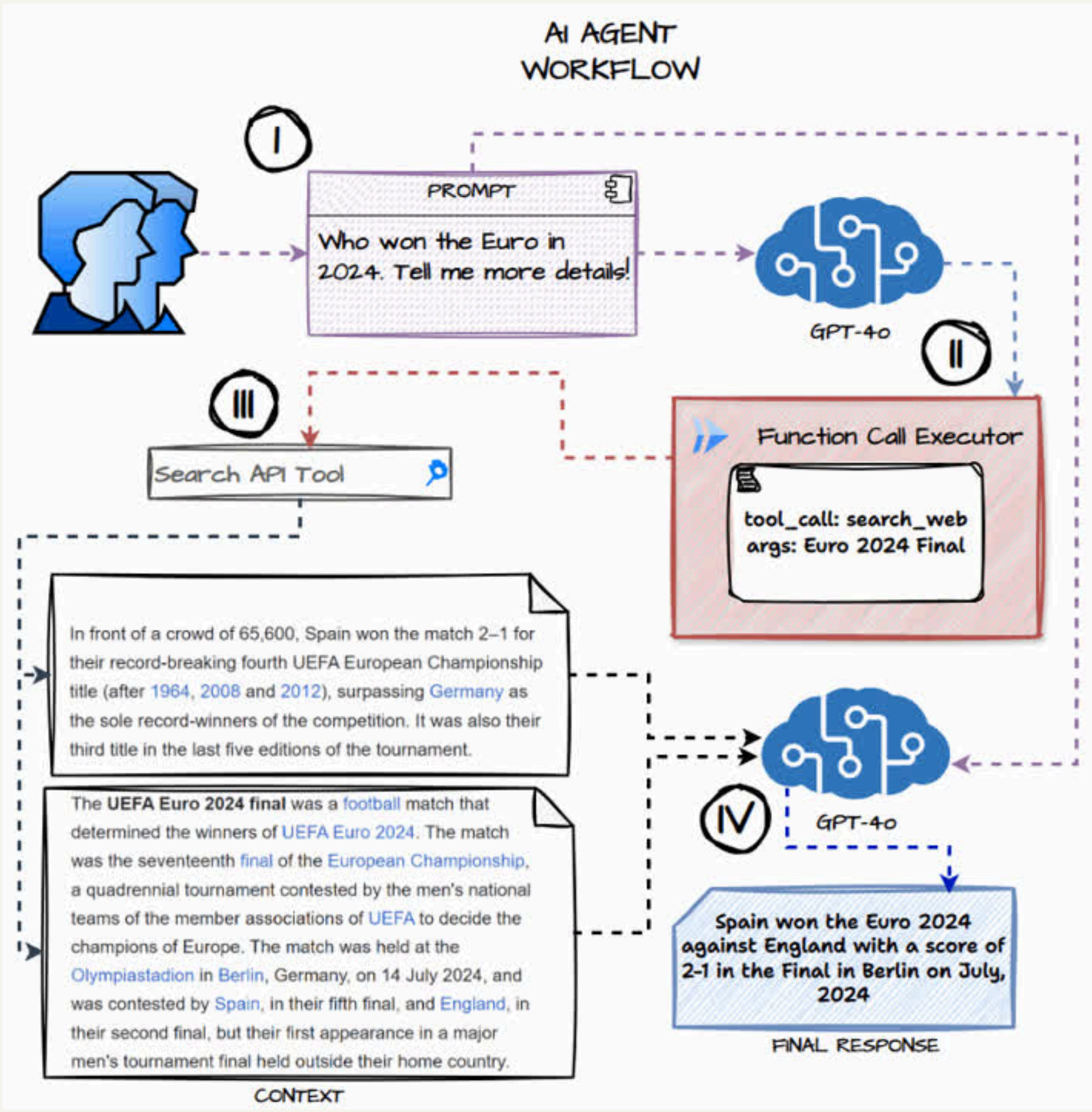
Guide to Building Effective AI Agents

Credits to: ANTHROP\C

Dipanjan (DJ)

What are Agents?

