* to use a tool? How is communication between different agents managed? How do you build resilience into the system to handle unexpected outcomes or errors?

Why Patterns Matter in Agent Development

This complexity is precisely why agentic design patterns are indispensable. They are not rigid rules, but rather battle-tested templates or blueprints that offer proven approaches to standard design and implementation challenges in the agentic domain. By recognizing and applying these design patterns, you gain access to solutions that enhance the structure, maintainability, reliability, and efficiency of the agents you build on your canvas.

Using design patterns helps you avoid reinventing fundamental solutions for tasks like managing conversational flow, integrating external capabilities, or coordinating multiple agent actions. They provide a common language and structure that makes your agent's logic clearer and easier for others (and yourself in the future) to understand and maintain. Implementing patterns designed for error handling or state management directly contributes to building more robust and reliable systems. Leveraging these established approaches accelerates your development process, allowing you to focus on the unique aspects of your application rather than the foundational mechanics of agent behavior.

This book extracts 21 key design patterns that represent fundamental building blocks and techniques for constructing sophisticated agents on various technical canvases.

Understanding and applying these patterns will significantly elevate your ability to design and implement intelligent systems effectively.

Overview of the Book and How to Use It

This book, "Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems," is crafted to be a practical and accessible resource. Its primary focus is on clearly explaining each agentic pattern and providing concrete, runnable code examples to demonstrate its implementation. Across 21 dedicated chapters, we will explore a diverse range of design patterns, from foundational concepts like structuring sequential operations (Prompt Chaining) and external interaction (Tool Use) to more advanced topics like collaborative work (Multi-Agent Collaboration) and self-improvement (Self-Correction).

The book is organized chapter by chapter, with each chapter delving into a single agentic pattern. Within each chapter, you will find:

- A detailed **Pattern Overview** providing a clear explanation of the pattern and its role in agentic design.
- A section on **Practical Applications & Use Cases** illustrating real-world scenarios where the pattern is invaluable and the benefits it brings.
- A **Hands-On Code Example** offering practical, runnable code that demonstrates the pattern's implementation using prominent agent development frameworks. This is where you'll see how to apply the pattern within the context of a technical canvas.
- **Key Takeaways** summarizing the most crucial points for quick review.
- **References** for further exploration, providing resources for deeper learning on the pattern and related concepts.

While the chapters are ordered to build concepts progressively, feel free to use the book as a reference, jumping to chapters that address specific challenges you face in your own agent development projects. The appendices provide a comprehensive look at advanced prompting techniques, principles for applying AI agents in real-world environments, and an overview of essential agentic frameworks. To complement this, practical online-only tutorials are included, offering step-by-step guidance on building agents with specific platforms like AgentSpace and for the command-line interface. The emphasis throughout is on practical application; we strongly encourage you to run the code examples, experiment with them, and adapt them to build your own intelligent systems on your chosen canvas.

A great question I hear is, 'With AI changing so fast, why write a book that could be quickly outdated?' My motivation was actually the opposite. It's precisely because things are moving so quickly that we need to step back and identify the underlying principles that are solidifying. Patterns like RAG, Reflection, Routing, Memory and the others I discuss, are becoming fundamental building blocks. This book is an invitation to reflect on these core ideas, which provide the foundation we need to build upon. Humans need these reflection moments on foundation patterns

Introduction to the Frameworks Used

To provide a tangible "canvas" for our code examples (see also Appendix), we will primarily utilize three prominent agent development frameworks. **LangChain**, along with its stateful extension **LangGraph**, provides a flexible way to chain together language models and other components, offering a robust canvas for building complex sequences and graphs of operations. **Crew AI** provides a structured framework specifically designed for orchestrating multiple AI agents, roles, and tasks, acting as a canvas particularly well-suited for collaborative agent systems. The **Google Agent Developer Kit (Google ADK)** offers tools and components for building, evaluating, and deploying agents, providing another valuable canvas, often integrated with Google's AI infrastructure.

These frameworks represent different facets of the agent development canvas, each with its strengths. By showing examples across these tools, you will gain a broader understanding of how the patterns can be applied regardless of the specific technical environment you choose for your agentic systems. The examples are designed to clearly illustrate the pattern's core logic and its implementation on the framework's canvas, focusing on clarity and practicality.

By the end of this book, you will not only understand the fundamental concepts behind 21 essential agentic patterns but also possess the practical knowledge and code examples to apply them effectively, enabling you to build more intelligent, capable, and autonomous systems on your chosen development canvas. Let's begin this hands-on journey!