Foundations of Agentic AI *for* Retail

Third Edition

Designing, Building, and Operating Retail AI Agents, from First Principles to Production

Fatih Nayebi

Foundations of Agentic AI *for* Retail

Designing, Building, and Operating Retail AI Agents, from First Principles to Production

THIRD EDITION

Fatih Nayebi

Gradient Divergence
Montréal, QC, Canada

Sample - Not for Redistribution

Foundations of Agentic AI for Retail
Designing, Building, and Operating Retail AI Agents, from First Principles to Production
Third edition

Copyright © 2025, 2026 Fatih Nayebi. All rights reserved.
No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the publisher, except in the case of brief quotations embodied in critical reviews and certain other noncommercial uses permitted by copyright law.

Published by Gradient Divergence, Montréal, Québec, Canada
For permissions and inquiries, write to contact@gradientdivergence.com.

First edition May 2025, second edition January 2026, third edition October 2026.
Version 3.0 (build 5a8a840).
ISBN 978-1-0694226-5-1 (hardcover)
ISBN 978-1-0694226-4-4 (paperback)

The code in this book is released under the MIT License at
https://github.com/gradient-divergence/agentic-retail-foundations.
Please report errors there as issues.

Models, interfaces, prices, and regulations are described as they stood in October 2026.
Nothing in this book is legal advice.
While every precaution has been taken in the preparation of this book, neither the author nor the publisher assumes any liability for errors or omissions, or for damages resulting from the use of the information contained herein.
Product and company names may be trademarks of their owners and are used here editorially.

Set in Source Serif 4 and Inter, with code in JetBrains Mono, mathematics in Latin Modern Math and the title pages in EB Garamond, using LuaLaTeX.

Sample – Not for Redistribution

Epigraph

"... the question of whether Machines Can Think, a question of which we now know that it is about as relevant as the question of whether Submarines Can Swim."

Edsger W. Dijkstra[1]

"If we're successful in building truly intelligent systems, we'll have the biggest opportunity in human history to make the world better for all of humanity. If we fail to build systems aligned with human values, however, we'll probably have the biggest catastrophe in human history."

Stuart Russell[2]

"The reinforcement learning problem is the AI problem, if you think AI is about an agent. An agent needs to interact with an environment, and learn from its interactions how to improve itself."

Richard S. Sutton[3]

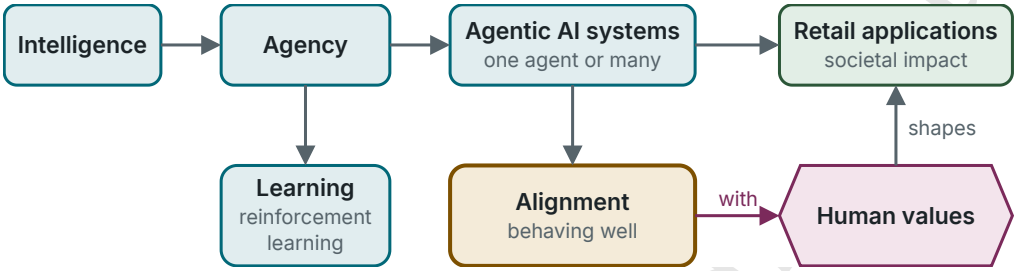
1 From "The threats to computing science" (EWD898), a keynote delivered at the ACM 1984 South Central Regional Conference, Austin, Texas, November 16 to 18, 1984.
2 Stuart Russell is a leading AI researcher and co-author of *Artificial Intelligence: A Modern Approach*.
3 Richard S. Sutton is a reinforcement learning pioneer and co-author of *Reinforcement Learning: An Introduction*.

Sample - Not for Redistribution

These quotes frame a central tension: agency has reach; alignment matters. They point to both the potential and the stakes in building Agentic AI systems that act in the world.

As you read, keep one question in view: **what does it mean for an agent to behave well** in a retail setting, not just be capable? The chapters ahead return to this idea through architectures, policies, evaluation, and human oversight.

An illustration of these ideas appears below.



Carry these ideas into the chapters ahead.

Points to Ponder

- How does Dijkstra's analogy (submarines and swimming) map to specific Agentic AI tasks in retail (e.g., inventory management, customer service bots)?
- In light of Russell's warning, what specific "human values" are most critical to embed in retail AI agents to avoid negative consequences?
- Based on Sutton's quote, what kinds of "interactions" might a retail agent learn from in a physical store versus an online store?

Sample - Not for Reproduction

Table of Contents

Foreword xv

Preface xvii

What's New in the Third Edition xxii

1 Introduction: The Age of Agency 1
- 1.1 What is Agentic AI? 1
- 1.2 Core Technologies and Architectures Enabling Agentic AI 6
- 1.3 Making the Retail Business Case 10
- 1.4 Conclusion 14

Part I: Foundations of Agentic AI 17

2 Agent Architectures and Operating Models 19
- 2.1 Defining the Modern AI Agent 20
- 2.2 RAOM: The Canonical Production Agent Loop 20
- 2.3 BDI Models: Human-Like Decision Making 24
- 2.4 OODA: Agile Decision Cycles for Dynamic Environments 34
- 2.5 LLMs Inside the Loop: Classical Architectures with Modern Patterns 39
- 2.6 Agent Skills and Capability Registry 40
- 2.7 Choosing and Evaluating the Right Architecture 43
- 2.8 Conclusion 44

3 Decision Frameworks I: Probabilistic Reasoning and Optimization 47
- 3.1 Decision-Making Process Overview 48
- 3.2 Optimization Models for Decision-Making 48
- 3.3 Bayesian Decision Theory 54
- 3.4 Causal Reasoning 62
- 3.5 Choosing a Decision Framework 63
- 3.6 Capstone Metrics and Causal Questions 64
- 3.7 Conclusion 65

4 Decision Frameworks II: Sequential Decision-Making 67
- 4.1 Two Canonical Sequential Problems 68
- 4.2 Markov Decision Processes (MDPs) 68
- 4.3 Partially Observable MDPs for Real-World Environments 79
- 4.4 Capstone Mapping: Inventory Risk as a Sequential Model 82
- 4.5 Evaluation and Guardrails 86
- 4.6 Conclusion 87

5 Decision Frameworks III: Learning and Planning 89
- 5.1 Planning First, RL When Justified 90
- 5.2 Reinforcement Learning: Learning Through Interaction 90

Sample Not for Redistribution

5.3 Policy Optimization and Actor-Critic Methods 92
5.4 Planning and Optimization in Decision-Making 98
5.5 Code Example: A Greedy Store Fulfillment Planner 101
5.6 Capstone Integration: Inventory Risk Simulation Harness 108
5.7 Conclusion 110

Part II: Enabling Technologies and Architectures 113

6 Core Technologies Enabling Agentic Retail 115
6.1 Core Technologies for Agentic Retail 116
6.2 LLM Reasoning Patterns and Implementation 120
6.3 Deep Dive: Computer Vision for Physical Store Awareness 131
6.4 Evaluation and Monitoring 138
6.5 Conclusion 138

7 Physical Perception and Sensor Intelligence 141
7.1 Sensor-to-Agent Contract 142
7.2 IoT and Sensor Networks 143
7.3 Knowledge Graphs as Agent Memory 156
7.4 Causal Reasoning and Counterfactual Analysis 165
7.5 Integration with Other Agent Systems 177
7.6 Conclusion 178

Part III: Multi-Agent Coordination and Interoperability 181

8 Multi-Agent Coordination and Interoperability 183
8.1 Applications of Multi-Agent Systems 185
8.2 Coordination, Orchestration, Collaboration, and Delegation 186
8.3 Contract-First Multi-Agent Systems 186
8.4 Operating Multi-Agent Systems in Production 187
8.5 Agent Communication Protocols 187
8.6 Multi-Agent System Architectures 191
8.7 Multi-Agent Collaboration Patterns 192
8.8 Coordination Mechanisms 197
8.9 Collaborative or Competitive Agent Systems 205
8.10 Conclusion 208

9 End-to-End Agent Integration 211
9.1 System Architecture Overview 212
9.2 Core Principles for End-to-End Integration 212
9.3 Events vs APIs vs Queues: A Decision Guide 213
9.4 Schema Governance and Compatibility 213
9.5 State Correctness 214
9.6 Agent Workflow Management 215
9.7 Event-Driven Architectures 218
9.8 API-Based Communication Between Agents 224
9.9 State Management Across Agent Systems 230
9.10 Human Interaction in Multi-Agent Systems 236
9.11 Real-Time Decision-Making and Feedback Loops 237
9.12 The Integration Journey 246
9.13 Conclusion 247

Sample - Not for Redistribution

Part IV: Implementation and Ethical Considerations 249

10 Building Production Agents 251
10.1 Implementation Workflow 252
10.2 Agent Development Methodologies 257
10.3 Monitoring and Maintaining Agent Systems 264
10.4 Enterprise Scaling Challenges for Agents 272
10.5 Capstone Build Sequence: From Stub to Production 273
10.6 Conclusion 275