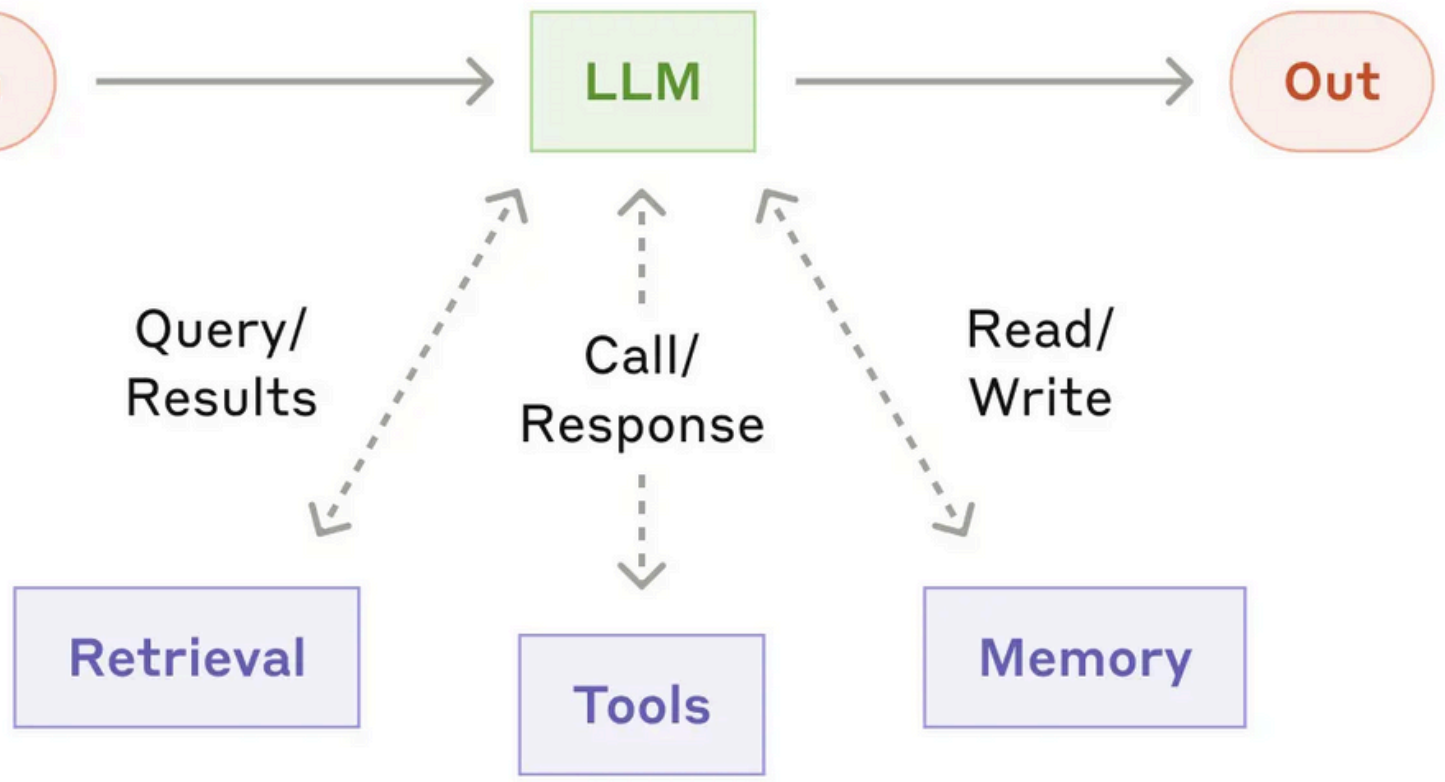


- **An Agentic AI System is usually an autonomous system that operates independently over extended periods, using various tools and flows to accomplish complex tasks**
- **Anthropic goes one step further to do a more fine-grained categorization:**
  - **Workflows** are systems where LLMs and tools are orchestrated through predefined code paths.
  - **Agents,** on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks.

