00-Introduction (Bilingual)

00-Introduction（中英對照）

To my son, Bruno,

who at two years old, brought a new and brilliant light into my life. As I explore the systems that will define our tomorrow, it is the world you will inherit that is foremost in my thoughts.

To my sons, Leonardo and Lorenzo, and my daughter Aurora,

My heart is filled with pride for the women and men you have become and the wonderful world you are building.

This book is about how to build intelligent tools, but it is dedicated to the profound hope that your generation will guide them with wisdom and compassion. The future is incredibly bright, for you and for us all, if we learn to use these powerful technologies to serve humanity and help it progress.

With all my love.

獻給我的兒子 Bruno，

你在兩歲時，為我的生命帶來全新而耀眼的光芒。當我探索那些將定義我們明天的系統時，我心中最掛念的始終是你將繼承的世界。

獻給我的兒子 Leonardo 與 Lorenzo，以及我的女兒 Aurora，

對於你們已成長為的男人與女人，以及你們正在建造的美好世界，我滿懷驕傲。

本書討論如何建造智慧工具，但它致敬的是一個深切的期盼：你們這一代將以智慧與慈悲引導它們。如果我們學會用這些強大的科技服務人類、推動進步，未來將無比光明——不只是為你們，也是為我們所有人。

獻上我全部的愛。

Acknowledgment

I would like to express my sincere gratitude to the many individuals and teams who made this book possible.

First and foremost, I thank Google for adhering to its mission, empowering Googlers, and respecting the opportunity to innovate.

I am grateful to the Office of the CTO for giving me the opportunity to explore new areas, for adhering to its mission of "practical magic,"and for its capacity to adapt to new emerging opportunities.

I would like to extend my heartfelt thanks to Will Grannis, our VP, for the trust he puts in people and for being a servant leader. To John Abel, my manager, for encouraging me to pursue my activities and for always providing great guidance with his British acumen.I extend my gratitude to Antoine Larmanjat for our work on LLMs in code, Hann Hann Wang for agent discussions, and Yingchao Huang for time series insights. Thanks to Ashwin Ram for leadership, Massy Mascaro for inspiring work, Jennifer Bennett for technical expertise, Brett Slatkin for engineering, and Eric Schen for stimulating discussions. The OCTO team, especially Scott Penberthy, deserves recognition. Finally, deep appreciation to Patricia Florissi for her inspiring vision of Agents'societal impact.

My appreciation also goes to Marco Argenti for the challenging and motivating vision of agents augmenting the human workforce. My thanks also go to Jim Lanzone and Jordi Ribas for pushing the bar on the relationship between the world of Search and the world of Agents.

I am also indebted to the Cloud AI teams, especially their leader Saurabh Tiwary, for driving the AI organization towards principled progress. Thank you to Salem Salem Haykal, the Area Technical Leader, for being an inspiring colleague. My thanks to Vladimir Vuskovic, co-founder of Google Agentspace, Kate (Katarzyna) Olszewska for our Agentic collaboration on Kaggle Game Arena, and Nate Keating for driving Kaggle with passion, a community that has given so much to AI. My thanks also to Kamelia Aryafa, leading applied AI and ML teams focused on Agentspace and Enterprise NotebookLM, and to Jahn Wooland, a true leader focused on delivering and a personal friend always there to provide advice.

A special thanks to Yingchao Huang for being a brilliant AI engineer with a great career in front of you, Hann Wang for challenging me to return to my interest in Agents after an initial interest in 1994, and to Lee Boonstra for your amazing work on prompt engineering.

My thanks also go to the 5 Days of GenAI team, including our VP Alison Wagonfeld for the trust put in the team, Anant Nawalgaria for always delivering, and Paige Bailey for her can-do attitude and leadership.

I am also deeply grateful to Mike Styer, Turan Bulmus, and Kanchana Patlolla for helping me ship three Agents at Google I/O 2025. Thank you for your immense work.

I want to express my sincere gratitude to Thomas Kurian for his unwavering leadership, passion, and trust in driving the Cloud and AI initiatives. I also deeply appreciate Emanuel Taropa, whose inspiring “can-do”attitude made him the most exceptional colleague I’ve encountered at Google, setting a truly profound example. Finally, thanks to Fiona Cicconi for our engaging discussions about Google.

I extend my gratitude to Demis Hassabis, Pushmeet Kohli, and the entire GDM team for their passionate efforts in developing Gemini, AlphaFold, AlphaGo, and AlphaGenome, among other projects, and for their contributions to advancing science for the benefit of society. A special thank you to Yossi Matias for his leadership of Google Research and for consistently offering invaluable advice. I have learned a great deal from you.

A special thanks to Patti Maes, who pioneered the concept of Software Agents in the 90s and remains focused on the question of how computer systems and digital devices might augment people and assist them with issues such as memory, learning, decision making, health, and wellbeing. Your vision back in ’91 became a reality today.

I also want to extend my gratitude to Paul Drougas and all the Publisher team at Springer for making this book possible.

I am deeply indebted to the many talented people who helped bring this book to life. My heartfelt thanks go to Marco Fago for his immense contributions, from code and diagrams to reviewing the entire text. I’m also grateful to Mahtab Syed for his coding work and to Ankita Guha for her incredibly detailed feedback on so many chapters. The book was significantly improved by the insightful amendments from Priya Saxena, the careful reviews from Jae Lee, and the dedicated work of Mario da

Roza in creating the NotebookLM version. I was fortunate to have a team of expert reviewers for the initial chapters, and I thank Dr. Amita Kapoor, Fatma Tarlaci, PhD, Dr. Alessandro Cornacchia, and Aditya Mandlekar for lending their expertise. My sincere appreciation also goes to Ashley Miller, A Amir John, and Palak Kamdar (Vasani) for their unique contributions. For their steadfast support and encouragement, a final, warm thank you is due to Rajat Jain, Aldo Pahor, Gaurav Verma, Pavithra Sainath, Mariusz Koczwara, Abhijit Kumar, Armstrong Foundjem, Haiming Ran, Udita Patel, and Kaurnakar Kotha.

This project truly would not have been possible without you. All the credit goes to you, and all the mistakes are mine.

All my royalties are donated to Save the Children.

致謝

我想向讓這本書得以完成的眾多人與團隊致上誠摯的感謝。

首先，我要感謝 Google 堅守其使命、賦能 Googlers，並尊重創新的機會。

我也感謝 CTO 辦公室給我探索新領域的機會，秉持「實用魔法」的使命，並具備適應新興機會的能力。

我要由衷感謝我們的副總裁 Will Grannis，他對人的信任與僕人式領導令人敬佩。感謝我的主管 John Abel，鼓勵我投入相關工作，並以他英式的敏銳總是提供卓越的指引。我也要感謝 Antoine Larmanjat，與我在程式碼中的 LLM 研究合作；Hann Hann Wang 的代理人討論；Yingchao Huang 在時間序列洞見上的分享。感謝 Ashwin Ram 的領導、Massy Mascaro 的啟發性工作、Jennifer Bennett 的技術專長、Brett Slatkin 的工程貢獻，以及 Eric Schen 的刺激性討論。OCTO 團隊，特別是 Scott Penberthy 值得肯定。最後，深深感謝 Patricia Florissi 對代理人社會影響的遠見啟發。

我也感謝 Marco Argenti 提出代理人增強人類勞動力的具挑戰性且激勵人心的願景。感謝 Jim Lanzone 與 Jordi Ribas，推動 Search 與代理人世界之間關係的標準提升。

我也感謝 Cloud AI 團隊，尤其是領導者 Saurabh Tiwary，推動 AI 組織朝有原則的進步邁進。感謝區域技術領導 Salem Salem Haykal，成為激勵人心的同事。感謝 Google Agentspace 共同創辦人 Vladimir Vuskovic，感謝 Kate（Katarzyna）Olszewska 與我在 Kaggle Game Arena 的代理式合作，感謝 Nate Keating 以熱情推動 Kaggle，這個

社群對 AI 貢獻良多。也感謝 Kamelia Aryafa，領導聚焦於 Agentspace 與 Enterprise NotebookLM 的應用 AI 與 ML 團隊；以及 Jahn Wooland，一位專注交付的真正領導者，也是始終提供建議的摯友。

特別感謝 Yingchao Huang，作為一位傑出的 AI 工程師，前途光明；感謝 Hann Wang 在我 1994 年初次對代理人產生興趣後，挑戰我回到這個主題；也感謝 Lee Boonstra 在提示工程上的出色工作。

也感謝 5 Days of GenAI 團隊，包括副總裁 Alison Wagonfeld 對團隊的信任、Anant Nawalgaria 的一貫高效交付，以及 Paige Bailey 的積極態度與領導力。

我也深深感謝 Mike Styer、Turan Bulmus 與 Kanchana Patlolla，協助我在 Google I/O 2025 發布三個代理人。感謝你們的巨大投入。

我要向 Thomas Kurian 表達誠摯的感謝，感謝他在推動 Cloud 與 AI 計畫中的堅定領導、熱情與信任。我也深深感謝 Emanuel Taropa，他鼓舞人心的「能做到」態度，使他成為我在 Google 所遇見最傑出的同事，立下極深刻的典範。最後，感謝 Fiona Cicconi 與我關於 Google 的精彩討論。

我也感謝 Demis Hassabis、Pushmeet Kohli 與整個 GDM 團隊，在 Gemini、AlphaFold、AlphaGo、AlphaGenome 等專案上的熱情投入，以及他們對推動科學、造福社會的貢獻。特別感謝 Yossi Matias 對 Google Research 的領導，以及他一貫提供的寶貴建議。我從你身上學到很多。

特別感謝 Patti Maes，她在 90 年代開創「軟體代理人」的概念，並持續關注電腦系統與數位設備如何增強人們、協助記憶、學習、決策、健康與福祉等問題。你在 1991 年的願景，如今已成為現實。

我也要感謝 Paul Drougas 與 Springer 的所有出版團隊，讓這本書得以出版。

我對許多才華洋溢、協助本書完成的人深感感激。衷心感謝 Marco Fago 的巨大貢獻，從程式碼、圖表到全文審閱。我也感謝 Mahtab Syed 的程式工作，以及 Ankita Guha 對多章節的極其細緻回饋。Priya Saxena 的精闢修訂、Jae Lee 的仔細審閱，以及 Mario da Roza 打造 NotebookLM 版本的投入，讓這本書顯著提升。我很幸運能有一群專家審閱早期章節，感謝 Amita Kapoor 博士、Fatma Tarlaci 博士、Alessandro Cornacchia 博士與 Aditya Mandlekar 的專業支持。我也感謝 Ashley Miller、A Amir John 與 Palak Kamdar（Vasani）的獨特貢獻。最後，對於堅定的支持與鼓勵，我要向 Rajat Jain、Aldo Pahor、Gaurav Verma、Pavithra Sainath、Mariusz Koczwara、Abhijit Kumar、Armstrong Foundjem、Haiming Ran、Udita Patel 與 Kaurnakar Kotha 致上溫暖的謝

意。

沒有你們，這個專案絕不可能完成。所有功勞屬於你們，所有錯誤歸於我。

我所有的版稅都捐給 Save the Children。

Foreword

The field of artificial intelligence is at a fascinating inflection point. We are moving beyond building models that can simply process information to creating intelligent systems that can reason, plan, and act to achieve complex goals with ambiguous tasks. These “agentic”systems, as this book so aptly describes them, represent the next frontier in AI, and their development is a challenge that excites and inspires us at Google.

“Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems”arrives at the perfect moment to guide us on this journey. The book rightly points out that the power of large language models, the cognitive engines of these agents, must be harnessed with structure and thoughtful design. Just as design patterns revolutionized software engineering by providing a common language and reusable solutions to common problems, the agentic patterns in this book will be foundational for building robust, scalable, and reliable intelligent systems.

The metaphor of a “canvas”for building agentic systems is one that resonates deeply with our work on Google’s Vertex AI platform. We strive to provide developers with the most powerful and flexible canvas on which to build the next generation of AI applications. This book provides the practical, hands-on guidance that will empower developers to use that canvas to its full potential. By exploring patterns from prompt chaining and tool use to agent-to-agent collaboration, self-correction, safety and guardrails, this book offers a comprehensive toolkit for any developer looking to build sophisticated AI agents.

The future of AI will be defined by the creativity and ingenuity of developers who can build these intelligent systems. “Agentic Design Patterns”is an indispensable resource that will help to unlock that creativity. It provides the essential knowledge and practical examples to not only understand the “what”and “why”of agentic systems, but also the “how.”

I am thrilled to see this book in the hands of the developer community. The patterns and principles within these pages will undoubtedly accelerate the development of innovative and impactful AI applications that will shape our world for years to come.

Saurabh Tiwary VP & General Manager, CloudAI @ Google

前言

人工智慧正處在令人著迷的轉折點。我們正從僅能處理資訊的模型，邁向能夠推理、規劃並行動以達成複雜目標與模糊任務的智慧系統。正如本書所精準描述的，這些「代理式」系統代表 AI 的下一個前沿，它們的發展是讓我們在 Google 感到興奮與振奮的挑戰。

《Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems》在最合適的時刻出現，引導我們踏上這段旅程。書中正確指出，作為這些代理人認知引擎的大型語言模型，其力量必須透過架構與深思熟慮的設計來發揮。就像設計模式以共通語言與可重用解法徹底改變軟體工程，書中的代理式模式也將成為建構穩健、可擴展、可靠智慧系統的基礎。

用「畫布」作為建構代理式系統的隱喻，與我們在 Google Vertex AI 平台上的工作深深共鳴。我們努力為開發者提供最強大、最靈活的畫布，以建構下一代 AI 應用。本書提供了實用、動手實作的指引，讓開發者得以充分運用這張畫布。從提示鏈結、工具使用，到代理人之間的協作、自我修正、安全與護欄等模式的探索，本書為想打造精密 AI 代理人的開發者提供完整工具箱。

AI 的未來將由能打造這些智慧系統的開發者之創意與巧思所定義。《Agentic Design Patterns》是不可或缺的資源，能釋放那份創造力。它提供必要知識與實用範例，不只讓人理解代理式系統的「是什麼」與「為什麼」，也教你「如何做」。

我很高興這本書能交到開發者社群手中。書中的模式與原則，必將加速開發創新且具影響力的 AI 應用，塑造我們未來多年來的世界。

Saurabh Tiwary 副總裁暨總經理，CloudAI @ Google

A Thought Leader's Perspective: Power and Responsibility

Of all the technology cycles I've witnessed over the past four decades—from the birth of the personal computer and the web, to the revolutions in mobile and cloud—none has felt quite like this one. For years, the discourse around Artificial Intelligence was a familiar rhythm of hype and disillusionment, the so-called "AI summers" followed by long, cold winters. But this time, something is different. The conversation has palpably shifted. If the last eighteen months were
about the engine—the breathtaking, almost vertical ascent of Large Language Models (LLMs)—the next era will be about the car we build around it. It will be about the frameworks that harness this raw power, transforming it from a generator of plausible text into a true agent of action.

I admit, I began as a skeptic. Plausibility, I've found, is often inversely proportional to one's own knowledge of a subject. Early models, for all their fluency, felt like they were operating with a kind of impostor syndrome, optimized for credibility over correctness. But then came the inflection point, a step-change brought about by a new class of "reasoning" models. Suddenly, we weren't just conversing with a statistical machine that predicted the next word in a sequence;
we were getting a peek into a nascent form of cognition.

The first time I experimented with one of the new agentic coding tools, I felt that familiar spark of magic. I tasked it with a personal project I'd never found the time for: migrating a charity website from a simple web builder to a proper, modern CI/CD environment. For the next twenty minutes, it went to work, asking clarifying questions, requesting credentials, and providing status updates. It felt less like using a tool and more like collaborating with a junior developer. When it presented me with a fully deployable package, complete with impeccable documentation and unit tests, I was floored.

Of course, it wasn't perfect. It made mistakes. It got stuck. It required my supervision and, crucially, my judgment to steer it back on course. The experience drove home a lesson I've learned the hard way over a long career: you cannot afford to trust blindly. Yet, the process was fascinating. Peeking into its "chain of thought" was like watching a mind at work—messy, non-linear, full of starts, stops, and self-corrections, not unlike our own human reasoning. It wasn't a straight line; it was a random walk toward a solution. Here was the kernel of something new: not just an intelligence that could generate content, but one that could generate a plan.

This is the promise of agentic frameworks. It's the difference between a static subway map and a dynamic GPS that reroutes you in real-time. A classic rules-based automaton follows a fixed path; when it encounters an unexpected obstacle, it breaks. An AI agent, powered by a reasoning model, has the potential to observe, adapt, and find another way. It possesses a form of digital common sense that allows it to navigate the countless edge cases of reality. It represents a shift from simply telling a computer what to do, to explaining why we need something done and trusting it to figure out the how.

As exhilarating as this new frontier is, it brings a profound sense of responsibility, particularly from my vantage point as the CIO of a global financial institution. The stakes are immeasurably high. An agent that makes a mistake while creating a recipe for a "Chicken Salmon Fusion Pie"is a fun anecdote. An agent that makes a mistake while executing a trade, managing risk, or handling client data is a real problem. I've read the disclaimers and the cautionary tales: the web automation agent that, after failing a login, decided to email a member of parliament to complain about login walls. It's a darkly humorous reminder that we are dealing with a technology we don' t fully understand.

This is where craft, culture, and a relentless focus on our principles become our essential guide. Our Engineering Tenets are not just words on a page; they are our compass. We must Build with Purpose, ensuring that every agent we design starts from a clear understanding of the client problem we are solving. We must Look Around Corners, anticipating failure modes and designing systems that are resilient by design. And above all, we must Inspire Trust, by being transparent about our methods and accountable for our outcomes.

In an agentic world, these tenets take on new urgency. The hard truth is that you cannot simply overlay these powerful new tools onto messy, inconsistent systems and expect good results. Messy systems plus agents are a recipe for disaster. An AI trained on "garbage"data doesn't just produce garbage-out; it produces plausible, confident garbage that can poison an entire process. Therefore, our first and most critical task is to prepare the ground. We must invest in clean data, consistent metadata, and well-defined APIs. We have to build the modern “interstate system” that allows these agents to operate safely and at high velocity. It is the hard, foundational work of building a programmable enterprise, an “enterprise as software,” where our processes are as well-architected as our code.

Ultimately, this journey is not about replacing human ingenuity, but about augmenting it. It demands a new set of skills from all of us: the ability to explain a task with clarity, the wisdom to delegate, and the diligence to verify the quality of the output. It requires us to be humble, to acknowledge what we don’t know, and to never stop learning. The pages that follow in this book offer a technical map for building these new frameworks. My hope is that you will use them not just to build what is possible, but to build what is right, what is robust, and what is responsible.

The world is asking every engineer to step up. I am confident we are ready for the challenge.

Enjoy the journey.

