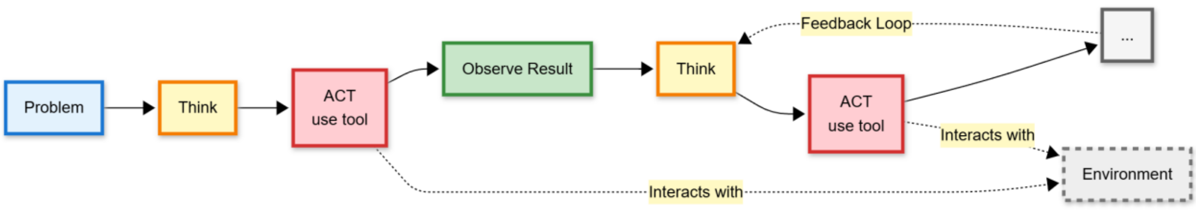
Top 25 GenAI Patterns in Production: Agentic and Non-Agentic

Contents

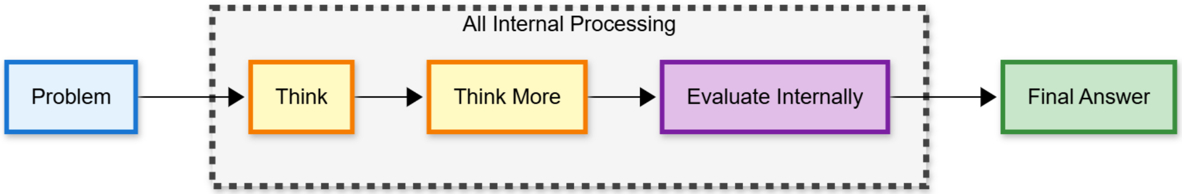
AI systems in production currently follow two dominant architecture, **Agentic** and **Non-Agentic**.

This collection shows the top 25 AI patterns that power real-world applications, explaining how both approaches shape modern AI systems with clear architecture and code.

Agentic Patterns: Agentic patterns focus on ***action-oriented intelligence*** where AI systems interact with external environments, use tools, receive feedback, and take actions that change state to achieve goals autonomously.



Non-Agentic Patterns: Non-Agentic patterns, on the other hand, represent ***cognitive intelligence,*** pure reasoning, prompting, and generation techniques that operate entirely within the LLM without external interaction.



Agentic Patterns

Systems that interact with the external world

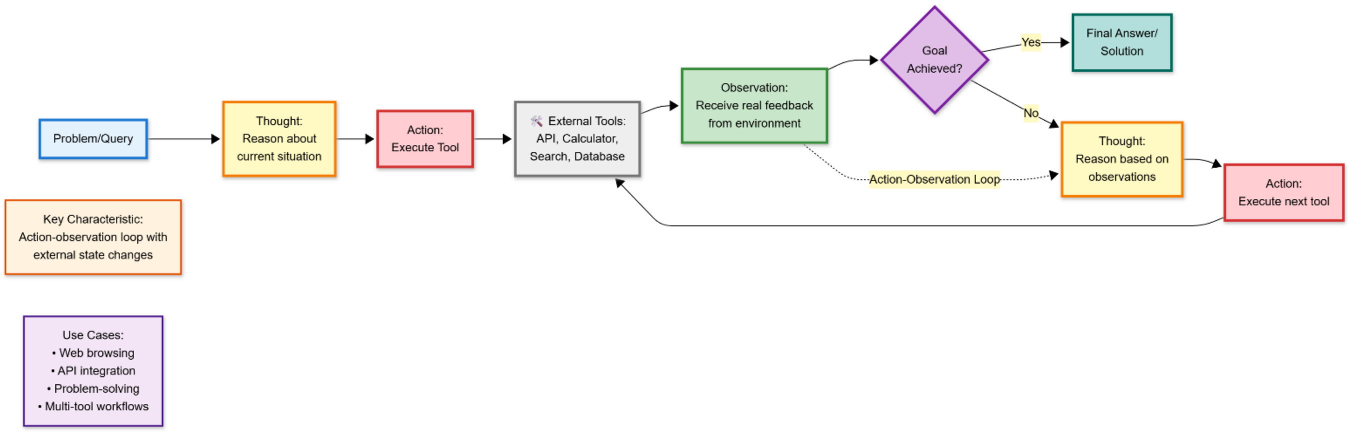
1. ReAct (Reasoning and Acting)

Paper: [ReAct: Synergizing Reasoning and Acting in Language Models (Yao et al., 2023)]
Source: Google Research, Princeton University

Why Agentic: ReAct agents bridge reasoning and execution, they don't just **think**, they ***act*** through real-world tools or APIs, observe feedback, and adjust their strategy dynamically based on outcomes.

Process:

1. **Thought:** The LLM reasons about the current state, analyzing context and planning the next step.
2. **Action:** Executes a real action using an external tool (e.g., web search, API call, calculator, database query).
3. **Observation:** Collects actual feedback or results from the external environment.
4. **Reflection:** Interprets observations and updates reasoning accordingly.
5. **Repeat:** Loops through reasoning-action-observation until the goal is achieved.



Key Characteristic: The agent observes what's happening around it, decides what to do next, and performs actions that update things in the outside system or environment.

Use Cases:

- **Web browsing and information gathering:** Agents explore the web, extract facts and summarize relevant data.
- **API integration and automation:** Connect with APIs or services to fetch, update or automate tasks.
- **Interactive problem solving:** Collaborate with users in real time to reason, plan, and refine solutions.
- **Multi tool workflows:** Coordinate multiple tools (e.g., browser, code executor, file manager) to complete complex goals end-to-end.

Code: [Colab Notebook]

---

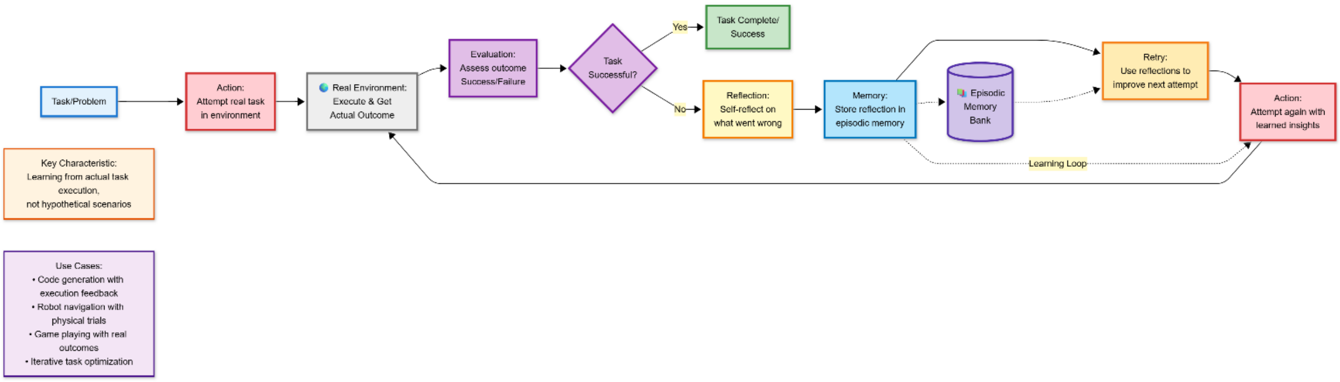
2. Reflexion

Paper: [Reflexion: Language Agents with Verbal Reinforcement Learning (Shinn et al., 2023)]
Source: Northeastern University, MIT

Why Agentic: Reflexion agents perform real-world actions, analyze the actual results, and iteratively refine future behaviour using self-generated verbal feedback. Unlike static reasoning models, Reflexion builds memory from lived experiences, learning how to improve through trial, failure, and correction.