Source: Building effective agents - Anthropic; Image by Author

Created by: Dipanjan (DJ)

Building Block for Agentic AI Systems - The Augmented LLM



- **The basic building block of Agentic AI Systems is an LLM enhanced with augmentations such as retrieval, tools, and memory**

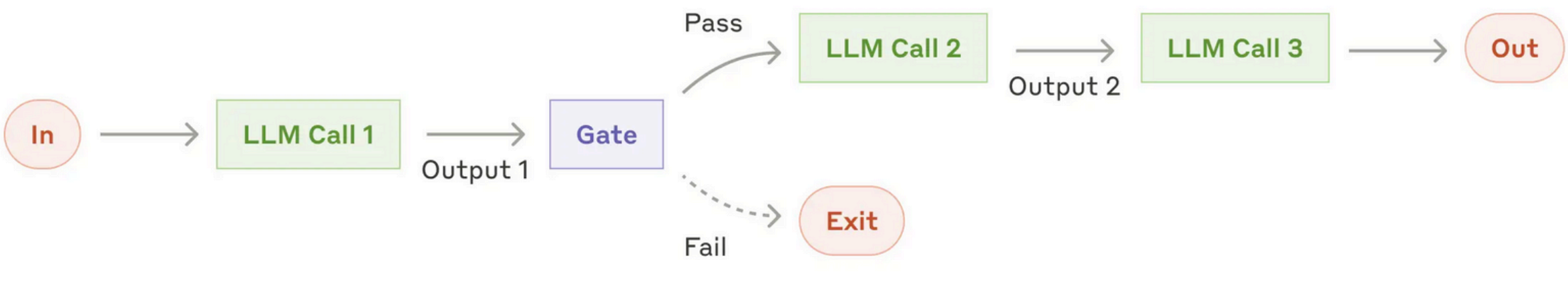
- **Powerful LLM platforms have these in-built. When using APIs you would need to connect the LLM with relevant tools, memory and databases so that they can generate their own search queries, select appropriate tools, and determine what information to retain.**

- **Anthropic recommends focusing on two key aspects of the implementation:**
  - **Tailoring these capabilities to your specific use case**
  - **Ensuring they provide an easy, well-documented interface for your LLM**

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Workflow: Prompt Chaining



- **Prompt Chaining decomposes a task into a sequence of steps, where each LLM call processes the output of the previous one (similar to least-to-most prompting)**

- **You can add programmatic checks (see "gate" in the diagram above) on any intermediate steps to ensure that the process is still on track.**

- **This workflow is ideal for situations where the task can be easily and cleanly decomposed into fixed subtasks**

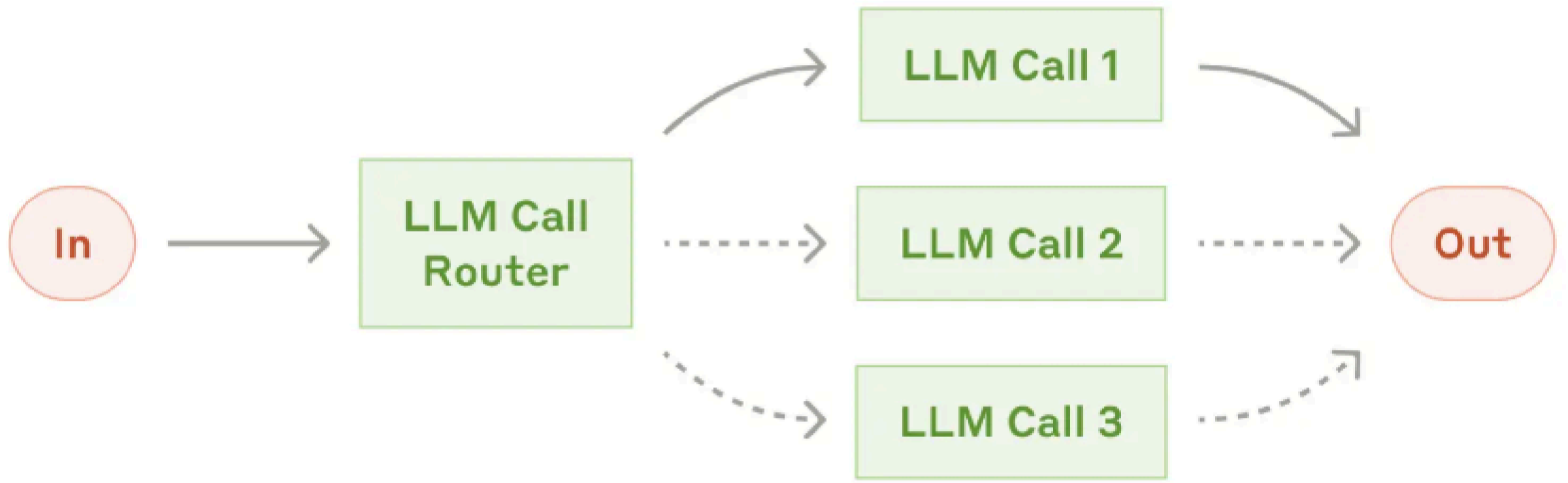
- **The main goal is to trade off latency for higher accuracy, by making each LLM process an easier task.**