11 Operating Agents (AgentOps) 277
11.1 DevOps Foundations: From Commit to Running Container 277
11.2 Agent SRE Model 279
11.3 Evaluation Lifecycle 280
11.4 Security and Compliance: Protecting Customer Trust 285
11.5 Capstone Integration: Tracing, Metrics, and Runbooks 286
11.6 Workflow Engines for Complex Processes 287
11.7 Observability: Seeing the Whole Elephant 287
11.8 The Interconnected Lifecycle: DevOps, DataOps, MLOps 289
11.9 Incident Response & Chaos Engineering 293
11.10 Cost Optimization & FinOps for Agents 294
11.11 Infrastructure as Code & Platform Engineering 295
11.12 SRE Playbooks & On-call Excellence 297
11.13 Self-Audit Checklist 298
11.14 Conclusion 299

12 Governance and Responsible Autonomy 301
12.1 Ethical Governance Framework 302
12.2 Policy-as-Code Patterns 303
12.3 Human Oversight Patterns 305
12.4 Compliance Specifics 305
12.5 Transparency and Explainability 306
12.6 Accountability for Agent Decisions 309
12.7 Human-in-the-Loop Approaches 314
12.8 Risk Management for Autonomous Systems 321
12.9 Capstone Governance Flows 326
12.10 Conclusion 328

Part V: Case Studies and Future Directions 331

13 Playbooks and Case Studies 333
13.1 Autonomous Inventory Management 334
13.2 Agentic Pricing and Promotion Systems 342
13.3 Retail Media Networks (RMN) 347
13.4 Customer-Facing Agents 347
13.5 Agentic Commerce 353
13.6 Conclusion 353

14 Strategy, Maturity, and Futures 357
14.1 Implementation Roadmap and Maturity Model 357
14.2 Key Takeaways for Implementers 361

Sample - Not for Redistribution

14.3 Build vs Buy vs Partner Decision Framework 363
14.4 Agentic AI Snapshot 364
14.5 Emerging Trends in Agentic AI 366
14.6 The Path to Fully Autonomous Systems 369
14.7 Conclusion: Charting the Course for Agentic Systems 373

Appendix A: Advanced Mathematical Foundations for Decision Frameworks **375**

Appendix B: Retail Domain Coverage Map **385**
Minimum viable agent playbooks for lighter-coverage domains 386

Appendix C: Teaching Resources **388**
Notation Guide 388
Glossary and Acronyms 389
Companion Resources 391
How to Run the Companion Code 391
Case Outline Index 391
How to Teach This Book 392

References **394**

About the Author **405**

Index **406**

Sample - Not for Redistribution

List of Figures

1.1	Agentic AI System Loop	4
1.2	Example of interaction between customers, Agentic AI, data systems, and external services	8
1.3	Layered Reference Architecture with Cross-Cutting Concerns	9
2.1	Belief-Desire-Intention (BDI) architecture	24
2.2	OODA loop of the dynamic pricing agent	34
3.1	Structured Approach to Decision Making Process	48
4.1	A Pricing Markov Decision Process	73
5.1	Reinforcement Learning Cycle	90
6.1	Core technologies integration	116
6.2	Typical LLM integration workflow	117
6.3	Integrated agentic system addressing an inventory shortage	119
6.4	LLM-powered customer service agent	130
6.5	Computer vision for store awareness	132
7.1	Multi-layer retail sensor network showing store sensors feeding edge processing, gateway routing, cloud analytics, and agent decisions	142
7.2	Privacy and latency tradeoffs (illustrative ranges)	146
7.3	Edge gateway that redacts or aggregates sensitive signals before sending only safe summaries to the cloud	147
7.4	Sensor data processing showing ingestion, type-specific processing, confidence scoring, discrepancy handling, and maintenance	149
7.5	Retail knowledge graph of product relationships with customers, stores, suppliers, categories, promotions, substitutes, and complements	156
7.6	Causal graph of the promotion analyzer	168
7.7	Causal reasoning layer connected to LLMs, computer vision, IoT, and knowledge graphs	178
8.1	FIPA-inspired message example	188
8.2	Implementation of agent communication	189
8.3	Coordination between different agents and external systems in a retail environment	192
8.4	Auction mechanism for supplier selection	204
8.5	Cooperative inventory sharing between stores	206

Sample - Not for Redistribution

9.1 End-to-end retail integration architecture 212
9.2 CQRS pattern in retail 219

10.1 Reference Architecture for a Retail Agentic System (cloud and edge) 256

11.1 The Interconnected Lifecycle: DevOps, DataOps, MLOps 289
11.2 CI/CD Pipeline Blueprint 292

12.1 Ethical governance framework: policy, implementation, governance structure, and stakeholder feedback 302
12.2 Governance workflow from design and testing through deployment and monitoring with feedback loops 313
12.3 Escalation workflow showing trigger, human review, approval or override, and audit logging 317

13.1 Inventory Management Cycle 334
13.2 A typical multi-agent workflow for autonomous inventory 339
13.3 Agents in a dynamic pricing system and their interactions 343
13.4 Interaction of a customer-facing chatbot agent with backend systems to personalize responses 349

Sample - Not for Redistribution

Foreword

By **Professor Alain Abran, Ph.D., Ing.**
Professor Emeritus, Department of Software Engineering and IT
École de technologie supérieure (ÉTS), Montréal

When I first met Fatih as a doctoral candidate in software engineering, his curiosity was already leaning toward the then-nascent field of machine learning. Back then, discussions of autonomous agents and large-scale AI systems were largely confined to research seminars and speculative conferences; few imagined the industrial impact we see today. Yet Fatih was convinced, even then, that rigorous engineering principles could (and should) underpin intelligent systems long before "AI" became a ubiquitous business acronym.

We worked together first during his Ph.D., co-supervised with my colleague Jean-Marc Desharnais. Later, as a post-doctoral researcher in our laboratory, he and I co-authored publications that blended empirical measurement with innovative uses of predictive models. Those collaborations affirmed a shared conviction: software engineering, when anchored in disciplined methods and strong bodies of knowledge, can adapt and thrive even as the underlying technologies evolve at breakneck pace.

That conviction lies at the heart of *Foundations of Agentic AI for Retail*. This book is more than a technical manual and more than an industry playbook; it bridges scientific rigor and real-world applicability, and the enduring principles codified in the *SWEBOK* with the frontier concepts reshaping commerce through autonomous agents. The Retail Agent Operating Model (RAOM) gives that rigor a production spine that practitioners can actually ship.

A rigorous lineage

In my own career, I have argued that software engineering must remain rooted in measurable evidence and systematic knowledge. The *Software Engineering Body of Knowledge (SWEBOK)* was conceived to provide practitioners with a stable, shared foundation, much as civil engineers rely on structural mechanics or physicians on anatomy. Fatih extends that philosophy into the realm of Agentic AI. From his lucid treatment of Belief-Desire-Intention (BDI) models and OODA loops, to his detailed guidance on reinforcement learning pipelines and event-driven architectures, he demonstrates that even the most sophisticated AI agents can and must be engineered with the same care we devote to any critical system.

Why retail, why now? Retail may seem, at first glance, an unlikely proving ground for Agentic AI. Yet few industries present a richer set of real-time signals such as prices, inventories, customer behaviors, and supply-chain events that demand rapid, decentralized decisions. Fatih's choice of retail as a proving ground is therefore inspired: it exposes every limitation of monolithic, rule-based software and makes a

Sample - Not for distribution

compelling case for autonomous, collaborative agents governed by clear objectives, guardrails, and feedback loops.

Readers will appreciate how the book weaves advanced theory with concrete practice. Chapter by chapter, Fatih moves from foundational concepts to decision-making frameworks, enabling technologies, multi-agent coordination, and finally to full end-to-end integration, including the ethical and governance considerations that responsible engineers must never overlook. It aims to guide C-suite executives, software architects, data scientists, and graduate students alike.

The human dimension

Underlying the algorithms is Fatih's conviction that technology ultimately serves human progress. His emphasis on Human-in-the-Loop safeguards, transparency, and rigorous evaluation echoes the broader movement toward responsible AI, an ethos that fits the scientific mindset we fostered at ÉTS. I am particularly pleased to see extensive attention given to explainability, accountability, and risk management, ensuring that Agentic AI advances do not outpace our capacity to govern them.

A glance toward the horizon

Agentic systems are likely to permeate domains far beyond retail, including healthcare, energy, transportation, and public services, wherever complex, dynamic environments require continuous adaptation. The frameworks articulated here will serve as a template for those future applications. More importantly, they remind us that even as AI models grow in capability, the disciplines of requirements engineering, measurement, validation, and ethical oversight remain indispensable.

Fatih has delivered a timely and engaging work. It is a reflection of his evolution from inquisitive graduate student to industry leader and educator, and it reflects the very principles we strived to instill: intellectual curiosity, methodological rigor, and an unwavering focus on practical impact.

I invite you, the reader, to dig into these pages with both critical attention and creative imagination. May you emerge not only informed but inspired to engineer the next generation of intelligent systems that honor the best traditions of our discipline while venturing boldly into new frontiers.

