Introduction to Agents

A comprehensive guide for developers, architects, and product leaders transitioning from proofs-of-concept to robust, production-grade agentic systems.

Authors: Alan Blount, Antonio Gulli, Shubham Saboo, Michael Zimmermann, and Vladimir Vuskovic

From Predictive AI to Autonomous Agents

Artificial intelligence is evolving from passive, discrete tasks to autonomous problem-solving. We're witnessing a paradigm shift from AI that predicts or creates content to a new class of software capable of independent task execution.

An agent is not simply an AI model in a static workflow—it's a complete application that makes plans and takes actions to achieve goals. It combines a Language Model's ability to **reason** with the practical ability to **act**, handling complex, multi-step tasks autonomously.

Core Anatomy

Model, Tools, and Orchestration Layer

Taxonomy

From simple to collaborative systems

Production

Agent Ops for enterprise scale

What Is an AI Agent?

An AI Agent combines models, tools, an orchestration layer, and runtime services, using the LM in a loop to accomplish a goal. These four elements form the essential architecture of any autonomous system.

The Model

The core language model serving as the agent's central reasoning engine to process information, evaluate options, and make decisions.

Tools

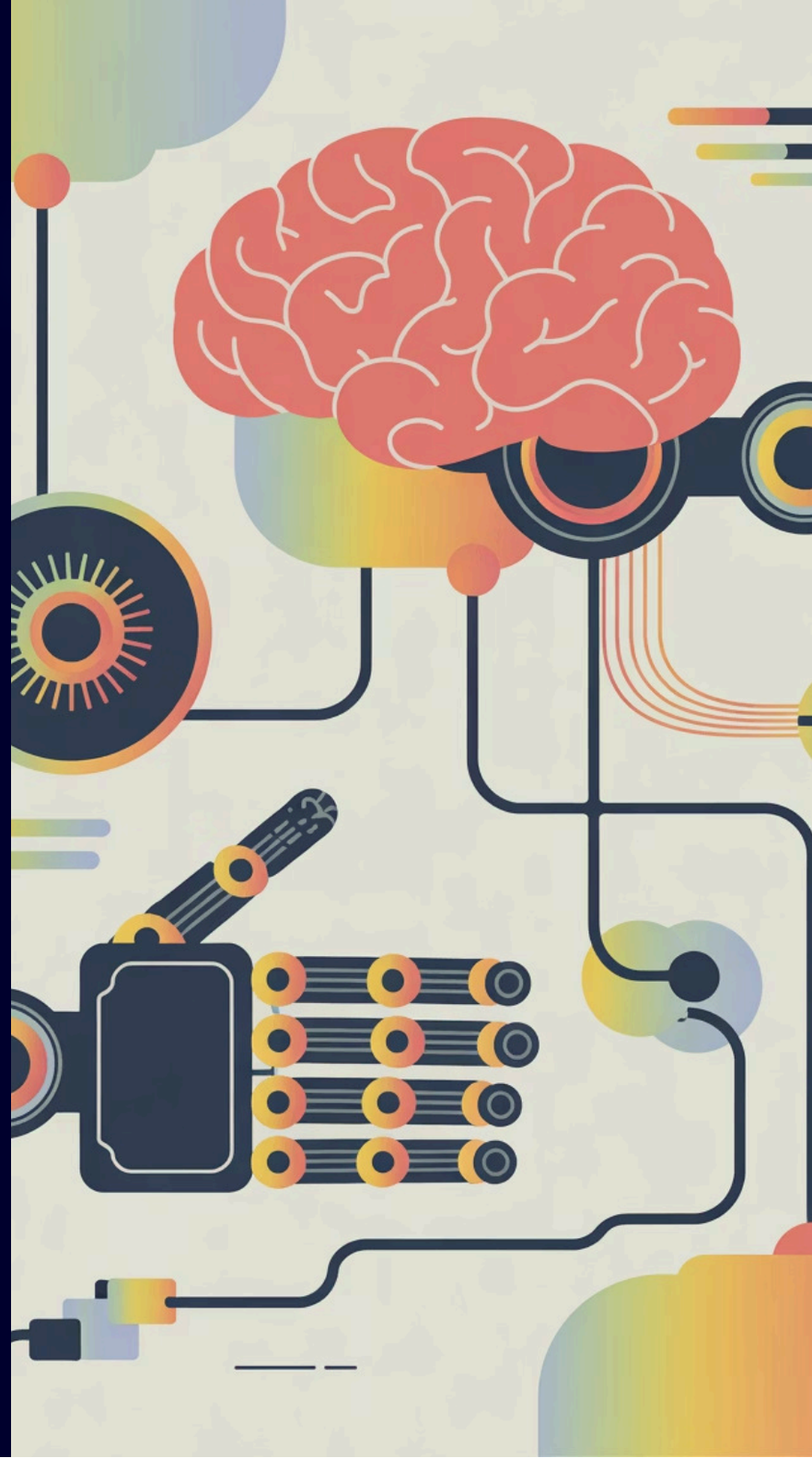
Mechanisms connecting the agent's reasoning to the outside world, including API extensions, code functions, and data stores.