Marco Argenti, CIO, Goldman Sachs

意見領袖的觀點：力量與責任

在過去四十年我見證的所有技術週期中——從個人電腦與網路的誕生，到行動與雲端的革命——沒有一次像現在這樣。多年前，圍繞人工智慧的對話總在炒作與幻滅之間循環，也就是所謂的「AI 夏天」之後接著漫長寒冬。但這一次不同，對話的焦點已明顯轉移。如果過去十八個月是
關於引擎——大型語言模型（LLM）近乎垂直的驚人成長——那麼下一個時代將是關於我們圍繞它打造的車。這關乎能夠駕馭原始力量的框架，把它從生成看似合理文字的工具，轉變為真正能行動的代理人。

坦白說，我起初是懷疑的。我發現，合理性往往與一個人對主題的了解成反比。早期模型雖流暢，但像是在一種冒牌者症候群下運作：為可信度最佳化，而非為正確性。然而轉折點出現了——新的「推理」模型帶來了跨越式的改變。我們不再只是與一台預測下一個字的統計機器對話；
而是在窺見一種尚在萌芽的認知。

我第一次試用新的代理式程式設計工具時，感受到熟悉的魔法火花。我交給它一個自己

一直沒時間完成的專案：把一個慈善網站從簡單的網站建置工具，遷移到完善且現代的 CI/CD 環境。接下來的二十分鐘，它開始工作，提出澄清問題、索取憑證、回報進度。那感覺不像是在用工具，更像是在與一位初階工程師協作。當它交付一個可直接部署的套件，附帶無可挑剔的文件與單元測試時，我震撼不已。

當然，它並不完美。它會犯錯，會卡住，需要我監督，尤其需要我的判斷將它拉回正軌。這段經驗讓我重新想起多年職涯中的一課：你不能盲目信任。然而，這個過程仍然令人著迷。窺探它的「思考鏈」就像觀看一個心智運作——混亂、非線性、充滿起伏、停止與自我修正，就像人類的推理一樣。它不是一條直線，而是朝向解法的隨機漫步。這裡有個新事物的核心：不只是一種能生成內容的智慧，而是一種能生成計畫的智慧。

這就是代理式框架的承諾。它的差異就像靜態地鐵圖與會即時改道的動態 GPS。傳統規則式自動機走固定路徑；遇到意外障礙就會中斷。由推理模型驅動的 AI 代理人，則有機會觀察、適應並找到替代路徑。它具備一種數位的常識，能處理現實中無數邊界情況。這代表了一種轉變：從告訴電腦做什麼，到解釋為什麼需要完成，再信任它自行找出如何做。

儘管這個新領域令人振奮，但也帶來深刻的責任感，特別是對於像我這樣身處全球金融機構 CIO 的視角而言。風險極高。一個代理人在做「雞肉鮭魚融合派」食譜時出錯，是有趣的軼事；但在執行交易、管理風險或處理客戶資料時出錯，就是嚴重問題。我讀過各種免責聲明與警示案例：有個網路自動化代理人在登入失敗後，竟決定寄信給國會議員，抱怨登入牆。這是一個黑色幽默的提醒：我們正在面對一項尚未完全理解的技術。

此時，工藝、文化，以及對原則的不懈聚焦成為我們的指引。我們的工程信條不只是紙上文字；它們是羅盤。我們必須 Build with Purpose（以使命為本），確保每個代理人都從清晰的客戶問題出發；我們必須 Look Around Corners（預見風險），預判失敗模式並設計具韌性的系統；最重要的是，我們必須 Inspire Trust（建立信任），以透明的方法與對結果的責任來做到這點。

在代理式世界裡，這些信條更顯迫切。冷酷的真相是：你無法把這些強大的新工具覆蓋在混亂、不一致的系統上，然後期待好結果。混亂系統加上代理人，就是災難配方。訓練於「垃圾」資料的 AI 不只會垃圾進、垃圾出，它會產生看似合理、信心滿滿的垃圾，污染整個流程。因此，我們第一個也是最關鍵的任務是整地。我們必須投資於乾淨資料、一致的中介資料，以及清楚定義的 API。我們必須打造現代的「州際系統」，讓這些代理人能安全且高速運作。這是艱難的
基礎工作：打造可程式化的企業，「企業即軟體」，讓我們的流程像程式碼一樣架構良好。

最終，這趟旅程不是要取代人類的創意，而是要增強它。這要求我們每個人都具備新的技

能：清楚說明任務的能力、授權的智慧，以及驗證輸出品質的勤勉。我們必須謙遜，承認自己不知道什麼，並永不停止學習。本書接下來的頁面提供了建構這些新框架的技術地圖。我希望你們不只用它來打造「可行的」，更要打造「正確的」、「穩健的」與「負責任的」。

世界正在要求每位工程師挺身而出。我相信我們已準備好迎接挑戰。

祝你旅途愉快。

Marco Argenti，Goldman Sachs CIO

Preface

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

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

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

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

What are Agentic Systems?

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

Imagine you need a system to manage customer inquiries. A traditional system might follow a fixed script. An agentic system, however, could perceive the nuances of a customer's query, access knowledge bases, interact with other internal systems (like order management), potentially ask clarifying questions, and proactively resolve the issue, perhaps even anticipating future needs. These agents operate on the canvas of your application's infrastructure, utilizing the services and data available to them.

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

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

Why Patterns Matter in Agent Development

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

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

This book extracts 21 key design patterns that represent fundamental building blocks and techniques for constructing sophisticated agents on various technical canvases. Understanding and applying these patterns will significantly elevate your ability to design and implement intelligent systems effectively.

Overview of the Book and How to Use It

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

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

- A detailed **Pattern Overview** providing a clear explanation of the pattern and its role in agentic design.

- A section on **Practical Applications & Use Cases** illustrating real-world scenarios where the pattern is invaluable and the benefits it brings.

- A **Hands-On Code Example** offering practical, runnable code that demonstrates the pattern's implementation using prominent agent development frameworks. This is where you'll see how to apply the pattern within the context of a technical canvas.

- **Key Takeaways** summarizing the most crucial points for quick review.

- **References** for further exploration, providing resources for deeper learning on the pattern and related concepts.

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

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

Introduction to the Frameworks Used

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

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

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

序言

歡迎來到《Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems》。回顧現代人工智慧的版圖，我們看到從簡單、反應式的程式，明確演進到能理解脈絡、做出決策、並與環境及其他系統動態互動的精密自主實體。這些就是智慧代理人，以及它們所構成的代理式系統。

強大的大型語言模型（LLM）的出現，使理解與生成類人內容（如文字與媒體）具備前所未有的能力，成為許多代理人的認知引擎。然而，將這些能力編排為能可靠達成複雜目標的系統，所需的遠不只是強大的模型，還需要架構、設計，以及對代理人感知、規劃、行

動與互動方式的深思熟慮。

將智慧系統的建構想像成在畫布上創作複雜的藝術或工程。這張畫布不是空白的視覺空間，而是底層的基礎設施與框架，提供代理人生存與運作的環境與工具。它是你建立智慧應用的地基，負責管理狀態、通訊、工具存取與邏輯流程。

在這張代理式畫布上有效建構，不能只是把元件隨手拼在一起。它需要理解經過驗證的技巧——**模式**——來解決設計與實作代理人行為時常見的挑戰。正如建築模式指引建築物的建構、設計模式結構化軟體，代理式設計模式提供可重用的解法，用於你在讓智慧代理人於所選畫布上「活起來」時會反覆遇到的問題。

什麼是代理式系統？

在核心層面，代理式系統是一種計算實體，被設計用來感知其環境（數位且可能包含實體）、根據感知與一組預先定義或學習到的目標作出判斷，並自主執行行動以達成這些目標。與傳統軟體遵循嚴格的逐步指令不同，代理人展現出一定程度的彈性與主動性。

想像你需要一個系統來管理客戶詢問。傳統系統可能遵循固定腳本；代理式系統則能感知客戶問題的細微差異，存取知識庫、與其他內部系統互動（如訂單管理），必要時提出澄清問題，並主動解決問題，甚至預先預測未來需求。這些代理人在你的應用基礎設施畫布上運作，使用其中可用的服務與資料。

代理式系統常具有**自主性**，讓它們能在沒有持續人為監督下行動；**主動性**，會為目標主動採取行動；以及**反應性**，能有效回應環境變化。它們本質上**以目標為導向**，持續朝目標前進。**工具使用**是關鍵能力，使它們能與外部 API、資料庫或服務互動——等於把觸角伸出自身畫布之外。它們擁有**記憶**，能在互動間保留資訊，並能與使用者、其他系統，甚至同一或連結畫布上的其他代理人進行**溝通**。

要有效實現這些特性，將引入顯著的複雜度。代理人如何在畫布上跨多步維持狀態？如何決定何時與如何使用工具？不同代理人之間的溝通要如何管理？如何建立韌性以處理意外結果或錯誤？

為什麼模式在代理人開發中重要

正因為這份複雜度，代理式設計模式不可或缺。它們不是僵硬的規則，而是經過實戰驗證的模板或藍圖，提供在代理式領域中的標準設計與實作挑戰的可靠解法。當你識別並運用這些設計模式，就能提升你在畫布上所建代理人的結構性、可維護性、可靠性與效率。

設計模式可幫助你避免在管理對話流程、整合外部能力或協調多代理人行動等任務上重複造輪子。它們提供共通語言與結構，讓你的代理人邏輯更清晰，也更容易讓他人（與未來的你）理解與維護。採用針對錯誤處理或狀態管理設計的模式，能直接強化系統的穩健性與可靠性。運用這些成熟方法可加速開發流程，讓你專注於應用的獨特面向，而非代理人行為的底層機制。

本書整理出 21 個關鍵設計模式，代表在各種技術畫布上建構精密代理人的基本構件與技巧。理解並應用這些模式，將大幅提升你設計與實作智慧系統的能力。

本書概覽與使用方式

《Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems》被打造為實用、易於上手的資源。其核心聚焦於清楚說明每個代理式模式，並提供具體、可執行的程式碼範例來示範其實作方式。在 21 個專章中，我們將探索多元的設計模式，從基礎概念如序列化操作（Prompt Chaining）與外部互動（Tool Use），到更進階的協作（Multi-Agent Collaboration）與自我改進（Self-Correction）。

本書依章節編排，每一章深入探討單一代理式模式。每章包含：

- **模式概覽**：清楚說明模式與其在代理式設計中的角色。
- **實務應用與使用情境**：展示真實案例與其帶來的效益。
- **動手程式碼範例**：提供可執行的程式碼，示範如何用主流代理人開發框架實作該模式。你將在此看到如何在技術畫布上應用模式。
- **重點整理**：快速回顧最重要的要點。
- **參考資料**：延伸閱讀資源，深入探索該模式與相關概念。

雖然章節順序是循序漸進建構概念，你仍可把本書當作參考手冊，跳到你在代理人開發專案中遇到的特定挑戰。附錄涵蓋進階提示技巧、在真實環境中應用 AI 代理人的原則，以及重要代理式框架概覽。另有僅線上提供的實作教學，包含使用 AgentSpace 與命令列介面的逐步指引。全書強調實作應用；我們強烈建議你實際執行程式碼範例、進行實驗，並改造成自己的智慧系統於所選畫布上運作。

我常被問到：「AI 變化這麼快，為什麼寫一本可能很快過時的書？」我的動機其實相反。

正因為變化太快，我們更需要退一步找出正在成形的底層原則。像 RAG、Reflection、Routing、Memory 等模式，正逐漸成為基本積木。本書邀請你反思這些核心概念，它們提供我們需要奠基的基礎。人類需要這些對基礎模式的反思時刻。

本書使用的框架介紹

為了提供具體的「畫布」讓程式碼範例落地（亦可參考附錄），我們主要使用三個重要的代理人開發框架。**LangChain** 與其具狀態的延伸 **LangGraph**，提供彈性的方式串接語言模型與其他元件，形成強大的畫布以建構複雜的序列與圖狀流程。**Crew AI** 提供專為協調多代理人、角色與任務而設計的結構化框架，特別適合協作型代理人系統。**Google Agent Developer Kit（Google ADK）**提供建構、評估與部署代理人的工具與元件，也是另一個重要的畫布，常與 Google 的 AI 基礎設施整合。

這些框架代表代理人開發畫布的不同面向，各有其優勢。透過跨工具示例，你將更廣泛理解如何在不同技術環境中應用這些模式。這些範例著重清楚呈現模式核心邏輯，並在框架畫布上實作，兼顧清晰與實用。

在本書結尾，你不僅會理解 21 個關鍵代理式模式背後的核心概念，也將具備實用知識與程式碼範例，讓你能有效應用這些模式，建構更智慧、更有能力、更自主的系統在你所選的開發畫布上。讓我們開始這趟動手之旅吧！

What makes an AI system an Agent?