Process:

1. **Action:** The agent executes a real task (e.g., calling an API, running code, performing a search).
2. **Evaluation:** Observes the true outcome, whether the action succeeded, failed, or caused unexpected results.
3. **Reflection:** The agent reasons verbally about mistakes ("Why did this fail?") and identifies improvements.
4. **Memory:** Stores insights and outcomes in **episodic memory*** for future reference.
5. **Retry:** On the next attempt, retrieves past reflections to avoid repeating errors and enhance performance.



***Episodic Memory** in Reflexion is a storage system that records the agent's past attempts, failures, and self-reflections.

It works like a personal experience log where the agent stores:

- What it tried
- What failed
- Why it failed
- What to do differently

When retrying a task, the agent consults this memory to avoid repeating the same mistakes, enabling it to learn and improve from actual execution experiences rather than just general knowledge.

Example: If code crashes with an error, the agent stores "avoided edge case X, caused bug Y" and uses this insight in the next attempt.

Key Characteristic: Learning from actual task execution, not hypothetical scenarios.

Use Cases:

- **Code generation with execution feedback:** The agent writes code, runs it, observes errors or outputs, and improves the code iteratively.
- **Robot navigation with physical trials**: Robots test different movements, observe their results in the real world, and adjust their path accordingly.
- **Game playing with real outcomes:** The agent plays games, learns strategies from wins and losses, and adapts its gameplay dynamically.
- **Iterative task optimization:** Continuously refines its approach based on feedback, improving performance or efficiency with each cycle.

Code: [Colab Notebook]

---

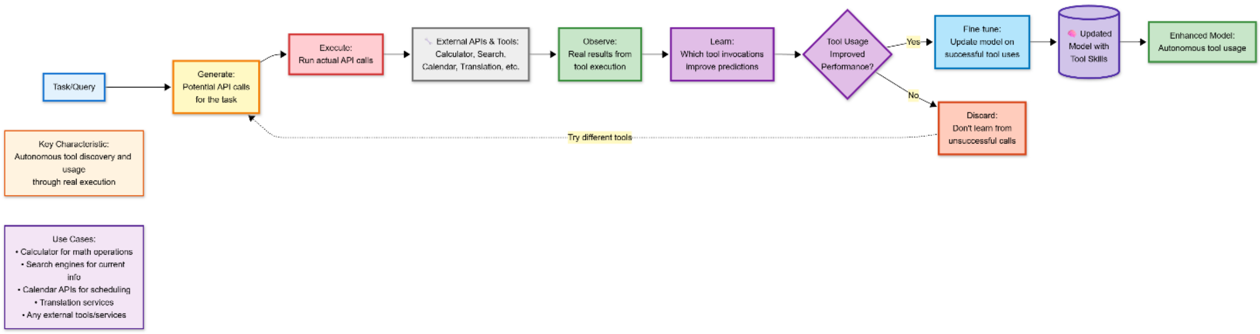
3. Toolformer

Paper: [Toolformer: Language Models Can Teach Themselves to Use Tools (Schick et al., 2023)]
Source: Meta AI

Why Agentic: Self-learns to invoke external APIs and tools through execution and feedback.

Process:

1. Model generates potential API calls
2. **Executes actual API calls** (calculator, search, calendar etc)
3. Observes real results
4. Learns which tool invocations improve predictions
5. Fine-tunes on successful tool uses



Key Characteristic: Autonomous tool discovery and usage through real execution.

Use Cases:

- **Calculator for mathematical operations:** Perform calculations, conversions, or complex math tasks.

- **Search engines for current information:** Fetch up-to-date data, news, or references from the web.
- **Calendar APIs for scheduling:** Check availability, create events, and manage appointments automatically.
- **Translation services:** Translate text between languages accurately and in context.
- **File and document tools:** Read, edit, or summarize documents and spreadsheets.
- **Communication tools:** Send emails, messages, or notifications through integrated platforms.
- **Data and analytics services:** Query databases, analyze data, and generate reports.
- **…and any other external tools/services:** Connect to APIs or services to expand functionality and automate workflows.

Code: [Colab Notebook]

---

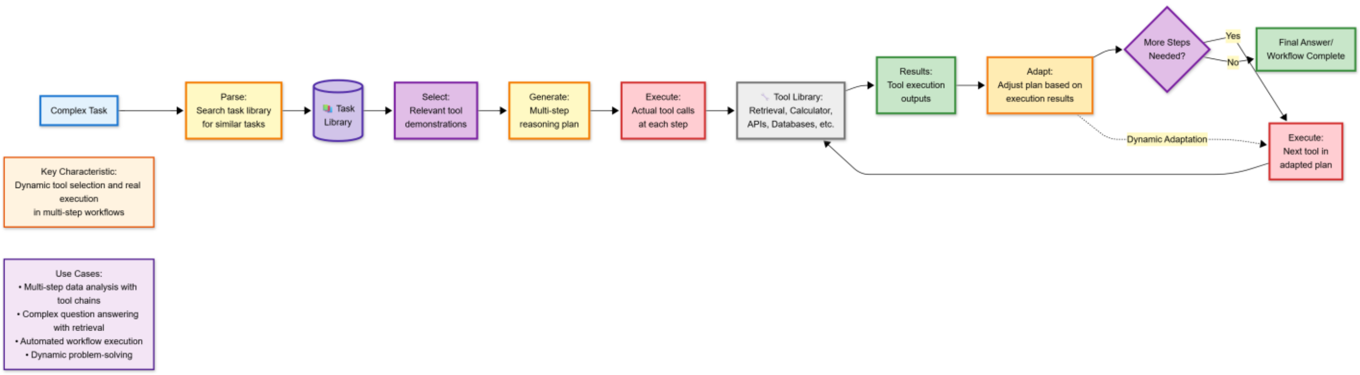
4. Automatic Reasoning and Tool-use (ART)

Paper: [ART: Automatic Multi-step Reasoning and Tool-use for Large Language Models (Paranjape et al., 2023)]
Source: Stanford University

Why Agentic: Automatically selects and executes tools from a library based on task requirements.