- **Examples where prompt chaining is useful:**
  - Generating Marketing copy, then translating it into a different language.
  - Writing an outline of a document, checking that the outline meets certain criteria, then writing the document based on the outline.

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Workflow: Routing

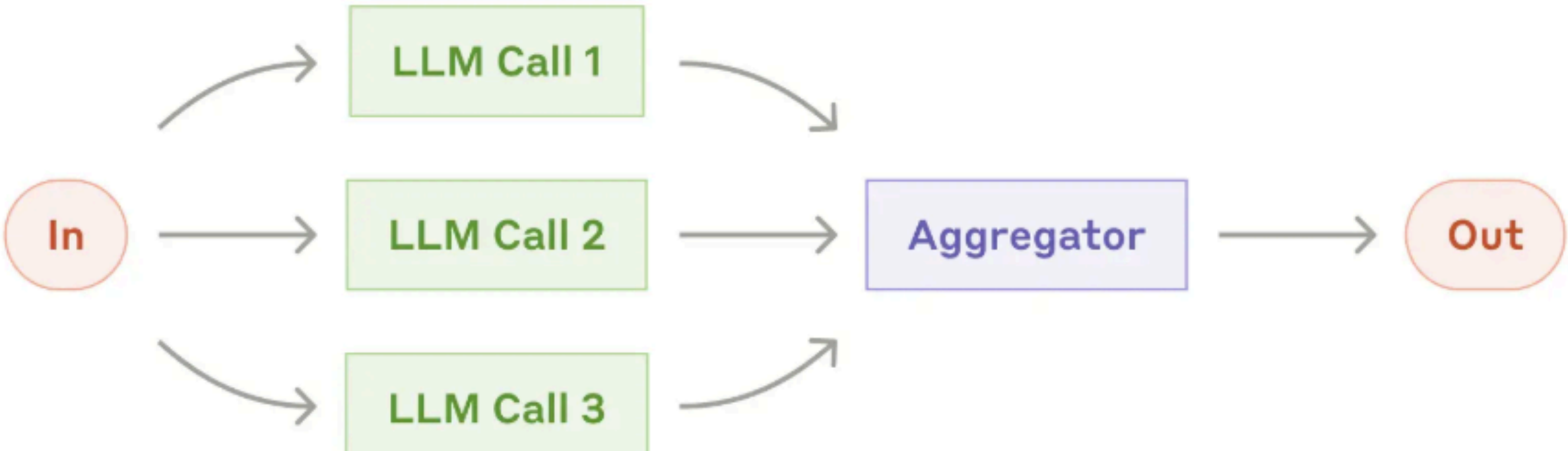


- **Routing classifies an input and directs it to a specialized followup task.**
- **The followup task could be a single LLM call or a chain of multiple steps involving LLM, tool calls, hits to a DB etc. depending on a specific task flow.**
- **This workflow allows for separation of concerns, and building more specialized prompts.**
- **Without this workflow, optimizing for one kind of input can hurt performance on other inputs.**
- **Routing works well for complex tasks where there are distinct categories that are better handled separately, and where classification can be handled accurately, either by an LLM or a more traditional classification model/algorithm.**
- **Popular open-source router frameworks include Route0x & Semantic Router**
- **Examples where routing is useful:**
  - Directing different types of customer service queries (general questions, refund requests, technical support) into different downstream processes, prompts, and tools.
  - Routing easy/common questions to smaller models like Claude 3.5 Haiku and hard/unusual questions to more capable models like Claude 3.5 Sonnet to optimize cost and speed.