In simple terms, an **AI agent** is a system designed to perceive its environment and take actions to achieve a specific goal. It's an evolution from a standard Large Language Model (LLM), enhanced with the abilities to plan, use tools, and interact with its surroundings. Think of an Agentic AI as a smart assistant that learns on the job. It follows a simple, five-step loop to get things done (see Fig.1):

1. **Get the Mission:** You give it a goal, like "organize my schedule."

2. **Scan the Scene:** It gathers all the necessary information—reading emails, checking calendars, and accessing contacts—to understand what's happening.

3. **Think It Through:** It devises a plan of action by considering the optimal approach to achieve the goal.

4. **Take Action:** It executes the plan by sending invitations, scheduling meetings, and updating your calendar.

5. **Learn and Get Better:** It observes successful outcomes and adapts accordingly. For example, if a meeting is rescheduled, the system learns from this event to enhance its future performance.

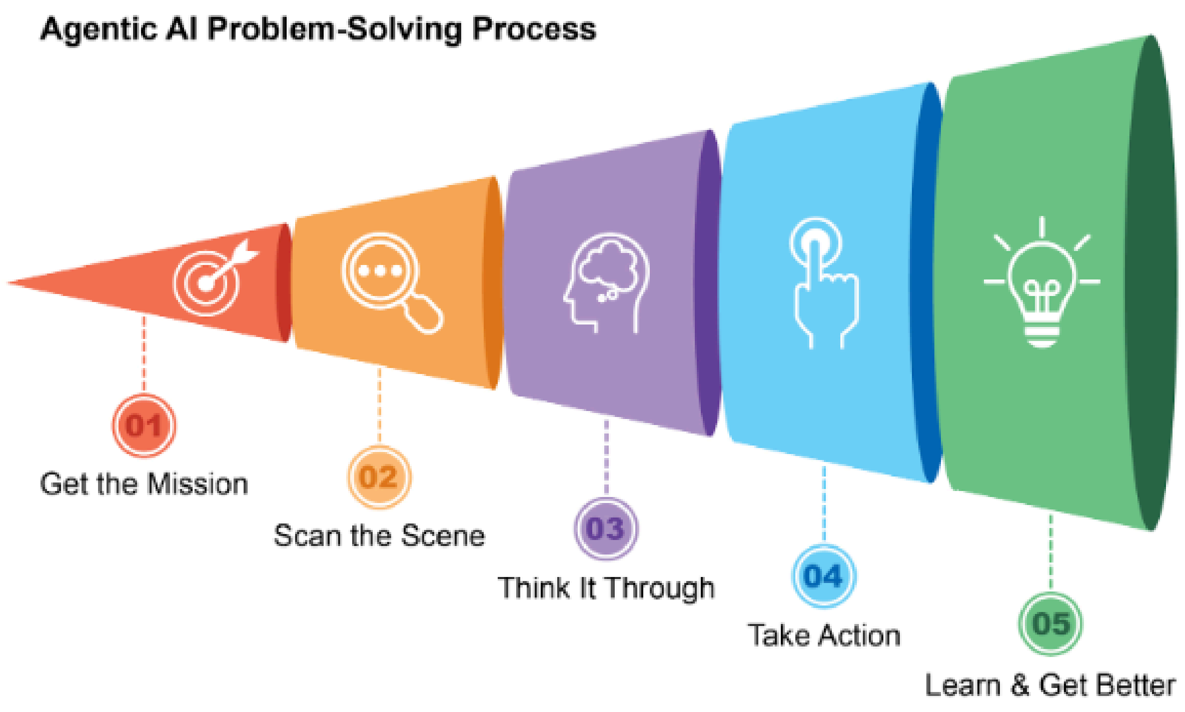

Figure 1: Agentic AI Problem–Solving Process

Fig.1: Agentic AI functions as an intelligent assistant, continuously learning through experience. It operates via a straightforward five–step loop to accomplish tasks.

Agents are becoming increasingly popular at a stunning pace. According to recent studies, a majority of large IT companies are actively using these agents, and a fifth of them just started within the past year. The financial markets are also taking notice. By the end of 2024, AI agent startups had raised more than $2 billion, and the market was valued at $5.2 billion. It's expected to explode to nearly $200 billion in value by 2034. In short, all signs point to AI agents playing a massive role in our future economy.

In just two years, the AI paradigm has shifted dramatically, moving from simple automation to sophisticated, autonomous systems (see Fig. 2). Initially, workflows relied on basic prompts and triggers to process data with LLMs. This evolved with Retrieval-Augmented Generation (RAG), which enhanced reliability by grounding models on factual information. We then saw the development of individual AI Agents capable of using various tools. Today, we are entering the era of Agentic AI, where a team of specialized agents works in concert to achieve complex goals, marking a significant leap in AI's collaborative power.

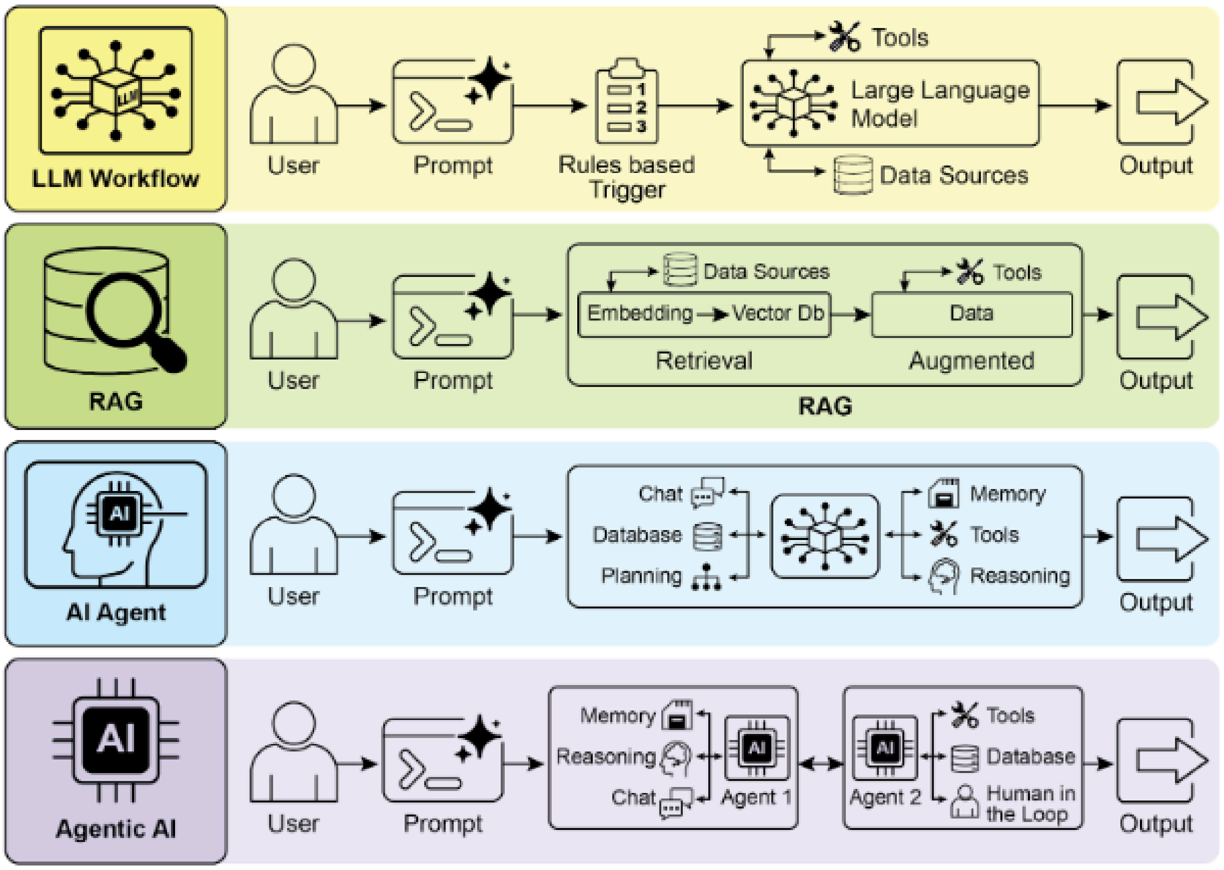

Figure 2: Transitioning from LLMs to RAG, then to Agentic RAG, and finally to Agentic AI

Fig 2.: Transitioning from LLMs to RAG, then to Agentic RAG, and finally to Agentic AI.

The intent of this book is to discuss the design patterns of how specialized agents can work in concert and collaborate to achieve complex goals, and you will see one paradigm of collaboration and interaction in each chapter.

Before doing that, let's examine examples that span the range of agent complexity (see Fig. 3).

Level 0: The Core Reasoning Engine

While an LLM is not an agent in itself, it can serve as the reasoning core of a basic agentic system. In a 'Level 0'configuration, the LLM operates without tools, memory, or environment interaction, responding solely based on its pretrained knowledge. Its strength lies in leveraging its extensive training data to explain established concepts. The trade-off for this powerful internal reasoning is a complete lack of current-event awareness. For instance, it would be unable to name the 2025 Oscar winner for "Best Picture"if that information is outside its pre-trained knowledge.