Montréal, April 2025
Alain Abran

Sample - Not for redistribution

Preface

A Meeting of Theory and Practice

Retail is changing quickly, driven by new technology and shifting customer expectations. As artificial intelligence (*AI*) moves from research labs into day-to-day operations, retailers are testing ideas like smart shelves that reorder, promotions that adapt in real time, and systems that anticipate demand. But many promising pilots stall before they reach durable, at-scale *Agentic AI* deployment.

I've seen that gap from two sides: industry, where teams iterate quickly, and academia, where theory and ethics are stress-tested. In both settings, one idea has stayed compelling: the "*autonomous agent*", a software entity capable of perceiving its environment, reasoning about complex states, and taking action. Agentic AI systems can use the other forms of AI (predictive, prescriptive and descriptive) as tools, orchestrate the effort to carry out the best decisions, and learn from their mistakes.

That gap is why this book moves from first principles to production practice. It connects retail decision models to the evidence, authority, and operating controls needed to act responsibly.

Why Now?

Customer expectations keep rising: speed, personalization, and consistent offline-to-online experiences. Human-driven decision loops and *heuristic-based approaches* are straining under that load.

At the same time, advances in *computer vision*, *natural language processing*, *reinforcement learning*, *edge computing*, and engineering disciplines like *context engineering*, *AI engineering*, and *prompt engineering* make more adaptive systems possible. Agentic design patterns turn those advances into workflows you can operate and govern at scale.

Who This Book Is For

The primary reader is a retail product or technical lead responsible for moving a bounded agent workflow into operation. The book also serves product managers and business analysts, data scientists and engineers, and students, instructors and researchers.

This book assumes familiarity with Python, basic machine learning and statistics, and core retail operations, and it does not teach them from scratch. If any of those are new, start with the narrative sections, keep Appendix C's notation guide and glossary at hand, and return to the deeper technical material when you need it. Some of my students skip the foundations and let coding agents do the work. A coding agent can build a notebook or a simple web app quickly, but the work needs many assumptions,

Sample - Not for Redistribution

decisions, and authorizations along the way, and without the foundations a student cannot get the best out of the agent.

Scope and Structure

After Chapter 1, five parts cover foundations, technologies, coordination, production, and case studies. The table gives the question each part answers and what it delivers.

Part	Core question	Value promise
I. Foundations of Agentic AI (Chapters 2–5)	*How do agents that share one low-stock product act within policy and choose among competing actions?*	Policy and tool boundaries around Belief-Desire-Intention (BDI), Observe-Orient-Decide-Act (OODA), and ReAct (Reasoning and Acting) agents, then optimization, Bayesian and causal reasoning, Markov decision processes (MDPs), reinforcement learning, and planning for deciding whether to cut the price, reorder, or wait.
II. Enabling Technologies and Architectures (Chapters 6–7)	*Which technologies supply the agent's evidence when a shelf shows a gap?*	Language models, computer vision, sensor networks, knowledge graphs, and causal reasoning, enough to sketch a shelf-availability stack with its contracts and in-store latency budget and to name the controls it still needs before production.
III. Multi-Agent Coordination and Interoperability (Chapters 8–9)	*Who owns an incident when several agents act on it?*	Roles, handoffs, and allocation mechanisms through which specialized agents share evidence and resolve conflicting recommendations, then the workflows, events, APIs, reservations, oversight, and feedback that carry decisions out.
IV. Implementation and Ethical Considerations (Chapters 10–12)	*How does the integrated system get to production, stay reliable while it changes, and stay accountable?*	Building (Chapter 10), operating (Chapter 11), and governing (Chapter 12) the production agent, with three readiness instruments: production readiness gates, a self-audit checklist, and go or no-go governance gates.
V. Case Studies and Future Directions (Chapters 13–14)	*Which controls hold up in retail deployments, and how does a retailer get from a first pilot to dependable autonomy?*	Five domain playbooks with rollout and failure lessons, drawn from published cases and anonymized composites, or labeled as design sketches, then a maturity model with gates between stages and emerging trends with their limits.

Each chapter opens with a running case and learning objectives and ends with a Conclusion and graded exercises. Most chapters also add to the book's capstone, an inventory-risk workflow with supplier coordination, pricing protection, and customer communication, whose agent roles Chapter 2 names and whose build sequence Chapter 10 sets out. Read straight through, or follow your route below.

NOTE **A route for each reader**

If you are a product manager or analyst, read all fourteen chapters in order, following each case, decision table, contract, and operational frame, and skim the listings and derivations. Slow down for Chapter 3's causal questions and capstone metrics, Chapter 8's approval contracts, Chapter 9's guide to events, APIs, and queues, Chapter 11's evaluation lifecycle, and Chapter 13's playbooks. Each playbook's five parts, from KPI goal to failure lessons, make a template for scoping your own use case.

If you are an engineer or data scientist, read Chapters 1 to 12 in order with their listings, and run the companion demos as you go (Appendix C shows how). Data scientists and

Copyright © Manning Publications
For online reading, not for distribution

NOTE CONTINUED **A route for each reader**

researchers should take Chapters 3 to 7 in full, derivations included, with Appendix A for the proofs and bounds. Engineers and architects can skim the derivations of Chapters 3 to 5 and spend longest on Chapters 2, 6, and 8 to 11. Then read Chapters 13 and 14, running `demos/capstone_scaffold_demo.py` for the approval-and-receipt record and `demos/capstone_gate_review_demo.py` for the simulated gate review. The companion code gives you the capstone's scaffold. Completing it through Chapter 10's seven-step build sequence and checking it against that chapter's readiness gates is left to you.

If you are a student, read the chapters in order, finish each chapter's worked case before its exercises, and work through the mathematics and the companion demos. If you teach, Appendix C maps the chapters to an MBA course of six to eight weeks and a technical course of ten to twelve. It assigns the Check exercises as short responses, its case outlines as lab starting points, and a final project that adapts the capstone to a chosen retail domain.

My Journey and Aspirations

I wrote *Foundations of Agentic AI for Retail* from the intersection of enterprise technology, academic research, and product delivery. As Founder & CEO of OODARIS AI, I work on agentic systems in retail and the realities of integration, governance, and measurement.

At McGill University, where I am an Assistant Professor, I teach and research *AI* and mentor practitioners with diverse backgrounds and a shared interest in applied innovation. My earlier roles leading data, analytics, and AI inside retail organizations ground the examples in operational reality.

My first production AI system, in 2008, was a recommender that suggested modules to the clients of an independent software vendor, and at times its recommendations were not fully appreciated. I have worked in the retail, banking, investment, health care and telecommunications domains and on their use cases, and I led AI at a bank in Canada.

People need to trust agentic systems before they let them be fully autonomous. To gain that trust, they need to understand how the systems are built, what they will do, how they make decisions, and how their actions and decisions can be tuned, reverted and defended. Practitioners, builders and users all need that understanding, and all of it requires knowledge of the foundations.

This book is both a record of what has worked and a guide for what comes next. My hope is that these pages demystify Agentic AI and help you move from proofs-of-concept to production responsibly. Done well, autonomous agents can reduce toil and tighten feedback loops, freeing teams for more valuable work.

Code Repository and Interactive Notebooks

Run the examples from the GitHub repository at https://github.com/gradient-divergence/agentic-retail-foundations. Use the short scripts under `demos/` for checks or automation and the *Marimo* notebooks for step-by-step exploration with adjustable parameters. Appendix C walks through the setup.

Sample - Not for Redistribution

Disclosure and Neutrality

I am an Assistant Professor at McGill University and Founder & CEO of OODARIS AI. I wrote this book as an independent academic and practitioner reference. Products, platforms, tools and vendors are named to illustrate patterns and trade-offs, not to endorse them or prescribe a specific solution. What the book says about platforms, vendors, protocols and standards is as of October 2026, unless the text gives another date.

Acknowledgments

This book reflects input from retail operators, data scientists, ethicists, supply-chain strategists, software engineers, and academic researchers. I am grateful to the thought leaders, students, and practitioners who challenged the ideas, flagged gaps, and improved early drafts.

A Community of Scholars and Innovators

I first thank the **faculty and research staff** at McGill University, especially within the Desautels Faculty of Management, for fostering a rigorous and intellectually stimulating environment. Their open forums, reading groups, and joint projects challenged and refined my thinking on autonomous systems. I am particularly grateful for the **interdisciplinary collaborations** that offered diverse perspectives on AI's role in retail.

Students: The Lifeblood of Inspiration

To the **graduate and undergraduate students** I've encountered: your curiosity and tenacity in courses like *Enterprise Data Science: Concepts and Algorithms*, *Enterprise Machine Learning in Production*, *Introduction to AI and Deep Learning*, *Applications and Architectures of Deep Learning*, and *Designing and Developing Agentic AI Systems*, as well as in hackathons and seminars, pushed me to re-examine assumptions and seek better solutions. This book benefited from our dialogues.