Orchestration Layer

The governing process managing the agent's operational loop, handling planning, memory, and reasoning strategy execution.

Deployment

Production hosting on secure, scalable servers with essential monitoring, logging, and management services.



The Developer's Role Is Changing

Traditional Developer

Acts as a **"bricklayer"**, precisely defining every logical step with explicit code for every action.

Agent Developer

Acts as a **"director"**, setting the scene with instructions, selecting tools and APIs, and providing context to guide the autonomous actor.

An LM's greatest strength—its incredible flexibility—is also your biggest challenge. What we call "context engineering" guides LMs to generate desired outputs. Agents are software that manage the inputs of LMs to get work done.

The Agentic Problem-Solving Process

At its core, an agent operates on a continuous, cyclical process to achieve objectives. This loop can be broken down into five fundamental steps:

01

Get the Mission

The process is initiated by a specific, high-level goal from a user or automated trigger.

02

Scan the Scene

The agent perceives its environment, accessing available resources like user requests, memory, and tools.

03

Think It Through

The agent's core "think" loop analyzes the mission against the scene and devises a plan using chain of reasoning.

04

Take Action

The orchestration layer executes the first concrete step, invoking the appropriate tool or API.

05

Observe and Iterate

The agent observes the outcome, adds it to memory, and repeats the cycle until the mission is complete.

Real-World Example: Customer Support Agent

Let's see how a Customer Support Agent operates through the 5-step cycle when a user asks: *"Where is my order #12345?"*

1 Think: Create Strategy

Agent reasons: "I need to find the order, extract tracking number, query carrier API, and synthesize response."

2 Act: Find Order

Calls `find_order("12345")` tool and observes result with tracking number "ZYX987".

3 Act: Check Status

Calls `get_shipping_status("ZYX987")` tool and observes result: "Out for Delivery".

4 Report: Final Response

Generates message: "Your order #12345 is 'Out for Delivery!'"

A Taxonomy of Agentic Systems

Agentic systems can be classified into broad levels, each building on the capabilities of the last. Understanding this taxonomy helps architects scope what kind of agent to build.

Level 0: Core Reasoning

LM operates in isolation with pre-trained knowledge only

Level 1: Connected Problem-Solver

Reasoning engine connects to external tools and APIs

Level 2: Strategic Problem-Solver

Plans complex, multi-part goals with context engineering

Level 3: Collaborative Multi-Agent

Team of specialized agents working in concert

Level 4: Self-Evolving System

Autonomously creates new tools and agents

Level 0 & 1: From Isolation to Connection

Level 0: Core Reasoning System

The LM operates in isolation, responding solely based on vast pre-trained knowledge without tools, memory, or real-time awareness.

Strength: Can explain established concepts and plan approaches with great depth.

