SoftEd North America | White Paper

Fundamentals of Agentic AI

How It Works and What It Means for Professionals

By David Mantica

Managing Director, SoftEd North America

2026

Executive Summary

Agentic AI is the next evolution of generative AI. Where a generative AI tool like ChatGPT, Claude, or Gemini responds to a single prompt from a human user, an agentic AI system operates autonomously over time. It perceives its environment, reasons about a goal, selects and calls tools, evaluates the results, and iterates until the work is done. The human is no longer the engine of the work. The human is the supervisor of work done by software.

This white paper is written for professionals who need to understand what agentic AI actually is, how it works under the hood, and how it will change their roles. It is grounded in the practical experience of building and running agentic systems, supplemented with technical detail verified against authoritative sources including Anthropic, Google Cloud, IBM, MIT Sloan, NVIDIA, AWS, and the MIT Project NANDA research.

The argument is simple. Agentic AI is not a replacement for generative AI. The two live together. Generative AI will continue to handle personal productivity. Agentic AI will handle autonomous, consistent, multi-step business processes. Both require human oversight, though of different kinds. The professionals who invest the time now to understand this shift — the loop, the stack, the economics, the guardrails, and the role implications — will be the ones shaping how it gets deployed in their organizations over the next three years.

1. What Agentic AI Actually Is

1.1 The Core Definition

An AI agent is a software program that uses a large language model (LLM) to achieve specific goals by observing its environment and iteratively acting upon it using the tools at its disposal. This definition matters because it frames every downstream decision: an agentic project is a software project first and an AI project second.

IBM defines agentic AI as an artificial intelligence system that can accomplish a specific goal with limited supervision, composed of AI agents that mimic human decision-making to solve problems in real time. Google Cloud describes it as an advanced form of AI focused on autonomous decision-making and action, distinguishing it from traditional AI that primarily responds to commands. MIT Sloan frames it as the next evolution of generative AI: semi- or fully autonomous systems able to perceive, reason, and act

on their own, integrating with other software systems to complete tasks independently or with minimal human supervision.

These definitions converge on four attributes that define an agent and distinguish it from a chatbot or a workflow:

- **Autonomy.** The agent operates without step-by-step human direction.
- **Reasoning.** The agent plans an approach and adjusts based on observed results.
- **Tool use.** The agent calls external systems, APIs, databases, and applications to act on the world.
- **Memory.** The agent retains context across steps and, in many systems, across sessions.

1.2 The Fundamental Shift: From Asking to Assigning

The cleanest way to understand agentic AI is to contrast it with the generative AI most professionals use daily. When you open ChatGPT or Claude and ask a question, you are the engine of the work. You decide what to ask, you interpret the answer, you decide what to do next. The model is reactive. It waits for you.

In an agentic system, the engine becomes code. The agent receives a goal, decides how to pursue it, takes action, checks the result, and iterates. You are no longer asking and receiving. You are assigning and overseeing. The model still sits at the heart of the system, but the orchestration happens in software that wraps around it.

This is the shift that matters. It changes the nature of the relationship between the professional and the tool — from conversation to delegation.

1.3 Where Agents Fit in the Evolution

Agentic AI is the fourth stage in a recognizable evolution of business automation:

Stage	Characteristics
Chatbots	Scripted responses, rule-based decision trees, no reasoning. Limited to the pre-defined questions they were built to answer.
RPA	Robotic Process Automation. Automates repetitive, deterministic tasks. Executes exactly what it was scripted to do with no ability to adjust to novel inputs.
AI Workflows	LLM-powered chains of steps designed by humans. Each step is pre-defined. The intelligence happens inside the steps, but the flow itself is fixed.
AI Agents	Autonomous goal-seeking systems. Dynamic planning. Tool use and memory. The flow itself is determined by the agent based on the goal and the current state.

The boundary between AI workflows and AI agents is blurring in practice. Many systems marketed as agents are really workflows with an LLM in the loop. The meaningful distinction is whether the system can plan its own steps or whether the steps are hardcoded.

2. How Agentic Workflows Operate

2.1 The Core Loop

Every well-designed agentic system runs a version of the same loop. NVIDIA, Google Cloud, IBM, and UiPath all document this loop with minor variations in terminology. The canonical four-step form:

- **Perceive.** The agent gathers information from its environment — documents, APIs, databases, tool outputs, previous agent messages. It builds a contextual understanding of the current state.
- **Reason.** An LLM serves as the reasoning engine. It interprets the goal, evaluates the current state, and plans the next action. Techniques like retrieval-augmented generation (RAG) are often used to pull in relevant context.
- **Act.** The agent executes a tool call — hitting an API, writing a file, sending an email, querying a database, or handing off to another agent. This is where the system interacts with the real world.
- **Evaluate.** The agent checks the outcome against the goal. If satisfied, it proceeds to the next task or terminates. If not, it loops back into perception with the new state and tries again.

