| **Day 1: What Are Agents Anyway?** **Generative AI** is great at understanding and generating content. ​**Agentic AI** goes a step further—it understands, generates content, *and performs actions.* |  |
| --- | --- |

|  | |  | | --- |  **Why does “taking action” matter?** Let’s rewind a bit. In 2022, ChatGPT blew up because, for the first time, AI felt conversational. You didn’t need to write code or train models—you could just ***talk*** *to it*.  Let’s compare:   * Traditional programming → Needed code to operate * Traditional ML → Needed feature engineering * Deep learning → Needed task-specific training * **ChatGPT** → Could reason across tasks and respond *without training*   This is known as **zero-shot learning** (no examples needed) or even **in-context learning** (understands tasks just from instructions).  But by 2024, people wanted *more*. Talking was cool—but what if the AI could actually ***do things***?  For example:   * Instead of just giving you a list of leads, could it *email them*? * Instead of summarizing a doc, could it *file it in the right folder* and *create a task* in your workflow? * Instead of suggesting a product to a user, could it *a****utomatically customize the landing page***?   That’s where **agents** came in. **So… how do agents take action?** The magic lies in the **tools**.  Most agents are paired with APIs, function calls, or plugins that let them *interact with external systems*. The LLM doesn’t just respond with text—it outputs structured commands like:  “Call the send\_email() function with the following inputs…” “Fetch records from the CRM using this query…” “Schedule a meeting for Tuesday at 2PM…”  This works because of a mechanism called **tool use** (or **function calling**). The agent is told what tools are available, and it figures out *when* and *how* to use them—either directly or through some planning mechanism.  In more advanced agents, this is enhanced with:   * **Memory** → To remember past steps or context * **Planning modules** → To decide what to do next, especially for multi-step tasks * **State management** → So the agent can track progress and avoid loops or failures   Think of the LLM as the *brain*, and tools as the *hands*. Without tools, an agent just talks. With tools, it acts. **Two ways to define agents:**  * **Technical view** → Agents = LLM + Tools + Planning + Memory (and all the above components discussed above) * **Business view** → Agents = Systems that *complete tasks end-to-end*   But don’t get confused: **Today's** **Agents are not AI innovations.**   * They are *engineering wrappers* around AI models. * The underlying intelligence still comes from the AI Models. The agent just helps *act* on that intelligence.  **So how do you actually build agentic AI applications?** Here’s where most people go wrong: They start with **“Let’s build an agent!”** instead of **“What real-world problem are we solving?”**  Flip the narrative.  Start with the real-world/enterprise **pain points**:   * A support team drowning in repetitive queries * An analyst switching between dashboards to find insights * A sales team manually logging and tracking customer activity  **A useful mental model: Autonomy vs. Control** Once you've identified the problem, the next decision is: ​**How autonomous should your agent be?**  Think of it as a tradeoff:  How much **autonomy** are you giving the agent vs. How much **control** do you want to retain on the human side?  This isn't a one-size-fits-all decision, it's contextual. Different problems demand different levels of agent involvement. |
| --- | --- | --- |

# **Day 2: The 4 Types of Agentic Systems (and When to Use What)**

​  
**Not all agents are built the same**, and not all problems need highly autonomous systems.

In this lesson, we’ll walk through **four types of agentic systems (as discussed yesterday)**, using a simple but powerful lens:

How much **autonomy** does the agent have?

How much **control** does the human or system retain?

​

|  |
| --- |

This balance impacts how the system behaves, how you evaluate it, and what infrastructure you need to build.

But first, a quick foundation.

### **The Tool-Augmented LLM**

At the core of most modern agents is an **LLM (Large Language Model)** acting as the brain of the system. Throughout this course, we use the term **LLM** to refer broadly to **generative AI models**—not just text-only models.

On its own, it’s capable of generating content, but to turn it into an agent, you augment it with:

* **Tools:** APIs, functions, databases it can call
* **Planning:** The ability to break a goal into multiple steps
* **Memory:** So it can track past actions and outcomes
* **State and Control Logic:** To know what’s done, what failed, and what to do next

When connected to these components, the LLM becomes more than a chatbot. It becomes a **goal-driven system** that can reason, take action, and adapt.

But depending on how much you trust it to act without supervision, you end up with **different types of agents**.

Let’s walk through them, starting from the least autonomous.

### **1. Rule-Based Systems/Agents**

**Low Autonomy, Low Control**

These systems don’t use LLMs at all. They’re built with traditional if-this-then-that logic. Every decision path is manually scripted. There’s no reasoning or learning. Rule based Agents have existed much before the LLM era.

Wait, aren’t we talking about AI agents? Well, yes—but not every problem needs an AI model. Remember: start with the problem, not the AI. If you can solve it without AI, don’t overcomplicate it.

**What problems do they solve?**​  
Well-structured, repetitive tasks with fixed inputs and outputs.

**Examples:**

* Automatically approve reimbursements under a fixed amount
* Rename files in a folder based on filename patterns
* Copy data from Excel sheets into form fields

**Pros:** Fast, auditable, predictable  
​**Cons:** Brittle to change, can’t handle ambiguity  
​**Best used when:** You know all the conditions ahead of time and there’s no need for flexibility.

### **2. Workflow Agents**

**Low Autonomy, High Control**

This is often the first step for enterprises introducing LLMs into their workflows. Here, the LLM enhances an existing workflow but doesn’t execute actions independently. A human stays in control.

**What problems do they solve?**​  
Repetitive tasks that benefit from natural language understanding, summarization, or generation, but still need human decision-making.

**Examples:**

* Suggesting first-draft responses in a support tool like Zendesk
* Generating summaries of meeting transcripts
* Translating natural language queries into structured search inputs for BI dashboards

**How the LLM is used:**​  
It reads input (text, tickets, documents), understands context, and generates useful content, but doesn’t act on it. A human still decides what to do.

**Pros:** Easy to deploy, low risk, quick value  
​**Cons:** Can’t execute or plan, limited end-to-end value  
​**Best used when:** You want to augment your team’s productivity without giving up oversight.

### **3. Semi-Autonomous Agents**

**Moderate to High Autonomy, Moderate Control**

These are true agentic systems. They not only understand tasks but can plan multi-step actions, invoke tools, and complete goals with minimal supervision. However, they often operate with some constraints or monitoring built in.

**What problems do they solve?**​  
Multi-step workflows that are well-understood but too tedious or time-consuming for humans.

**Examples:**

* A lead follow-up agent that drafts, personalizes, and sends emails based on CRM data, while logging results
* A document automation agent that extracts details from contracts and updates internal systems
* A research agent that pulls data from multiple sources, compares findings, and sends a structured report

**How the LLM is used:**​  
The LLM plans the steps, calls APIs to fetch or push data, keeps track of progress, and adapts if something goes wrong. It often includes fallback paths or checkpoints for human review.

**Pros:** Automates complex workflows, saves time, higher ROI  
​**Cons:** Needs infrastructure (planning, memory, tool calling), harder to test  
​**Best used when:** You want to automate well-bounded business workflows while retaining some control.