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Workflow: Parallelization



- **LLMs can sometimes work simultaneously on a task and have their outputs aggregated programmatically**

- **There are two variations of the parallelization workflow:**
  - **Sectioning:** Breaking a task into independent subtasks run in parallel.
  - **Voting:** Running the same task multiple times to get diverse outputs. (like in self-consistency prompting)

- **Parallelization is effective when the divided subtasks can be parallelized for speed, or when multiple perspectives or attempts are needed for higher confidence results**

- **For complex tasks with multiple considerations, LLMs generally perform better when each consideration is handled by a separate LLM call, allowing focused attention on each specific aspect.**

- **Examples where parallelization is useful:**
  - **Sectioning:**
    - Implementing guardrails where one model instance processes user queries while another screens them for inappropriate content or requests.
    - Automating evals for evaluating LLM performance, where each LLM call evaluates a different aspect of the model's performance on a given prompt.
  - **Voting:**
    - Reviewing a piece of code for vulnerabilities, where several different prompts review and flag the code if they find a problem.
    - Evaluating whether a given piece of content is inappropriate, with multiple prompts evaluating different aspects or requiring different vote thresholds to balance false positives and negatives.

Source: Building effective agents - Anthropic

Workflow: Orchestrator-Workers

- **In the orchestrator-workers workflow, a central LLM dynamically breaks down tasks, delegates them to worker LLMs, and synthesizes their results.**

- **This workflow is well-suited for complex tasks where you can’t predict the subtasks needed (in coding, for example, the number of files that need to be changed and the nature of the change in each file likely depend on the task).**

- **The key difference from parallelization is its flexibility—subtasks aren't pre-defined, but determined by the orchestrator based on the specific input (often used in agentic systems with branching chains).**

- **Example where orchestrator-workers is useful:**
  - Coding products that make complex changes to multiple files each time.
  - Search tasks that involve gathering and analyzing information from multiple sources for possible relevant information.

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Workflow: Evaluator-Optimizer

- **In the evaluator-optimizer workflow, one LLM call generates a response while another provides evaluation and feedback in a loop (just like reflection agents).**

- **This workflow is particularly effective when we have clear evaluation criteria, and when iterative refinement provides measurable value.**

- **The two signs of good fit are:**
  - LLM responses can be demonstrably improved when a human articulates their feedback
  - The LLM can provide such feedback.