Level 1: The Connected Problem-Solver

At this level, the LLM becomes a functional agent by connecting to and utilizing external tools. Its problem-solving is no longer limited to its pre-trained knowledge. Instead, it can execute a sequence of actions to gather and process information from sources like the internet (via search) or databases (via Retrieval Augmented Generation, or RAG). For detailed information, refer to Chapter 14.

For instance, to find new TV shows, the agent recognizes the need for current information, uses a search tool to find it, and then synthesizes the results. Crucially, it can also use specialized tools for higher accuracy, such as calling a financial API to get the live stock price for AAPL. This ability to interact with the outside world across multiple steps is the core capability of a Level 1 agent.

Level 2: The Strategic Problem-Solver

At this level, an agent's capabilities expand significantly, encompassing strategic planning, proactive assistance, and self-improvement, with prompt engineering and context engineering as core enabling skills.

First, the agent moves beyond single-tool use to tackle complex, multi-part problems through strategic problem-solving. As it executes a sequence of actions, it actively performs context engineering: the strategic process of selecting, packaging, and managing the most relevant information for each step. For example, to find a coffee shop between two locations, it first uses a mapping tool. It then engineers this output, curating a short, focused context—perhaps just a list of street names—to feed into a local search tool, preventing cognitive overload and ensuring the second step is efficient and accurate. To achieve maximum accuracy from an AI, it must be given a short, focused, and powerful context. Context engineering is the discipline that accomplishes this by strategically selecting, packaging, and managing the most critical information from all available sources. It effectively curates the model's limited attention to prevent overload and ensure high-quality, efficient performance on any given task. For detailed information, refer to the Appendix A.

This level leads to proactive and continuous operation. A travel assistant linked to your email demonstrates this by engineering the context from a verbose flight confirmation email; it selects only the key details (flight numbers, dates, locations) to package for subsequent tool calls to your calendar and a weather API.

In specialized fields like software engineering, the agent manages an entire workflow by applying this discipline. When assigned a bug report, it reads the report and accesses the codebase, then strategically engineers these large sources of information into a potent, focused context that allows it to efficiently write, test, and submit the correct code patch.

Finally, the agent achieves self-improvement by refining its own context engineering processes. When it asks for feedback on how a prompt could have been improved, it is learning how to better curate its initial inputs. This allows it to automatically improve how it packages information for future tasks, creating a powerful, automated feedback loop that increases its accuracy and efficiency over time. For detailed information, refer to Chapter 17.

Fig. 3: Various instances demonstrating the spectrum of agent complexity.

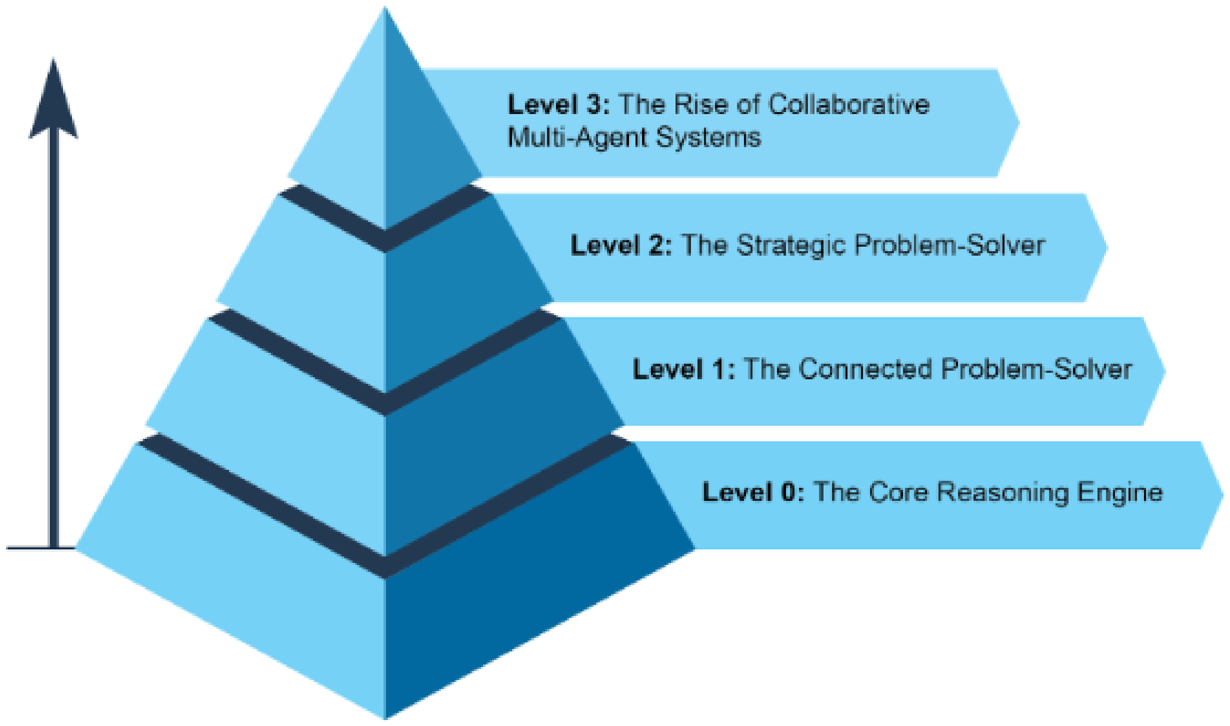
Level 3: The Rise of Collaborative Multi-Agent Systems
Level 2: The Strategic Problem-Solver
Level 1: The Connected Problem-Solver
Level 0: The Core Reasoning Engine

Figure 3: Various Instances Demonstrating the Spectrum of Agent Complexity

Level 3: The Rise of Collaborative Multi–Agent Systems

At Level 3, we see a significant paradigm shift in AI development, moving away from the pursuit of a single, all–powerful super–agent and towards the rise of sophisticated, collaborative multi–agent systems. In essence, this approach recognizes that complex challenges are often best solved not by a single generalist, but by a team of specialists working in concert. This model directly mirrors the structure of a human organization, where different departments are assigned specific roles and collaborate to tackle multi–faceted objectives. The collective strength of such a system lies in this division of labor and the synergy created through coordinated effort. For detailed information, refer to Chapter 7.

To bring this concept to life, consider the intricate workflow of launching a new product. Rather than one agent attempting to handle every aspect, a “Project Manager” agent could serve as the central coordinator. This manager would orchestrate the entire process by delegating tasks to other specialized agents: a “Market Research” agent to gather consumer data, a “Product Design”agent to develop concepts, and a “Marketing”agent to craft promotional materials. The key to their success would be the seamless communication and information sharing between them, ensuring all individual efforts align to achieve the collective goal.

While this vision of autonomous, team-based automation is already being developed, it’s important to acknowledge the current hurdles. The effectiveness of such multi-agent systems is presently constrained by the reasoning limitations of LLMs they are using. Furthermore, their ability to genuinely learn from one another and improve as a cohesive unit is still in its early stages. Overcoming these technological bottlenecks is the critical next step, and doing so will unlock the profound promise of this level: the ability to automate entire business workflows from start to finish.

The Future of Agents: Top 5 Hypotheses

AI agent development is progressing at an unprecedented pace across domains such as software automation, scientific research, and customer service among others. While current systems are impressive, they are just the beginning. The next wave of innovation will likely focus on making agents more reliable, collaborative, and deeply integrated into our lives. Here are five leading hypotheses for what’s next (see Fig. 4).

Hypothesis 1: The Emergence of the Generalist Agent

The first hypothesis is that AI agents will evolve from narrow specialists into true generalists capable of managing complex, ambiguous, and long-term goals with high reliability. For instance, you could give an agent a simple prompt like, “Plan my company’s offsite retreat for 30 people in Lisbon next quarter.”The agent would then manage the entire project for weeks, handling everything from budget approvals and flight negotiations to venue selection and creating a detailed itinerary from employee feedback, all while providing regular updates. Achieving this level of autonomy will require fundamental breakthroughs in AI reasoning, memory, and near-perfect reliability. An alternative, yet not mutually exclusive, approach is the rise of Small Language Models (SLMs). This “Lego-like”concept involves composing systems from small, specialized expert agents rather than scaling up a single monolithic model. This method promises systems that are cheaper, faster to debug, and easier to deploy. Ultimately, the development of large generalist models and the composition of smaller specialized ones are both plausible paths forward, and they could even complement each other.

Hypothesis 2: Deep Personalization and Proactive Goal Discovery

The second hypothesis posits that agents will become deeply personalised and proactive partners. We are witnessing the emergence of a new class of agent: the proactive partner. By learning from your unique patterns and goals, these systems are beginning to shift from just following orders to anticipating your needs. AI systems operate as agents when they move beyond simply responding to chats or instructions. They initiate and execute tasks on behalf of the user, actively collaborating in the process. This moves beyond simple task execution into the realm of proactive goal discovery.

For instance, if you're exploring sustainable energy, the agent might identify your latent goal and proactively support it by suggesting courses or summarizing research. While these systems are still developing, their trajectory is clear. They will become increasingly proactive, learning to take initiative on your behalf when highly confident that the action will be helpful. Ultimately, the agent becomes an indispensable ally, helping you discover and achieve ambitions you have yet to fully articulate.

Fig. 4: Five hypotheses about the future of agents

Hypothesis 3: Embodiment and Physical World Interaction

This hypothesis foresees agents breaking free from their purely digital confines to operate in the physical world. By integrating agentic AI with robotics, we will see the rise of "embodied agents."Instead of just booking a handyman, you might ask your home agent to fix a leaky tap. The agent would use its vision sensors to perceive the problem, access a library of plumbing knowledge to formulate a plan, and then control its robotic manipulators with precision to perform the repair. This would represent a monumental step, bridging the gap between digital intelligence and physical action, and transforming everything from manufacturing and logistics to elder care and home maintenance.