I also acknowledge the *Retail Gen AI Hackathon and Capstone project* teams at McGill University. Your drive to apply theory to practice informed many *use cases* and *architectural frameworks* in this book.

Industry-Academia Synergy

My experiences at the **ALDO Group** reinforced what it takes to deploy AI in retail. Collaborating with *data scientists*, *engineers*, and *strategists* provided practical insight into integration, governance, and operational constraints. Special thanks to the Data, Analytics, and AI team for their exploratory spirit and feedback.

Gratitude also extends to the broader ecosystem of **industry partners, research consortiums, and AI conferences** whose shared experiences shaped the *code snippets*, *decision frameworks*, and *multi-agent coordination* strategies presented.

Technical Reviewers and Early Readers

This book has been strengthened by the critical eyes of the **technical reviewers and early readers** who generously devoted their time to dissecting initial drafts. Their

Sample - Not for Redistribution

attention to detail, pointed questions, and calls for clarity improved both accuracy and readability.

Notable contributions include:

- **Arial Huang:** Reviewed and drew distinctions between traditional AI and *Agentic AI*, sharpening the narrative.
- **Armen Momejian:** Provided feedback on structure and organization, plus suggestions across multiple chapters.
- **Arthur Pentecoste:** Delivered chapter-by-chapter reviews of narrative flow, context, technical accuracy, and LaTeX corrections.
- **Basant Mounir:** Contributed input on overall structure and chapter organization, with feedback on several sections.
- **Chiara Liu:** Shared insights on governance, practical code implementation, and advanced LLM techniques.
- **Joseph and Roonie Corera:** Offered general feedback that improved clarity and presentation.
- **Laurence Audrey Vincent:** Provided detailed feedback on the chapter covering *Governance and Responsible Autonomy*.
- **Matthieu Houle:** Provided comprehensive feedback across multiple chapters, focusing on conceptual clarity and figure/example improvements.
- **Necmiye Genc:** Offered thoughtful commentary and a fresh lens on structure and substance.
- **Onur Erkin Sucu:** Reviewed mathematical components and source code, enhancing clarity and precision.
- **Yael Kochman:** Gave feedback on clarity, structure, and accessibility of introductory concepts.
- **Yash Joshi:** Suggested improvements to modern agent architectures, frameworks, deployment patterns, and industry case studies.

A special mention goes to reviewers who stress-tested the ideas against real-world scenarios. Your vantage point, at the intersection of academic experimentation and brick-and-mortar realities, kept the text grounded.

Looking Ahead

I view this book not as a static endpoint but as part of a living conversation about the evolution of AI in retail. Progress depends on open exchange, interdisciplinary work, and inclusive dialogue. I hope readers will take these concepts, challenge them, refine them, and extend them in practice.

To all those students, researchers, industry colleagues, and academic peers who have fueled my passion for teaching and learning, thank you. Your contributions guided me in bringing together the theoretical and practical sides of *Agentic AI*. I am grateful for your support and energized by the work still ahead in retail.

Fatih Nayebi

Montréal, 2026

Sample - Not for Redistribution

What's New in the Third Edition

The third edition keeps the second's fourteen chapters, in the same order and under the same titles. What changed is inside them.

Each chapter now picks up the question the one before it left open and hands its own to the next, and running retail cases carry each part. In Part I, a pricing agent and an inventory agent share a low-stock product and must decide whether to cut the price, reorder or wait, until the supplier warns mid-promotion that new orders will take a week longer to arrive. In Part II, a camera flags an empty shelf: was the product sold or only moved? Part III asks who owns that incident when several agents act on it. Part IV readies the capstone system for production, with Black Friday as its stress test and a winter-coat markdown that draws complaints as its governance test. Part V follows a hypothetical retailer's holiday pilot and asks what it must show to earn more autonomy.

In every chapter I now say what I have seen in my courses at McGill University or in the agents my team builds at OODARIS AI. Chapter 3 describes our forecasting tournament, which simple statistical models win for some product categories. Chapters 4 and 5 say that our models learn offline, from logged decisions, and that I have not run a learned policy on live traffic. Chapter 6 introduces decision models, learned models that each make one narrow decision for an agent. Their vendor trains them with an unpublished method it calls RLCD (reinforcement learning for calibrated decisions), so the chapter describes only what they do in our system. Chapter 9 reports the failures we met between workflow steps, Chapter 11 our worst deployment day, Chapter 12 how an approval queue becomes a rubber stamp, and Chapter 14 a prediction I got wrong.

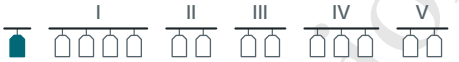
In October 2026, the companion's dependencies were updated to their latest compatible releases and its code was improved, and every runnable listing in print now comes from it. A demo that needs an API key or a running service says which one and stops. The regression suite grew from 36 test files to 112, and long listings are shortened in print, with each cut marked and the full file named.

In print, the book is now 7 × 10 inches and in full color. Each part opens on two pages, with a map of how its chapters carry the running case, and each chapter ends with one Conclusion and one list of exercises, graded Check, Apply or Analyze against its learning objectives.

If you read the second edition, your chapter references still hold, though the exercises are renumbered. To see what changed, read the part openers, each chapter's opening and close, and Chapter 13. Appendix C's assignments now use the Check exercises.

Sample — Not for Redistribution

1



Introduction: The Age of Agency

15 pages, 1 code listing

A product is running low in one store. Reorder too late and the shelf is empty before the delivery arrives, and the next customers leave without it. Reorder too much and the surplus sits in the stockroom as overstock. Someone has to make that call, product by product and store by store, and in retail, delay is costly. Retail AI is now shifting from systems that chat to systems that act, and a reorder like this one is the kind of decision such systems take on. This chapter asks what makes a system an agent and why a retailer should want one. Behind that question sits the one the whole book answers: when should a retailer hand a decision to software, and what evidence and controls make the handover worthwhile? We start with a clear definition and the core loop that carries the rest of the book, and a small agent that reorders one product carries that loop through the chapter. What that agent leaves undone sets the agenda for the chapters after it.

We define **Agentic AI** and set it apart from generative AI and traditional automation. Next, we walk through the **Perceive–Reason–Act–Learn** loop that drives these systems, then the technologies, data, and architecture it runs on. Finally, we turn to the **retail domain**, to the operating model that keeps agents safe and the use cases that matter most.

OBJECTIVES

By the end of this chapter, you will be able to:

1. Define **Agentic AI**, distinguish it from **generative AI** and traditional automation, and identify its enabling technologies.
2. Explain the **Perceive–Reason–Act–Learn** loop and outline the data, architecture, and controls needed for retail deployments.
3. Explain why **retail** is a strong fit for agentic systems.

1.1 What is Agentic AI?

The low-stock product is at risk again. A system detects the stockout risk, triggers a replenishment order within policy limits, and alerts the store manager with a short summary of its rationale. Before a retailer lets software do that, it needs to know what kind of system it is trusting. **Agentic AI** refers to AI systems, often called *AI agents*, that autonomously perform tasks on behalf of a user or another system (Stryker 2024; Russell and Norvig 2020). Rather than return fixed responses, they plan workflows and

Sample Content for Peer Review Not for Distribution

use tools to make decisions and take actions with minimal human input, perceiving their environment, reasoning about what they observe, and acting to achieve specified goals. They can also interact with external data sources and services beyond the data they were trained on (Stryker 2024), which lets them adjust to real-time information and unforeseen situations.

Agentic AI is not the same as generative AI. Generative AI produces content (text, images, audio, video, and code) in response to prompts. Agentic AI is *goal-directed* and can operate autonomously over extended periods. It adds planning, tool invocation, state management, and controls so the system can execute multi-step work. Agentic systems often combine several techniques (large language models, or LLMs, with traditional algorithms and tools) to reach a precision that generative models alone may not deliver (Finn et al. 2025).

The idea of an agent is not new. Classic AI literature describes an *intelligent agent* as an entity that perceives its environment and acts to achieve goals. Agentic AI builds on that foundation with advances such as LLMs and reinforcement learning (learning from the results of actions) (Sutton and Barto 2018). Deep reinforcement learning, for instance, learned to play 49 Atari games from screen pixels and game scores (Mnih et al. 2015).

Agents appear across industries, from autonomous vehicles to automated trading, but they suit complex, high-velocity environments like **retail**, this book's subject, where **shopping assistants** serve customers and **supply chain agents** optimize logistics and inventory.

1.1.1 Agentic AI vs. Traditional AI, Generative AI, and Automation

Agents are easiest to define against the systems a retailer already runs. Traditional AI systems and many machine learning (ML) deployments are typically bounded pipelines that run within a predefined workflow and stop. Effective as they are, they do not usually plan multi-step actions or invoke tools across systems without explicit orchestration. Agentic AI differs in *autonomy* and *adaptability*, because it can adjust its behavior using fresh context (retrieval, tool outputs, policies, and feedback). The effect is a shift from passive tools to operational partners, which lets a business scale operations and respond to complexity without a proportional increase in human labor.

