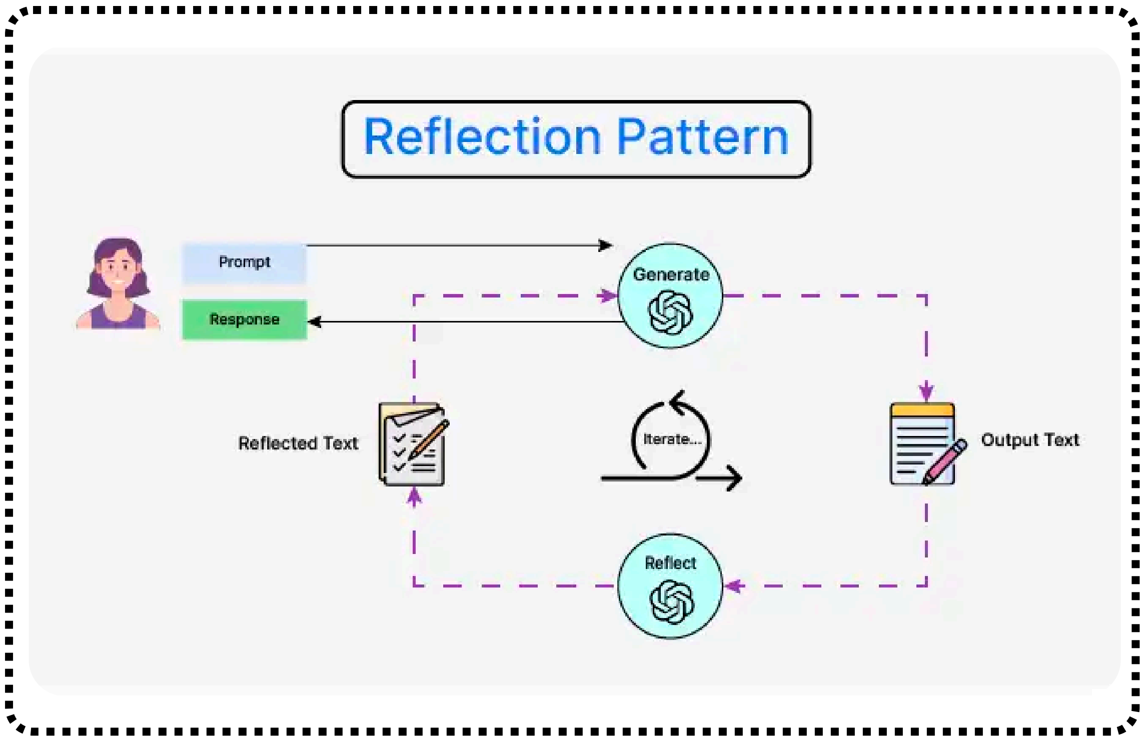
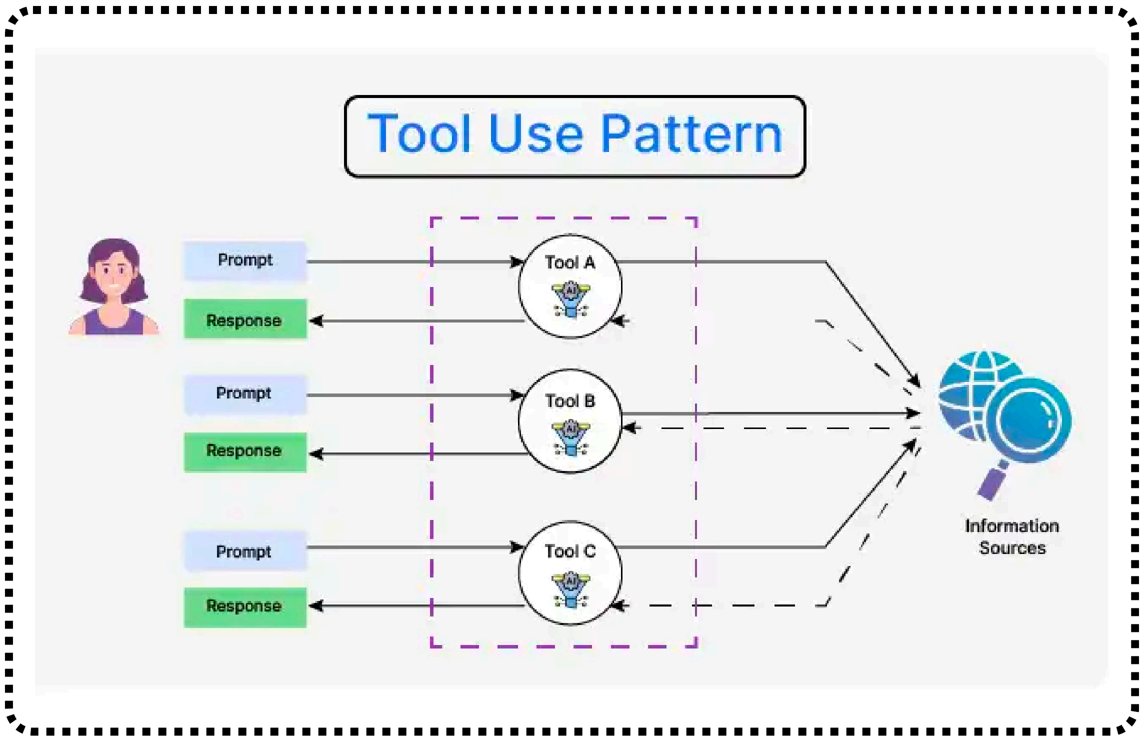
Analytics Vidhya