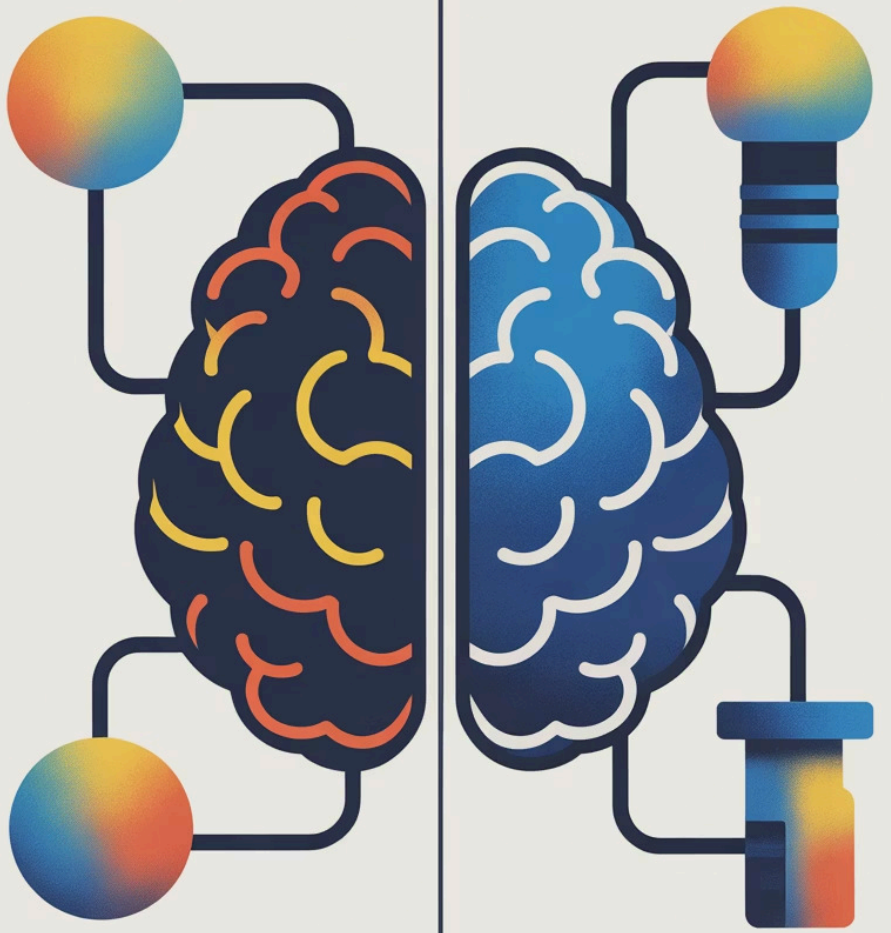
Limitation: Completely "blind" to events outside training data. Cannot answer "What was the Yankees game score last night?"

Level 1: Connected Problem-Solver

The reasoning engine becomes functional by connecting to external tools—the "Hands" component.

Capability: Can invoke Google Search API, financial APIs, or databases via RAG to access real-time information.

Example: Now can answer the Yankees question by searching for current game results.



Level 2: The Strategic Problem-Solver

Level 2 marks a significant expansion from executing simple tasks to strategically planning complex, multi-part goals. The key skill is context engineering—actively selecting and managing the most relevant information for each step.

Think & Act

Find halfway point between two addresses using Maps tool

Think & Act

Search for 4+ star coffee shops in Millbrae using focused query

Synthesize

Present "Millbrae Coffee" and "The Daily Grind" to user

This strategic planning enables proactive assistance, like reading a flight confirmation email, extracting key details, and automatically adding it to your calendar.

Level 3 & 4: Collaboration and Evolution

Level 3: Collaborative Multi-Agent System

A "team of specialists" working in concert, mirroring human organizations. A Project Manager agent delegates to MarketResearchAgent, MarketingAgent, and WebDevAgent—each handling specialized tasks for launching "Solaris" headphones.

Key Insight: Agents treat other agents as tools, enabling division of labor and collective strength.

Level 4: Self-Evolving System

The system identifies gaps in its own capabilities and dynamically creates new tools or agents. The Project Manager realizes it needs social media monitoring, invokes AgentCreator, and a new SentimentAnalysisAgent is created on the fly.

Key Insight: The system evolves from using fixed resources to actively expanding them—a truly learning organization.

Core Agent Architecture: The Model

The LM is the reasoning core, and its selection is critical. Real-world success demands a model that excels at **agentic fundamentals**: superior reasoning for complex problems and reliable tool use.

Define the Problem

Start by defining the business problem, then test models against metrics that directly map to that outcome.