The loop repeats until the goal is met, a stop condition is hit, or the agent escalates to a human. This is what is meant when vendors describe agents as "autonomous" — the loop runs without you telling it to run.

2.2 LLMs Versus Agents: A Technical Comparison

Feature	LLMs (Passive AI)	AI Agents (Agentic AI)
Interaction	Prompt → Response	Goal-directed, continuous reasoning
Initiative	Waits for user input	Initiates actions proactively
Task Scope	Single-turn tasks	Multi-step, multi-turn tasks
Memory	Typically stateless	Stateful with short- and long-term memory
Autonomy	Fully user-dependent	Capable of autonomous decision-making
Example	Summarize this document	Research a topic, summarize multiple sources, draft and send an email

> *Think of it this way: an LLM is a brilliant librarian. You ask a question and it gives you the best answer from its knowledge. An AI agent is a project manager who delegates, follows up, and gets the job done through the tools at its disposal.*

2.3 Memory and Feedback Loops

Memory is what distinguishes a workflow from a true agent. In a stateless LLM call, every prompt starts from scratch. In an agent, the system maintains context — short-term memory of the current task, and often long-term memory of past interactions, user preferences, and organizational knowledge.

This matters more than it sounds. The MIT Project NANDA research on enterprise AI adoption identified the lack of memory and learning as the single most significant factor separating failed deployments from successful ones. As one executive quoted in the MIT report put it: the tool "doesn't retain knowledge of client preferences or learn from previous edits. It repeats the same mistakes and requires extensive context input for each session." Memory is not a feature. It is a requirement for any agent intended to do meaningful work.

2.4 Tool Use and the Model Context Protocol

An agent is only as capable as the tools it can call. Tools are the hands of the agent — the APIs, functions, databases, and systems that let it affect the world. Historically, connecting an LLM to a tool required a custom integration for each pair of model and service, producing what Anthropic has called the "N×M integration problem": the combinatorial cost of integrating N different LLMs with M different tools.

In November 2024, Anthropic released the Model Context Protocol (MCP) as an open standard to solve this. MCP uses a client-server architecture and JSON-RPC 2.0 messaging. AI applications act as MCP clients; tool providers expose MCP servers. Servers expose three primitives — Tools (executable functions), Resources (structured data), and Prompts (templates). Clients expose two — Roots (filesystem access) and Sampling (requesting LLM completions from the client-side model).

MCP has since been adopted by major AI providers including OpenAI and Google DeepMind, and in December 2025 was donated to the Agentic AI Foundation under the Linux Foundation. Reference implementations exist in Python, TypeScript, C#, Java, Ruby, and others. For professionals, the practical significance is straightforward: MCP is becoming the standard way agents connect to enterprise systems. Understanding it — or at least knowing the term — is increasingly non-optional.

2.5 Multi-Agent Systems

A single agent is rarely the whole solution. Production agentic systems typically involve multiple specialized agents that coordinate with each other. Three common patterns:

- **Sequential.** Agent A completes a task and passes its output to Agent B, which passes to Agent C. Works when the workflow is linear and well-defined.
- **Hierarchical.** A supervisor or orchestrator agent delegates sub-tasks to specialist agents and integrates their outputs. This is the most common enterprise pattern.
- **Parallel.** Multiple agents work on different aspects of the same problem simultaneously, with their outputs merged at the end.

A practical example: a blog post production system might use a planning agent to draft an outline, a research agent to gather sources, a writing agent to produce a draft, and a review agent that critiques the draft and sends it back for revision. The planning agent acts as the supervisor, looping the system until the output meets its quality bar.

3. The Agentic Stack

Every agentic system runs on a layered stack. Understanding the stack is critical for any professional who will be scoping, evaluating, or governing an agentic project. The four layers, from top to bottom:

Tier	Layer	What It Does and Representative Tools
1	**The Agent**	The reasoning framework. Defines agent behavior, prompts, tool routing, and orchestration logic. Tools: Microsoft Copilot Studio, CrewAI, LangChain/LangGraph, AutoGen.
2	**Orchestration**	Low-code glue that connects agents to external applications. Tools: Zapier, n8n, Microsoft Power Automate.
3	**Model and AI Services**	The LLMs and agent-building services from the major AI providers. Tools: OpenAI Agents SDK, Google Agent Development Kit (ADK), Azure OpenAI Service.
4	**Cloud Infrastructure**	Compute, storage, identity, security, and data. Where everything actually runs. Providers: AWS, Microsoft Azure, Google Cloud.