Table 1.1 sets automation, generative AI, and agentic systems side by side in a retail context.

TABLE 1.1 *Automation, generative AI, and agentic AI in retail*

System	What it does	Where it breaks	Why it matters in retail
Automation	Executes fixed rules	Fails when context shifts	Reliable for routine tasks only
Generative AI	Produces content and summaries	Hallucinates or drifts without guardrails	Best for language, needs guardrails for action
Agentic AI	Plans and executes actions with policies	Requires strong data and controls	Enables safe autonomy at scale

1.1.2 From Automation to Agents: The Evolution of AI in Retail

Retail AI reached that last row in three waves, each adding to the one before rather than replacing it.

Sample - Not for Redistribution

The **first wave** automated routine tasks. Early retail technology was transactional and handled repetitive work such as inventory management, point-of-sale transactions, and basic data processing. These systems followed rules fixed in advance, such as reordering when stock falls below a set point, as the agent in this chapter's listing does, and anything outside those rules went to people. Rule-based replenishment long predates modern AI. Arrow's account of the 1951 research on inventory policy describes ordering when inventory reaches a threshold and replenishing to a target level (Arrow 2002).

The **second wave** brought predictive intelligence through machine learning. Systems could now identify patterns and make forecasts, and retailers began using them for demand forecasting, personalized recommendations, and pricing optimization. Even so, many retail deployments remained confined to narrow workflows and needed additional orchestration for cross-functional decisions.

The **third wave** is Agentic AI. It builds on earlier analytics, ML, and *generative AI* advances, such as LLMs that perform language tasks from a few examples placed in the prompt (Brown et al. 2020), and what it adds to them is autonomy. It joins the versatility of LLMs to the structured decision-making of traditional software (Stryker 2024), much as neuro-symbolic AI combines neural methods with symbolic reasoning (Sarker et al. 2021), so an agent can decide *what* needs to be done next instead of waiting for a human prompt at each step. In February 2025, Bernard Marr described autonomous shopping, dynamic pricing, and automated supply-chain management as potential retail applications (Marr 2025). Agentic systems have the four classic agent properties that Wooldridge and Jennings set out in 1995 (Wooldridge and Jennings 1995).

- **Autonomy**: Agents make decisions aligned with broader business objectives without continuous human oversight.
- **Reactivity**: They detect and respond to real-time changes within their operational environment.
- **Proactivity**: They initiate strategies and actions toward predefined objectives instead of only reacting.
- **Social Ability**: Agents communicate, collaborate, and coordinate actions with other systems, agents, and humans toward shared objectives.

These properties say what an agent does, and its loop shows how.

1.1.3 How Agentic AI Works

Each simulated day in this chapter's listing, the agent for the low-stock product reads its stock, compares it with a threshold, and orders or waits. Those are three steps of the **Perceive–Reason–Act–Learn** loop this book uses as its teaching model (related to sense–think–act and perceive–decide–act cycles, with feedback for evaluation and adaptation), and the listing leaves the fourth empty. Figure 1.1 shows the full loop. The agent perceives signals from the retail environment and turns them into state, reasons over that state to plan and decide, and acts through tools and APIs. Its actions change prices, orders, or replies in the store. Outcomes then feed the Learn step, which evaluates them and sends updates back to reasoning (NVIDIA, n.d.).

1. **Perception (Sensing)**. The agent **gathers data** from its environment and inputs, including real-time signals from databases, APIs, user interactions, or physical sensors, to *perceive* the state relevant to its objectives. For example, an agent might

Sample - Not for Redistribution

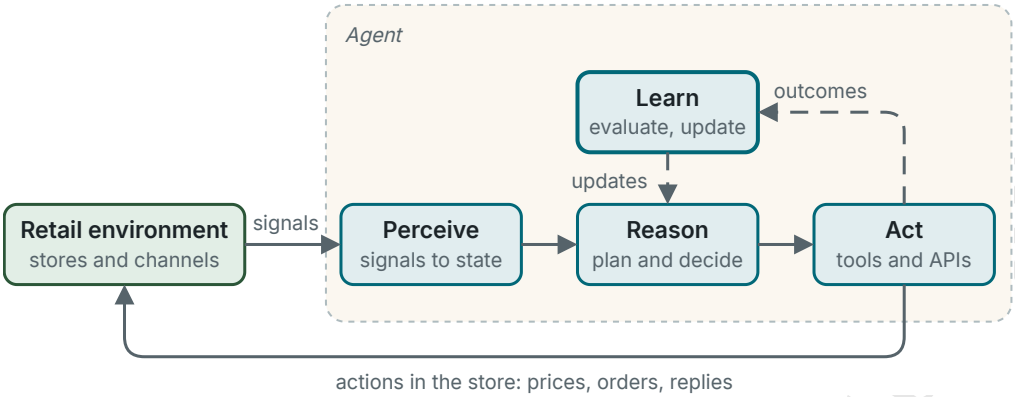


FIGURE 1.1 *Agentic AI System Loop*

ingest market data, system logs, or user emails, then normalize those inputs into facts, trends, or anomalies.

2. **Reasoning (Planning/Deciding).** The agent **analyzes the information, formulates a plan, and makes decisions.** In modern Agentic AI, an LLM or another AI model often acts as the *brain* of the agent. Given goals and perceived state, the agent generates possible actions, predicts outcomes, evaluates options, and selects a course of action. For complex tasks, it can break the work into sub-tasks, use specialized skills or tools (for example, a pricing optimization algorithm), and assemble a solution. Techniques like retrieval-augmented generation (RAG) let an agent retrieve relevant information before it generates a response or plan, which improves accuracy. Later chapters introduce context engineering, the discipline of shaping what the agent retrieves and remembers before each decision. The result of this phase is a decision or an action plan (e.g., *"reduce the price of item X by 10% for the next 48 hours and send a restock order for 500 units"*). A merchant would check such a plan before acting on it, and so should the agent: the margin plan has to fund the discount, and the restock has to arrive before the discounted demand empties the shelf.
3. **Action (Execution).** Once a plan is in place, the agent **acts** through the necessary systems or tools. In software, that could mean calling APIs, updating databases, sending messages, or triggering workflows. Actions range from modifying a configuration to executing a financial transaction or controlling a robotic arm. Agentic AI frameworks integrate with external tools so that digital decisions have real-world effects. Because they do, developers can enforce **guardrails** to keep actions safe (e.g., requiring approval for high-stakes actions).
4. **Learning (Feedback Loop).** After acting, the agent observes the new state of the environment and evaluates the outcome (Sutton and Barto 2018). Did the action succeed? Did it move closer to the goal? This feedback informs the next round of planning and, when warranted, gated updates to prompts, policies, or models.

Learning comes with a caution. An agent that learns through live trial and error can encounter unsafe states and cause harm, so reinforcement learning must address safety during learning as well as in deployment (García and Fernández 2015). In the deployment pattern this book uses, those updates are evaluated offline and versioned before release, and online adaptation runs only within explicitly tested limits. The

Sample Not for Distribution

long-term aim is an agent that automates tasks and continuously **optimizes** how it does so.

1.1.4 Code Example: Implementing a Simple Agent Loop

Return now to the low-stock product from the opening, with an agent small enough to read in full. It watches that product's stock and decides when to reorder, the loop behind a low-stock alert that requests replenishment before the shelf goes empty. With one fixed rule, it is an agent only in the classic sense and sits in the automation row of Table 1.1, which keeps its loop easy to see. The simulation also assumes immediate receipt, so an order adds stock the same day. A real replenishment workflow keeps ordered stock on order until receipt, then adds it to on-hand stock, and bases reordering on inventory position, defined as on-hand plus on-order stock minus backorders (Pound 2013).

NOTE **Code implementation note**

The listing comes from `demos/intro_inventory_book_demo.py` in the companion repo, and `notebooks/intro.py` runs the same seven-day scenario (see Preface).

The class has one method for each step of the loop, with `learn()` left empty, and the lines below it run the agent for seven simulated days. If you build agents, trace those days in the code. If you are reading for the business case, read from the paragraph after the listing that begins *Run as written* through the decision record, then skip to *Making the Retail Business Case* and its investment test.

```
class InventoryAgent:
    def __init__(self, reorder_threshold, max_capacity):
        self.reorder_threshold = reorder_threshold
        self.max_capacity = max_capacity
        self.current_stock = 0
    def perceive(self, external_data):
        self.current_stock = external_data.get("stock_level", self.current_stock)
    def decide(self):
        if self.current_stock < self.reorder_threshold:
            order_quantity = self.max_capacity - self.current_stock
            return {"action": "reorder", "amount": order_quantity}
        return {"action": "wait"}
    def act(self, decision):
        if decision["action"] == "reorder":
            amount = decision["amount"]
            print(f"Placing order for {amount} units.")
            self.current_stock += amount
    def learn(self, feedback):
        pass
agent = InventoryAgent(reorder_threshold=50, max_capacity=100)
environment = {"stock_level": 60}
for day in range(1, 8):
    print(f"\nDay {day}: Stock level = {environment['stock_level']}")
    agent.perceive(environment)
    decision = agent.decide()
    agent.act(decision)
    sales = 15 if day == 3 else 5
    environment["stock_level"] = max(agent.current_stock - sales, 0)
    agent.learn(feedback={"sales": sales, "stock_level":
environment["stock_level"]})
```

Each day the agent reads the stock level in `perceive()`. In `decide()`, the Reason step, it compares that level with the threshold of 50 and returns either a reorder or a wait. In `act()` it simulates the order by adding the units to its own stock count.

