MULTI-AGENT AI SYSTEMS IN C#

BUILDING AUTONOMOUS AGENTS WITH HARNESS, HERMES, AND LOOP ENGINEERING

HARNESS
BATTERIES-INCLUDED RUNTIME

HERMES
OPEN-SOURCE ORCHESTRATION

LOOP ENGINEERING
SELF-PROMPTING AUTONOMOUS WORKFLOWS

KF WONG

Multi-Agent AI Systems in C#

Building Autonomous Agents with Harness, Hermes, and Loop Engineering

Steve T. Publications

This book is available at https://leanpub.com/multi-agentaisystemsinc

This version was published on 2026-07-13

Contents

Building Autonomous Agents with Harness, Hermes, and Loop Engineering

About this book: This book teaches you how to design, build, and deploy production-grade multi-agent AI systems using C# and the .NET platform. Through three complementary approaches – the Microsoft Agent Framework's batteries-included Harness runtime, Hermes Agent's open-source orchestration layer, and the Loop Engineering paradigm for self-prompting autonomous workflows – you will learn to create intelligent agent systems that reason, collaborate, and operate independently. Every chapter includes complete, production-ready code examples, architectural patterns, and hands-on projects designed to take you from your first agent to a fully deployed multi-agent application.

Introduction: The Agent Revolution

In early 2026, the landscape of artificial intelligence shifted in a way that few predicted. It was not the release of a smarter model or a breakthrough in training methodology. It was something subtler and more profound: agents stopped being toys and started becoming infrastructure.

A year earlier, most developers who worked with large language models built chatbots. You typed a prompt, the model generated a response, and you moved on to the next prompt. The interaction was fundamentally reactive – the model waited for you to ask. By mid-2026, that pattern had evolved into something entirely different. Agents began observing their environment, reasoning about what to do next, taking actions through tools, and repeating this cycle autonomously until a goal was achieved. They did not wait for you to prompt them. They prompted themselves.

This transition from reactive chatbots to autonomous agents represents one of the most significant shifts in software development since the move from monolithic applications to microservices. And just as microservices required new patterns for service discovery, communication, and orchestration, autonomous agents demand a fundamentally different approach to system design.

Why Multi-Agent?

A single agent, however capable, has inherent limitations. It has a finite context window, a single perspective, and a sequential mode of operation. When you face a complex problem – analyzing a codebase for security vulnerabilities, coordinating a customer support operation across multiple time zones, or building a software feature from specification to deployment – no single agent can efficiently handle all dimensions of the task simultaneously.

Multi-agent systems solve this by decomposing complex problems into specialized roles. One agent researches while another analyzes. One writes code while another reviews it. One monitors for anomalies while another

generates reports. These agents communicate, coordinate, and collaborate to produce results that no single agent could achieve alone.

The analogy to human organizations is deliberate. A company does not hire one person to do everything. It creates specialized roles – engineers, designers, product managers, testers – and establishes communication channels and processes that allow them to work together effectively. Multi-agent systems apply the same principle to AI: specialization, coordination, and collaboration.

The Three Pillars

This book is organized around three complementary approaches that, together, form a complete stack for building multi-agent AI systems in C#.

Harness refers to the runtime scaffolding that turns a language model into an agent capable of real action. A model on its own generates text. A harness wraps that model with tool calling, context management, memory, planning, approval gates, and observability. The Microsoft Agent Framework provides a batteries-included HarnessAgent that gives you all of these capabilities out of the box, tuned for long-running autonomous work in C# and .NET.

Hermes represents the orchestration layer that coordinates multiple agents working together. Built by Nous Research as an open-source project, Hermes Agent provides multi-agent coordination through patterns like the Kanban board – a shared state mechanism where tasks are represented as cards, agents claim and process them in parallel, and results flow back to an orchestrator. Hermes is Python-first but integrates seamlessly with C# applications through its API server, which exposes OpenAI-compatible endpoints and REST APIs.

Loop Engineering is the design paradigm that makes agents truly autonomous. Coined by industry leaders like Addy Osmani and Boris Cherny in 2026, loop engineering describes the practice of designing systems that prompt and supervise AI agents automatically, rather than requiring a human to type every prompt. A loop consists of automations that discover work on a schedule, skills that codify project knowledge, connectors that integrate with external tools, sub-agents that split maker and checker roles, and state management that persists across runs.