3.1 The Agent Layer

The agent layer is where you define the behavior of the agent itself: its role, its tools, its memory strategy, and how it coordinates with other agents. The frameworks in this layer differ in their design philosophy and target audience.

- **LangChain and LangGraph.** The most widely adopted open-source framework. LangGraph, built on LangChain, models agent workflows as state machines with explicit nodes and edges. Used in production by Klarna, Uber, LinkedIn, JPMorgan, and hundreds of other companies. Steepest learning curve, but the most mature ecosystem for observability and production deployment.
- **CrewAI.** Role-based abstraction. You define agents with roles, goals, and tools, and organize them into "crews" that collaborate on tasks. Among the fastest paths from idea to working prototype, particularly suited to multi-agent systems that model human team structures.
- **Microsoft Copilot Studio.** Microsoft's native agent-building environment. Tight integration with Microsoft 365, Power Platform, and the broader Azure ecosystem. The default choice for organizations already standardized on Microsoft.
- **AutoGen.** Microsoft Research's multi-agent framework built around conversational collaboration between agents. Best suited when agents need to negotiate decisions through dialogue.

3.2 The Orchestration Layer

Orchestration platforms are the low-code bridge between agents and the thousands of applications they need to work with. These tools let non-developers prototype agentic workflows quickly, and they let developers avoid writing custom connectors for every SaaS tool.

- **Zapier.** The most widely adopted no-code automation platform, with an integration library in the thousands. In 2025 Zapier launched Zapier Agents — autonomous AI teammates that handle

multi-step tasks across apps — alongside Zapier Copilot for plain-language workflow creation. Best for non-technical users and simple-to-medium complexity workflows.

- **n8n.** Open-source, developer-oriented, self-hostable. Ships with dedicated AI agent nodes, native LangChain integration, and MCP support. Preferred when data privacy requires on-premises deployment or when workflows are complex enough to need code-level customization. Sits between pure no-code tools and full-code frameworks.
- **Microsoft Power Automate.** Microsoft's automation platform, tightly integrated with Copilot Studio, Teams, and the rest of the Microsoft 365 stack. Common in enterprise and government environments already standardized on Microsoft.

An important caveat: orchestration platforms excel at prototyping and internal automation. Production-grade, customer-facing agents that require per-user authentication, strict tenant isolation, and non-deterministic decision-making typically outgrow these platforms and migrate to code-first frameworks.

3.3 The Model and AI Services Layer

The large AI providers have each released their own agent-building services that sit above their models. These are not the frameworks from Tier 1 — they are the providers' native offerings designed for teams that want to build on a single vendor stack.

- **OpenAI Agents SDK.** OpenAI's production-ready framework for building agents. A successor to the experimental Swarm project, it provides minimal abstractions for agents, handoffs, tools, and guardrails. Tight integration with GPT-4o, o3, and other OpenAI models; supports non-OpenAI models with some configuration.
- **Google Agent Development Kit (ADK).** Announced at Google Cloud NEXT 2025, ADK is an open-source, code-first Python framework for building and deploying multi-agent systems. Optimized for Gemini but model-agnostic via LiteLLM. Ships with bidirectional audio and video streaming, native MCP support, and seamless deployment to Vertex AI.
- **Azure OpenAI Service.** Microsoft's managed service for OpenAI models deployed within Azure, with enterprise-grade security, compliance certifications (including FedRAMP), and data residency controls. Often the default for regulated industries and government agencies.

3.4 The Cloud Infrastructure Layer

Every agentic system ultimately runs on cloud infrastructure. This is not optional. The compute required to run LLMs at scale, the storage required for memory and retrieval, and the identity and security controls required for enterprise deployment all depend on a hyperscaler foundation. For government organizations, this layer is also where the compliance and data residency conversations land — FedRAMP authorization, StateRAMP, and agency-specific security frameworks all operate at the cloud layer.

> *Your agent is only as strong as the stack beneath it. The decisions you make about model provider, orchestration platform, and cloud vendor will shape what your agent can do, how much it costs to run, and whether it can pass your security review.*

4. Token Economics: The Hidden Cost Driver

Tokens are the fundamental unit of text processed by an LLM. Roughly one token equals three to four characters, or about three-quarters of a word. Every interaction with an LLM is measured in input tokens (what the system sends to the model) and output tokens (what the model returns). Commercial models charge per million tokens at rates that vary by model and provider.