Day 6 of Mastering AI Agents

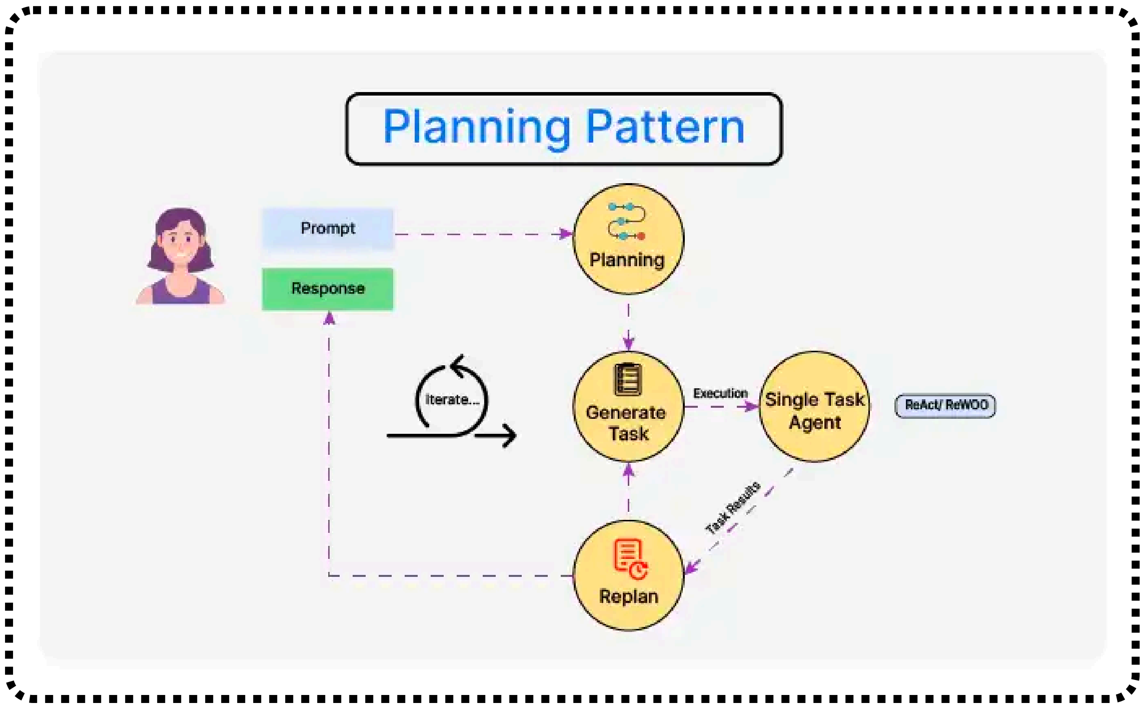
Agentic AI Design Patterns



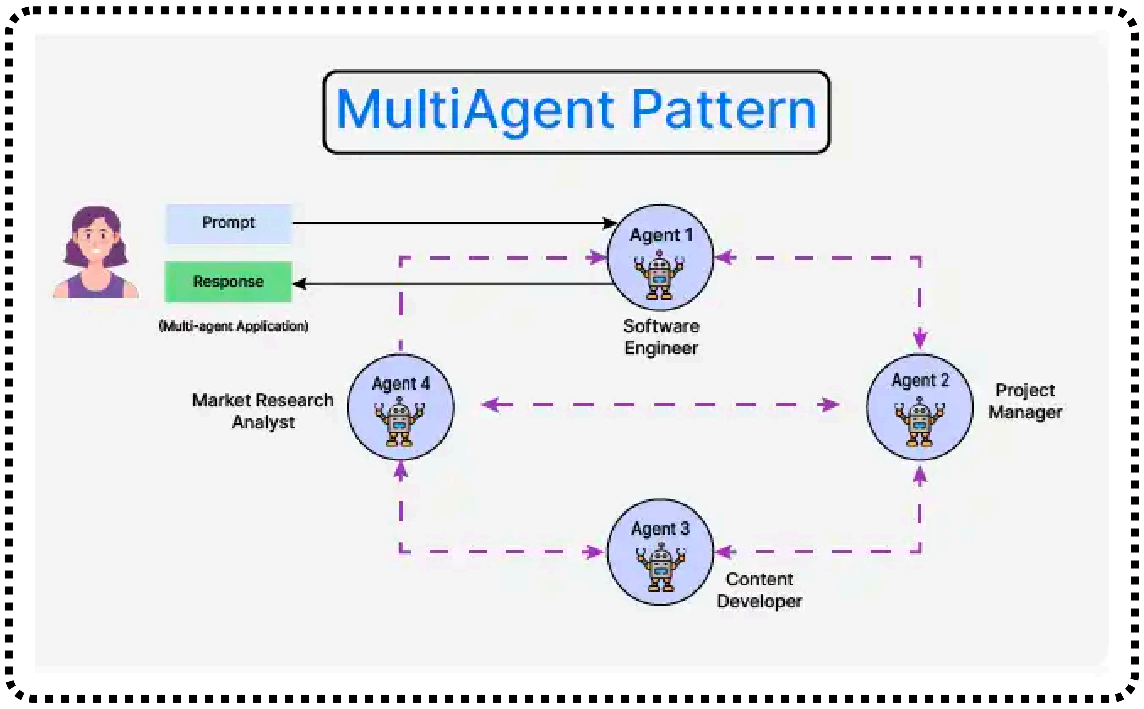
Reflection Pattern



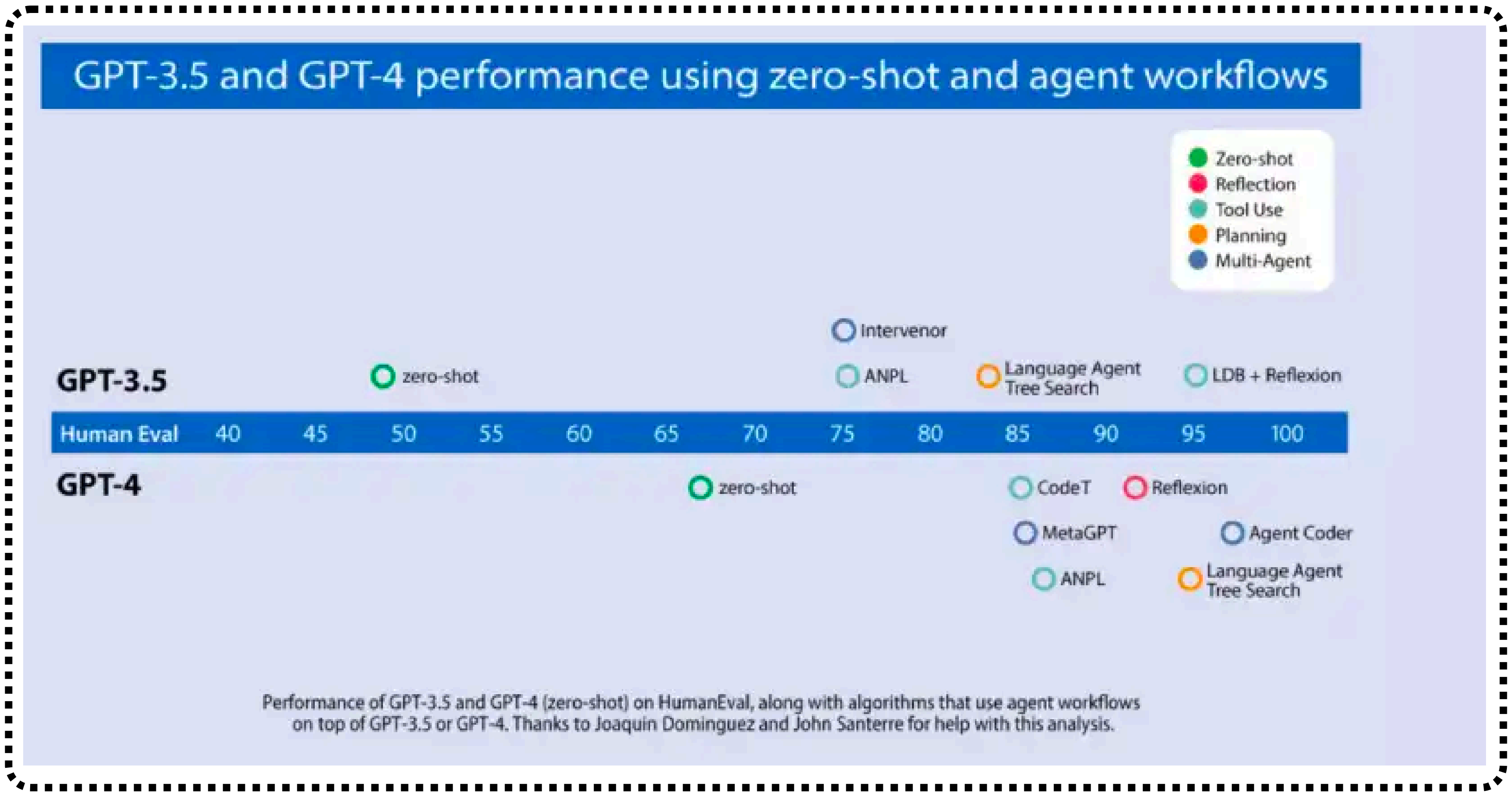
Tool Use Pattern



Planning Pattern



MultiAgent Pattern



GPT-3.5 and GPT-4 performance using zero-shot and agent workflows

Performance of GPT-3.5 and GPT-4 (zero-shot) on HumanEval, along with algorithms that use agent workflows on top of GPT-3.5 or GPT-4. Thanks to Joaquin Dominguez and John Santerre for help with this analysis.

What is Agentic AI?