These three pillars are not competing approaches. They address different layers of the same problem. The harness gives an individual agent its capabilities. Hermes provides patterns for coordinating multiple agents. Loop

engineering makes the entire system self-sustaining. Together, they form a complete architecture for production-grade multi-agent AI applications.

Who This Book Is For

This book is written for C# developers who want to build multi-agent AI systems. You should be comfortable with C#, async/await, and the .NET ecosystem. You do not need prior experience with AI or machine learning, though familiarity with concepts like embeddings, vector search, or prompt engineering will help you move through the early chapters more quickly.

If you are a beginner, start from Chapter 1 and work through every chapter in order. Each chapter builds on the previous one, and the code examples are designed to be cumulative – you will extend the same project throughout the book, adding capabilities as you learn them.

If you are an experienced developer who already works with AI agents, use this book as a reference. Jump to the chapters that address your specific needs: Chapter 4 for Harness deep dive, Chapter 7 for Hermes integration, Chapter 9 for loop engineering patterns, or Chapter 12 for durable workflows and human-in-the-loop design.

What You Will Build

Throughout this book, you will build a progressively more sophisticated multi-agent application. It starts as a simple chat agent in Chapter 2 and evolves into a complete customer support platform by Chapter 17, featuring:

- Multiple specialized agents (triage, research, resolution, escalation)
- Tool integration via the Model Context Protocol
- Retrieval-Augmented Generation over a knowledge base
- Durable workflows with human approval gates
- Observability through OpenTelemetry tracing
- Deployment to Azure Container Apps

Each chapter contributes a piece of this puzzle. By the end, you will have a production-ready application that demonstrates every concept covered in the book.

How This Book Is Organized

The book is divided into seven parts:

Part I: Foundations (Chapters 1-3) establishes the theoretical and practical groundwork. You will learn what multi-agent systems are, set up your development environment, and understand the core concepts that underpin everything that follows.

Part II: Harness Deep Dive (Chapters 4-6) covers the Microsoft Agent Framework's HarnessAgent in detail. You will learn to create agents with tool calling, memory, context compaction, RAG, and structured outputs.

Part III: Hermes Deep Dive (Chapters 7-8) introduces Hermes Agent and its multi-agent orchestration capabilities. You will learn to coordinate multiple specialized agents using the Kanban board pattern and integrate Hermes from C#.

Part IV: Loop Engineering (Chapters 9-10) explains the loop engineering paradigm and shows you how to implement self-prompting agent loops in C# using MAF's looping evaluators and .NET background services.

Part V: Advanced Orchestration (Chapters 11-13) covers the five built-in orchestration patterns in MAF, durable workflows with checkpointing, human-in-the-loop approval gates, and cross-framework communication protocols.

Part VI: Production Engineering (Chapters 14-16) addresses the concerns that separate prototypes from production systems: security, testing, observability, deployment, and scaling.

Part VII: Capstone Project (Chapter 17) brings everything together in a comprehensive multi-agent customer support platform.

A Note on Code Examples

All code examples in this book are written for C# and the .NET platform, targeting .NET 8 or later. The Microsoft Agent Framework packages used are based on version 1.x, released as generally available in April 2026. Where package APIs differ between preview and stable releases, I note the differences explicitly.

Code examples assume you have access to a large language model API, such as OpenAI, Azure OpenAI, or a local model through Ollama. Environment variables are used for API keys throughout the examples, and I never hardcode credentials in source code.

Every code listing is designed to be complete and runnable. If you copy the code from a listing into your project, it should compile and run with minimal modification (typically just adding your own API key). Where a listing omits boilerplate for brevity, I note what was omitted and where to find the complete version in the accompanying sample repository.

The Road Ahead

The field of multi-agent AI is evolving rapidly. New frameworks, protocols, and best practices emerge monthly. This book captures the state of the art as of mid-2026, but the underlying principles – specialization, coordination, autonomy, observability – will remain relevant regardless of which tools are in fashion.

My goal is not just to teach you how to use specific libraries, but to give you a deep understanding of *why* multi-agent systems work the way they do, what trade-offs you face at each architectural decision point, and how to evaluate new tools and patterns as they emerge.

Let us begin.

Chapter 1: The Age of Multi-Agent Systems

What Are AI Agents and Why Multi-Agent?

An AI agent is a system that uses a large language model as its reasoning engine to perceive its environment, make decisions, take actions through tools, and pursue goals autonomously. Unlike a traditional chatbot that generates text in response to a prompt, an agent operates in a loop: it observes, reasons, acts, evaluates the result, and repeats until its task is complete.