Process:

1. Parse task library for similar tasks
2. Select relevant tool demonstrations
3. Generate reasoning steps
4. **Execute actual tool calls** at appropriate steps
5. Adapt based on tool execution results



Key Characteristic: Dynamic tool selection and real execution in multi-step workflows.

Use Cases:

- **Multi-step data analysis with tool chains:** Combine multiple tools to clean, process, visualize, and interpret data in sequence.

- **Complex question answering with retrieval:** Gather information from documents, databases, or APIs to answer detailed or multi-part questions.
- **Automated workflow execution:** Coordinate multiple steps across different applications or services to complete tasks without manual intervention.
- **Dynamic problem-solving:** Adapt strategies in real time based on changing inputs, constraints, or feedback.
- **Simulation and scenario testing:** Run "what-if" analyses to predict outcomes and optimize decisions.
- **Resource management and orchestration:** Allocate tasks, tools, or services efficiently to achieve goals.
- **Continuous learning from results:** Refine actions and workflows based on past performance or outcomes.

Code: [Colab Notebook]()

---

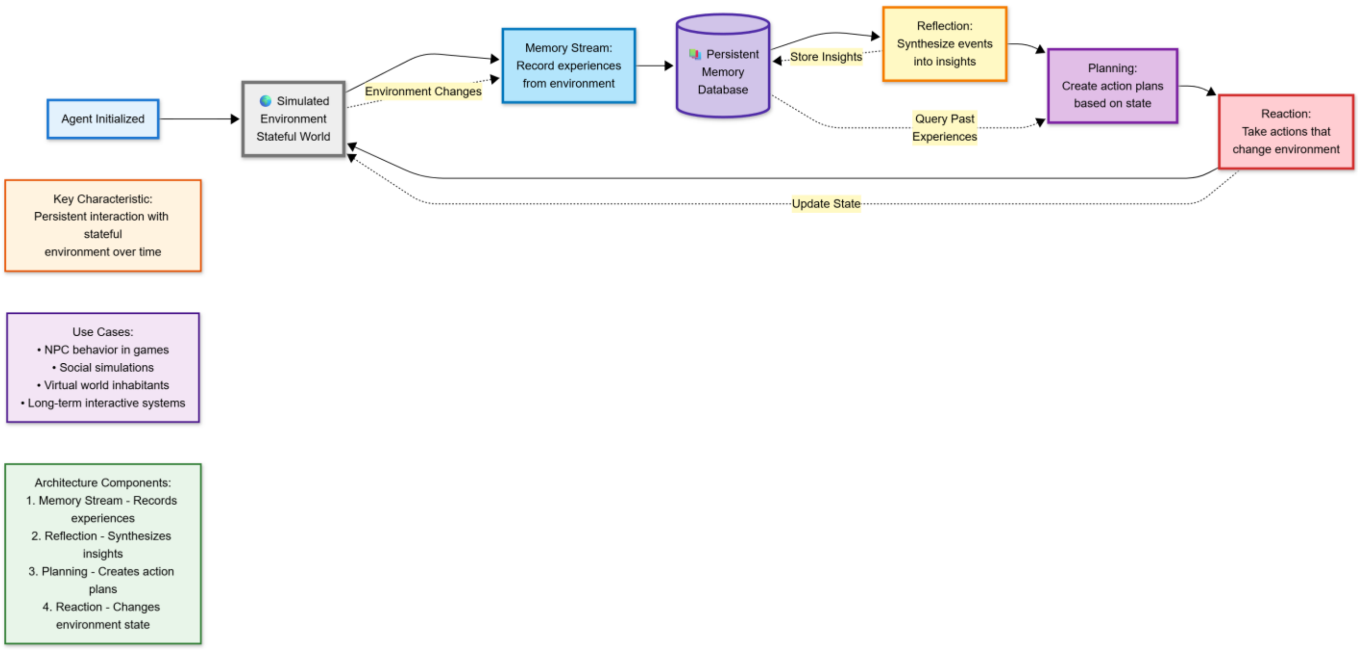
5. Generative Agents

Paper: [Generative Agents: Interactive Simulacra of Human Behavior (Park et al., 2023)]()
Source: Stanford University, Google Research

Why Agentic: Agents act in simulated environments, observe changes, and maintain persistent memory.

Architecture:

1. **Memory Stream**: Records experiences from environment
2. **Reflection**: Synthesizes observed events into insights
3. **Planning**: Creates action plans based on environment state
4. **Reaction**: Takes actions that change environment state



Key Characteristic: Persistent interaction with stateful environment over time.

Use Cases:

- **Multi-step data analysis with tool chains:** Combine multiple tools to clean, process, visualize, and interpret data in sequence.
- **Complex question answering with retrieval:** Gather information from documents, databases, or APIs to answer detailed or multi-part questions.
- **Automated workflow execution:** Coordinate multiple steps across different applications or services to complete tasks without manual intervention.
- **Dynamic problem-solving:** Adapt strategies in real time based on changing inputs, constraints, or feedback.
- **Simulation and scenario testing:** Run "what-if" analyses to predict outcomes and optimize decisions.
- **Resource management and orchestration:** Allocate tasks, tools, or services efficiently to achieve goals.
- **Continuous learning from results:** Refine actions and workflows based on past performance or outcomes.

Code: Colab Notebook

---

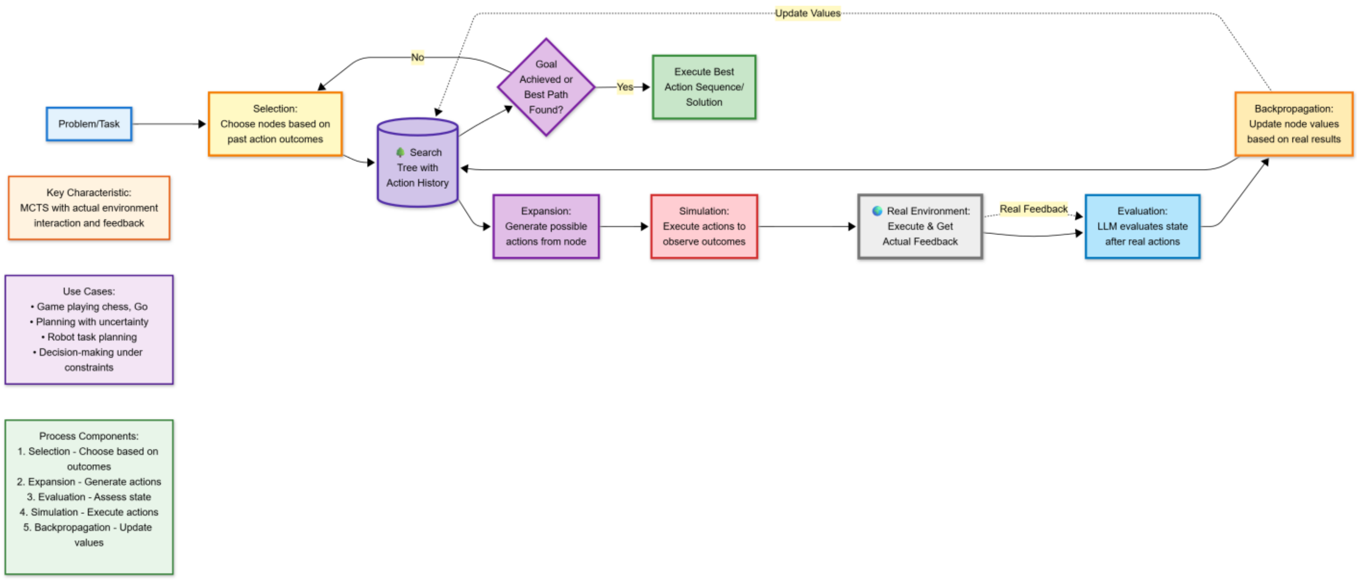
6. LATS (Language Agent Tree Search)