### **4. Autonomous Agents**

**High Autonomy, Low Control**

These agents are fully goal-driven. You give them a broad objective, and they figure out what to do, how to do it, when to retry, and when to escalate. They act independently, often across systems and over time.

**What problems do they solve?**​  
High-effort, async, or long-running tasks that span multiple systems or steps and don’t need constant human input.

**Examples:**

* A competitive research agent that pulls data over days, summarizes updates, and generates weekly insight briefs
* An ops automation agent that detects issues in pipelines, diagnoses root causes, and files tickets with suggested fixes
* A testing agent that autonomously runs product flows, logs results, and suggests new edge-case scenarios

**How the LLM is used:**​  
The LLM is the planner, decision-maker, tool-user, memory tracker, and communicator. It manages retries, evaluates whether goals are met, and decides when to stop or adapt.

**Pros:** Extremely scalable, can handle complex tasks  
​**Cons:** High risk if not monitored, hard to evaluate or trace, infra-heavy  
​**Best used when:** The task is high-leverage, async, and doesn’t require human feedback at every step.

### **So how do you decide what type of agent to build?**

**Not by picking your favorite architecture**

**You start with the problem.**

* Is it repetitive and structured?
* Does it involve language understanding or generation?
* Is it a multi-step task that needs decision-making?
* Do you trust an AI system to execute the entire task, or do you want a human in the loop?

Here’s a simple mapping to guide you:

|  |
| --- |

**Before we wrap up, two important points:**

1. These approaches aren’t mutually exclusive, a single system can use a mix of them. Some parts might require high control, while others can benefit from high autonomy. Think of them as options you can apply to different parts of a problem.
2. Each problem type can be tackled by either a single agent or a group of collaborating agents.

We’ll dive deeper into single-agent vs. multi-agent design later in the course. But for now, just remember:

Don’t start with “How do I build a multi-agent system?”  
Start with “What’s the problem I’m solving, and what kind of autonomy does it require?”

Let the **problem shape the agentic design**, not the other way around.

Tomorrow, at the same time, we’ll dive deeper into the role of **tools** in agentic systems. They’re the reason AI has become far more *usable*—and we’ll break down exactly how and why in our deep dive.