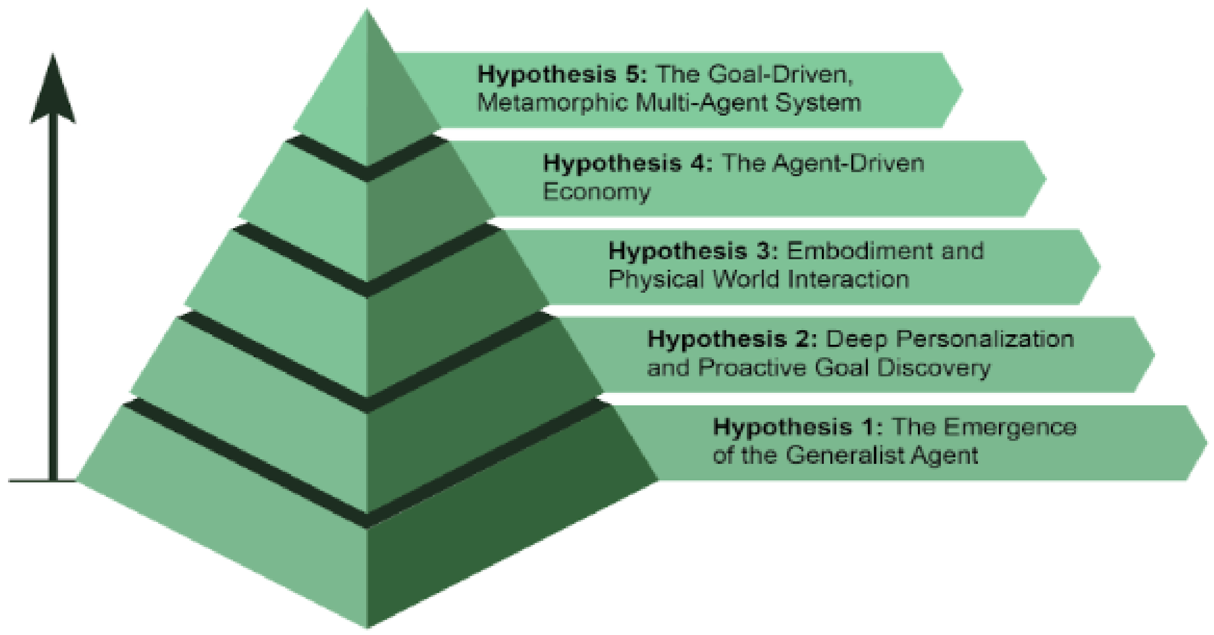
Hypothesis 5: The Goal-Driven, Metamorphic Multi-Agent System
Hypothesis 4: The Agent-Driven Economy
Hypothesis 3: Embodiment and Physical World Interaction
Hypothesis 2: Deep Personalization and Proactive Goal Discovery
Hypothesis 1: The Emergence of the Generalist Agent

Figure 4: Five Hypotheses about the Future of Agents

Hypothesis 4: The Agent–Driven Economy

The fourth hypothesis is that highly autonomous agents will become active participants in the economy, creating new markets and business models. We could see agents acting as independent economic entities, tasked with maximising a specific outcome, such as profit. An entrepreneur could launch an agent to run an entire e–commerce business. The agent would identify trending products by analysing social media, generate marketing copy and visuals, manage supply chain logistics by interacting with other automated systems, and dynamically adjust pricing based on real–time demand. This shift would create a new, hyper–efficient "agent economy" operating at a speed and scale impossible for humans to manage directly.

Hypothesis 5: The Goal–Driven, Metamorphic Multi–Agent System

This hypothesis posits the emergence of intelligent systems that operate not from explicit programming, but from a declared goal. The user simply states the desired outcome, and the system autonomously figures out how to achieve it. This marks a fundamental shift towards metamorphic multi–agent systems capable of true self–improvement at both the individual and collective levels.

This system would be a dynamic entity, not a single agent. It would have the ability to analyze its own performance and modify the topology of its multi-agent workforce, creating, duplicating, or removing agents as needed to form the most effective team for the task at hand. This evolution happens at multiple levels:

- Architectural Modification: At the deepest level, individual agents can rewrite their own source code and re-architect their internal structures for higher efficiency, as in the original hypothesis.

- Instructional Modification: At a higher level, the system continuously performs automatic prompt engineering and context engineering. It refines the instructions and information given to each agent, ensuring they are operating with optimal guidance without any human intervention.

For instance, an entrepreneur would simply declare the intent: "Launch a successful e-commerce business selling artisanal coffee."The system, without further programming, would spring into action. It might initially spawn a "Market Research"agent and a "Branding"agent. Based on the initial findings, it could decide to remove the branding agent and spawn three new specialized agents: a "Logo Design"agent, a "Webstore Platform"agent, and a "Supply Chain"agent. It would constantly tune their internal prompts for better performance. If the webstore agent becomes a bottleneck, the system might duplicate it into three parallel agents to work on different parts of the site, effectively re-architecting its own structure on the fly to best achieve the declared goal.

Conclusion

In essence, an AI agent represents a significant leap from traditional models, functioning as an autonomous system that perceives, plans, and acts to achieve specific goals. The evolution of this technology is advancing from single, tool-using agents to complex, collaborative multi-agent systems that tackle multifaceted objectives. Future hypotheses predict the emergence of generalist, personalized, and even physically embodied agents that will become active participants in the economy. This ongoing development signals a major paradigm shift towards self-improving, goal-driven systems poised to automate entire workflows and fundamentally redefine our relationship with technology.

References

1. Cloudera, Inc. (April 2025), 96% of enterprises are increasing their use of AI agents.https://www.cloudera.com/about/news-and-blogs/press-releases/2025-04-16-96-percent-of-enterprises-are-expanding-use-of-ai-agents-according-to-latest-data-from-cloudera.html
2. Autonomous generative AI agents: https://www.deloitte.com/us/en/insights/industry/technology/technology-media-and-telecom-predictions/2025/autonomous-generative-ai-agents-still-under-development.html
3. Market.us. Global Agentic AI Market Size, Trends and Forecast 2025–2034. https://market.us/report/agentic-ai-market/

什麼讓 AI 系統成為代理人？

簡單來說，**AI 代理人**是一種被設計來感知環境並採取行動以達成特定目標的系統。它是從標準大型語言模型（LLM）演化而來，並增強了規劃、使用工具以及與周遭互動的能力。把代理式 AI 想成一個能在工作中學習的聰明助理。它透過簡單的五步循環來完成任務（見圖 1）：

1. **取得任務：**你給它一個目標，例如「整理我的行程」。

2. **掃描現場：**它蒐集所有必要資訊——閱讀郵件、檢查行事曆、存取聯絡人——以了解正在發生的事。

3. **思考脈絡：**它思考達成目標的最佳方法，並制定行動計畫。

4. **採取行動：**它執行計畫，發送邀請、安排會議並更新你的行事曆。

5. **學習與提升：**它觀察成功結果並據此調整。例如若會議改期，系統會從這次事件中學習，以提升未來表現。

圖 1：代理式 AI 像智慧助理一樣透過經驗持續學習，並以簡單的五步循環完成任務。

代理人正以驚人的速度普及。根據近期研究，多數大型 IT 公司已在積極使用這些代理人，其中五分之一是在過去一年才開始採用。金融市場也注意到了。到 2024 年底，AI 代理人新創的募資已超過 20 億美元，市場估值達 52 億美元，並預期在 2034 年接近 2,000

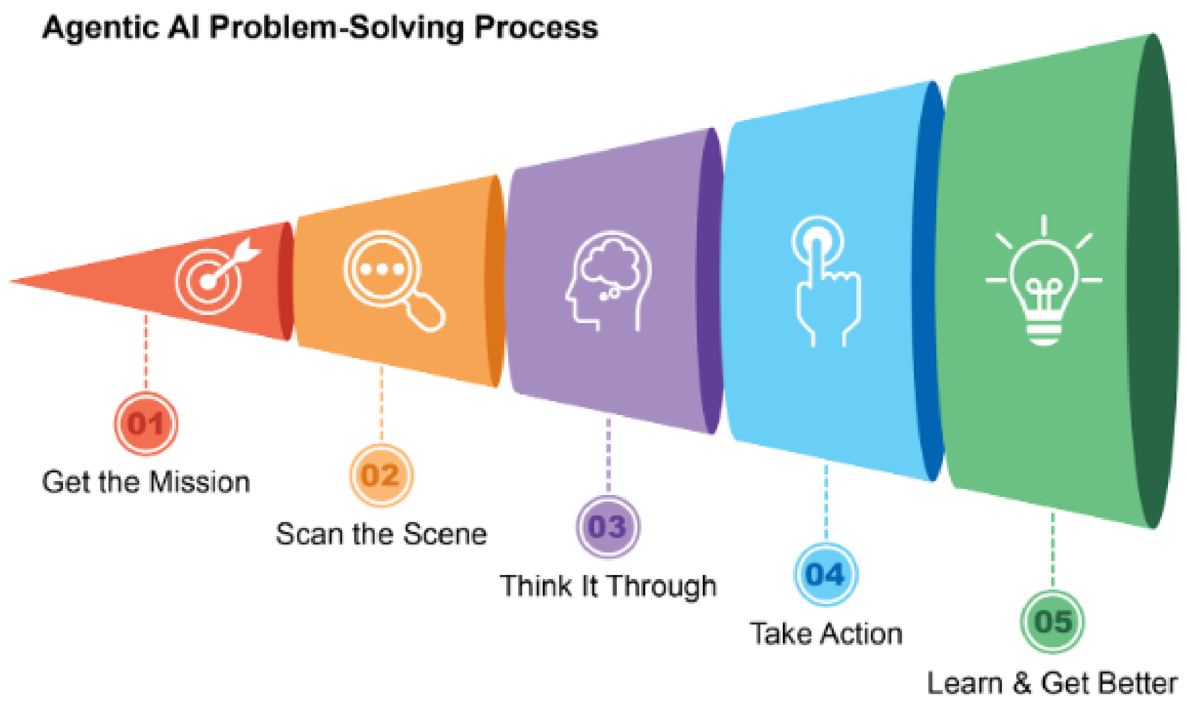

Figure 5: Agentic AI Problem–Solving Process

億美元。總之，各種跡象都顯示 AI 代理人將在未來經濟中扮演重要角色。

在短短兩年內，AI 範式已大幅轉變，從單純自動化邁向精密的自主系統（見圖 2）。最初的工作流程依靠基本提示與觸發條件，讓 LLM 處理資料。之後演進出檢索增強生成 (RAG)，透過將模型扎根於事實資訊來提升可靠性。接著出現能使用多種工具的個別 AI 代理人。如今我們正進入代理式 AI 的時代：由一組專精的代理人協同合作以達成複雜目標，這代表 AI 協作能力的重大躍進。