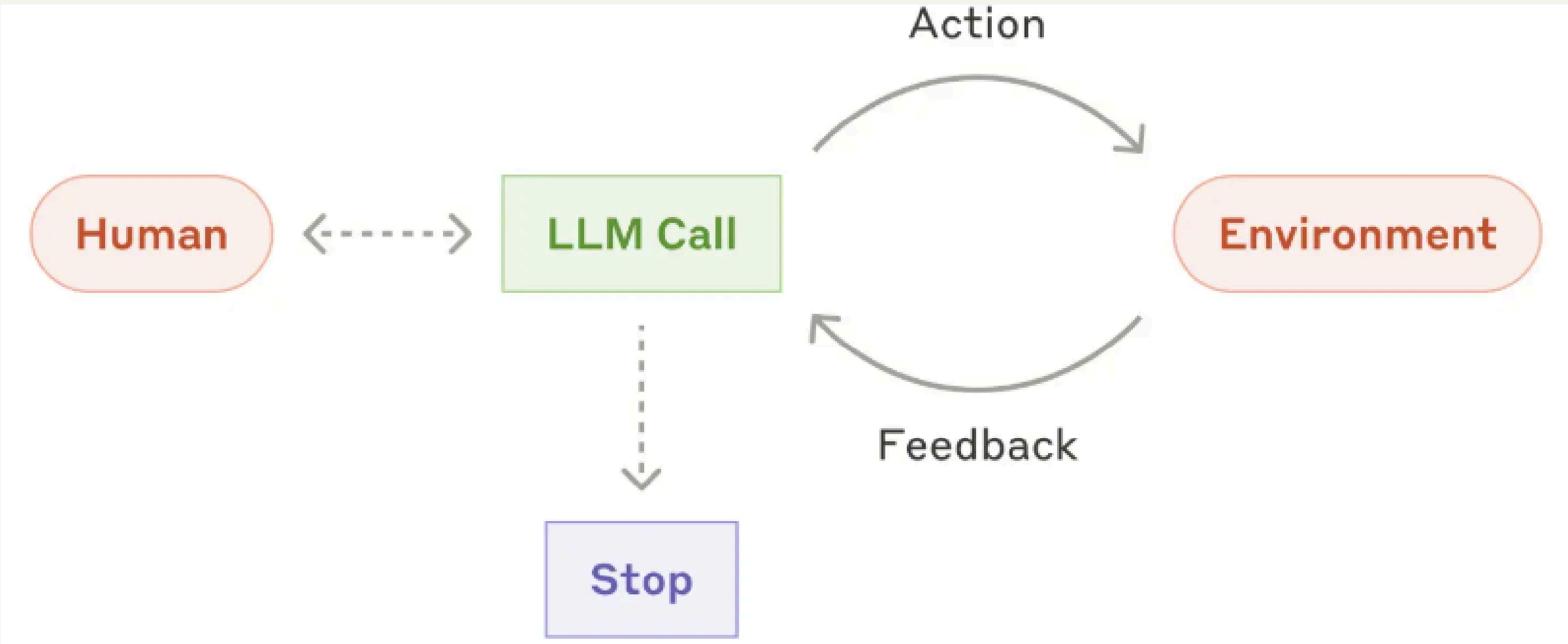
- **This is analogous to the iterative writing process a human writer might go through when producing a polished document or when developed code is reviewed, tested and then improved.**

- **Examples where evaluator-optimizer is useful:**
  - Literary translation where there are nuances that the translator LLM might not capture initially, but where an evaluator LLM can provide useful critiques.
  - Complex search tasks that require multiple rounds of searching and analysis to gather comprehensive information, where the evaluator decides whether further searches are warranted.