This definition matters because it distinguishes agents from the simpler AI applications that preceded them. A chat completion API call takes a prompt and returns text. Retrieval-Augmented Generation (RAG) adds a knowledge retrieval step before generation. Tool calling lets the model request actions from external systems. All of these are still fundamentally reactive: they respond to a single input with a single output.

An agent changes the dynamic entirely. It initiates its own actions. It plans multi-step strategies. It remembers what it has done and adapts when things go wrong. It can hand work off to other agents and wait for their results. The transition from reactive to autonomous is not incremental; it is architectural.

The case for *multi-agent* systems follows naturally from the limitations of single agents. Consider a software development task: analyzing a pull request for bugs, security vulnerabilities, and style violations. A single agent could attempt all three analyses sequentially, but it would face several problems:

- **Context window limits**: Loading an entire codebase plus analysis instructions for three different review types may exceed the model's context window.
- **Cognitive overload**: Asking one agent to be simultaneously a security expert, a performance reviewer, and a style checker dilutes its effectiveness in each role.

- **Sequential bottlenecks**: Running three analyses one after another takes three times as long as running them in parallel.
- **Confirmation bias**: The same model that wrote code is often too forgiving when reviewing it.

A multi-agent system addresses each concern. Three specialized agents – a security reviewer, a performance analyzer, and a style checker – can run in parallel, each with its own focused context and expert instructions. Their results are aggregated by an orchestrator that produces a unified review. The total time is determined by the slowest agent, not the sum of all three.

This pattern – decompose, specialize, parallelize, aggregate – is the fundamental insight behind multi-agent systems. It mirrors how human organizations solve complex problems, and it turns out to be equally effective for AI.

From Chatbots to Autonomous Systems: The Evolution

The evolution from chatbots to autonomous agents has proceeded through several distinct phases, each building on the capabilities of the previous one.

Phase 1: Conversational Interfaces (2022-2023). The first wave of generative AI applications were conversational interfaces – essentially advanced autocomplete systems that could maintain context across multiple turns. ChatGPT demonstrated the power of this approach in late 2022, and within months every major technology company had released their own chat-based AI product. These systems were impressive but fundamentally limited: they could only generate text, and they required a human to drive every interaction.

Phase 2: Tool-Augmented Models (2023-2024). The second wave added tool calling – the ability for models to request actions from external systems. OpenAI introduced function calling in March 2023, and by mid-2024 most major model providers supported some form of structured tool invocation. This was a critical step because it gave models *agency*: they could do things, not just talk about doing things. A model could look up the weather, query a database, send an email, or run a code analysis tool. But the model still needed a human to initiate each interaction and interpret the results.

Phase 3: Single-Agent Autonomy (2024-2025). The third wave introduced agent loops – systems where a model repeatedly calls tools, observes results, and decides its next action without human intervention. Frameworks like

LangChain, Semantic Kernel, and AutoGen provided the scaffolding for building these loops. A single agent could now plan a multi-step task, execute it tool by tool, and produce a final result. This was where the concept of an "agent" became real rather than theoretical.

Phase 4: Multi-Agent Systems (2025-2026). The current wave is about coordination between multiple agents. Microsoft's Agent Framework (released in preview form in late 2025 and reaching general availability in April 2026) provides a production-ready platform for building multi-agent workflows in .NET and Python. Hermes Agent by Nous Research offers open-source multi-agent orchestration with its Kanban board pattern. The Model Context Protocol (MCP), standardized across the industry, enables agents to discover and use tools from any compatible server. And the A2A (Agent-to-Agent) protocol, backed by a consortium including AWS, Google, Microsoft, and others, provides cross-framework agent communication.

This evolution is not linear; each phase coexists with the ones that preceded it. Many production systems today combine all four phases: conversational interfaces for user interaction, tool-augmented models for specific tasks, single-agent loops for autonomous work, and multi-agent orchestration for complex operations. Understanding where your system fits on this spectrum is the first step in designing it effectively.

Introducing Harness, Hermes, and Loop Engineering

The three approaches covered in this book address different layers of the multi-agent stack. Understanding their relationship is essential before diving into implementation.

Harness: The Agent Runtime

A harness is the scaffolding that surrounds a language model and turns it into an agent capable of real action. The term comes from horse harnesses – the equipment that connects a horse to a carriage, giving it something useful to pull. Similarly, an agent harness connects a model to tools, memory, planning capabilities, and safety controls, giving it something useful to do.