|  | |  | | --- |  **Day 3: What are Tools?** But here's a shared trait across all agent types, no matter how simple or complex:  They rely on **tools** to perform actions.  So what are tools anyway? **What Are “Tools” in AI?** In the context of agentic AI, **tools are external capabilities the LLM can invoke,** things like:   * APIs * Database queries * Internal services * Third-party systems * Internal functions written in code   They turn the LLM from something that *just talks* into something that can **act**.  Remember, LLMs on their own are stateless, have no access to real-time systems, and can’t take action. ​  But give them tools, and they can:   * Fetch data from your internal systems * Trigger events (e.g., send an email, create a JIRA ticket) * Access structured data like calendars, dashboards, or CRMs * Run pre-written logic based on business rules   This is how **generation turns into execution.** **Why Tools Matter** 1. **They unlock execution: W**ithout tools, your agent is just an assistant that makes suggestions. With tools, it can complete workflows end-to-end.  2. **They increase precision:** Rather than hallucinating, the LLM can ask the right system directly—“What’s the actual order status?” instead of making up a delay reason.  3. **They let you control risk:** You define what’s exposed. The LLM can’t do anything outside of the tools you register.  4. **They enable composability:** If you want to combine your CRM, calendar, and email stack into one assistant, you can expose each of those as tools and let the LLM orchestrate them. **A Step-by-Step Example: End-to-End Agent Task Using Tools** Let’s say you want to build a simple agent to handle the task: ​**“Inform a customer that their order is delayed and offer a new delivery time.”**  Here’s how the system works with tools:   1. **Input:** A human types: “Hey, can you let John know his order is delayed and reschedule it for tomorrow?” 2. **Planning:** The LLM breaks it down:    * Check the order status    * If delayed, check delivery slots    * Draft an email    * Send the email    * Log the interaction 3. **Tool calls:**    * get\_order\_status(order\_id=12345)    * get\_available\_slots(date=today+1)    * send\_email(to=john@example.com, content=...)    * log\_event(event\_type="reschedule", status="completed") 4. **Text generation:**​ The LLM composes the message: ​*“Hi John, just letting you know your order has been delayed. We’ve rescheduled it for tomorrow. Thanks for your patience.”* 5. **Execution:**​ The system runs the actions, logs the output, and optionally sends a status update to a dashboard.  **How This Works: Visual** Let’s introduce a helpful visual model that explains how these steps unfold. This is a really nice visual from [LangChain](https://click.convertkit-mail2.com/27u49vk0v5u9uvv586pi3hg658744hghg3lox/g3hnh5hmo0x2n7ir/aHR0cHM6Ly9wcmV2aWV3LmNvbnZlcnRraXQtbWFpbDIuY29tL2NsaWNrL2RwaGVoMGh6aG0vYUhSMGNITTZMeTl3ZVhSb2IyNHViR0Z1WjJOb1lXbHVMbU52YlM5Mk1DNHhMMlJ2WTNNdmRYTmxYMk5oYzJWekwzUnZiMnhmZFhObEx3PT0=).   |  | | --- |   Here’s what’s happening:   1. The user asks a question or gives a task 2. The **LLM understands what needs to be done** and plans its next step 3. A **parser** converts the LLM’s idea into a structured format (like: get\_order\_status(order\_id=12345)) 4. The agent then **calls the right tool,** this could be an API, a database query, or an internal function 5. The tool returns a result, this is called an **observation** 6. The **LLM looks at the result**, decides what’s missing or what comes next 7. This loop continues until it has enough information to **generate the final answer or complete the task**   The LLM is using each tool’s result to guide its next decision.  But remember, the LLM itself is still just generating text. That text is structured into tool calls, executed externally, and the results are fed back into the LLM, creating a loop of reasoning, action, and reflection (a.k.a, an agent). This is your **aha** moment!  This structure is used by frameworks like LangChain, CrewAI, AutoGen, and even custom orchestration setups in production teams.. **What Makes a Tool Usable by an LLM?** To register a tool with an agent system, you typically define:   * A **name** (e.g., create\_meeting) * A **description** (so the model knows when to use it) * Input **parameters** (and types) * Output structure (so the model can use the result)   That metadata is what allows the LLM to reason about which tool to use and how. **A Note on Parsing and Structured Outputs** The **parser** plays a key role in converting the LLM’s response into a structured tool call — something the system can reliably execute (like get\_order\_status(order\_id=12345)).  But in many modern setups, **you don’t always need a separate parser**.  Most popular LLMs, especially those designed for tool use, can directly **produce structured outputs**, like JSON or function calls, that can be consumed by your backend as-is. This reduces the need for manual parsing and helps simplify orchestration.  Similarly, well-designed tools return structured data making it easier for the LLM to reason about what to do next.  The structure on both sides (input and output) is what makes agent loops **robust, traceable, and production-grade.**  A lot of this will feel familiar if you've built or worked with APIs before. But if you're not from that world, **don’t overthink the wiring**.  Just remember this:  AI models on their own can understand and generate.  But when they’re connected to software, tools, APIs, internal systems — they can **actually do things.**   | ***Day 4: What Is RAG, and What Does It Mean to Make It Agentic?*** *The model needs to* ***think with your data****. That’s where* ***RAG*** *comes in.* ***What Is RAG?*** ***RAG*** *stands for* ***Retrieval-Augmented Generation****. It’s a system design where the model retrieves relevant information from your own data — just before generating a response.*  *Instead of relying only on what the model was trained on, RAG gives it access to live, contextual information from your enterprise systems. This makes answers more accurate, grounded, and auditable.*  *You might be wondering, “Why not just give all the data to the model directly?”*  *The problem is, models can only process a limited amount of text at a time — and even within that limit, they struggle when too much irrelevant or noisy information is included. It makes their responses less focused and more error-prone.* ***The RAG Process (at a Glance)*** *Here’s what it looks like in practice:*   |  | | --- |   ***Image Source****:* [*https://hyperight.com/7-practical-applications-of-rag-models-and-their-impact-on-society/*](https://hyperight.com/7-practical-applications-of-rag-models-and-their-impact-on-society/)   1. ***Data*** *– Your internal content (PDFs, emails, notes, wikis)* 2. ***Chunking*** *– It’s broken into smaller parts for better indexing* 3. ***Prompt + Context*** *– At query time, the system retrieves relevant pieces (this is also called the retrieval phase)* 4. ***LLM*** *– The model uses that context to generate a response* 5. ***Output*** *– The result is based on your data, not just what the model "knows"*  ***Why RAG Is Everywhere in Enterprise AI*** *You’ll often hear this number:*  *From what I’ve seen across clients and systems, 70% of enterprise GenAI use-cases use RAG.*  ***Why RAG is invaluable to :***   * *Enterprise knowledge changes frequently* * *Fine-tuning models is expensive and slow* * *Retrieval is faster, safer, and easier to control* * *It brings structure and traceability into LLM systems* * *It works on both unstructured (docs) and semi-structured (dashboards, notes) data*   *So instead of asking: “How do I teach the model everything we know?” Most teams ask: “How do I let the model fetch what we already have?”* ***RAG = LLM + Additional Retrieved Data*** *RAG became the dominant pattern in 2024 for a reason: it bridged the gap between general-purpose LLMs and private, task-specific enterprise knowledge.*  *At its core,* ***RAG is simple****:*  *You take an LLM and feed it additional, retrieved information right before generation.*  *This makes the model more accurate, more context-aware, and less reliant on memorized facts. It was super useful for tasks like Q&A, summarization, and policy lookups — especially in data-rich environments like legal, finance, and support.*  *No wonder* ***2024 was dubbed “the year of RAG.”*** ***But Now We’re Moving Into the Agentic Era*** *RAG isn’t going away, but it’s evolving.*  *Today’s systems don’t just retrieve once and generate an answer. In agentic workflows, retrieval becomes part of a broader,* ***dynamic reasoning loop****.*  *Agents plan, retrieve, reflect, and retrieve again — not just once, but as many times as needed throughout a task.*  *That’s where* ***Agentic RAG*** *comes in.* ***What Is Agentic RAG?*** ***Traditional RAG*** *looks like this:*   * *One query* * *One retrieval* * *One response*   *It works well for standalone questions like:*  *“What’s our policy on PTO rollover?”*  *But most real-world enterprise workflows aren’t one-shot.*   |  | | --- |   *Let’s say you’re building a* ***deal assistant*** *for your sales team. In a single task, the agent may need to:*   1. *Pull the customer’s CRM history* 2. *Retrieve current pricing for their segment* 3. *Look up regional legal terms* 4. *Reference past contract clauses* 5. *Generate a custom proposal* 6. *Double-check facts* 7. *Log the interaction*   *In agentic systems, retrieval isn’t just a setup step. It’s how the agent gathers missing context, checks its assumptions, and adapts mid-task. That means RAG becomes:*   * *A tool for* ***in-task learning*** * *A method for reducing hallucinations* * *A mechanism for handling dynamic workflows* * *A bridge between reasoning and grounded enterprise knowledge*   *Agentic RAG turns retrieval into a* ***first-class decision-making loop*** *by using retrieval as part of the model's thinking process.*  *If you think about it, RAG is also a kind of tool, but instead of triggering an action, it helps the agent pull the right information from a large volume of data. In practice, agents often combine RAG + tools + planning to complete complex tasks reliably and contextually.