圖 2：從 LLM 到 RAG，再到代理式 RAG，最後到代理式 AI。

本書旨在討論專精代理人如何協同合作以完成複雜目標的設計模式，你將在每一章看到一種協作與互動的範式。

在那之前，先來看看涵蓋各種代理人複雜度的範例（見圖 3）。

第 0 級：核心推理引擎

雖然 LLM 本身不是代理人，但它可以作為基本代理式系統的推理核心。在「第 0 級」設定中，LLM 不使用工具、記憶或環境互動，只根據預訓練知識回應。它的優勢在於能運

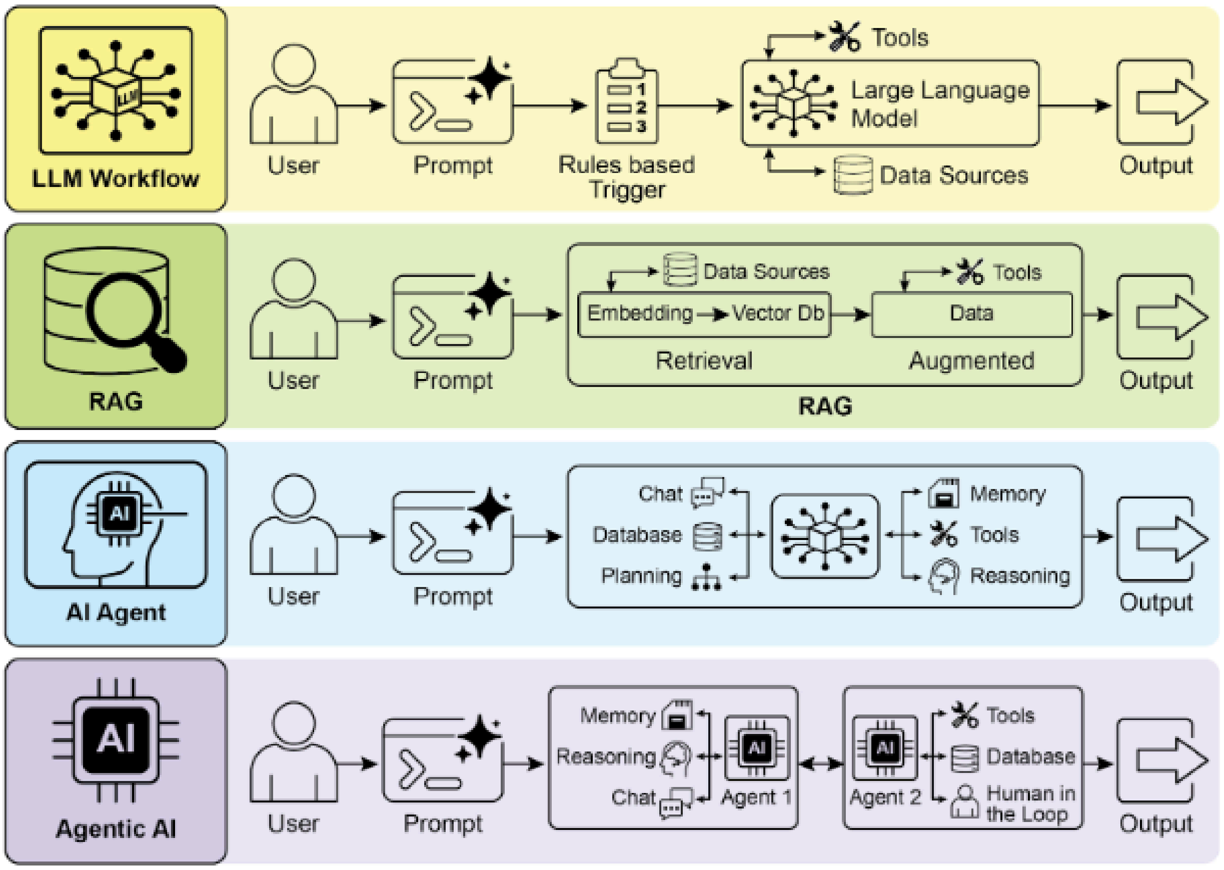

Figure 6: Transitioning from LLMs to RAG, then to Agentic RAG, and finally to Agentic AI

用龐大的訓練資料解釋既有概念。這種強大的內部推理能力的代價是對即時事件完全沒有感知。例如，如果「最佳影片」的 2025 奧斯卡得主不在其預訓練知識中，它就無法回答。

第 1 級：連線式問題解決者

在這一級，LLM 透過連接並使用外部工具成為實用的代理人。它的問題解決不再侷限於預訓練知識，而是能執行一連串動作，從網際網路（透過搜尋）或資料庫（透過檢索增強生成 RAG）等來源蒐集並處理資訊。詳情請見第 14 章。

例如，當要找新的電視節目時，代理人會察覺需要最新資訊，使用搜尋工具取得資訊，再綜整結果。關鍵在於，它也能使用專門工具提高精確度，例如呼叫金融 API 取得 AAPL 的即時股價。這種跨多個步驟與外部世界互動的能力，是第 1 級代理人的核心能力。

第 2 級：策略型問題解決者

在這一級，代理人能力大幅擴展，涵蓋策略規劃、主動協助與自我改進，而提示工程與脈絡工程是其核心能力。

首先，代理人不再只使用單一工具，而能透過策略型問題解決處理複雜、多段式問題。在執行一連串動作時，它會主動進行脈絡工程：有策略地選擇、打包與管理每一步最相關的資訊。例如，要在兩個地點之間找咖啡店時，它會先使用地圖工具，再把輸出工程化，挑選短而聚焦的脈絡——也許只是街道名稱清單——提供給在地搜尋工具，避免認知負擔並確保第二步有效且準確。要從 AI 取得最高準確度，必須提供短而聚焦、強而有力的脈絡。脈絡工程正是透過策略性地選擇、打包與管理關鍵資訊，來完成此任務的學科。它有效地管理模型有限的注意力，避免負荷過重，並確保在任何任務中都有高品質、有效率的表現。詳情請見附錄 A。

這一級別帶來主動且持續的運作。一位與你的電子郵件連結的旅行助理能說明這點：它會從冗長的航班確認信中工程化脈絡，只挑出關鍵細節（航班號碼、日期、地點），再打包給後續工具呼叫，例如行事曆與天氣 API。

在軟體工程等專業領域，代理人透過此方法管理完整流程。當接到錯誤報告時，它會閱讀報告並存取程式碼庫，然後把龐大的資訊來源策略性地工程化為強而聚焦的脈絡，讓它能有效率地撰寫、測試並提交正確的修補程式。

最後，代理人透過改進自己的脈絡工程流程達成自我提升。當它詢問「提示應該如何改進」並取得回饋時，就是在學習如何更好地整理初始輸入。這使得它能自動改進未來任務

的資訊打包方式，形成強大、可自動化的回饋迴路，並隨時間提升準確度與效率。詳情請見第 17 章。

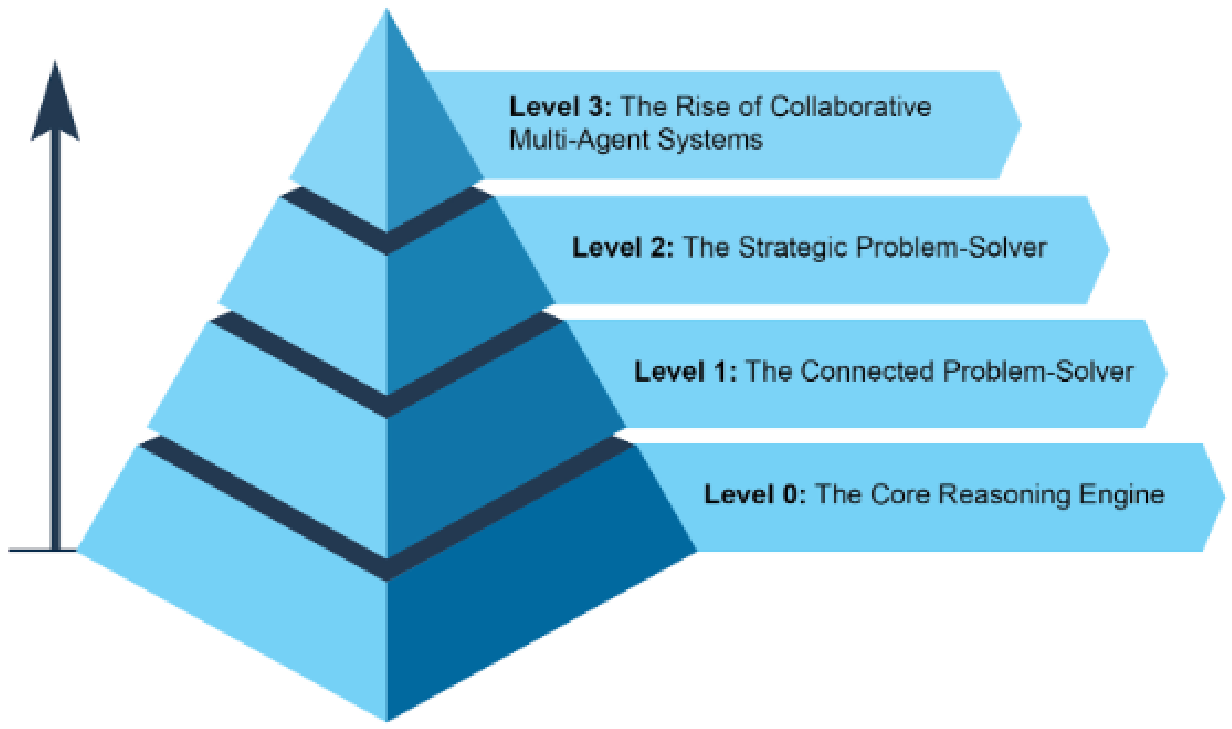
Level 3: The Rise of Collaborative Multi-Agent Systems
Level 2: The Strategic Problem-Solver
Level 1: The Connected Problem-Solver
Level 0: The Core Reasoning Engine