For generative AI use, where a human issues one prompt at a time, token costs are modest. For agentic systems, where each task may trigger ten to thirty LLM calls in a single reasoning loop, token costs become the dominant line item. Gartner's 2026 analysis found that agentic models consume between five and thirty times more tokens per task than a standard generative AI chatbot.

The real-world implications are significant. Published case studies document enterprise pilots that cost roughly $50 in token spend during proof-of-concept and would have scaled to hundreds of thousands or millions of dollars per month at full production volume. One reported case saw a $500 pilot scale to $847,000 per month at production — a 717× increase. These are the pattern, not the exception.

4.1 What Drives Agentic Token Cost

- **Reasoning loops.** Each iteration of the perceive-reason-act-evaluate loop triggers at least one LLM call, often several.
- **Context accumulation.** Agents pass context forward between steps. Every tool output, every retrieved document, every intermediate decision becomes input tokens for the next call. Input token volume grows rapidly.
- **Tool descriptions.** When an agent has access to many tools, the descriptions of those tools are sent to the model on every call. Systems with hundreds of tools can consume hundreds of thousands of tokens in tool definitions alone before any work is done.
- **Reasoning tokens.** Modern reasoning models (OpenAI's o-series, Claude's extended thinking, Gemini's thinking mode) consume tokens for internal reasoning that never appears in the response. You still pay for them.
- **Multi-agent overhead.** Multi-agent systems multiply token consumption. Each agent's context, each handoff, each coordination message adds tokens.

4.2 Practical Implications for Government and Enterprise

Any organization planning to deploy agentic AI in production needs a token budget as explicit as its infrastructure budget. The economics will shift over time — per-token pricing has fallen consistently as models become more efficient, and providers are investing heavily in optimization techniques like prompt caching and semantic routing. But the underlying fact remains: agentic AI costs are variable, non-deterministic, and driven by usage in ways that traditional SaaS pricing is not. A CFO cannot look at an agentic project and ask "what does this cost per month?" without also asking "what is the expected token burn per task, and how many tasks will run per day?"

5. The Human-in-the-Loop Imperative

Agentic systems act autonomously. That autonomy is exactly why they require human oversight. The LLMs at the heart of these systems are non-deterministic — the same prompt can produce different outputs on different runs, and the system can take unexpected paths when the environment changes. In a low-stakes context, this is a feature. In a high-stakes context, it is a risk.

"Human in the loop" is the pattern that addresses this. The term originated in machine learning, where supervised training requires human review of model outputs, and has been extended in the agentic context to mean any architectural pattern that keeps a human involved in critical decision points of an autonomous system.

5.1 Why This Matters More in Government

- Public trust is the product. There is no margin for error when decisions affect citizens.
- Regulatory and compliance requirements are non-negotiable. Agents operate within the same legal boundaries as human employees.
- Decisions affect real citizens and real outcomes, often with limited recourse for those affected.
- Transparency and auditability are mandatory. Every agent decision must be explainable and reconstructable after the fact.

5.2 What Good Guardrails Look Like

A well-governed agentic system implements guardrails at multiple layers:

- **Approval gates.** Human sign-off is required before the agent takes any high-impact action — sending external communications, modifying records of record, committing financial transactions, publishing content.
- **Audit trails.** Every agent decision — every perception, every reasoning step, every tool call, every output — is logged in a form that can be inspected later. Observability is not optional.
- **Scope boundaries.** The agent is explicitly told what it can and cannot do: which tools it can call, which data sources it can access, which topics are out of scope.
- **Escalation paths.** Clear rules define when the agent must stop and hand off to a human. Low confidence in an output, unexpected errors, or actions outside its scope all trigger escalation.
- **Bias and quality monitoring.** Continuous evaluation of agent outputs for fairness, accuracy, and drift. This is an ongoing operational function, not a pre-launch checklist.

5.3 Where to Start: Internal Before External

The prudent path for any organization new to agentic AI — and especially for government agencies — is to deploy internally before deploying externally. Agents that analyze internal documents, triage internal tickets, or support internal research teams generate operational value without exposing citizens or customers to the risks of an autonomous system making mistakes in public. The internal track also generates the operational knowledge needed to govern external deployments responsibly when the time comes.

6. How Professional Roles Are Evolving

Agentic AI does not eliminate the roles in the software development lifecycle. It changes what those roles actually do. The pattern is consistent across roles: less time spent doing the work, more time spent directing, validating, and governing work done by agents.