*  *​*  *RAG is a deep and rapidly evolving space, honestly, it could be its own course. If you're curious to explore further, I’ve curated a GitHub repo of* [*key RAG papers*](https://click.convertkit-mail2.com/wvud84n64mt6u55zk75u7hnv28kxxs8h4vk9z/n2hohvhv546k6wa6/aHR0cHM6Ly9naXRodWIuY29tL2Fpc2h3YXJ5YW5yL2F3ZXNvbWUtZ2VuZXJhdGl2ZS1haS1ndWlkZS9ibG9iL21haW4vcmVzb3VyY2VzL1JBR19yb2FkbWFwLm1k) *that covers the landscape well and I have a* [*101 guide on Agentic RAG too*](https://click.convertkit-mail2.com/wvud84n64mt6u55zk75u7hnv28kxxs8h4vk9z/48hvhehmq36n65ux/aHR0cHM6Ly9naXRodWIuY29tL2Fpc2h3YXJ5YW5yL2F3ZXNvbWUtZ2VuZXJhdGl2ZS1haS1ndWlkZS9ibG9iL21haW4vcmVzb3VyY2VzL2FnZW50aWNfcmFnXzEwMS5tZA==)*.* |  | | --- | --- | --- | --- |  |  | *All the frameworks and abstractions you'v* | | --- | --- |  ***Day 5: What Is MCP and Why Should You Care?*** *So far, we’ve covered how LLMs can call tools, retrieve context with RAG, and act as agents that reason, reflect, and adapt. Today, we’re zooming in on a concept that’s been gaining some* ***serious*** *traction:* ***Model Context Protocol (MCP).*** ***First, a Quick Recap*** *In Day 3, we learned that tools let models do things. In Day 4, we saw that RAG helps models find relevant info before answering.*  *These are external supports, they help the model act smarter, but the coordination still sits outside the model. But what if you could pass all the context a model needs, tools, retrieved data, memory, instructions, in one clean, structured format?*  *That’s what* ***Model Context Protocol*** *is trying to solve.* ***So What Is MCP?*** *At its core,* ***Model Context Protocol (MCP)*** *is a standardized way to give an LLM everything it needs to reason and respond.*   |  | | --- |   *Think of it like packaging up:*   * *The task you want the model to do* * *The tools/APIs it can use* * *The documents or memory it might need* * *The prior messages in the conversation*   *…and then handing all of that over in one go.*  *It’s not a tool, library, or product. It’s a* ***protocol****, a structure for communication between the model and the outside world.*  *If you’re from the tech world, equivalents would be things like* ***HTTP, TCP/IP, or SMTP****.*  *If you’re not, don’t sweat it. Just remember:* ***tech folks love standardization*** *— it makes things easier to reuse and plug together.* ***Why Does This Matter?*** *Let’s say you’re building an agent.*  *Right now, you’re probably juggling:*   * *Sending a prompt* * *Passing retrieved documents* * *Registering tools* * *Managing state* * *Keeping track of what happened before*   ***MCP says:***  *“Let’s standardize how we give all of that to the model, so we don’t reinvent the wheel for every use case.”*  *And for enterprises, this matters a lot. As agents get more complex, coordinating tools, RAG, memory, and outputs becomes... messy.*  *MCP makes that orchestration* ***composable****,* ***modular****, and easier to plug into other systems.*  *If you’ve ever worked with APIs, think of MCP like a* ***well-defined request schema****. Instead of tossing everything into one long string and hoping the model figures it out, the model still sees text, but it’s structured, with clear context, options, and grounding.* ***Why Did MCP Catch On So Fast?*** *Given that MCP is just a protocol, you might be wondering: ​****What makes it better, and why did everyone jump on board?***  *Here’s what (I think) helped:*   1. *It’s AI-Native, MCP was built for AI agents. It makes space for everything agents use today: tools, prompts, memory, documents, and more.* 2. ***It launched with strong docs and examples*** *which makes it easy for developers to adopt quickly. Anthropic (the creators of MCP) even released* [*many examples and code*](https://click.convertkit-mail2.com/68uqz58k5gfoukkq5rruohpxdqnkkh9hno8e2/25h2hoh3050dm5b3/aHR0cHM6Ly9naXRodWIuY29tL21vZGVsY29udGV4dHByb3RvY29s)*. They didn’t just publish a spec, they shipped clients, SDKs, testing tools, and real-world demos.* 3. ***Network effect kicked in*** *– MCP was released quietly in Nov 2024… and most people slept on it. But in 2025, it exploded. Tools, startups, and even OpenAI began supporting it. Now it’s showing up everywhere.*  ***Common Misunderstandings*** *A few things people often get wrong:*   * ***MCP isn’t a new API or product*** *– It’s just a pattern, a clean way to frame what you send to the model.* * ***It doesn’t magically make models smarter*** *– It just gives them better, more structured context.* * ***It’s not just for agents*** *– Even simple assistants benefit from better context management.*  ***So… Should You Care?*** *If you’re building toy prompts or quick demos — probably not (yet).*  *But if you’re working on:*   * *Enterprise-grade agents* * *Multi-tool workflows* * *LLMs that need to access memory + RAG + planning* * *Systems where context management is becoming a bottleneck*   *Then yes, you should care. MCP is about getting better at passing evolving, structured context into models.*  ***But also, keep this in mind:****​ MCP is just a protocol. And like all standards, it only works if it gets adopted widely. If something better than MCP comes along before it becomes “the HTTP of agents”, the ecosystem might move again.* ***Day 6: Planning in Agents + Reasoning Models******Hi Deepti,******Woah! we’re more than halfway through our course!******Just checking in to make sure you've signed up for our live sessions as a part of this course, you can find them here (if the date has passed, please watch recordings)******Live Sessions (Please sign up individually):******Session 1:*** [***Building Agentic AI Applications 2025 (Q2 Updates)***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/e0hph7h7wxzvxxf8/aHR0cHM6Ly9tYXZlbi5jb20vcC9iODQ3MGMvMS1idWlsZGluZy1hZ2VudGljLWFpLWFwcGxpY2F0aW9ucy0yMDI1LXEyLXVwZGF0ZXM=)***​******Session 2:*** [***AI Protocols 101: What You Should Know About MCP, A2A etc***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/7qh7h8h9pge5vvfz/aHR0cHM6Ly9tYXZlbi5jb20vcC9lMmI1ZGIvMi1haS1wcm90b2NvbHMtMTAxLXdoYXQteW91LXNob3VsZC1rbm93LWFib3V0LW1jcC1hMmEtZXRj)***​******Session 3:*** [***Single vs. Multi-Agent AI Systems***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/owhkhqhwlzed3kav/aHR0cHM6Ly9tYXZlbi5jb20vcC8wZTBlMTUvMy1zaW5nbGUtdnMtbXVsdGktYWdlbnQtYWktc3lzdGVtcw==)***​******Session 4:*** [***Emerging Trends Shaping the AI-Native Products***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/z2hghnhergzx9kup/aHR0cHM6Ly9tYXZlbi5jb20vcC8xMGFmZmIvNC1lbWVyZ2luZy10cmVuZHMtc2hhcGluZy10aGUtYWktbmF0aXZlLXByb2R1Y3Rz)***​******You can also sign up for Agentic AI in Enterprise Use Cases – Demo Day, where you’ll see real projects built by students from*** [***our top-rated Maven course***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/p8heh9h45veokdcq/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)***. It’s completely free and a great way to understand how agentic AI can be applied in enterprise settings.******​***[***Demo Day Session 1***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/x0hph6hedgm08df5/aHR0cHM6Ly9tYXZlbi5jb20vcC80NjZlMjIvMS1kZXNpZ25pbmctYWdlbnRpYy1haS1zeXN0ZW1zLWZvci1lbnRlcnByaXNlLXVzZS1jYXNlcw==)***​******​***[***Demo Day Session 2***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/6qheh8hl6q4evdbo/aHR0cHM6Ly9tYXZlbi5jb20vcC9hMGNkZjEvMi1kZXNpZ25pbmctYWdlbnRpYy1haS1zeXN0ZW1zLWZvci1lbnRlcnByaXNlLXVzZS1jYXNlcw==)***​******Over the past few days, we talked about what agents can do. They can use tools, retrieve information through RAG, and pass everything in a clean format using MCP.******But all of that assumes something fundamental.******That the agent actually knows what to do next.******And that’s where things often break.******Today, we shift focus from tools and inputs to how agents think. More specifically, how modern models are starting to plan and why that changes how we design real-world systems.******Why Planning Matters in Agentic Systems******Here are a few examples to start with.******If you ask an agent "What’s 13 multiplied by 47", it can either solve it or call a calculator. This is a direct one-step task. There’s no real planning needed.******Now imagine asking "Find all our Q1 clients in the healthcare sector, check which ones are overdue on payments, and draft personalized emails with new payment links."******In this case, the agent needs to understand the instruction, break it down into manageable parts, retrieve the right data, choose tools, perform the steps in order, handle any exceptions, and know when the task is done.******That loop of interpreting, sequencing, and acting is what we mean by planning.******The agent, meaning the model, is expected to figure this out on its own. That includes selecting which tools to use and how to apply the information it has.******Here’s the challenge. Most general-purpose language models were never trained to do this.******Why Traditional LLMs Struggle With Planning******To understand this limitation, it helps to zoom out and remember how LLMs are trained. They are trained to predict the next token based on the previous context. That’s it.******They are great at continuing sentences, generating summaries, answering direct questions. That’s because those are the kinds of tasks their training optimized for.******In that sense, LLMs behave more like short-sighted generators. They complete what’s in front of them but are not inherently wired to think ahead.******So when you ask them to act as agents, especially in tasks that need decision-making over time or handling errors, they tend to break. Some common failure modes are skipping steps, repeating actions, overcomplicating simple things, or completely losing the plot halfway through.******Early Attempts to Improve Reasoning******To patch this gap, early builders started using prompting techniques that nudged the model to behave like a planner.******One popular example was chain-of-thought prompting. You prompt the model with phrases like “let’s think step by step” and it starts to break the task into stages.******This worked in many situations, especially logic puzzles or structured Q and A. But it wasn’t enough for real agents operating with tools, unpredictable inputs, or constantly changing state.******Because underneath, these models still weren’t trained for planning. They were just responding to prompt tricks.