The Microsoft Agent Framework provides `HarnessAgent`, a batteries-included implementation that wraps any `IChatClient` with a complete agentic pipeline. Out of the box, a HarnessAgent includes:

- Automatic tool-calling with configurable iteration limits
- Context-window compaction to prevent overflow during long conversations
- A persistent todo list for multi-step planning
- Plan/execute mode tracking that structures how the agent works
- File-based session memory for notes and artifacts
- Read/write file tools scoped to a working directory
- Approval gates with "don't ask again" standing rules
- Built-in OpenTelemetry observability
- Web search capabilities
- Shell command execution (optional)
- Background sub-agent delegation (optional)

Each of these capabilities can be individually enabled, disabled, or customized. You supply your own chat client and configure only the parts you want to change. Everything else has sensible defaults.

The harness is foundational because agent quality depends heavily on the environment in which the model operates. A strong model with poor tools, weak context, and no controls will produce poor results. A well-designed harness helps the agent act with the right information, the right capabilities, and the right boundaries.

Hermes: The Orchestration Layer

If a harness gives an individual agent its capabilities, Hermes provides the coordination layer that allows multiple agents to work together. Hermes Agent, built by Nous Research and released as open-source in February 2026, is designed as a self-hosted, persistent AI agent that grows with you over time.

Hermes introduced multi-agent orchestration in version 0.6.0 through its Kanban board pattern. In this model, tasks are represented as cards on a shared board. Each card has a title, an assignee (the agent profile that should work on it), and a status. A dispatcher loop claims ready cards and spawns the assigned agent in its own clean workspace. Agents read from and write to the board rather than communicating directly with each other.

The Kanban board pattern has five advantages over naive multi-agent setups:

1. **Durability**: The board survives restarts and crashes. If an agent dies mid-task, the dispatcher reclaims the card and respawns the agent.

2. **Parallelism**: Many agents work simultaneously, coordinated by the shared board.
3. **Event-driven design**: Work flows through the dependency graph without polling loops.
4. **Self-healing**: Dead tasks are automatically reclaimed and respawned.
5. **Auditability**: Every claim, comment, and completion is logged, creating a complete trace of what happened.

Hermes is Python-first, but its API server exposes OpenAI-compatible endpoints (`/v1/chat/completions` and `/v1/responses`) plus REST APIs under `/api/sessions/*`. This makes it straightforward to integrate Hermes from C# applications using standard HTTP clients. A C# application can spawn Hermes agents, monitor their progress through the Kanban board API, and aggregate results, all without writing any Python code.

Loop Engineering: The Autonomy Paradigm

Loop engineering, as a named discipline, emerged in early 2026 through the writings of Addy Osmani, Boris Cherny, and others. It describes the practice of designing systems that prompt and supervise AI agents automatically, rather than requiring a human to type every prompt.

The insight behind loop engineering is simple but profound: the bottleneck in AI-assisted development is not the model's intelligence, it is the human who has to keep prompting it. If you can design a system that generates its own prompts, checks its own work, and iterates until a goal is satisfied, you multiply your leverage dramatically.

A loop consists of five core primitives:

1. **Automations**: Scheduled tasks that discover work and triage it on a cadence. An automation might run every morning, scan for CI failures, read open issues, and write findings to a task board.
2. **Worktrees**: Isolated working directories that allow multiple agents to work in parallel without stepping on each other's files. Git worktrees are the canonical implementation, but any filesystem isolation works.
3. **Skills**: Codified project knowledge stored in structured documents (typically `SKILL.md` files). Skills capture the conventions, build steps, and institutional knowledge that an agent would otherwise have to guess at every session.

4. **Plugins and Connectors**: Integrations with external tools – issue trackers, databases, APIs, messaging platforms. The Model Context Protocol is the standard for connectors, allowing any MCP-compatible tool to be discovered and used by any MCP-compatible agent.
5. **Sub-agents**: Specialized agents that split roles within a loop. The most common pattern is maker/checker separation: one agent writes code, a different agent reviews it against specifications and tests. The model that wrote the code is too lenient grading its own work.

A sixth element ties these together: **state management**. A loop must persist its progress somewhere outside the conversation context, because the model forgets everything between runs. This state can be a markdown file, a Linear board, a database, or any durable storage. The key principle is that the state lives on disk, not in the agent's context window.