Paper: LATS: Language Agent Tree Search Unifies Reasoning, Acting, and Planning (Zhou et al., 2023)
Source: Ohio State University

Why Agentic: Combines search algorithms with real action execution and environmental feedback.

Process:

1. **Selection**: Choose nodes based on past action outcomes
2. **Expansion**: Generate possible actions
3. **Evaluation**: LLM evaluates state after real actions
4. **Simulation**: Execute actions to observe outcomes
5. **Backpropagation**: Update values based on real results



Key Characteristic: Monte Carlo Tree Search (MCTS) with actual environment interaction and feedback.

Use Cases:

- **Game playing (chess, Go):** Strategically explore moves and outcomes to choose the best actions based on past results.
- **Planning with uncertainty:** Make robust plans when outcomes are unpredictable or incomplete information is available.
- **Robot task planning:** Decide sequences of actions for robots to achieve goals while adapting to real-world feedback.
- **Decision-making under constraints:** Optimize choices considering limits like time, resources, or safety.
- **Resource allocation:** Efficiently distribute resources in dynamic environments using feedback-driven planning.
- **Multi-agent coordination:** Plan actions when multiple agents interact, ensuring collaboration or competition strategies.
- **Dynamic strategy adaptation:** Continuously update strategies based on observed results and evolving conditions.

Code: [Colab Notebook]

---

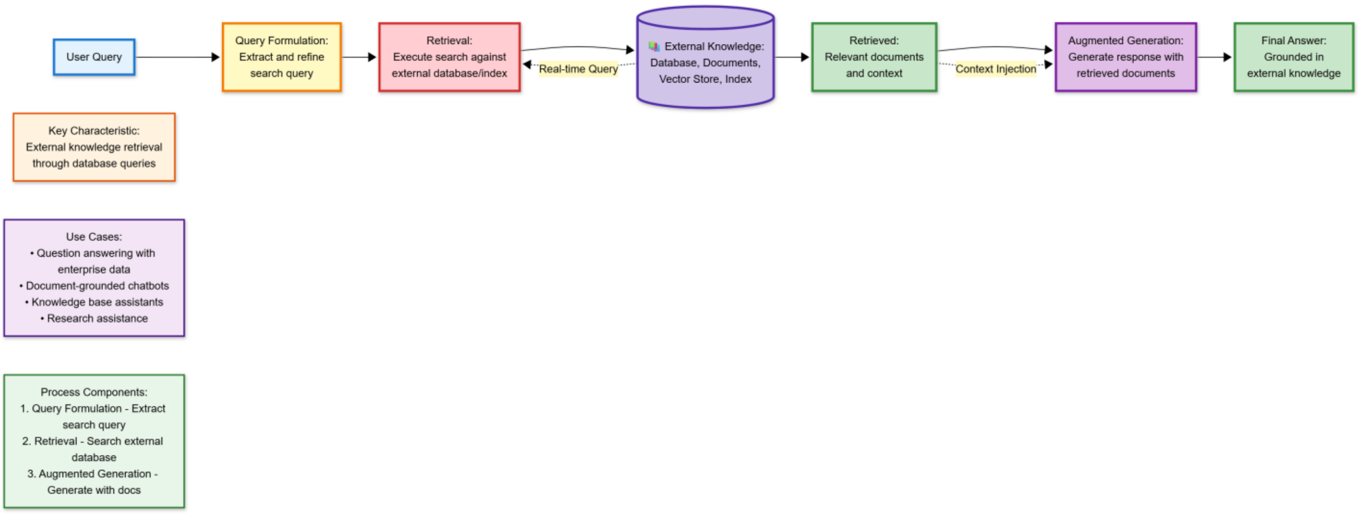
7. Retrieval-Augmented Generation (RAG)

Paper: [Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (Lewis et al., 2020)]
Source: Facebook AI, UCL

Why Agentic: Actively retrieves relevant information from external knowledge sources to generate informed, context-aware responses.

Process:

1. **Query Formulation**: Extract or generate a search query from user input.
2. **Retrieval**:
    a. Can use traditional keyword search (Elasticsearch, SQL, APIs) without embeddings.
    b. Or use semantic search with embedding models + vector DBs (Pinecone, Weaviate, Milvus) for context-aware retrieval.
3. **Augmented Generation:** Feed retrieved documents into an LLM to produce accurate and grounded responses.



Key Characteristic: External knowledge retrieval enhances LLM outputs; embeddings/vector DBs are optional but improve semantic relevance.

Use Cases:

- **Question answering with enterprise data:** Provide accurate answers using internal knowledge bases.
- **Document-grounded chatbots:** Generate responses backed by real documents for reliability.
- **Knowledge base assistants:** Help users navigate and extract insights from large information repositories.
- **Research assistance:** Summarize, analyze, or contextualize research materials efficiently.
- **Legal and compliance support:** Retrieve relevant laws, regulations, or policies for informed guidance.
- **Customer support automation:** Access FAQs or manuals to deliver precise solutions to user queries.

Code: Colab Notebook

---

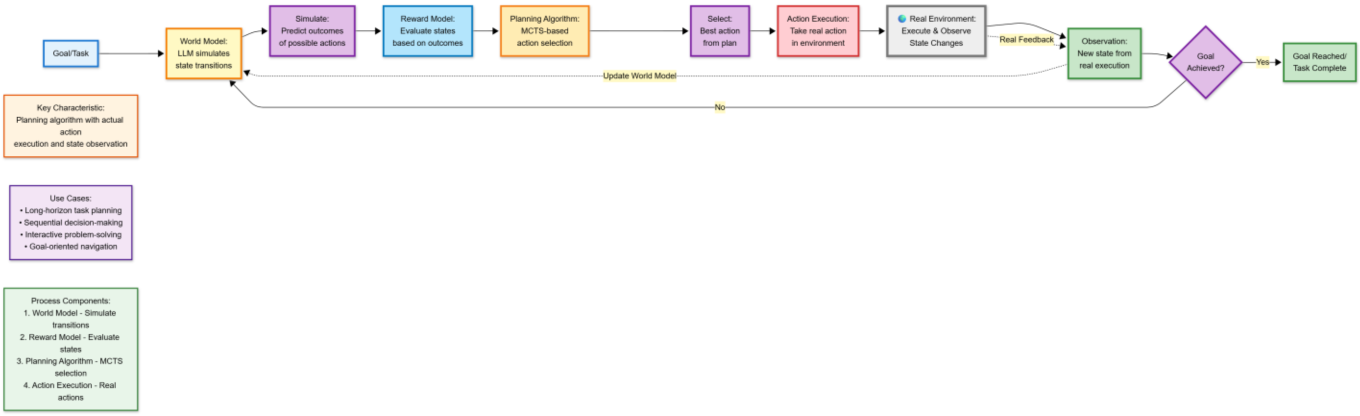
8. Reasoning via Planning (RAP)