******Then Came Reasoning Models******That led to a new shift in how models were trained. What if we didn’t just prompt for planning, but trained models to do it by design?******This gave rise to a new category known as Large Reasoning Models (LRMs)***  | ***No photo description available.*** | | --- |  ***​***[***Image Source***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/kkhmh6hn27pvg9cl/aHR0cHM6Ly93d3cudGhyZWFkcy5uZXQvQG1vY2thcGFwZWxsYS9wb3N0L0RGUWlwUjdTSVI3)***​******Here’s the rough difference:******LLMs: ​input → LLM → output statement******LRMs: ​input → LRM → plan step + output statement******All still text, but the pattern is different. LRMs are nudged during training to think before acting.******OpenAI’s o-series (like o1 and o3) were the first public examples. Shortly after, DeepSeek released DeepSeek-R1, another model tuned for tool-augmented reasoning and planning.Now most foundation model providers are releasing their own reasoning variants. Gemini has thinking models, Claude 3.7 has reasoning built into its flow, and some of these are even hybrids that activate reasoning only when needed.******How They Fit in Agentic Design******The main value of reasoning models is in improving the planning component of your system. This is the part that asks, what should I do next and why.******In most enterprise use cases, planning is where agents usually fall short.******Use Them With Caution******Reasoning models are still very new and under active development, and they come with real tradeoffs.******Research studies and real world usage shows that they often overthink simple tasks, generate longer outputs, and increase both latency and cost. If not carefully managed, they can also hallucinate planning steps that sound logical but are completely wrong.******Knowing what’s new is one thing. Using it blindly is another. ​Problem first — always.******As a rule of thumb, we never begin a project with a reasoning model.******I usually start with a mid-size base model and check if it can handle the task.******Only when I see clear signs of planning failure do I consider a reasoning model — and even then, I evaluate what it actually adds to the system.******Tomorrow (same time), we’ll shift to another core component of agents: memory — specifically, how agents can remember things effectively and why that matters.******-***[***Aish***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/58hvh7hgoew2p5f6/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL2FyZWdhbnRpLw==)***,*** [***Kiriti***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/25h2hoh305xwr2b3/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL3NhaS1raXJpdGktYmFkYW0v)***​******All the frameworks and abstractions you've seen above are original, developed through hands-on experience building over 100 real-world AI applications across enterprise settings. You won’t find them packaged like this anywhere else online.******If this approach resonates with you, and you're serious about learning how to design agents that actually work in production, not just hype demos, check out our top-rated 6-week course "***[***Building Agentic AI Applications with a Problem-First Approach***](https://click.convertkit-mail2.com/38u75wmow8fduoow2m8urh47dzonnh7h6lwpd/p8heh9h45veokdcq/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)***"******It's built for all backgrounds, whether you're a Product Manager, Architect, Director, C-suite leader, or someone exploring agentic AI with intent. Let’s move beyond the buzzwords and into systems that solve real problems!******Use the code 10DAY for a limited time 10% off!******Day 7: Memory in Agents*** *Hi Deepti,*  *Over the past few days, we’ve explored what makes agents act — from tools and RAG, to MCP and reasoning models.*  *Today, we shift gears to something that determines how well they act over time:* ***memory****.*  *Because here’s the baseline:*  ***AI models don’t have memory inherently.*** *They’re stateless by design. Every input is treated independently unless you architect memory into the system.* ***Why Memory Matters***  |  | | --- |   *Image source:* [*https://arxiv.org/html/2502.12110v1*](https://arxiv.org/html/2502.12110v1)  *If an agent is helping you draft emails, summarize long threads, or manage workflows over days or weeks — it needs to remember things like the email format, the user's name, or what tone to use.*  *Sure, you could pass that information again and again with every prompt… But wouldn’t it be better if the agent could retrieve the right information on its own, at the right time, from an external database?*  *That’s exactly where memory comes in.* ***“Wait… isn’t this just like Agentic RAG (Day 4)?”*** *Fair question. And you’re not wrong, managing memory often looks a lot like doing Agentic RAG.*  *You:*   * ***Write*** *structured or unstructured memories (facts, logs, past outputs)* * ***Store*** *them with metadata, tags, or embeddings* * ***Retrieve*** *the relevant slice when needed* * ***Ground*** *the model’s next action using that context*   *So yes, similar plumbing. But the* ***intent*** *is different.*  *RAG helps answer questions with knowledge.*  *Memory helps agents behave coherently over time.* ***Two Types of Memory in Agents*** *When designing real-world agent systems, you typically deal with two kinds of memory:*   |  | | --- |   *Image Source:* [*https://langchain-ai.github.io/langgraph/concepts/memory/#what-is-memory*](https://langchain-ai.github.io/langgraph/concepts/memory/#what-is-memory) ***1. Short-Term Memory*** *This is scoped to a single session or task.*  *It includes:*   * *The conversation so far* * *Tools used* * *Responses generated* * *Documents retrieved*   *Think of it as a raw log* ***user-agent conversations.***  *LangGraph, Autogen, and similar frameworks treat this as part of the agent’s "****state****." But state grows fast, and most agents don’t perform well when buried under too much irrelevant history.*  *That’s where you need strategies like:*   * *Trimming stale messages* * *Summarizing the past into key points* * *Filtering based on what’s still relevant*   *It’s a balancing act, context length vs clarity vs cost.* ***2. Long-Term Memory*** *This memory lives across sessions, days, weeks, or even forever.*  *It helps agents remember:*   * *Who the user is* * *How they prefer to interact* * *What’s already been done* * *Important context from the past*   *Examples:*   * *“User prefers neutral tone”* * *“User name is X and stays in the city Y”* * *​*   *But again, more data isn’t better by default. It’s about retrieving the right thing at the right time.* ***Types of Long-Term Memory to Consider*** *Borrowing from cognitive science, here are the common categories (not comprehensive, just a few popular ones):*   * ***Semantic Memory*** *→ Facts and info (Objective) ​“User speaks English and prefers Excel files.”* * ***Episodic Memory*** *→ Past actions ​“Agent already generated a summary yesterday.”* * ***Procedural Memory*** *→ Preferences (Subjective) ​“Avoid passive voice. Prioritize action items.”*  | *diagram*  *How agents can improve themselves using memory* | | --- |   *Not every agent needs all of them. It depends on your use-case.*  *Some examples:*   * ***User-facing chatbots*** *→ need* ***semantic memory*** *for personalisation* * ***Process automation agents*** *→ need* ***episodic memory*** *to avoid retries or loops* * ***Adaptive assistants*** *→ benefit from* ***procedural memory****, enabling them to rewrite their own prompts based on feedback*   *So before saying “we need memory,” ask:*   * *What kind?* * *Why is it needed?* * *How will it be stored, retrieved, and kept fresh?*  ***Managing Memory in Practice*** *Managing memory often feels like managing RAG. But here’s the hard part:* ***deciding what to store and what to retrieve.****​ Just stuffing more text into the agent input won’t help. In fact, it often hurts performance.*  *You need to* ***design memory intentionally****, based on:*   * *The agent’s job* * *What it needs to recall* * *When it should recall it* * *And how to keep it useful over time*  ***A Few Enterprise Examples*** *Let’s ground this in real life.*  ***Customer Support Agent****​ Needs to remember: recent support history, known bugs, user sentiment → episodic + semantic memory*  ***Sales Copilot****​ Needs to remember: previous pitches, user objections, close status → semantic + procedural memory*  ***Compliance Auditor Agent****​ Needs to recall: flagged items, prior exceptions, policy changes → episodic memory*  *In all these, what matters isn’t the quantity of data but the structure and relevance of memory. And yes, I’ve said this painfully many times, but I’ll say it again:*  *The answer lies in your use case. The memory strategy, just like tools or planning, depends entirely on the problem you’re trying to solve.*  *So what do we say to AI hype creators?* ***Problem-first, always :)***  *Tomorrow (same time), we’ll shift gears and talk about* ***multi-agent systems*** *— what they are, how they coordinate, and whether you actually need more than one agent at all.*  *-*[*Aish*](https://click.convertkit-mail2.com/qdunl0r705s8u77x4p4ilh87pxrkkc4h865z7/g3hnh5hmk7qpkqir/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL2FyZWdhbnRpLw==)*,* [*Kiriti*](https://click.convertkit-mail2.com/qdunl0r705s8u77x4p4ilh87pxrkkc4h865z7/9qhzhnhdlk59lea9/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL3NhaS1raXJpdGktYmFkYW0v)*​*  *All the frameworks and abstractions you've seen above are original, developed through hands-on experience building over 100 real-world AI applications across enterprise settings. You won’t find them packaged like this anywhere else online.*  *If this approach resonates with you, and you're serious about learning how to design agents that actually work in production, not just hype demos, check out our top-rated 6-week course "*[*Building Agentic AI Applications with a Problem-First Approach*](https://click.convertkit-mail2.com/qdunl0r705s8u77x4p4ilh87pxrkkc4h865z7/3ohphkh3n6pzn8ar/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)*"*  *It's built for all backgrounds, whether you're a Product Manager, Architect, Director, C-suite leader, or someone exploring agentic AI with intent. Let’s move beyond the buzzwords and into systems that solve real problems!*  *Use the code 10DAY for a limited time 10% off!* |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