Run as written, the listing reports stock of 60, 55, and 50 on days 1 to 3 and places no order. Day 3 is the one to study. Stock equals the threshold, and the rule reorders only *below* it, so the agent waits, and that day's 15 sales, three times the usual 5, leave 35. On day 4 the agent reads 35 and orders 100 − 35 = 65 units, and because receipt is immediate, stock is back at 100 the same day. Days 5 to 7 report 95, 90, and 85. Had the rule also reordered *at* 50, the agent would have ordered 50 units on day 3, before the spike instead of after it. The logic mirrors a simple (s, S) inventory policy with reorder threshold $s = 50$ and order-up-to level $S = 100$, which reorders when stock drops below s and orders up to S. The empty `learn()` marks where a real system's evaluation could adjust that threshold or estimate better order quantities in later updates.

In production, the source of truth is the external system state, tool calls need idempotency (a repeated call must not repeat its effect), failures must be retried safely, and every action should be auditable. Rolling back the software does not undo an order a supplier has accepted, so each order needs a cancellation deadline, a compensating action, and an accountable owner (Garcia-Molina and Salem 1987). This example strips those controls to keep the loop visible.

The listing is one demonstration with its own numbers, and later chapters' demonstrations have theirs. The case itself is written down as a **decision record**, which Chapter 2 revisits for the same low-stock product. For day 4 of the listing, it reads as follows.

- **Product**: one low-stock product at one store, the product Part I follows.
- **Stock state**: 35 units on hand and none on order, against a threshold of 50 and an order-up-to level of 100.
- **Decision horizon**: one simulated day, with the order received the same day.
- **Permitted actions**: reorder to the 100-unit level, or wait.
- **Decision**: reorder 65 units.
- **Unresolved**: whether 50 is the right threshold, what checks the order against policy, and what happens when another agent acts on the same product.

Later chapters supply the evidence and controls the record lacks, and Chapter 2 starts from its last question.

1.2 Core Technologies and Architectures Enabling Agentic AI

The decision record's first open question, whether 50 is the right threshold, is one a fixed rule cannot ask. Agentic systems run the same loop on four AI technologies that can help answer it.

Machine learning (ML) is the backbone of Agentic AI because it lets systems improve through experience. It learns patterns from historical data, so an agent can forecast demand for the low-stock product instead of trusting a fixed threshold, or detect anomalies as they occur. **Natural language processing (NLP)** lets it interpret and generate the language of emails, tickets, and documents. **Cognitive architectures** combine memory, reasoning, and learning in one design, so it can recall past events and plan multi-step tasks. **Decision-making algorithms** compare feasible actions.

Sample Not for Redistribution

They include planning, optimization, reinforcement learning policies, rule engines and knowledge graphs that apply logical constraints, and probabilistic reasoning that weighs which action is likely to succeed. Modern agents often combine them.

1.2.1 The Role of Data in Powering Agentic AI

The listing's agent perceives a single number, its stock level, so all it can do is compare that number with 50. All four technologies run on **data**, the fuel for learning and the context for decisions, and a retail agent may draw on many data sources, such as point-of-sale (POS) transaction records, inventory levels, supplier lead times, e-commerce website analytics, customer reviews, social media sentiment, and real-time video feeds from in-store cameras. By **integrating multiple data streams**, the agent builds a fuller picture of the business. Combining weather data with sales data, for instance, might let an agent predict a spike in demand for certain products (like raincoats or cold drinks) and move stock ahead of the spike, provided a transfer or delivery can land before the weather does.

The edge comes from **processing data continuously and using it in evaluation and updates**, which often takes pipelines that deliver fresh data in real time or near real time and handle large volumes, cleaning, retrieval refresh, and controlled retraining. An agent's **adaptability ties directly to its data inputs**, so missing or poor data degrades its performance, and new or more granular sources can improve it.

⚠ WARNING Data quality considerations

High-quality, diverse data is important for Agentic AI success. Ensure:

- Real-time data access
- Clean and preprocessed data pipelines
- Proper data governance
- Privacy and security compliance
- Data shared across silos

Data also enables **personalization**. A retail agent might use customer-specific data (purchase history, browsing behavior) to tailor recommendations or marketing outreach. Retailers already hold much of this data. Their large databases of products and customer interactions serve as a training ground for agents that optimize pricing, promotions, and supply chain decisions at scale. Sensitive data brings responsibility, and privacy and security come first when an agent ingests and acts on it. We return to security and ethical considerations in a later chapter, but autonomous agents require a solid data governance strategy from the start.

Data and models are still only part of the system, and two terms keep the rest in view.

NOTE AI Engineering vs. ML Engineering

AI engineering in this book refers to building the full system around the model, that is, the tool contracts (each tool's schema, limits, and error handling), workflows, evaluation, and operational controls that make agent decisions safe to ship. **ML engineering** focuses on training and serving models. Agentic systems require both, but the engineering emphasis shifts toward system behavior, policy enforcement, and traceable outcomes rather than model accuracy alone.

Sample Manuscript Copyright

1.2.2 Agentic AI System Architecture

Suppose the low-stock product's order had to reach a real supplier, stay within a budget, and leave a record an auditor could follow. Those are architecture questions, and safe autonomy in retail sets the architecture four requirements.

- Tool calling and structured outputs must be audit-ready.
- Multimodal signals (images, video, and sensor readings) matter for retail decisions alongside text.
- Cost and latency constraints must be designed into workflows.
- Governance and evaluation are required.

The simplest view of the architecture is a single interaction, and Figure 1.2 traces one purchase, the loop in miniature, from the first browse to the profile update.

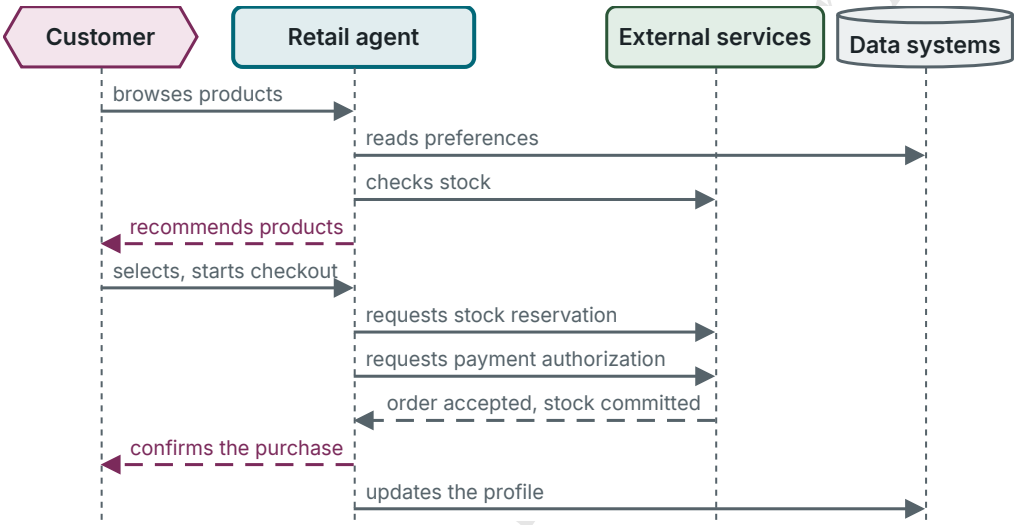


FIGURE 1.2 *Example of interaction between customers, Agentic AI, data systems, and external services*

The sequence is illustrative, and two of its steps are easy to misread. Payment authorization is distinct from capture: it places a hold on the payment method, and the funds are captured later (Stripe, n.d.). Committing stock sets it aside for one order, and only the accepted order commits it, before the agent confirms the purchase and updates the profile. As of October 2026, Shopify includes committed stock in on-hand inventory and excludes it from available inventory (Shopify, n.d.-b); the listing instead deducts simulated sales from on-hand stock.

A single interaction hides most of the system. The **layered reference model** in Table 1.2 is a checklist that groups responsibilities for building and operating an agent into six layers and two cross-cutting concerns.