Balance Trade-offs

Cross-reference quality with cost and latency. The "best" model sits at the optimal intersection for your specific task.

Use Multiple Models

Route complex reasoning to Gemini 2.5 Pro, simple tasks to Gemini 2.5 Flash for cost optimization.

The AI landscape evolves rapidly. Build a nimble "Agent Ops" practice with CI/CD pipelines that continuously evaluate new models, ensuring your agent always uses the best brain available.

Core Agent Architecture: Tools

Tools are the "hands" that connect reasoning to reality, enabling agents to retrieve information and take action. A robust tool interface is a three-part loop: defining capabilities, invoking them, and observing results.

Retrieval-Augmented Generation (RAG)

Query external knowledge from Vector Databases, Knowledge Graphs, or Google Search. Grounds the agent in fact, dramatically reducing hallucinations.

Natural Language to SQL

Query structured databases to answer analytic questions like "What were our top-selling products last quarter?"

API Integration

Wrap existing APIs as tools to send emails, schedule meetings, or update customer records—transforming the agent into an autonomous actor.

Code Execution

Write and execute code on the fly in secure sandboxes to generate SQL queries or Python scripts for complex calculations.

Human in the Loop (HITL)

Pause workflow to ask for confirmation or request specific information, ensuring people are involved in critical decisions.

Function Calling

Use OpenAPI specifications or Model Context Protocol (MCP) to provide structured contracts describing tool purpose, parameters, and responses.

The Orchestration Layer: The Nervous System

The orchestration layer is the central nervous system connecting the brain (model) and hands (tools). It runs the "Think, Act, Observe" loop and governs the agent's behavior.

Core Design Choices

Determine autonomy level: from deterministic workflows with LM as a tool, to LM in the driver's seat dynamically planning and executing.

Implementation Method

No-code builders for speed and accessibility, or code-first frameworks like Google's ADK for deep control and customization.

Production Framework

Must be open (any model/tool), provide precise control, and built for observability with detailed traces and logs.

The developer's most powerful lever is the **system prompt**—the agent's constitution containing domain knowledge, persona, constraints, rules of engagement, and explicit tool guidance.

Memory and Context Management

Short-Term Memory

The agent's active "scratchpad" maintaining the running history of the current conversation.

- Tracks sequence of (Action, Observation) pairs
- Provides immediate context for next decisions
- Implemented as state, artifacts, sessions or threads

Long-Term Memory

Provides persistence across sessions, typically implemented as a specialized RAG tool.

- Connected to vector database or search engine
- Agent can query its own history
- Remembers user preferences and past outcomes
- Enables personalized, continuous experience

Multi-Agent Design Patterns

As tasks grow complex, a "team of specialists" approach becomes more efficient. Proven agentic design patterns enable dynamic, scalable workflows.

Coordinator Pattern

A "manager" agent analyzes complex requests, segments tasks, and routes to specialist agents (researcher, writer, coder), then aggregates responses.

Sequential Pattern

Digital assembly line where output from one agent becomes direct input for the next—ideal for linear workflows.

Iterative Refinement

Generator agent creates content, critic agent evaluates against quality standards in a continuous feedback loop.

Human-in-the-Loop

Deliberate pause in workflow to get human approval before significant actions—critical for high-stakes tasks.

Agent Deployment and Services

After building a local agent, deploy it to a server where it runs continuously and can be accessed by users and other agents. Deployment is the "body and legs" of your agent.

Purpose-Built Platforms

Vertex AI Agent Engine supports runtime and services in one integrated platform for streamlined deployment.

Custom Infrastructure

Deploy agents in Docker containers on Cloud Run or GKE for direct control within existing DevOps infrastructure.

CI/CD Integration

Implement automated testing and deployment pipelines for secure, production-ready environments at scale.

Essential services include session history, memory persistence, logging, security measures for data privacy, and regulatory compliance.

Agent Ops: Managing the Unpredictable

Traditional software testing doesn't work with stochastic, agentic systems. Agent Ops is the disciplined approach to managing this new reality—an evolution of DevOps and MLOps tailored for AI agents.