Agentic AI is introduced as a solution for making LLMs more autonomous. Instead of just giving the model one prompt and expecting a final answer (like writing an essay in one go), an agent-like approach involves prompting the LLM multiple times, step by step. Each step refines the task, with the model improving its output iteratively.

To understand this better, let's look at it like this:

When we prompt an LLM in zero-shot mode, it's like asking someone to write a story in one go without revising. LLMs do well at this, but they can do even better. By using an agent-like workflow, we can prompt the LLM multiple times in steps. Each step builds on the previous one, refining the response.

Think of it like asking the LLM to go over the essay multiple times, improving it with each pass.

By each step, I meant:

Let's take the example of writing a code using Agentic workflow:

- **Plan an outline for the code**: Break down the task into smaller modules or functions.

- **Gather information and content**: Research libraries, algorithms, or existing solutions. Do web searches or check the documentation if needed.

- **Write the first draft of the code**: Implement the basic functionality, focusing on structure over perfection.

- **Review the code for inefficiencies or errors**: Check for unnecessary code, bugs, or logic flaws.

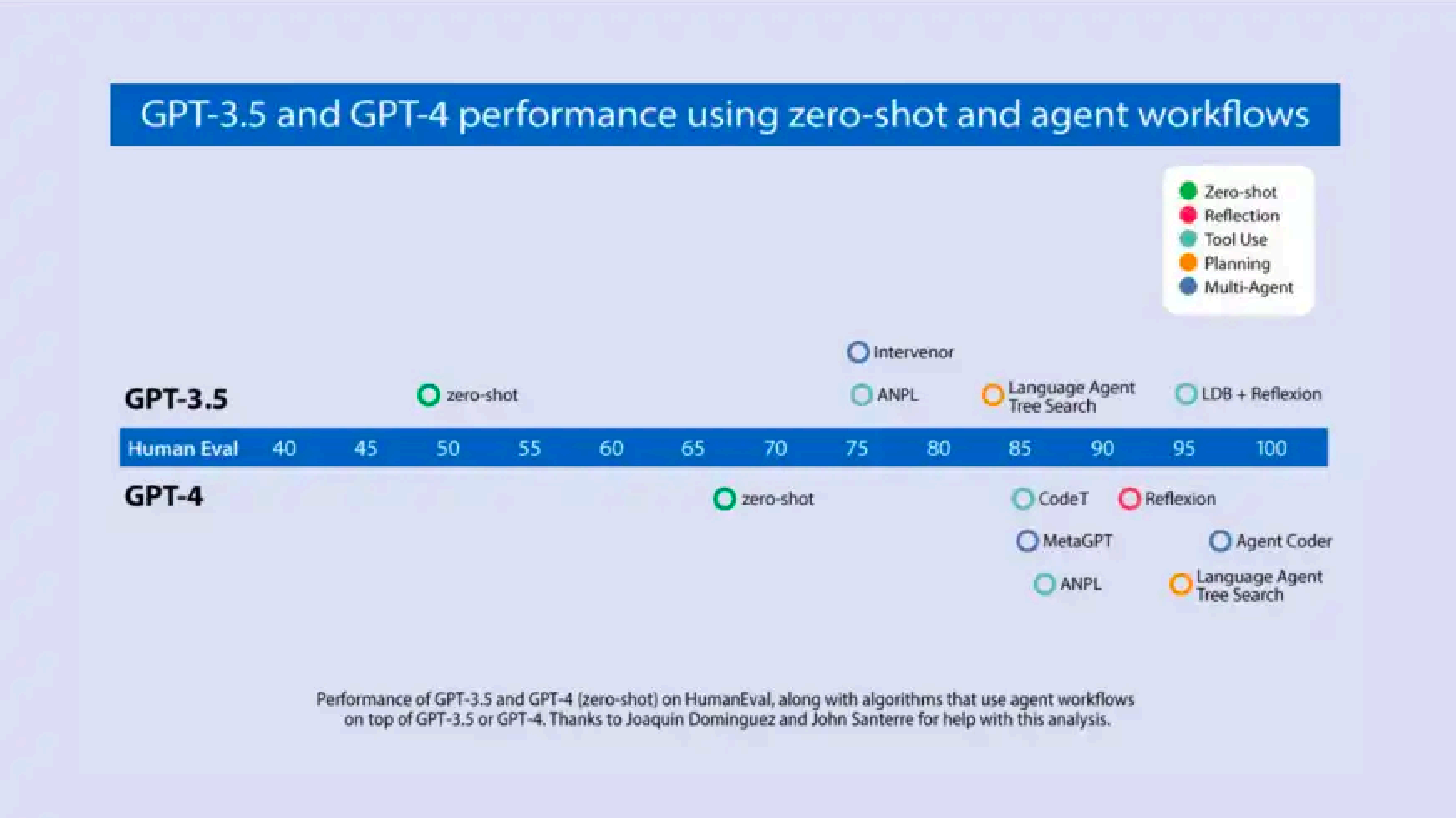
- **Revise the code**: Refactor, optimise, or add comments for clarity.

Rinse and repeat until the code is efficient and clean.
By allowing the model to work through these steps independently, the agentic design pattern enhances both human-like reasoning and efficiency.

This is similar to how humans break down complex tasks, gather information, make improvements, and iterate until the final result is satisfactory.