Sample Not for Distribution

TABLE 1.2 *Layered reference architecture and cross-cutting concerns for agentic AI systems*

Layer / concern	Description and role in Agentic AI system
Layer 1: Foundation Models	Pre-trained language and vision models whose base capabilities, such as understanding text, recognizing images, or predicting trends, higher layers turn into task-specific intelligence.
Layer 2: Data Operations	Pipelines that ingest, clean, transform, and store retail data (sales, stock, etc.), feed it to the agent in real time, and maintain its knowledge bases.
Layer 3: Agent Frameworks	The libraries and runtimes in which the agent's logic is written, combining data (Layer 2) and models (Layer 1) through abstractions such as "goals", "actions", and "tools".
Layer 4: Development Tools	Simulation environments for safe testing, monitoring dashboards, and integration tools for building and debugging agents and connecting them to existing software such as POS systems or databases.
Layer 5: Deployment Infrastructure	Cloud services, in-store edge devices, container orchestration, and APIs that run agents reliably and efficiently in production, across every location of a chain and with many concurrent interactions.
Layer 6: Agent Ecosystem	The applications through which agents deliver value to employees, customers, or other agents (customer service bots, inventory robots, recommendation systems, etc.), plus any marketplace for adding new agent capabilities.
Cross-cutting: Monitoring & Observability	Continuous monitoring of performance, behavior, and system health through metrics, decision logs, and anomaly and failure detection across all layers, so issues surface before they affect business outcomes.
Cross-cutting: Governance, Security & Compliance	Authentication, authorization, data privacy (e.g., GDPR in customer data handling), regulatory compliance, and threat protection, designed into each layer throughout the system lifecycle because retail agents may handle sensitive information and make financial decisions.

Figure 1.3 draws the same model, with Layers 1 to 3 grouped as core intelligence, Layers 4 and 5 as engineering, and Layer 6 as the ecosystem, and the two cross-cutting concerns alongside.

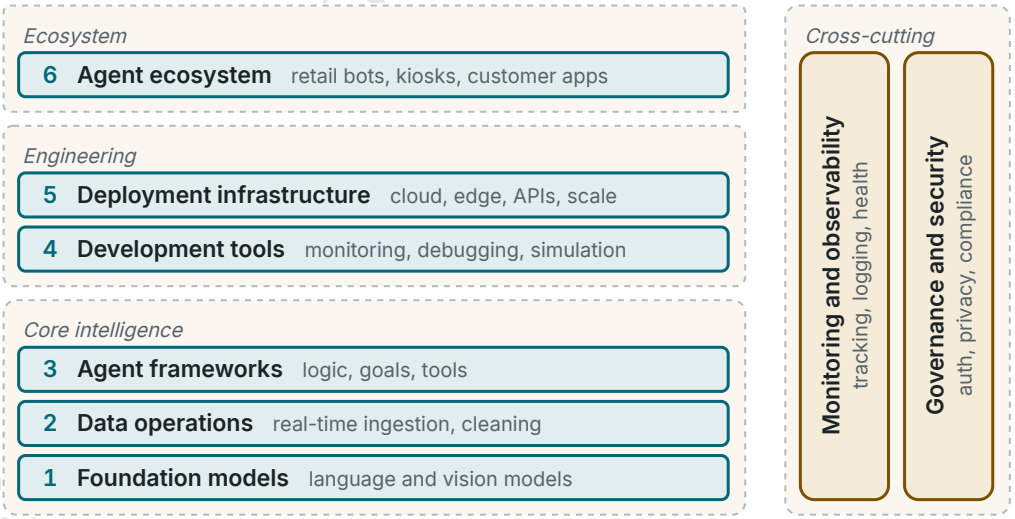


FIGURE 1.3 *Layered Reference Architecture with Cross-Cutting Concerns*

Sample not for Redistribution

Real systems rarely separate this cleanly, but the checklist still finds gaps. Read the listing's agent against it. Its reorder rule is Layer 3 logic, its stock reading stands in for a Layer 2 pipeline, it calls no Layer 1 model, and with neither cross-cutting concern, nothing stops an agent like it from overspending or breaking policy. Skipping a layer creates risk of this kind, and a pilot that works in the lab can stall for want of data, infrastructure, monitoring, or security. The model is also one arrangement among several. Some architectures emphasize **modularity**, with skills packaged as modules that can be recombined, and others **orchestration**, where a central controller manages sub-agents (one might specialize in price optimization and another in restocking).

Orchestration matters because a real deployment often runs several loops at once. A *multi-agent system* divides responsibilities among such **collaborating agents**, which share information through shared memory or knowledge bases. Each agent typically follows the perceive–reason–act–learn cycle internally, and agents can negotiate or coordinate during the reasoning phase. A marketing agent may ask a supply chain agent whether stock is available before it launches a promotion, for instance. This design lets Agentic AI solutions scale across functions while each agent keeps its autonomy and flexibility (Arsanjani 2024). Chapter 8 returns to multi-agent systems.

1.3 Making the Retail Business Case

Agentic AI, the third wave, reaches beyond retail. On the first day of class, I tell my students that this is the most important topic they will learn. The software industry and knowledge work are going through a revolution as we build intelligent systems that perceive, orient, decide, act, and learn. The opportunities are large, because all the software that is not intelligent today will be transformed by agentic approaches. User experiences, websites, and mobile apps will become more agentic and more conversational too. The useful question for a retailer is narrower. Which decisions justify autonomy, and what evidence would make the investment worthwhile?

1.3.1 Why Agentic Approaches Matter in Retail

Day 3 illustrates when a fixed threshold triggers an order. The demonstration assumes immediate receipt and records no stockout. Compare replenishment policies using the same available information, lead times, costs, contribution, and service constraints before claiming an advantage. The business itself is volatile, with shifting consumer demands, rapid technology advances, and competitive pressure, and traditional retail systems, often inflexible and manually intensive, struggle to keep pace.

Agentic AI addresses these limitations with system-level properties that retail needs on top of the four classic ones. **Adaptability** matters when disruptions occur. After an unexpected weather event, for instance, an agentic logistics system can reroute shipments and prioritize essential goods where carrier capacity and lead times allow, and engage other agents to manage customer expectations and limit the damage. **Resilience** lets agentic systems handle scenarios that deviate from historical patterns. **Scalability** can come through centralized orchestration or through distributed agents, and either way, scaling across markets, channels, and products requires explicit coordination, shared contracts, and operational controls.

Surveys separate AI use from agent deployment. In McKinsey's cross-industry survey of June and July 2025, 88% of respondents reported regular AI use in at least one business function, 23% reported scaling agentic AI in at least one function, and an

Sample - Not for Redistribution

additional 39% reported experimenting with AI agents (Singla et al. 2025). These are self-reports across industries, not retail measurements. Adoption is uneven, results depend on the use case, data maturity, integration, and governance, and the right pace differs for each retailer. Such surveys suggest possible benefits, but they cannot establish the value of any one deployment, which takes the investment test at the end of this chapter. To keep autonomy from outrunning governance meanwhile, the book uses a single operating backbone.

1.3.2 The Blueprint: The Retail Agent Operating Model (RAOM)

Before the listing's agent sends a real order for 65 units, someone has to say which policy permits it, what evidence supports it, and where it escalates if it goes wrong. The backbone, the Retail Agent Operating Model (RAOM), exists to tie every action to policy, evidence, and clear escalation paths. RAOM pairs a decision loop with a control plane. The decision loop is this chapter's loop with its steps spelled out (observe, orient, decide, act, measure, learn), and the control plane is the set of policies, tool contracts, budgets, and monitoring that keeps the loop safe. Observe is the Perceive step, orient and decide divide Reason between them, act is Act, and measure and learn divide Learn.

Across the book, RAOM is expressed through three planes, a capability plane (models, tools, and data), a control plane (policies, evaluation, approvals), and an integration plane (events, state, and handoffs). The six layers list what a system is built from, and the planes ask what each part is for: powering the decision loop, keeping it safe, or connecting it to retail systems. In the listing's agent, the stock reading and the reorder rule sit in the capability plane, a real order to a supplier would travel on the integration plane, and the control plane is missing. RAOM and its planes recur in the implementation and operations chapters, where systems are built and governed.

1.3.3 Applications of Agentic AI in Retail

RAOM governs how an agent acts, and the next question is where acting creates value. Retail already uses **Artificial Intelligence (AI)** to analyze and suggest, and what agents change is the last step, the move from analysis and content generation to actions executed inside policy. Shelf-scanning robots that detect on-shelf gaps and chatbots that handle customer queries become agents once they act on what they find. **Agentic AI for Retail** chains decisions across steps, touching every stage of the customer journey from Awareness and Consideration through Purchase, Service, and Loyalty. Five application areas stand out.

- **Autonomous Shopping Assistants:** One of the most visible applications is the customer-facing shopping assistant. Like a personal shopper, it can guide customers through product selection, answer complex queries, find an item across stores, compare prices, and place an order. Unlike a static chatbot, it can offer recommendations unprompted and handle multi-step tasks such as checkout with saved payment details. Such **autonomous shopping agents** can already research products and support purchases with the user's confirmation, as ChatGPT agent demonstrated in July 2025 (OpenAI 2025g). They can operate 24/7, serve many customers at once, and learn preferences over time.
- **Dynamic Pricing and Merchandising:** Prices, promotions, and placements change within commitments the agent does not set, such as advertised prices, promotion terms, and the margin plan, and optimization problems like these suit agents

Sample - Not for Redistribution

well. An **AI pricing agent** could analyze competitor prices, supply levels, demand signals, and events to adjust pricing in real time, in channels where prices can change that fast. Agents can also manage merchandising tasks by monitoring performance, experimenting with placement or recommendations, and rolling out changes that improve outcomes. Autonomy opens the door to **automated promotion engines**, **planogram optimizers** (a planogram is a shelf layout plan), and **assortment planners**. Set separate approved decision windows for digital repricing, promotions, markdowns, seasonal buys, and shelf resets. Continuous monitoring should then authorize changes only within each workflow's schedule, commitments, and approvals. A March 2025 vendor article by Mike Troy of SymphonyAI names planogram analysis, assortment optimization, pricing compliance across channels, and competitive product analysis as top merchandising use cases (Troy 2025).