# **Day 8: Multi-Agent AI Systems**

Hi Deepti,

So far, we’ve talked a lot about *what makes a single agent act* — from tools and RAG to memory and planning.

What if your agentic pipeline needs to:

* **Parallelize** tasks to speed things up
* **Use different agent personas** for different parts of a task
* Or **break up complexity** across specialized units, like in a team

That’s where **multi-agent systems** come in.

### **Why Use Multi-Agent Systems?**

Sometimes, a single agent just can’t cut it because the problem itself might demand scale, specialization, or parallel thinking.

Let’s take a few examples:

* You're generating a **marketing strategy** that needs market insights, legal review, and creative suggestions.
* You're building a **compliance assistant** that needs to extract information, flag risks, and cross-check policies.
* You're automating a **sales process** where one agent talks to the user, another enriches data, and a third handles follow-ups.

Could you do this with one beefy agent? Maybe.

But when you split it into multiple, **specialized agents**, you can enable:

* **Parallelization**: Agents work on parts of a task simultaneously.
* **Specialization**: One agent is good at legalese, another at writing emails.
* **Tooling Independence**: Each agent can have its own tools and memory.

### **Flat vs Hierarchical Agent Coordination**

All multi-agent systems need some way to coordinate. This is where communication patterns come in — and there are two common ones:

|  |
| --- |

​

#### **1. Hierarchical Patterns (More Controllable)**

An orchestrator agent delegates subtasks to others. It sees the full picture and controls the flow.

You’d use this when:

* Tasks can be clearly decomposed
* You want tight control
* You have known agent roles (e.g. summarizer, generator, checker)

Think: enterprise workflows, tool suites, parallel pipelines

#### **2. Flat Patterns (More Dynamic)**

Agents talk to each other as peers — no boss.

You’d use this when:

* Tasks need creativity or debate
* You want agents to evaluate each other
* There’s no one “correct” answer path

Think: brainstorming, ranking options, multi-view reasoning

### **What Nobody Tells You: Multi-Agent Systems Are a Pain**

On paper, this all sounds great. You can also build quick multi-agent prototypes and have fun with them, but for customer/enterprise use-cases…**It’s painful.**

Most people read a blog on multi-agent systems and get excited about modularity — “like microservices!” they say. But **AI agents are not microservices.**

Unlike code, **AI models are non-deterministic.**

They don’t always behave the same way, and adding more agents means:

* **More non-determinism** (variation across agents, not just within one)
* **More memory and state complexity** (who knows what, and when?)
* **Higher latency and cost**
* **More coordination bugs and failure points**
* And yes — **more collusion**, where agents agree when they shouldn't (it happens more than you think)

Honestly, we could write a book on how painful it is to get multi-agent systems working reliably.

### **So… Should You Use Them?**

Here’s my personal rule:

**In the Enterprise, don’t start with multi-agents. Start with one.**

Let that one agent fail, either *empirically* (via eval metrics) or *operationally* — before you scale.

Most enterprise use cases (honestly, 70%+) work just fine with a **single well-designed agent** — one that uses tools, memory, RAG, and planning.

Multi-agent systems shine only when:

* The task is **big enough to need parallel execution**
* You need **clear specialization**
* Or you want **creative debate, evaluation, or distributed decision-making**

Even then, you need strong design, especially around **memory**, **state**, and **communication protocols**.

### **Final Word: Problem First, Always**

This has been our mantra from Day 1:

Don’t build a multi-agent system because it sounds “agentic.”