Now, let us understand Agentic Design Patterns in detail.

Agentic Design Patterns: Evaluations


GPT-3.5 and GPT-4 performance using zero-shot and agent workflows

Performance of GPT-3.5 and GPT-4 (zero-shot) on HumanEval, along with algorithms that use agent workflows on top of GPT-3.5 or GPT-4. Thanks to Joaquin Dominguez and John Santerre for help with this analysis.

- Andrew Ng's analysis, shared in a letter on Deeplearning.ai, noted advancements in AI-driven code generation, particularly focusing on the performance of models like GPT-3.5 and GPT-4. The evaluation was centred on these models' capabilities to perform on the widely recognized HumanEval coding benchmark, a common standard for assessing an algorithm's proficiency in writing code.

- The data presented shows the evolution in AI coding abilities using AI agents. GPT-3.5, when tested in a zero-shot setting (i.e., without any prior examples), achieved a correctness rate of 48.1%. GPT-4, also evaluated in a zero-shot manner, demonstrated a significant improvement, with a 67.0% success rate. However, what stood out in the analysis was how integrating these models into an iterative agent workflow (Agentic workflow) drastically boosted their performance. When GPT-3.5 was wrapped in such an agent loop, its accuracy soared to an impressive 95.1%, far surpassing its baseline and even approaching human-level coding proficiency.

- This finding underscores the transformative potential of iterative workflows (Agentic workflow) in enhancing AI model performance, suggesting that the future of AI-assisted coding may heavily rely on these more advanced, adaptive frameworks rather than on model size or architecture improvements alone.

- But what are Agentic design patterns that complete the delegation of autonomy to AI systems, enabling them to act more independently and effectively? These patterns structure AI agents to perform tasks, make decisions, and communicate with other systems in a more human-like and autonomous manner, ultimately creating both savvy and dependable applications.≣

Why Do We Need Agentic Design Patterns?

Complexity Management

- Agentic patterns simplify the complexity involved in building intelligent, adaptive software by providing tested structures and methodologies. This avoids repetitive solutions and ensures consistency across different implementations.

Scalability and Modularization

- Using standardized patterns, agent-based software systems become easier to scale and modularize. Patterns facilitate encapsulation, which enables straightforward enhancements or addition of new features and agents.

Enhanced Autonomy and Flexibility

- Traditional software design can be rigid, relying heavily on direct human oversight. Agentic patterns allow software agents greater flexibility in operating autonomously, reacting quickly and intelligently to changing environmental conditions.

Advantages

Reusable and Proven Solutions: Agentic patterns offer reusable templates derived from proven solutions, significantly reducing design complexity and development time.

Improved Maintainability: Structured design patterns enable clearer separation of concerns. Maintenance and enhancements become more manageable, reducing long-term costs.

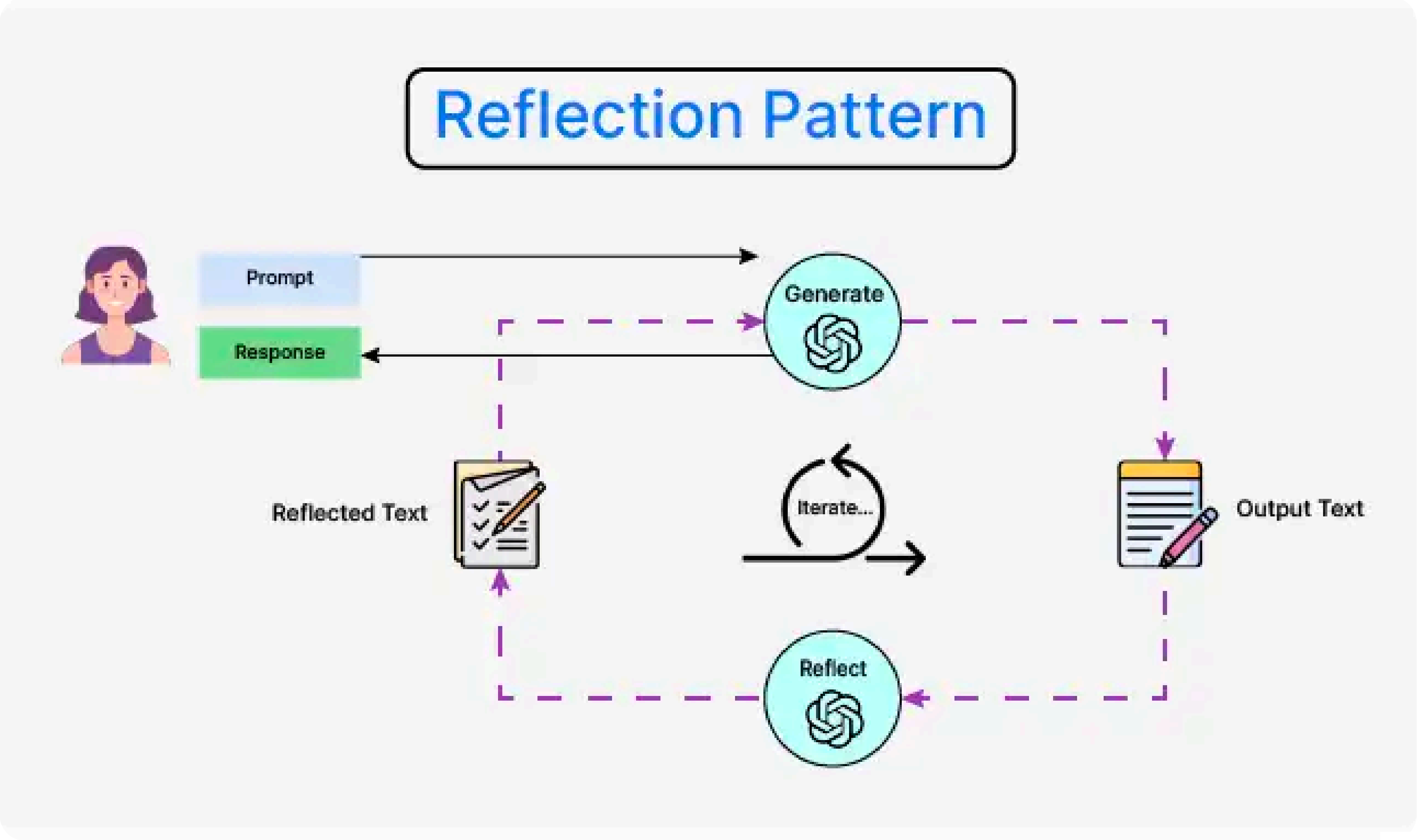
Higher Reliability and Robustness: Using tested and refined patterns enhances the reliability and robustness of software systems by systematically handling known challenges.