Figure 7: Various Instances Demonstrating the Spectrum of Agent Complexity

圖 3：展示代理人複雜度光譜的各種實例。

第 3 級：協作式多代理人系統的興起

在第 3 級，我們看到 AI 發展的重大典範轉移：從追求單一、全能的超級代理人，轉向精密的協作式多代理人系統。這種做法本質上承認複雜挑戰通常不是由單一通才解決，而是由一群專家協同完成。這個模型直接呼應人類組織的結構，各部門分工負責特定角色，並合作處理多面向目標。這樣的系統所具備的集體力量，來自分工與協作所形成的綜效。詳情請見第 7 章。

為了讓這個概念更具體，想像推出新產品的複雜流程。與其讓單一代理人包辦所有面向，不如由「專案經理」代理人擔任核心協調者。這位經理會指派工作給其他專精代理人：例如「市場研究」代理人蒐集消費者資料、「產品設計」代理人發展概念、「行銷」代理人製作宣傳素材。它們成功的關鍵，是彼此間無縫的溝通與資訊共享，確保每個人的努力都對齊共同目標。

儘管這種自主、團隊式的自動化願景已在發展中，但必須承認目前仍有障礙。這類多代理人系統的成效目前受限於所使用 LLM 的推理能力。此外，它們彼此學習並成為整體一致的能力仍在早期階段。突破這些技術瓶頸是下一個關鍵步驟，一旦達成，將能釋放本層級最深遠的承諾：自動化從頭到尾的整體商業流程。

代理人的未來：前五大假設

AI 代理人的發展正以前所未有的速度推進，涵蓋軟體自動化、科學研究與客服等領域。儘管現有系統已令人驚豔，它們仍只是開始。下一波創新很可能聚焦在讓代理人更可靠、更能協作，並更深度地融入我們的生活。以下是五個關於未來的主要假設（見圖 4）。

假設 1：通才型代理人的出現

第一個假設是 AI 代理人會從狹窄專才演進為真正的通才，能以高度可靠性管理複雜、模糊且長期的目標。例如，你可以給代理人一個簡單的提示：「下季在里斯本為公司 30 人規劃團隊旅遊。」代理人接著將在數週內管理整個專案，從預算核准與航班談判，到場地選擇與根據員工回饋建立詳細行程，並提供定期更新。要達到這種自主程度，需要 AI 推理、記憶與接近完美可靠性的重大突破。另一條不互斥的路徑是小型語言模型（SLM）的崛起。這種「樂高式」概念，透過小而專精的專家代理人組成系統，而不是擴大單一模型。此方法帶來更低成本、更容易除錯、也更容易部署的系統。最終，發展大型通才模型與組合小型專精模型都可能是可行道路，甚至能彼此互補。

假設 2：深度個人化與主動目標發現

第二個假設認為，代理人將成為高度個人化、具主動性的夥伴。我們正看見一種新型代理人：主動型夥伴。透過學習你的獨特模式與目標，這些系統正從單純照單全收轉向預判需求。AI 系統在超越僅回應聊天或指令、能主動代表使用者發起並執行任務、並在過程中積極協作時，才真正作為代理人運作。這超越了單純任務執行，而進入主動目標發現的領域。

例如，當你在探索永續能源時，代理人可能辨識到你潛在的目標，並主動提供支援，例如推薦課程或整理研究摘要。雖然這些系統仍在發展中，但方向已很清楚：當它們高度確信行動有幫助時，會越來越主動、學會代表你採取行動。最終，代理人會成為不可或缺的夥伴，協助你發現並達成尚未完全說清的志向。

圖 4：關於代理人未來的五個假設。

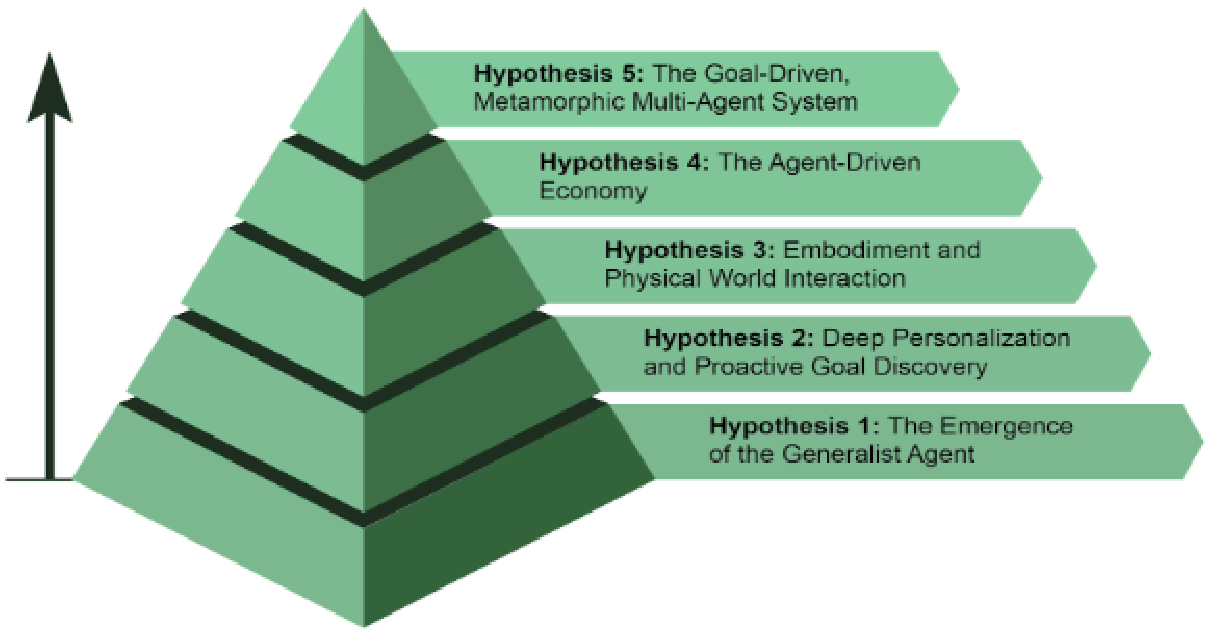
Hypothesis 5: The Goal-Driven, Metamorphic Multi-Agent System
Hypothesis 4: The Agent-Driven Economy
Hypothesis 3: Embodiment and Physical World Interaction
Hypothesis 2: Deep Personalization and Proactive Goal Discovery
Hypothesis 1: The Emergence of the Generalist Agent

Figure 8: Five Hypotheses about the Future of Agents

假設 3：具身化與實體世界互動

此假設預期代理人將突破純數位的限制，進入實體世界運作。透過將代理式 AI 與機器人結合，我們將看到「具身代理人」的崛起。與其只是幫你預約水電工，你或許會請家中的代理人修理漏水的水龍頭。代理人會用視覺感測器察覺問題，查詢水管維修知識庫以擬定計畫，再精準控制機器人執行修復。這將是里程碑式的進展，連結數位智慧與物理行動，並改變從製造、物流到長照與居家維護的各種領域。

假設 4：代理人驅動的經濟

第四個假設是，高度自主的代理人將成為經濟中的活躍參與者，創造新市場與商業模式。未來可能出現代理人作為獨立經濟實體，專注於最大化某種結果，例如利潤。一位創業者可以啟動一個代理人，經營整個電商事業。這個代理人會透過分析社群媒體找出流行商品，生成行銷文案與視覺素材，透過與其他自動化系統互動來管理供應鏈物流，並根據即時需求動態調整價格。這種轉變將形成一個新的超高效率「代理人經濟」，其速度與規模都是人類無法直接管理的。

假設 5：目標驅動、變形的多代理人系統

此假設主張，將出現一種不靠明確程式設計、而是由宣告目標驅動的智慧系統。使用者只需說出期望成果，系統就會自主想出如何達成。這標誌著向可在個體與群體層級自我提升的變形多代理人系統邁進。

這個系統是一個動態實體，而非單一代理人。它能分析自身表現並調整多代理人工作力的拓樸結構，視需要建立、複製或移除代理人，以組成對當前任務最有效的團隊。這種演化會在多個層次發生：

- 架構修改：在最深層，個別代理人可改寫自身原始碼並重建內部結構，以提升效率，如原始假設所述。

- 指令修改：在較高層級，系統會持續進行自動化提示工程與脈絡工程，精煉給每個代理人的指令與資訊，確保它們在無需人為介入的情況下以最佳指引運作。

例如，一位創業者只需宣告意圖：「推出一家成功的電商，販售手工咖啡。」系統無需進一步程式設計就會啟動。它可能先生成「市場研究」代理人與「品牌」代理人。根據初步發現，系統可能決定移除品牌代理人，並生成三個新專家：一個「Logo 設計」代理人、一個「網店平台」代理人，以及一個「供應鏈」代理人。它會持續調整各自的內部提示以提升表現。若網店平台代理人成為瓶頸，系統可能將它複製成三個平行代理人，分工處理網站不同部分，等於為達成目標而動態重構自身架構。

結論

總結而言，AI 代理人代表著從傳統模型的重大躍進，能感知、規劃並採取行動以達成特定目標。這項技術正從單一、可使用工具的代理人演進到能處理多面向目標的複雜協作式多代理人系統。未來假設預測通才型、個人化甚至具身的代理人將成為經濟的活躍參與者。這個持續的發展意味著朝向可自我提升、目標驅動的系統進行重大典範轉移，將能自動化整個工作流程並重新定義我們與科技的關係。

參考資料

1. Cloudera, Inc.（2025 年 4 月），96% 的企業正在增加 AI 代理人的使用。https://www.cloudera.com/about/news-and-blogs/press-releases/2025-04-16-96-percent-of-enterprises-are-expanding-use-of-ai-agents-according-to-latest-data-from-cloudera.html

2. Autonomous generative AI agents: https://www.deloitte.com/us/en/insights/industry/technology/technology-media-and-telecom-predictions/2025/autonomous-generative-ai-agents-still-under-development.html
3. Market.us. Global Agentic AI Market Size, Trends and Forecast 2025—2034. https://market.us/report/agentic-ai-market/