The Three-Layer Architecture: Runtime, Orchestration, Autonomy

These three approaches form a coherent stack when combined:

```
+-------------------------------------------------+
|  Layer 3: Loop Engineering (Autonomy)           |
|  Automations, Skills, Connectors, Sub-agents    |
|  State Management, Self-Healing                 |
+-------------------------------------------------+
|  Layer 2: Hermes / Orchestration (Coordination) |
|  Multi-Agent Patterns, Kanban Board,            |
|  Parallel Execution, Result Aggregation         |
+-------------------------------------------------+
|  Layer 1: Harness (Runtime Capabilities)        |
|  Tool Calling, Memory, Compaction,              |
|  Approval Gates, Observability                  |
+-------------------------------------------------+
|  Foundation: Language Model                     |
|  OpenAI, Azure OpenAI, Anthropic, Ollama, etc.  |
+-------------------------------------------------+
```

Layer 1 (the harness) gives each individual agent the capabilities it needs to do useful work. Layer 2 (orchestration) coordinates multiple agents working on a shared goal. Layer 3 (loop engineering) makes the entire system self-sustaining, running autonomously without human intervention.

You do not need all three layers for every application. A simple single-agent chatbot only needs Layer 1. A multi-agent research pipeline benefits from Layers 1 and 2. An autonomous coding assistant that runs overnight and delivers results in the morning needs all three.

The key insight is that these layers are composable. You can start with a harness and add orchestration when you need multiple agents. You can add loop engineering on top when you want autonomy. Each layer adds capability without invalidating the ones below it.

What This Book Will Build

Throughout this book, you will build a multi-agent customer support platform called **SupportHub**. The final system will include:

- A **Triage Agent** that classifies incoming support requests by category, urgency, and complexity
- A **Research Agent** that searches a knowledge base using RAG to find relevant documentation
- A **Resolution Agent** that drafts responses based on research findings and company policies
- An **Escalation Agent** that identifies cases requiring human intervention and routes them appropriately
- A **Human Reviewer** approval gate where support managers can approve or modify automated responses before they are sent

The system will use the Microsoft Agent Framework for agent runtime capabilities, integrate with Hermes-style orchestration patterns for multi-agent coordination, and implement loop engineering principles for autonomous operation. It will be deployed to Azure Container Apps with OpenTelemetry observability and structured outputs throughout.

You will build this system incrementally, adding one capability per chapter. By the end, you will have a production-ready application that demonstrates every concept covered in the book.

Chapter Summary

In this chapter, we explored the evolution from chatbots to autonomous multi-agent systems. We introduced the three pillars of modern agent development: Harness (the runtime), Hermes (orchestration), and Loop Engineering (autonomy). We described how these layers compose into a complete architecture and outlined the SupportHub application you will build throughout the book.

Key Takeaways:

- AI agents operate in loops of observe-reason-act-evaluate, unlike reactive chatbots
- Multi-agent systems decompose complex problems into specialized roles for parallel execution
- The three-layer stack (runtime, orchestration, autonomy) is composable and incremental
- The SupportHub application will demonstrate all concepts through a real-world use case

Review Questions:

1. What distinguishes an AI agent from a traditional chatbot?
2. Why are multi-agent systems more effective than single agents for complex tasks?
3. What are the five core primitives of loop engineering?
4. How do the three layers (harness, orchestration, autonomy) relate to each other?

Further Reading:

- Microsoft Agent Framework documentation: https://learn.microsoft.com/en-us/agent-framework/
- Addy Osmani on Loop Engineering: https://addyosmani.com/blog/loop-engineering/
- Hermes Agent by Nous Research: https://hermes-agent.nousresearch.com/docs/
- The Model Context Protocol specification: https://modelcontextprotocol.io/

Exercises:

1. Identify a complex task in your current work that could benefit from multi-agent decomposition. Sketch how you would break it into specialized agent roles.
2. Research the OWASP Agentic Top 10 security risks. Which ones are most relevant to the SupportHub application?
3. Set up a free account with an LLM provider (OpenAI, Azure OpenAI, or Ollama for local models) and make your first API call using the provider's SDK.