Paper: Reasoning with Language Model is Planning with World Model (Hao et al., 2023)
Source: Peking University, UC Berkeley

Why Agentic: Uses an LLM as a world model to simulate, plan, and execute actions, learning from real environmental feedback.

Process:

1. **World Model**: LLM simulates state transitions from actions
2. **Reward Model**: Evaluate states based on outcomes
3. **Planning Algorithm**: Monte Carlo Tree Search (MCTS) based action selection
4. **Action Execution**: Take real actions in environment



Key Characteristic: Combines simulation, planning, and real action execution to solve complex, long-horizon tasks.

Use Cases:

- **Long-horizon task planning:** Plan multi-step actions for projects, robotics, or automated workflows.
- **Sequential decision-making:** Handle tasks where each action depends on previous outcomes.
- **Interactive problem-solving:** Collaborate with humans or other agents to iteratively reach solutions.
- **Goal-oriented navigation:** Robots or agents navigate environments while adjusting to changes dynamically.
- **Multi-agent coordination:** Plan actions when multiple agents interact to achieve shared or competing goals.
- **Resource management:** Optimize allocation and sequencing under constraints.
- **Simulation-based training:** Test strategies in a virtual or real-world environment before committing to critical actions.

Code: [Colab Notebook]

---

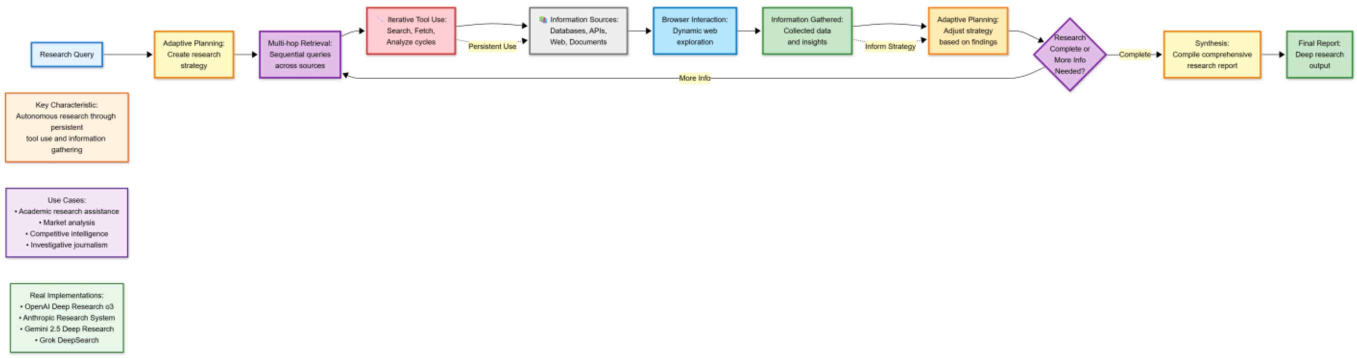
9. Deep Research Agents

Paper: [Deep Research Agents: A Systematic Examination And Roadmap (Huang et al., 2025)]
Source: Multiple research institutions

Why Agentic: Autonomous multi-turn information gathering with tool execution and adaptive exploration.

Core Capabilities:

1. **Multi-hop Information Retrieval**: Sequential queries across sources
2. **Adaptive Planning**: Adjust strategy based on findings
3. **Iterative Tool Use**: Repeated search, fetch, analyze cycles
4. **Browser Interaction**: Dynamic web exploration



Key Characteristic: Autonomous research through persistent tool use and information gathering.

Real Implementations:

- OpenAI Deep Research (o3 model)
- Anthropic Research System
- Gemini 2.5 Deep Research
- Grok DeepSearch

Use Cases:

- **Academic research assistance:** Summarize papers, extract key findings, suggest citations, and identify research gaps.
- **Market analysis:** Gather and analyze market trends, consumer sentiment, and emerging opportunities from multiple sources.
- **Competitive intelligence:** Monitor competitors' activities, product launches, and strategic moves to inform decision-making.
- **Investigative journalism:** Collect, verify, and cross-reference information from diverse sources to uncover insights and stories.
- **Regulatory and policy research:** Track changes in laws, guidelines, and policies to support compliance or strategy.
- **Patent and innovation analysis:** Identify technological trends, patent landscapes, and potential areas for innovation.

- **Scientific hypothesis exploration:** Generate and evaluate hypotheses by integrating data from multiple research sources.
- **Multi-source synthesis:** Combine structured data, publications, reports, and web content to produce actionable insights.

---

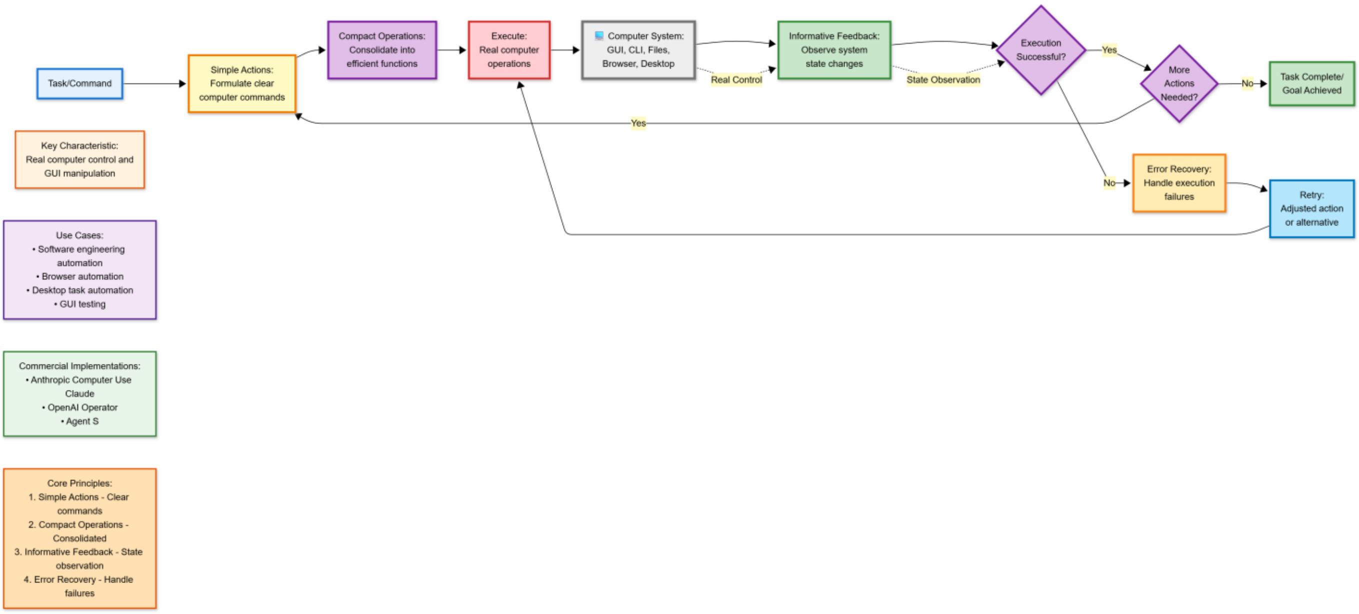
10. Agent-Computer Interface (ACI)