Efficiency in Problem-Solving: Agent-based approaches, guided by established patterns, handle complex tasks more efficiently, especially in dynamic and uncertain environments where traditional centralized solutions struggle.

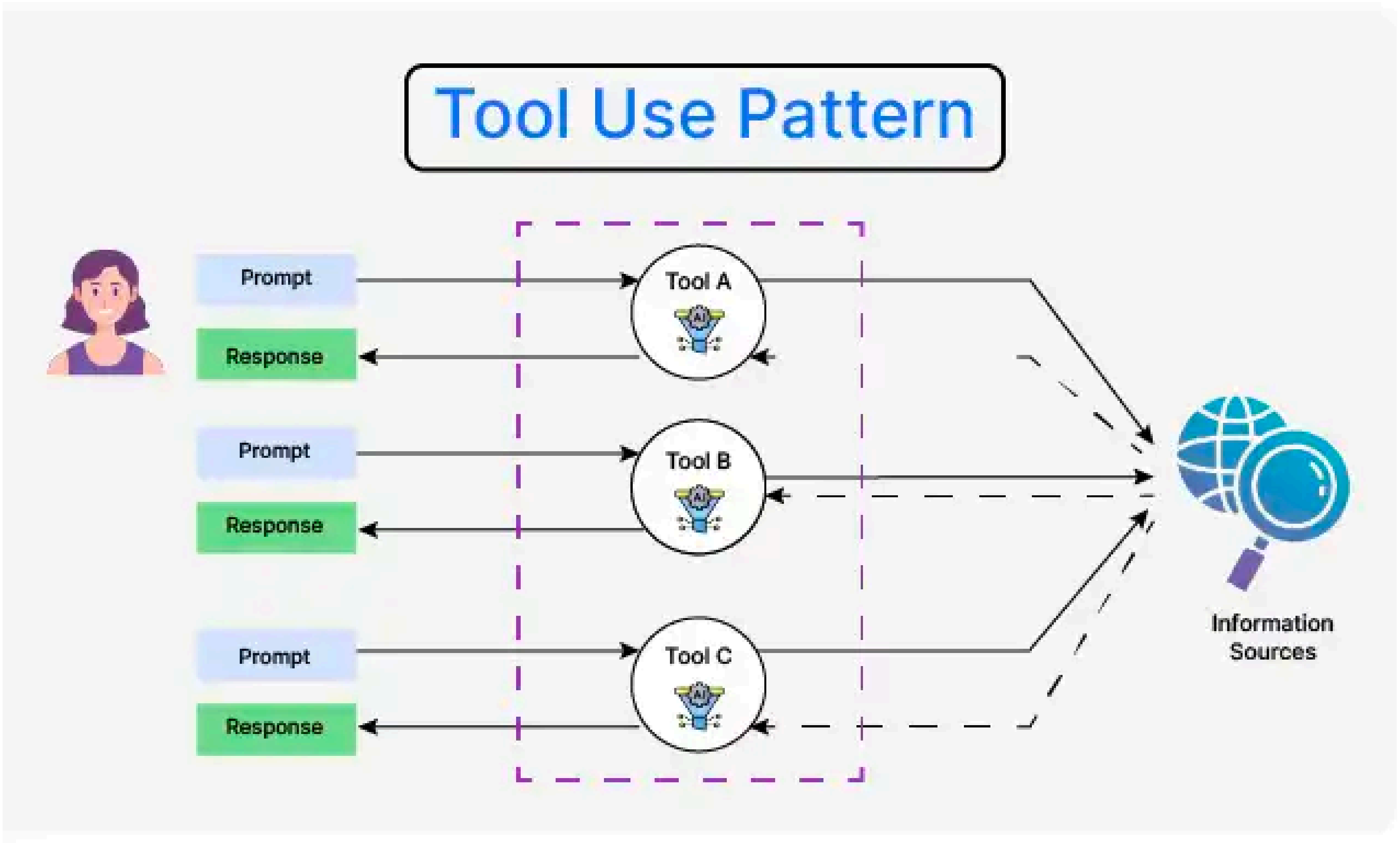
Scalable and Distributed Operation: Agentic patterns inherently support distributed operations, enabling systems to scale effectively as more agents or computational resources become necessary.