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Autonomous Agents

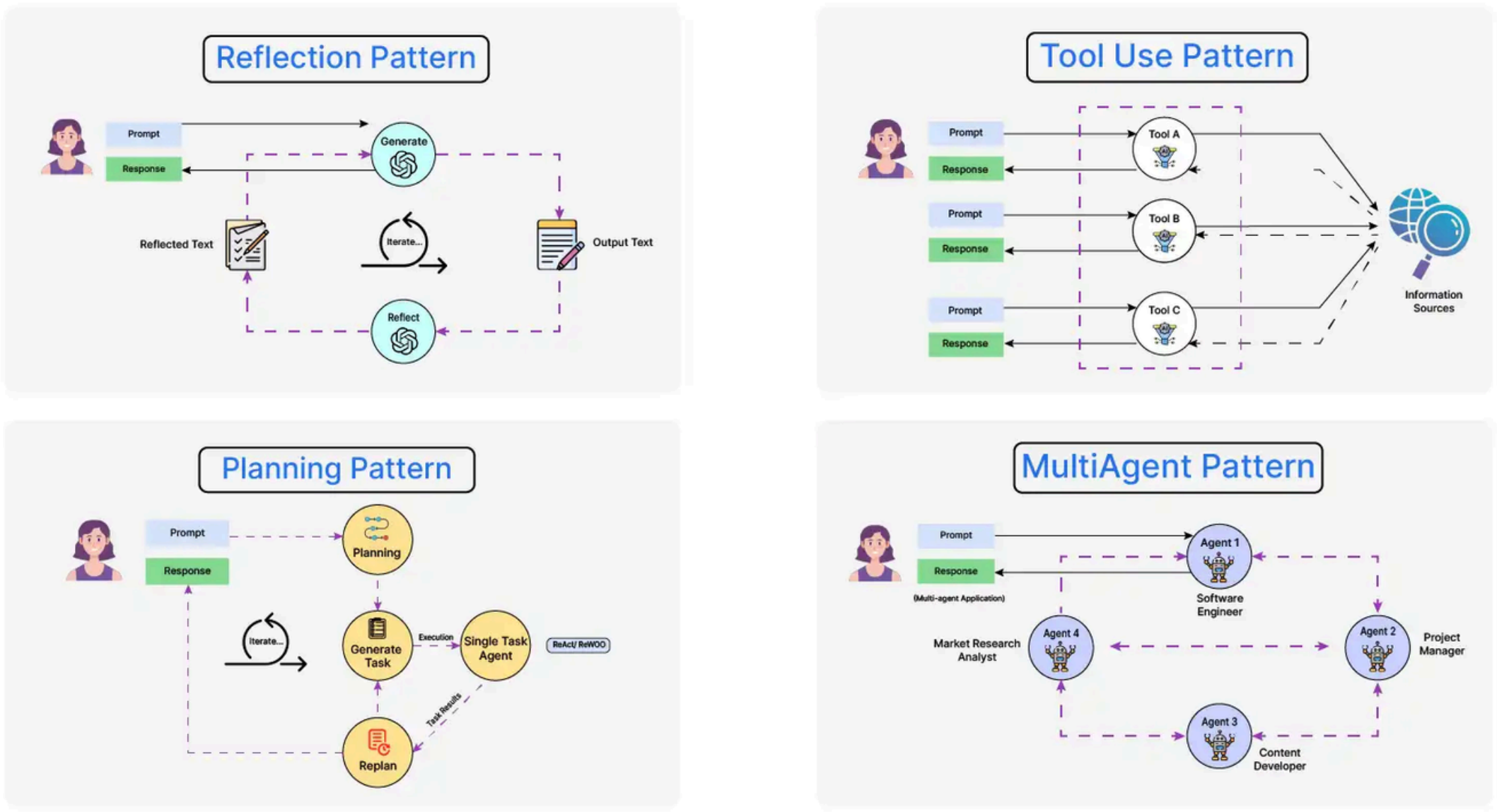


- **Autonomous agents focus on understanding complex inputs, engaging in reasoning and planning, using tools reliably, and recovering from errors**
- **Agents begin their work with either a command from, or interactive discussion with, the human user.**
- **Once the task is clear, agents plan and operate independently or sometimes collaboratively, potentially returning to the human for further information or judgement.**
- **Agents can then pause for human feedback at checkpoints or when encountering blockers.**
- **The task often terminates upon completion, but it's also common to include stopping conditions (such as a maximum number of iterations) to maintain control.**
- **In simple terms, agents are just LLMs using tools based on environmental feedback in a loop and following certain flows as necessary**

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)