Paper: [SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering (Yang et al., 2024)]
Source: Princeton University

Why Agentic: Direct GUI and system interaction, executing real computer operations.

Core Principles:

1. **Simple Actions**: Clear computer commands
2. **Compact Operations**: Consolidated functions
3. **Informative Feedback**: System state observation
4. **Error Recovery**: Handle execution failures



Key Characteristic: Real computer control and GUI manipulation.

Commercial Implementations:

- Anthropic Computer Use (Claude)
- OpenAI Operator
- Agent S

Use Cases:

- Software engineering automation
- Browser automation
- Desktop task automation
- GUI testing

---

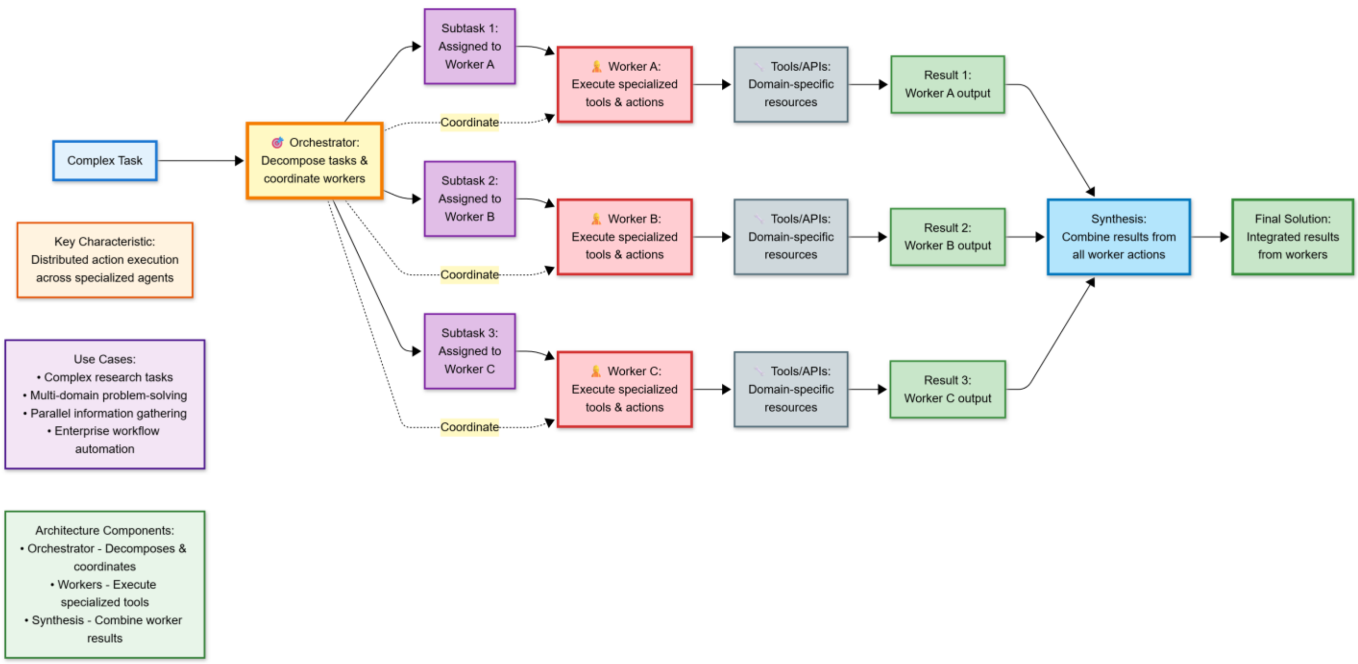
11. Orchestrator-Worker Pattern

Source: Andrew Ng (2024), Anthropic (2025)

Why Agentic: Multi-agent system where workers execute tools and gather information.

Architecture:

- **Orchestrator**: Decomposes tasks, coordinates workers
- **Workers**: Execute specialized tools and actions
- **Synthesis**: Combine results from worker actions



Key Characteristic: Distributed action execution across specialized agents.

Use Cases:

- **Complex research tasks:** Divide literature review, data collection, and analysis across multiple worker agents.
- **Multi-domain problem-solving:** Solve problems spanning different fields or data types simultaneously.
- **Parallel information gathering:** Scrape, query, or retrieve data from multiple sources at once.

- **Enterprise workflow automation:** Automate cross-department processes involving multiple tools or systems.
- **Data pipeline orchestration:** Coordinate ETL, data validation, and reporting tasks across services.
- **Customer support triage:** Distribute inquiries to specialized bots for faster and more accurate responses.
- **Multi-step project execution:** Break down projects into subtasks handled concurrently by different agents.
- **Collaborative decision-making:** Aggregate insights from diverse workers to support strategic decisions.

Code: [Colab Notebook]()

---

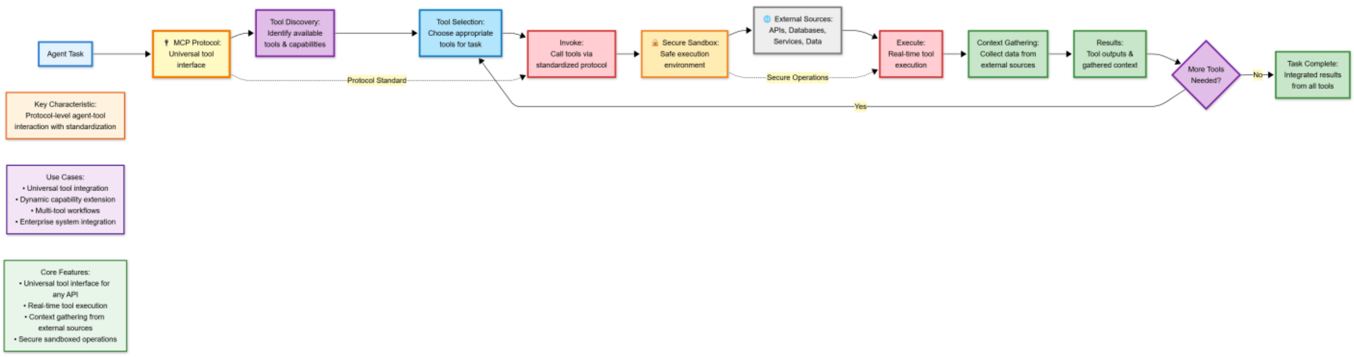
12. Model Context Protocol (MCP)

Year: 2024-2025
Source: Anthropic

Why Agentic: Standardized protocol for agents to invoke external tools and data sources.