Build it if — and only if — your **problem** needs it.

And the only way to know that? Have the **right metrics**, test, and let simpler systems fail first.

Tomorrow (same time), we’ll talk about some real world agents and how they function!

-[Aish](https://click.convertkit-mail2.com/75u5vn86n0f2ukkg657azhwzgmo66tnh53ekv/kkhmh6hn3dkp78sl/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL2FyZWdhbnRpLw==), [Kiriti](https://click.convertkit-mail2.com/75u5vn86n0f2ukkg657azhwzgmo66tnh53ekv/58hvh7hgzvrwemi6/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL3NhaS1raXJpdGktYmFkYW0v)​

*All the frameworks and abstractions you've seen above are original, developed through hands-on experience building over 100 real-world AI applications across enterprise settings. You won’t find them packaged like this anywhere else online.*

*If this approach resonates with you, and you're serious about learning how to design agents that actually work in production, not just hype demos, check out our top-rated 6-week course "*[*Building Agentic AI Applications with a Problem-First Approach*](https://click.convertkit-mail2.com/75u5vn86n0f2ukkg657azhwzgmo66tnh53ekv/25h2hoh3evkx53f3/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)*"*

*It's built for all backgrounds, whether you're a Product Manager, Architect, Director, C-suite leader, or someone exploring agentic AI with intent. Let’s move beyond the buzzwords and into systems that solve real problems!*

*Use the code 10DAY for a limited time 10% off!*

|  | | Logo | | --- |  **Day 9: Real-World Agentic Systems** Hi Deepti,  So far, we’ve covered all the ingredients that make up an agent: tools, planning, RAG, memory, structure, and even coordination in multi-agent setups.  But you might be thinking: “Where does all this actually show up in the real world?”  Let’s walk through a few public-facing systems that exhibit agentic behavior — as far as we can tell.  ⚠️ A quick note:  These aren’t open source. We don’t know their exact internals. What follows is an informed simplification based on how they behave externally — just enough to understand how the agentic stack might show up in practice. **NotebookLM (Google): Agentic Search on Your Own Data** Google’s [NotebookLM](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/p8heh9h4l2ex5euq/aHR0cHM6Ly9ub3RlYm9va2xtLmdvb2dsZS8=) acts like a personal research assistant. You upload your files, and it helps you work with them — summarizing, answering questions, even generating audio versions or study guides.  Let’s focus on its core: Q&A over your content — essentially a scaled-up, personal RAG system.  How it likely works:   1. User uploads files (PDFs, notes, slides, etc.) 2. Preprocessing The system stores them in a way that allows for retrieval later. 3. User asks a question e.g., “What were the key insights from my Q2 strategy deck?” 4. Planning    * Interprets task type — summary, Q&A, comparison?    * Identifies relevant docs or sections. 5. RAG Retrieves only the most useful parts of your documents. 6. LLM Generation Responds clearly and grounded in your actual content. 7. Memory    * *Short-term:* Keeps track of the current conversation.    * *Long-term:* Likely minimal or none. 8. Tools Might include file viewers, summarization modules, etc.   ​  What makes it agentic: It interprets user goals, searches across your data, and composes clear responses — rather than just giving static outputs. **Perplexity: Agentic Search on the Open Web** ​[Perplexity](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/x0hph6helpm3dgb5/aHR0cHM6Ly93d3cucGVycGxleGl0eS5haS8=) aims to do what traditional search engines like Google don’t, give you a direct, answer-like response with sources, instead of a page of links.  How it likely works:   1. User asks a question e.g., “What’s the latest research on Alzheimer’s treatments?” 2. Planning (implicit or explicit) Interprets the intent — What does "latest" mean? What counts as credible? 3. Tool Use Makes search queries via web APIs (search tool). 4. RAG Retrieves relevant content from web pages to feed into the model. 5. LLM Response Synthesizes a coherent answer from the retrieved content, often with citations. 6. Memory    * *Short-term:* Likely remembers session context.    * *Long-term:* May store preferences (e.g., “always use WSJ for news”).   What makes it agentic: It fetches info, chooses what to use, and constructs an answer, all in a multi-step reasoning loop.  ​ **DeepResearch (OpenAI): Deep Agentic Workflows** ​[DeepResearch](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/6qheh8hld54odvho/aHR0cHM6Ly9vcGVuYWkuY29tL2luZGV4L2ludHJvZHVjaW5nLWRlZXAtcmVzZWFyY2gv) is designed for open-ended, complex research tasks, things like market analysis, competitive landscapes, or technical deep dives.  This is where the agent stack really stretches its legs.  How it likely works:   1. User asks a broad task e.g., “Analyze the generative AI landscape for education startups.” 2. Planning    * Breaks it into subtasks: funding, trends, companies, risks    * Forms a plan or flow of actions — a mini execution graph 3. Tools Likely integrates:    * Web search    * Document readers (e.g., PDFs)    * Data tools (e.g., spreadsheets, graphs)    * Report generation modules 4. Agentic RAG This isn’t one-shot retrieval. It fetches, reflects, and re-fetches as the task evolves. 5. Memory    * *Episodic:* Tracks which parts of the task are done    * *Semantic:* Stores key facts, names, insights 6. Multi-step Reasoning This is the “agent loop” in action: Plan → retrieve → read → rethink → generate → refine → repeat   What makes it agentic: Heavy planning, iterative tool use, and the ability to drive a task forward on its own.  If you play around with the above agentic systems they feel like very different products but actually rely on the same foundational components we’ve been learning — tools, planning, memory, and RAG!   |  | | --- |  |  | | --- |   Can you guess what *level* of agents they are? Based on what we covered back in Day 2, here’s a rough guess (since none of these are open source, we can’t say for sure):   * **NotebookLM** looks like it sits between level 2 and level 3. It behaves more like a high-control workflow agent. It retrieves well, maybe does complex RAG behind the scenes, but it doesn’t make decisions on your behalf. You’re still directing it step by step. * Perplexity leans towards level 3 and maybe even level 4. You give it a query and it plans what to search, how to organize sources, and how to answer. It has more decision-making power baked in. You’re delegating the task, not just asking for help. * DeepResearch feels like a strong level 4 system. It takes a high-level task, breaks it down, searches, reasons, and builds out a full report. It reflects, re-asks, and loops. You’re not guiding it through each step, it acts much more independently.​   If you’re curious, try them out. They all have free versions. Watch for how much control you still have and how much the system is deciding on its own. It’s a good way to build that instinct for agent design.  Tomorrow, at the same time (for one last time), we’ll wrap things up, summarise what we’ve learned, go over some best practices, and take a quick look at where agentic AI is headed.  -[Aish](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/kkhmh6hn35pm3dfl/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL2FyZWdhbnRpLw==), [Kiriti](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/58hvh7hgznw7zna6/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL3NhaS1raXJpdGktYmFkYW0v)​  *All the frameworks and abstractions you've seen above are original, developed through hands-on experience building over 100 real-world AI applications across enterprise settings. You won’t find them packaged like this anywhere else online.*  *If this approach resonates with you, and you're serious about learning how to design agents that actually work in production, not just hype demos, check out our top-rated 6-week course "*[*Building Agentic AI Applications with a Problem-First Approach*](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/25h2hoh3e8xdegs3/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)*"*  *It's built for all backgrounds, whether you're a Product Manager, Architect, Director, C-suite leader, or someone exploring agentic AI with intent. Let’s move beyond the buzzwords and into systems that solve real problems!*  *Use the code 10DAY for a limited time 10% off!*   | [Register Now!](https://click.convertkit-mail2.com/qdunl0r705s8u77x454blh87pxrkkc4h865z7/25h2hoh3e8xdegs3/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu) | | --- | |
| --- | --- | --- | --- | --- | --- |