Autonomous Agents

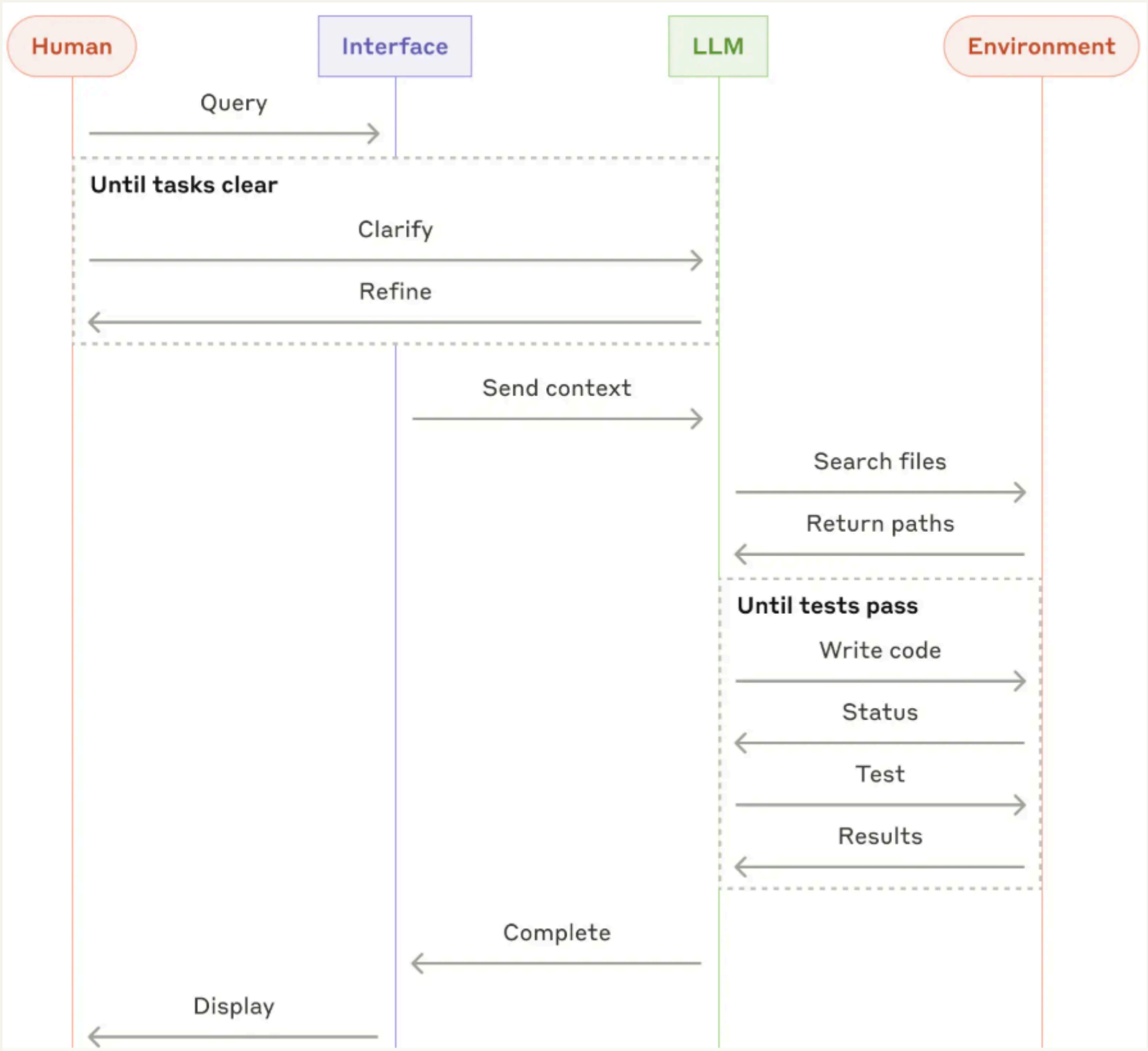


- **Agents can be used for open-ended problems where it's difficult or impossible to predict the required number of steps, and where you can't hardcode a fixed path.**
- **The LLM will potentially operate for many turns, and you must have some level of trust in its decision-making**
- **The autonomous nature of agents means higher costs, and the potential for compounding errors.**
- **Try using specific design patterns as mentioned above to build and architect robust Agentic AI systems which follow well-defined flows.**
- **Check out this article for more detailed information**

Source: Building effective agents - Anthropic & Agentic AI Design Patterns - Analytics Vidhya

Created by: Dipanjan (DJ)

Summary & Final Words



High-level flow of a coding agent

- **Success in the LLM space isn't about building the most sophisticated system. It's about building the right system for your needs.**

- **Start with simple prompts, optimize them with comprehensive evaluation, and add multi-step agentic systems only when simpler solutions fall short.**

    - **Maintain simplicity in your agent's design.**
    - **Prioritize transparency by explicitly showing the agent's planning steps.**
    - **Carefully craft your agent-computer interface (ACI) through thorough tool documentation and testing.**

- **Frameworks can help you get started quickly, but don't hesitate to reduce abstraction layers and build with basic components as you move to production.**

Source: Building effective agents - Anthropic

Created by: Dipanjan (DJ)