Core Features:

- **Universal tool interface:** Connect with any API or service seamlessly.
- **Real-time tool execution:** Perform actions and retrieve results immediately.
- **Context gathering:** Collect relevant information from external sources to inform decisions.
- **Secure sandboxed operations:** Execute tools safely without affecting the underlying system.



Key Characteristic: Protocol-level agent-tool interaction allowing flexible, secure, and standardized integrations.

Use Cases:

- **Universal tool integration:** Connect with diverse APIs, databases, and services without custom adapters.
- **Dynamic capability extension:** Enable agents to adopt new tools or services on-the-fly.
- **Multi-tool workflows:** Coordinate multiple tools in a single task for complex operations.

- **Enterprise system integration:** Bridge AI agents with CRM, ERP, or other internal enterprise systems.
- **Automated IT operations:** Execute scripts, monitor systems, and respond to alerts through standard protocols.
- **Knowledge retrieval and synthesis:** Pull data from various sources and combine it into actionable insights.
- **Cross-platform task execution:** Operate across cloud services, SaaS tools, and on-prem systems using one protocol.
- **Rapid experimentation:** Safely test new integrations and workflows without changing core agent architecture.

Code: Colab Notebook

Non-Agentic Patterns

Pure reasoning and prompting techniques

---

1. Chain-of-Thought (CoT)

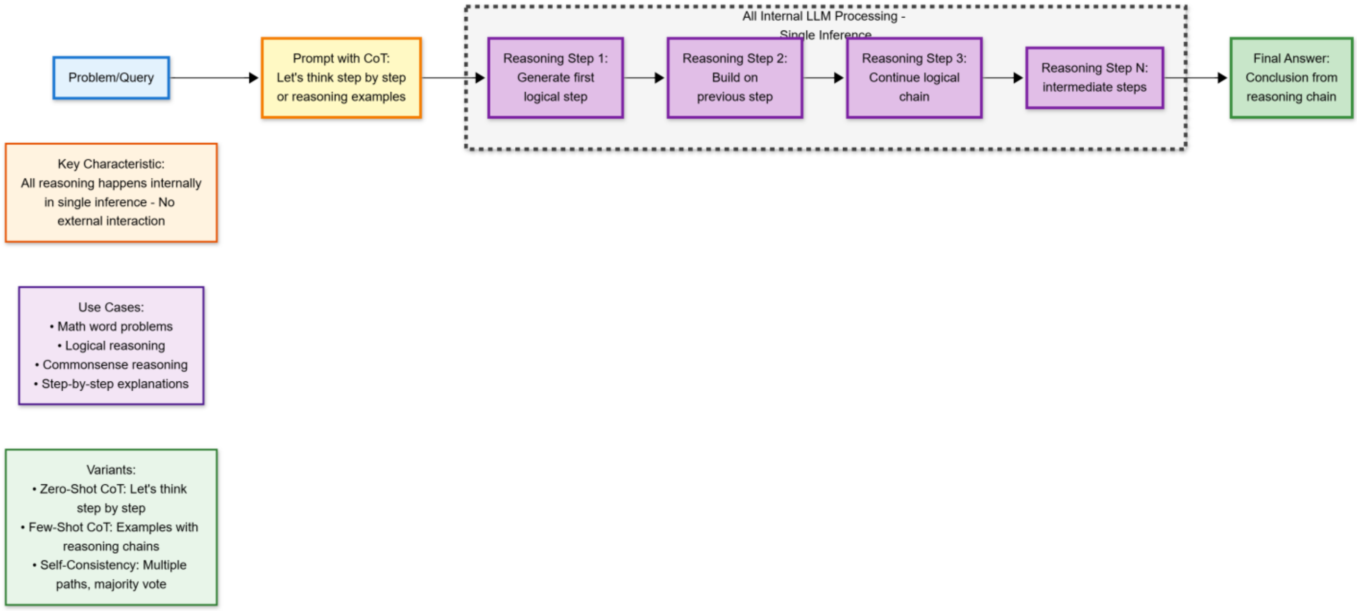
Paper: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (Wei et al., 2022)
Source: Google Research

Why Non-Agentic: Pure internal reasoning without external interaction.

Description: Prompts model to generate intermediate reasoning steps.

Variants:

- Zero-Shot CoT: "Let's think step by step"
- Few-Shot CoT: Examples with reasoning chains
- Self-Consistency: Multiple paths, majority vote



Key Characteristic: All reasoning happens internally in single inference.

Use Cases:

- Math word problems
- Logical reasoning
- Commonsense reasoning
- Step-by-step explanations

Code: Colab Notebook

---

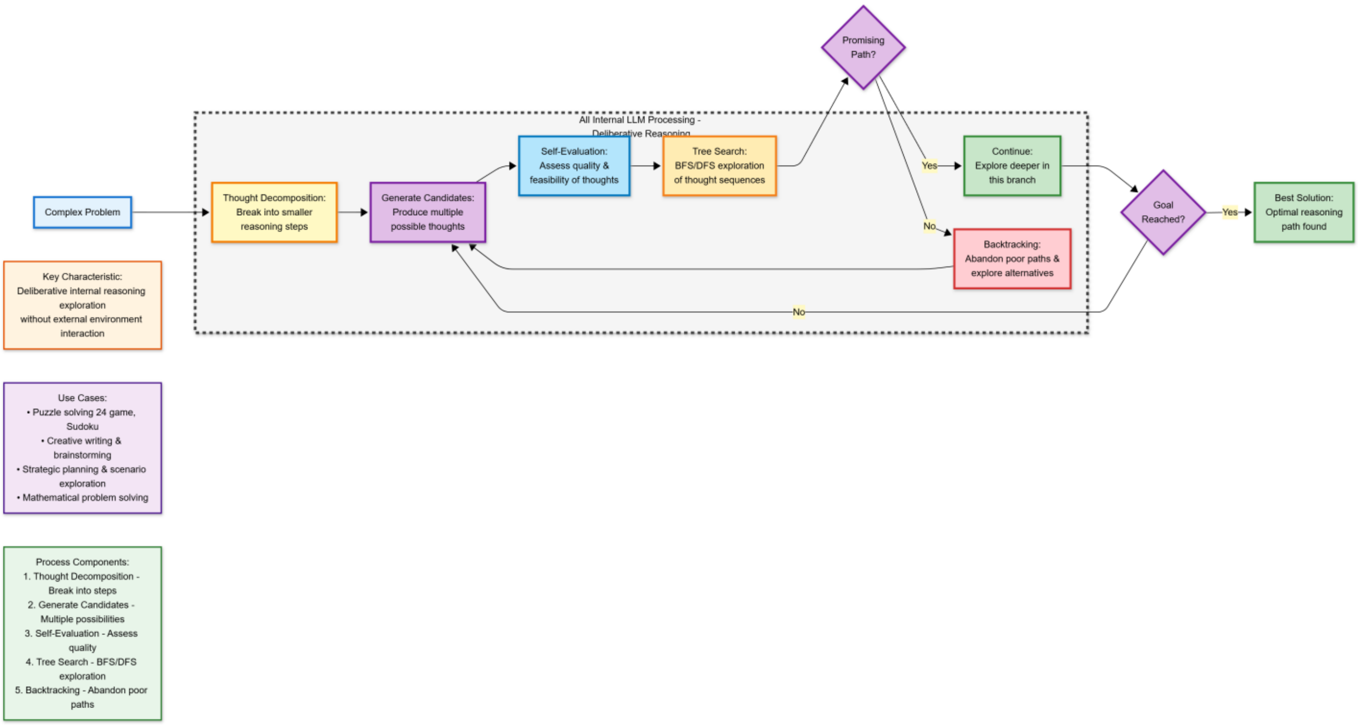
2. Tree of Thoughts (ToT)