## **Quick Recap**

Here’s what we covered:

* **What agents are** (Day 1): not just chatbots that generate text, but systems that can decide and act.
* **Types of agents** (Day 2): from tightly controlled workflow agents to fully autonomous ones, all depending on how much decision-making you hand over.
* **Tools and RAG** (Days 3–4): the bread and butter of agent action and knowledge grounding.
* **MCP** (Day 5): a clean way to structure everything an agent needs, tools, memory, prior messages, in one payload.
* **Planning and reasoning models** (Day 6): why plain LLMs aren’t enough for complex decisions, and how newer models are built for multi-step tasks.
* **Memory** (Day 7): short-term vs. long-term memory, what to store, how to retrieve, and why it matters for continuity.
* **Multi-agent systems** (Day 8): orchestration, peer-to-peer collaboration, and the messiness of coordination.
* **Real-world systems** (Day 9): how Perplexity, NotebookLM, and DeepResearch likely use these patterns in different ways.

We’ve covered the moving parts that show up in real-world systems. But all of it falls apart if you’re not thinking about two things: **observability and evaluation**.

## **What’s Still Hard**

### **Observability**

Observability means tracking what your agent is actually doing — at every step. You’ll want:

* Logs of tool calls, decisions, retries
* Metrics to spot bottlenecks in latency and cost
* Visibility into when things go off-rail
* Step-wise traceability for debugging

Tools like [Comet Opik](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/owhkhqhw7newz8tv/aHR0cHM6Ly93d3cuY29tZXQuY29tL3NpdGUvcHJvZHVjdHMvb3Bpay8=) help with this. You need to design observability into the system from day one, especially for high-autonomy agents.

### **Evaluation**

Agents are non-deterministic. So you need continuous evaluation, not just manual testing. At a minimum, track:

* Goal or task completion rates
* Tool call success and failure
* RAG quality and hallucination metrics
* Model overthinking or inefficiency
* Latency and token usage at each step

Evaluation is how you understand and improve your system.

This is also where most teams get stuck: doing vibe checks instead of real evals, and ending up in PoC purgatory.

Think of evals and observability as your testing pipeline — the agentic equivalent of software QA. The above metrics aren’t exhaustive; what you track will depend heavily on the specifics of your use case.

## **Where Things Are Headed in Agentic AI**

This space is still early, but here’s where things are clearly moving.

### **1. Protocols > Prompts**

| **No alternative text description for this image** |
| --- |

Image Source: Reuven’s [LinkedIn post](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/z2hghnhe87zegotp/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL3Bvc3RzL3JldXZlbmNvaGVuX2dvb2dsZWRlZXBtaW5kcy1uZXctYWdlbnQyYWdlbnQtYTJhLXByb3RvY29sLWFjdGl2aXR5LTczMTU4MTQ2MDAwNzQwMDY1MjgtOW0xOT9yY209QUNvQUFCaWhzaEFCREZWRnExa1owdllHYUhpYzRiOVg1U3E4MERFJnV0bV9tZWRpdW09bWVtYmVyX2Rlc2t0b3AmdXRtX3NvdXJjZT1zaGFyZQ==)​

As systems get more complex, we’ll move away from handcrafted prompts and toward shared standards. **MCP** (Model Context Protocol) standardizes how we package up structured context (tools, memory, RAG, and prior instructions) into the model.  
​**A2A**, released by Google just a few days back, focuses on agent-to-agent communication, allowing agents across platforms to collaborate with a shared schema.  
Expect cleaner abstractions going forward, though it’ll take time before any of these become as standard as HTTP.

### **2. Hybrid Reasoning Models**

Reasoning models are evolving. The future isn’t using one type of model for everything, but having models that know when to plan and when to act fast.  
We’re already seeing this with Claude 3.7 and similar models. The goal is to balance intelligence with efficiency — without overthinking everything.

### **3. Better Memory Systems**

Memory today is mostly duct-taped in. Going forward, expect tighter, smarter, and more context-aware memory integration, memory that knows what to recall, when, and why. Think memory scoped to tasks, sessions, or personas, and easier to manage.

### **4. Tool Ecosystem Maturity**

Right now, everyone’s building custom tools and wrappers. But over time, expect a more robust ecosystem, trusted, plug-and-play APIs, better abstraction layers, and shared security practices.  
The same way microservices matured in traditional software, tools will mature in the agentic stack too.

## **A Final Word**

If you’ve followed along, you’ve probably noticed the theme:

**We didn’t start with architecture. We started with problems.**

That’s the real mindset shift. Don’t chase agents for the hype. Build them when they make solving a problem easier, faster, or smarter.

Start simple. Measure everything. Scale when needed.

Agent-first thinking breaks. Problem-first thinking scales.

Thanks for reading, sharing, replying, and thinking along over these 10 days. Thanks again for following along. If you take away one thing from this series — let it be this: **problem first, always.**

**See you in the live sessions (if you're taking this course before July 2025 !**

-[Aish](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/p8heh9h4lde4vztq/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL2FyZWdhbnRpLw==), [Kiriti](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/x0hph6helzmegeu5/aHR0cHM6Ly93d3cubGlua2VkaW4uY29tL2luL3NhaS1raXJpdGktYmFkYW0v)​

*If this was helpful, feel free to forward it to someone looking to learn in this space.*

*And if you’d like to go deeper, our**​*[*full 6-week course covers system design*](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/dpheh0hemkxe3khm/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu)*, applied agentic concepts, and real evaluation workflows, the kind that support production-grade applications.*

*The course is built for everyone, whether you’re a Product Manager, Architect, Director, C-suite leader, or someone seriously exploring agentic AI.*

*P.S. Sharing two of our favorite cohort reviews below. You can find 60+ more on the course site. We’re the only AI course on the platform with a full 5-star rating.*

|  |
| --- |

*Use the code 10DAY for a limited time 10% off!*

| [Register Now!](https://click.convertkit-mail2.com/d0uq24974vfmu443og6imhzpvrq44clhovq7z/dpheh0hemkxe3khm/aHR0cHM6Ly9tYXZlbi5jb20vYWlzaHdhcnlhLWtpcml0aS9nZW5haS1zeXN0ZW0tZGVzaWdu) |
| --- |