Measure What Matters

Define KPIs that prove business value: completion rates, satisfaction scores, latency, cost, and impact on revenue.

Human Feedback

Convert user reports into new test cases, closing the loop and vaccinating against entire classes of errors.

LM as Judge

Use powerful models to evaluate quality against rubrics: correct answer, factual grounding, instruction following.

Metrics-Driven Development

Run new versions against evaluation datasets, compare scores to production, and use A/B deployments for safety.

Debug with Traces

OpenTelemetry traces provide step-by-step execution paths to diagnose root causes of issues.

Agent Interoperability: Connecting the Ecosystem

Agents must connect with humans, other agents, and commerce systems. This is the "face" of the agent—how it interacts with the wider world.

Agents and Humans

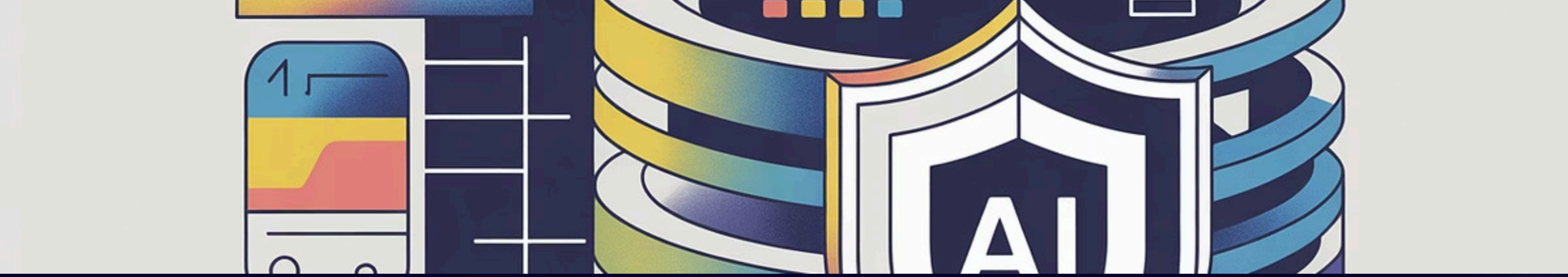
User interfaces from simple chatbots to rich dynamic experiences. Computer use enables agents to control UI. Live mode with Gemini Live API enables real-time, multimodal conversation with bidirectional streaming—speaking, listening, and seeing like humans.

Agents and Agents

Agent2Agent (A2A) protocol provides universal handshake for discovery and communication. Agent Cards advertise capabilities, endpoints, and credentials. Task-oriented architecture enables asynchronous, long-running collaboration between specialized agents.

Agents and Money

Agent Payments Protocol (AP2) uses cryptographically-signed mandates for verifiable proof of user intent. x402 protocol enables frictionless machine-to-machine micropayments. Together, these build the foundational trust layer for the agentic economy.



Security: The Trust Trade-Off

Every ounce of power granted to an agent introduces corresponding risk. The primary concerns are rogue actions and sensitive data disclosure. Success requires a hybrid, defense-in-depth approach.

Traditional Guardrails

Hardcoded rules acting as security chokepoints outside the model's reasoning.

- Policy engine blocking purchases over $100
- Requiring explicit user confirmation for external APIs
- Predictable, auditable hard limits on power

Reasoning-Based Defenses

Using AI to help secure AI through specialized guard models.

- Adversarial training for resilience to attacks
- Guard models examining proposed plans before execution
- Contextual awareness combined with rigid certainty

The Path Forward: Building the Future

Generative AI agents mark a pivotal evolution, shifting AI from passive tool to active, autonomous partner. This framework provides the foundation for production-grade agentic systems.

5	3	4
Levels of Capability	**Core Components**	**Design Patterns**
From core reasoning to self-evolving systems	Model, Tools, and Orchestration Layer	Coordinator, Sequential, Refinement, HITL

Success lies not in the initial prompt alone, but in engineering rigor applied to the entire system: robust tool contracts, resilient error handling, sophisticated context management, and comprehensive evaluation.

As this technology matures, this disciplined, architectural approach will be the deciding factor in harnessing the full power of agentic AI—building not just workflow automation, but truly collaborative, capable, and adaptable new members of our teams.