4 Types of Agentic Design Patterns

Agentic design patterns provide foundational blueprints for developing sophisticated autonomous software agents. Understanding these patterns is critical for building intelligent, robust, and scalable agent-based systems. Below, we discuss four essential agentic design patterns you must know:



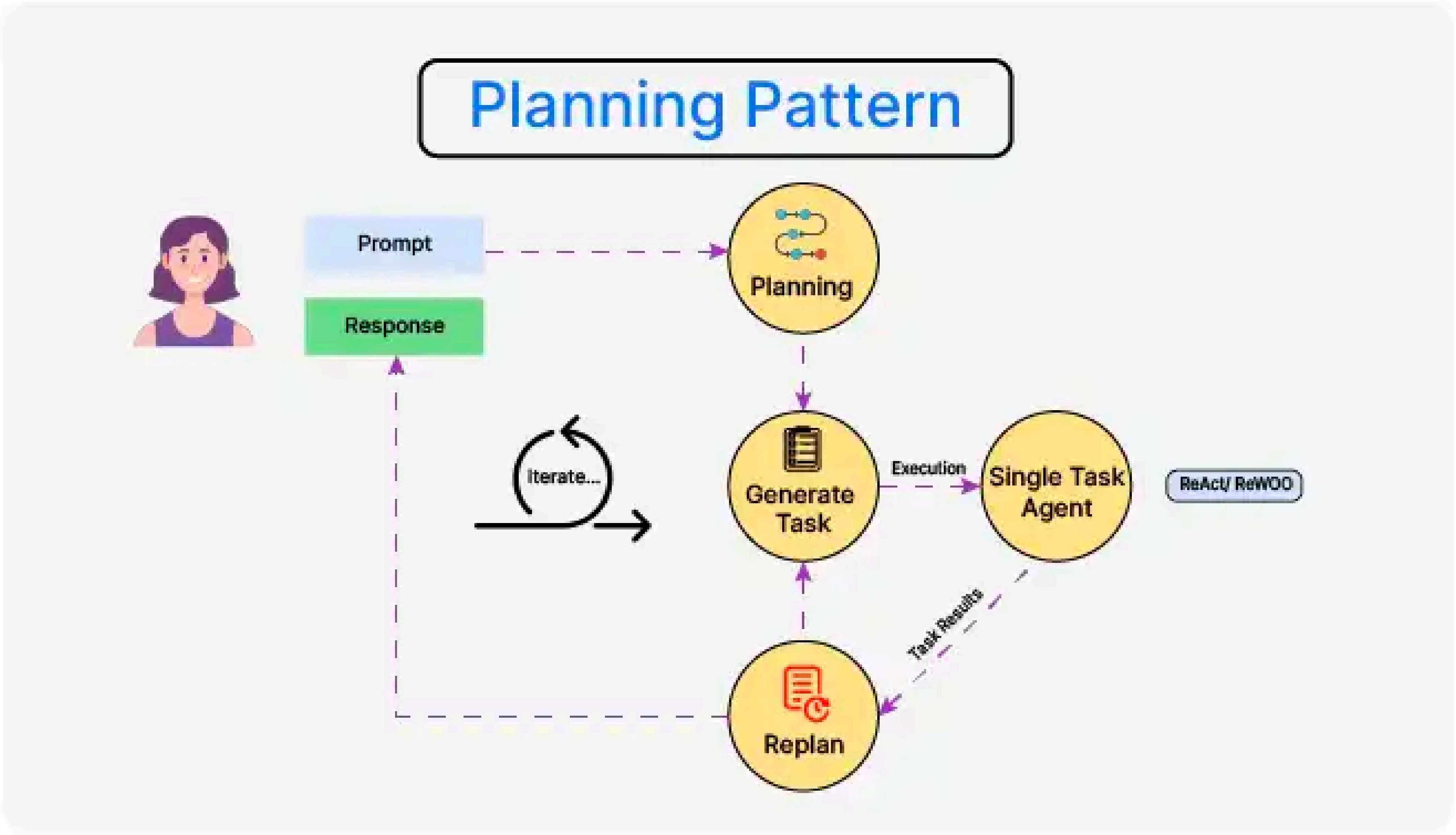
The Reflection Pattern allows agents to assess, reason about, and adjust their own internal states and behaviors dynamically. Agents employing reflection monitor their operations, identify issues



The Tools Use Pattern refers to agents capable of identifying, selecting, and employing external tools or resources to achieve their goals. This pattern involves intelligent agents integrating external software services, APIs, databases, or physical resources dynamically as "tools."