Role	Today	With Agentic AI

Business Analysts	Manually gather requirements, document use cases, conduct stakeholder interviews.	Direct agents to analyze policies and source documents, draft requirements, then validate the output. Deeper focus on the "what" and "why" rather than producing the artifact.
Project Managers	Chase status updates, compile reports, manage budgets and schedules.	Set agent objectives, review auto-generated status reports, focus on decisions and risks. Manage token budgets alongside traditional P&L.
Developers	Write every line of code. Implement specifications from BAs.	Architect solutions. Use coding agents for generation and testing. Focus shifts to system design, integration, and code review of agent output.
IT Leadership	Approve projects, manage vendor relationships, set strategic direction.	Define governance frameworks, set agent scope boundaries, measure AI-driven outcomes, establish audit and compliance practices.

The deeper implication is a shift in what constitutes skilled work. The act of producing a requirements document, a code file, or a status report becomes less valuable. The act of thinking clearly about what needs to be built, directing the agent to build it, and verifying that it was built correctly becomes more valuable. For professionals, the practical advice is straightforward: spend time with these tools now, build fluency in directing agents, and develop the judgment required to validate their outputs.

7. Practical Starting Points

For professionals who want to move from awareness to capability, the path is practical and incremental.

Step 1: Build generative AI fluency. Before working with agents, become genuinely fluent with generative AI. Use Claude, ChatGPT, or Gemini as a first resort for tasks rather than a curiosity. Ethan Mollick, in *Co-Intelligence*, argues that you should try everything with AI first — this is the fastest way to develop the intuition required to work with agents.

Step 2: Take a structured fundamentals course. Agentic AI has enough moving parts that self-study alone is inefficient. A structured fundamentals course covering the loop, the stack, and the tooling shortens the learning curve significantly.

Step 3: Join user groups and communities. The field is moving quickly. User groups around specific tools — Zapier, n8n, LangChain, CrewAI — are where practitioners share working patterns. The wisdom of a practicing community will outpace any single guru or vendor pitch.

Step 4: Automate one small thing. The fastest way to build real understanding is to build a working system. Start with something small and low-risk — email triage, meeting note summarization, a simple

research assistant — using a low-code platform like n8n or Zapier. The friction points you encounter will teach you more than any reading.

Step 5: Protect time for practice. Individual and organizational fluency with these tools comes from deliberate practice. Blocking recurring time to experiment is what separates the people who become capable from the people who remain curious.

8. Key Takeaways

1. Agents are not chatbots. They plan, reason, call tools, and iterate autonomously toward goals. Understanding this shift is the foundation for everything else.

2. The agentic loop is the engine. Perceive, reason, act, evaluate. Every production agentic system runs a version of this loop, and understanding it is the basis for scoping and governing agentic projects.

3. The stack matters. Agent framework, orchestration, model service, and cloud infrastructure. Decisions at any layer constrain decisions at the others.

4. Token economics are real. Agentic systems consume tokens at multiples of what generative AI consumes. Budget accordingly.

5. Human-in-the-loop is non-negotiable. In government and regulated contexts especially, approval gates, audit trails, scope boundaries, and escalation paths are prerequisites for deployment, not afterthoughts.

6. Professional roles are evolving, not disappearing. The work shifts from producing artifacts to directing and validating agent-produced artifacts. The value moves upstream to thinking clearly about what needs to be built.

7. Start internal, stay practical. Deploy agentic AI on internal, low-stakes processes first. Build organizational competence before exposing citizens, customers, or business-critical workflows to autonomous systems.

> *The goal is not to replace people. The goal is to free people up to do what only people can do — think, create, adjust, test, and iterate in ways that no agent can, at least not yet.*

Sources and Further Reading

- Anthropic. "Introducing the Model Context Protocol." November 2024. anthropic.com/news/model-context-protocol
- Model Context Protocol Specification. modelcontextprotocol.io
- Google Cloud. "What is Agentic AI?" cloud.google.com/discover/what-is-agentic-ai
- IBM. "What is Agentic AI?" ibm.com/think/topics/agentic-ai
- MIT Sloan. "Agentic AI, explained." mitsloan.mit.edu/ideas-made-to-matter/agentic-ai-explained
- NVIDIA. "What Is Agentic AI?" blogs.nvidia.com/blog/what-is-agentic-ai

- AWS. "What is Agentic AI?" aws.amazon.com/what-is/agentic-ai
- MIT Project NANDA. "The GenAI Divide: State of AI in Business 2025." July 2025.
- Google Cloud. Agent Development Kit documentation.
- OpenAI. Agents SDK documentation. openai.github.io/openai-agents-python
- Microsoft. Copilot Studio documentation. learn.microsoft.com/microsoft-copilot-studio
- Ethan Mollick. Co-Intelligence: Living and Working with AI. Portfolio, 2024.