Chapter 2: Development Environment Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Prerequisites and Toolchain Installation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Creating Your First .NET Agent Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Installing Microsoft Agent Framework Packages

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Configuring Model Providers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Your First Running Agent in Five Lines of Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Understanding the Project Structure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Configuring Dependency Injection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Verifying Your Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 3: Theoretical Foundations of Multi-Agent Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Agent Architecture Patterns: Reflex, Deliberative, Hybrid

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Communication Models: Message Passing, Shared Memory, Blackboard

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Coordination Theory and Emergent Behavior

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

The BDI Model: Belief, Desire, Intention in AI Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Why Multi-Agent Systems Outperform Single Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 4: Agent Harness Fundamentals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

What Is an Agent Harness and Why It Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

The HarnessAgent Class and Default Capabilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Creating and Configuring a Harness Agent in C#

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Enabling Context Compaction for Long Conversations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Customizing and Disabling Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

The Plan/Execute Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 5: Tools, MCP, and Function Calling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Function Calling and Tool Invocation in MAF

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building Custom Tools with C# Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Model Context Protocol: Connecting to MCP Servers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building Your Own MCP Server in C#

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Exposing Agents as MCP Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 6: Memory, Context, and State Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

AgentSession and Multi-Turn Conversations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Context Providers and Custom Memory

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Compaction Strategies for Long-Running Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Retrieval-Augmented Generation with Vector Stores

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Structured Outputs and JSON Schema

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 7: Hermes Agent Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

What Is Hermes Agent and Its Design Philosophy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Hermes Architecture: The AIAgent Orchestration Engine

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Multi-Agent Orchestration with the Kanban Board

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Scheduling, Cron, and Autonomous Task Execution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Integrating Hermes from C# via API Server

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 8: Building Multi-Agent Workflows with Hermes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Defining Agent Roles and Specialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Orchestrator Pattern: Decomposing Complex Tasks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Parallel Execution and Resource-Aware Scheduling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Shared Context and Selective Information Sharing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building a Research Pipeline with Multiple Hermes Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 9: The Loop Engineering Paradigm

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

From Prompting to Loop Design: A Paradigm Shift

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

The Five Primitives of Loop Engineering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Automations: The Heartbeat of Autonomous Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Skills and Plugins: Codifying Project Knowledge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

State Management: Memory That Survives the Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 10: Building Self-Prompting Agent Loops in C#

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Implementing Agent Loops with LoopEvaluators

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Goal-Driven Execution: Run Until Done

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Scheduled Automations with Background Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Self-Healing and Error-Recovery Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building a Complete Autonomous Agent Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 11: Multi-Agent Orchestration Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Sequential Orchestration: Step-by-Step Pipelines

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Concurrent Execution: Fan-Out and Fan-In Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Handoff Orchestration: Contextual Agent Delegation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Group Chat: Multi-Agent Dialogue Spaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Magentic Orchestration: Dynamic Sub-Agent Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 12: Durable Workflows and Human-in-the-Loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Making Workflows Durable with Durable Task Scheduler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Checkpointing and Recovery from Failures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Human-in-the-Loop: Approval Gates and Request Ports

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Hosting on Azure Functions for Serverless Scaling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Conditional Routing and Sub-Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Shared State in Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 13: Agent Communication Protocols

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Agent-to-Agent (A2A) Protocol: Cross-Framework Communication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

AG-UI: Real-Time Streaming to Frontends

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building a Multi-Agent Dashboard with AG-UI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Interoperability Between MAF, Hermes, and Other Frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 14: Security, Guardrails, and Governance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

The OWASP Agentic Top 10 Threat Landscape

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Middleware-Based Security Guards

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Content Filtering and Harmful Output Prevention

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Authentication, Authorization, and Agent Identity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Policy Enforcement with the Agent Governance Toolkit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 15: Testing, Debugging, and Observability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Unit Testing Agents and Executors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Integration Testing Workflows End-to-End

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

DevUI: Interactive Agent Testing Without Overhead

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

OpenTelemetry Tracing for Multi-Agent Systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Metrics, Logging, and Performance Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 16: Deployment, Scaling, and Production Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Choosing Your Deployment Target

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Deploying to Azure Container Apps with Aspire

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Auto-Scaling and Performance Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Cost Management: Token Budgets and Rate Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Production Runbooks and Incident Response

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter 17: Building a Complete Multi-Agent Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Architecture Design: The Customer Support Platform

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Building Specialized Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Implementing the Orchestration Workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Adding RAG, MCP Tools, and Human Approval Gates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Deploying and Monitoring the Complete System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Chapter Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Conclusion: The Road Ahead

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Key Takeaways: The Three-Layer Stack

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Where Multi-Agent Systems Are Headed

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Your Roadmap for Continued Learning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

Final Thoughts on Building Responsible AI Agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/multi-agentaisystemsinc.