- **Inventory Management and Supply Chain Optimization:** PwC's September 2026 guidance describes agents that **automate narrowly scoped supply-chain decisions** within established decision boundaries, given trusted data and clear accountability (PwC 2026). An inventory agent might detect faster-than-expected sales in one region, predict a stockout, and trigger a transfer or replenishment. It can also coordinate with a pricing agent to manage demand until new stock arrives. In the supply chain, agents can route shipments dynamically, switch to an approved backup supplier with confirmed capacity if a disruption is detected, or reschedule deliveries based on real-time logistics data. A longer-term aim is a **self-regulating supply chain**, where agents balance supply and demand with minimal human intervention. The intended payoff is less overstock, fewer stockouts, and better availability.
- **Customer Service:** Beyond conventional chatbots, **agentic customer service agents** can handle complex service workflows. If a customer reports a defective product, an agent could verify the purchase, check warranty terms, initiate a replacement shipment, and issue a return label in one interaction, faster than manual approvals allow. Such agents can pull order histories and issue refunds through tools (Schluntz and Zhang 2024), and their effect on resolution rates and customer satisfaction should be measured in each deployment.
- **Marketing Automation:** An agent can automate campaign management, personalizing content for customer segments, scheduling and launching campaigns, and adjusting strategies based on performance data. A marketing agent might detect that an email promotion is underperforming, for example, and autonomously A/B test a new message or switch the offer for a subset of customers, within the funding, advertised terms, and channel rules already agreed. Handing these tasks to an agent frees marketers to focus on strategy.

Beyond these five, notable retail applications include **fraud detection and finance** (agents that monitor transactions in real time and act on suspicious activity), **store operations** (agents that manage workforce scheduling or maintenance tasks), and **product design/R&D** (agents that analyze customer feedback and coordinate rapid prototyping). Not every repetitive or data-intensive process needs an agent, though. Where the steps are fixed, use a workflow and compare alternatives on cost and auditability. An agent earns its place where choosing the next step from changing evidence adds measurable value, the goals are clear, and governance is in place.

Sample - Not for Redistribution

New product launch evaluation shows where that line falls. Done manually, analyzing a batch of launches across stores can take a team of analysts days. The same vendor article reports that an agent can automate the data pulls, run comparisons, identify underperformers, and draft a short brief, cutting that time from days to minutes (Troy 2025). That is the vendor's account, not independent evidence, and any gain depends on a well-scoped workflow and accessible data. The pulls and comparisons are fixed steps a workflow can run, and the agent's part is choosing what to check next. A fair comparison sets each launch against others of the same age, distribution, availability, and selling exposure, so before it treats weak sales as weak demand, the agent checks whether the product reached fewer stores or ran out of stock. Faster feedback helps teams adjust marketing or inventory sooner, while humans stay responsible for the final call.

Clear goals and working governance are easier to state than to meet. In my experience, two silent killers show up early in agent projects.

⚠ WARNING The two silent killers of agent projects

1. **Automation without a control plane**: actions ship faster than policy or auditability. Prevention is straightforward but non-negotiable. Every action path needs policy checks, trace IDs, and rollback hooks.
2. **Feedback blind spots**: teams ship agents without a real evaluation loop and miss slow drift. Prevention requires a minimal eval harness plus rollouts gated on key performance indicators (KPIs), even for small pilots.

In the loop's terms, the first is an Act step with nothing to trace an action or roll it back, and the second is a loop with no working Learn step. The listing's inventory agent has both faults by design, since it simulates its orders with no policy check or trace ID and its `learn()` is empty.

The safeguards that address both are forms of **governance and oversight**: human oversight for critical decisions, clear escalation paths, appropriate decision boundaries, transparency and auditability, continuous monitoring of AI decisions, a standing evaluation harness, and rollouts gated on KPIs. Retailers must ensure that agents follow ethical guidelines, comply with regulations (pricing agents should not engage in illegal price discrimination, for example), and maintain brand trust. How much autonomy an agent should have depends on the use case, the risks, the domain, and the impact its decisions and actions will have on the business, on people, and on the environment. Humans supervise the system and review or override decisions in ambiguous or high-stakes scenarios. Boundaries and escalation paths take concrete form as **guardrails**, such as a limit on the discount an agent can offer or a required human sign-off for unusual recommendations. For the listing's agent, they would be a cap on each order's quantity and spend and a human sign-off for unusual orders. For **transparency**, an agent should explain its reasoning, or at least be auditable, so its actions can be understood, improved, and checked for reliability and for data or bias issues that could affect its decisions.

The KPIs that gate a rollout still have to be chosen, and choosing them is the investment test that forecasts cannot supply.

Sample - Not for Distribution

IMPORTANT **The investment test**

Before funding a pilot, agree on four things:

- A baseline, measured the way the pilot will be measured
- An outcome the business already tracks, such as the analysis cycle for new product launches, operating cost, revenue in the selected use case, or customer satisfaction
- The implementation cost, including integration and the people who review the agent's actions
- A stop rule, the result at which the pilot ends

For an inventory agent like the listing's, the four might read as follows.

- **Baseline**: the product's current stockout and overstock rates.
- **Outcome**: a fall in both, with no order that breaks policy.
- **Cost**: the integration with the ordering system and the time of whoever reviews its orders.
- **Stop rule**: a stockout rate no better than the baseline after an agreed period.

The evidence is still emerging. As data infrastructure improves and AI models mature, agents can take on more routine decisions under guardrails.

1.4 Conclusion

Back in the store, the agent has ordered 65 units and stock is at 100 again. It ordered after the spike instead of before it, because a rule fixed at 50 cannot ask whether 50 is still right. So what makes a system an agent, and why should a retailer want one? **Agentic AI builds on predictive analytics and generative AI by adding autonomy.** An agent runs the Perceive–Reason–Act–Learn loop, and in retail that moves software from informing and suggesting to taking actions, governed by guardrails. It can decide what to do next from changing evidence without waiting for a human prompt at each step, and a retailer should want one where delay is costly and that choice adds measurable value.

Model quality is only part of the story, because the surrounding engineering of policies, tool contracts, observability, and human escalation matters just as much. What the inventory agent left out, controls on the Act step and a working Learn step, are the two silent killers of agent projects, and fixing them comes first. An agent acts safely only when every action is tied to policy, evidence, and an escalation path, which is the job of RAOM. Whether it pays off is then a matter of measurement against a baseline, cost, and stop rule, use case by use case. Agentic AI will play a growing role in retail, and responsible adoption will decide its long-term success.

The capstone is the listing's problem at realistic size, an inventory-risk workflow with supplier coordination, pricing protection, customer communication, and an audit trail added. Its data model and event schema are introduced in the implementation chapter in Part IV, and RAOM runs through the book on the way.

The listing's agent decided alone, and its decision record ends on what that leaves open. A retail deployment often runs several agents, including pricing and inventory agents that can act on the same product at nearly the same moment. How do you build agents like these so that they stay fast and still accountable? That question opens Chapter 2.

Sample Not for Redistribution

Exercises

1. (Check; objectives 1 and 3.) How do traditional AI, generative AI, and Agentic AI differ? How does autonomy change retail operations?
2. (Check; objective 2.) Describe the Perceive–Reason–Act–Learn loop with a retail inventory management example.
3. (Check; objective 1.) What four technology pillars enable Agentic AI and how does each contribute?
4. (Check; objective 3.) Identify three high-impact applications of Agentic AI in retail.
5. (Check; objective 2.) Outline the layered architecture for agentic systems. Why are monitoring and security essential?
6. (Apply; objective 2.) **Agent Design:** Extend `perceive()` in the InventoryAgent example to read a supplied seasonal reorder threshold, then demonstrate how different thresholds change the reorder decision.
7. (Apply; objective 2.) **Autonomous Loop Simulation:** Create a flowchart for an autonomous shopping assistant that helps customers find products and complete purchases.
8. (Check; objective 2.) **Data Requirements:** List essential data sources for a dynamic pricing agent and their contributions to decision-making.
9. (Analyze; objective 3.) **Use Case Analysis:** Analyze how Agentic AI could change a retail process. Outline benefits and challenges.
10. (Analyze; objectives 2 and 3.) **Architecture Blueprint:** Design a system architecture for a retail Agentic AI solution with appropriate guardrails.

Sample - Not for Redistribution