Key Components:

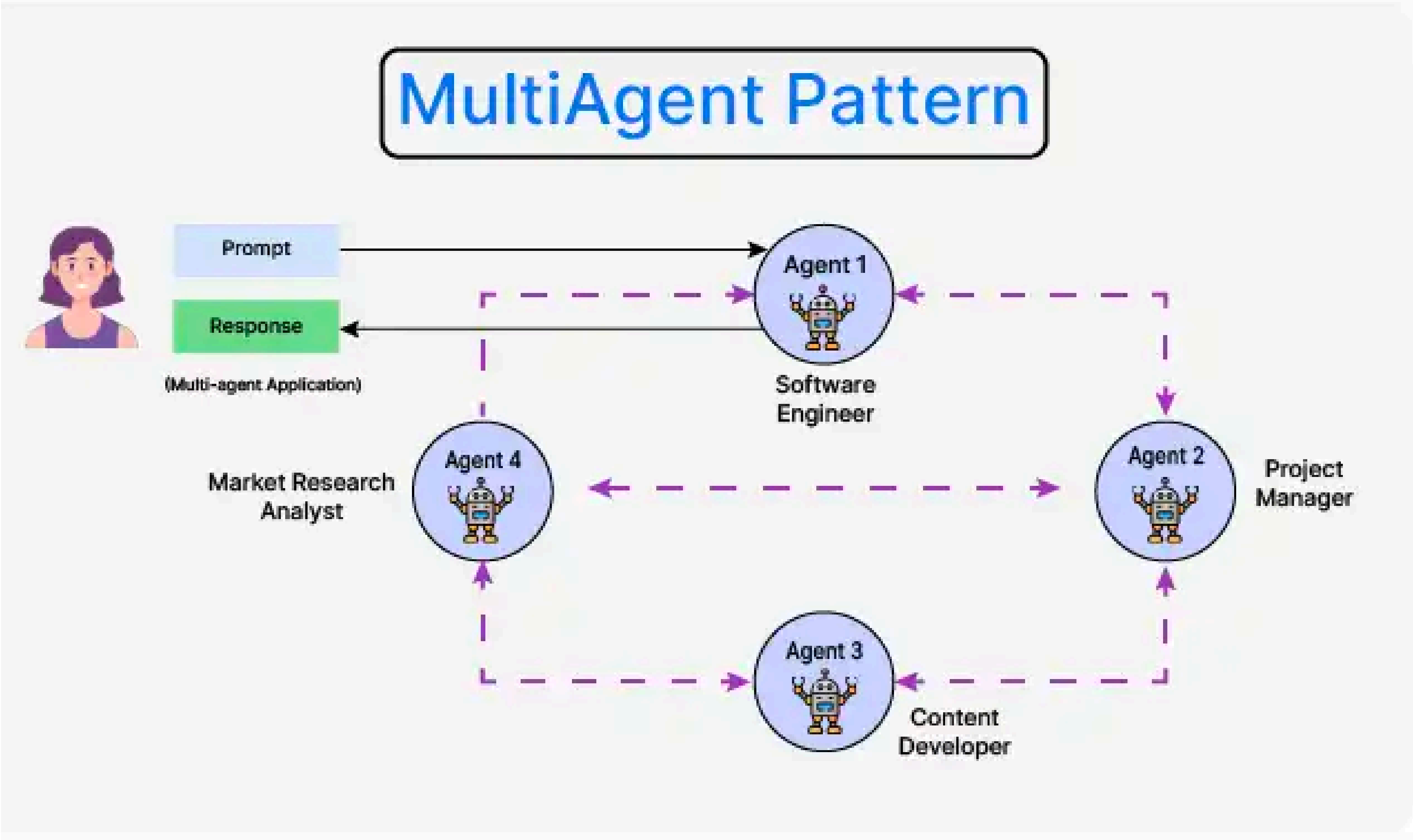
- **Tool Selection**: Agents identify suitable external tools or services.
- **Dynamic Integration**: Agents dynamically integrate and invoke selected tools into their workflow.
- **Abstraction Layer**: An intermediary layer simplifies communication between agents and external tools.



The Planning Pattern equips agents with capabilities to generate sequences of actions systematically aimed at achieving clearly defined goals. Agents using planning patterns explicitly represent their goals, tasks, and available resources to determine optimal action sequences proactively.

Key Components:

- **Goal Representation**: Explicitly defined objectives guiding agent actions.
- **Plan Generation**: Algorithms (such as STRIPS, PDDL planners, or heuristic planners) produce action sequences.
- **Plan Execution and Monitoring**: Continuous monitoring of plan execution with dynamic re-planning when deviations occur.



The Multi-Agent Planning Pattern extends single-agent planning by facilitating coordinated, collaborative planning among multiple autonomous agents. It emphasizes joint strategy formulation, communication, task allocation, and synchronization across multiple independent entities.

Key Components:

- **Distributed Planning**: Decentralized or partially centralized planning approaches, enabling agents to contribute individually or collectively to a global plan.
- **Communication Protocols**: Structured message-passing frameworks (such as Contract Net Protocol) for negotiation, collaboration, and conflict resolution.