Paper: Tree of Thoughts: Deliberate Problem Solving with Large Language Models (Yao et al., 2023)
Source: Princeton University, Google DeepMind

Why Non-Agentic: Explores multiple reasoning paths internally without interacting with the external environment.

Process:

1. **Thought Decomposition:** Break complex problems into smaller reasoning steps.
2. **Generate Candidate Thoughts:** Produce multiple possible ideas or solutions for each step.
3. **Self-Evaluation:** Assess the quality, relevance, or feasibility of each thought.
4. **Tree Search (BFS/DFS):** Explore sequences of thoughts systematically through the tree.
5. **Backtracking:** Abandon poor or low-value paths and explore alternatives.



Key Characteristic: Deliberative, internal reasoning exploration focused on evaluating and chaining thoughts.

Use Cases:

- **Puzzle solving (24 game, Sudoku, logic puzzles):** Explore all possible move sequences internally to find solutions.
- **Creative writing:** Generate, evaluate, and refine narrative ideas, dialogue, or storylines.
- **Strategic planning:** Map out possible plans, assess consequences, and select optimal strategies.
- **Complex problem decomposition:** Break multifaceted problems into smaller solvable steps internally.
- **Mathematical problem solving:** Stepwise reasoning for proofs or multi-step calculations.
- **Brainstorming and idea generation:** Generate diverse solution paths before selecting the best approach.
- **Game strategy analysis:** Evaluate internal strategies in board games or simulations without interacting with an environment.
- **Scenario exploration:** Consider multiple hypothetical outcomes to improve decision-making or planning.

Code: [Colab Notebook]

---

3. Graph of Thoughts (GoT)

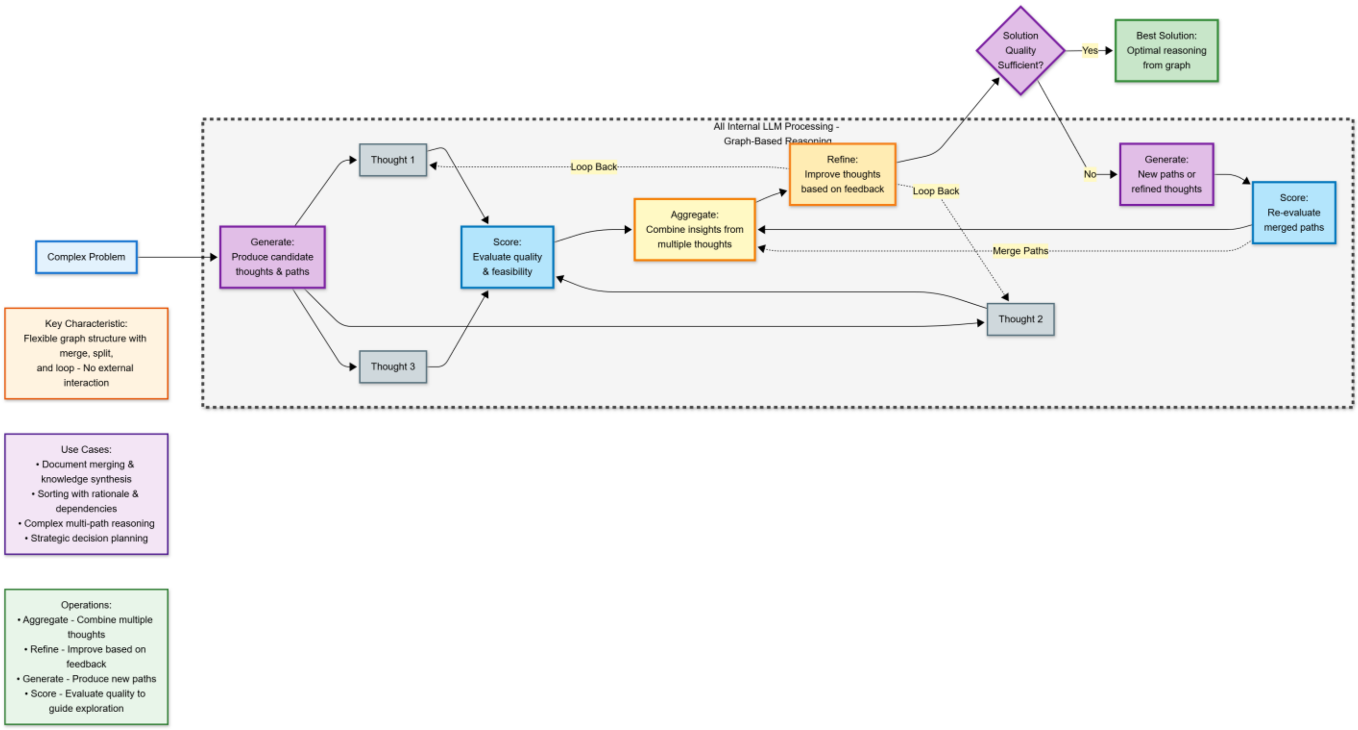
Paper: [Graph of Thoughts: Solving Elaborate Problems with Large Language Models (Besta et al., 2023)]
Source: ETH Zurich

Why Non-Agentic: Extends Tree of Thoughts (ToT) with a graph-based structure, but still operates internally without external environment interaction.

Description: Models reasoning as a flexible graph where thoughts can merge, split, and loop, enabling richer exploration of possibilities than linear trees.

Operations:

- **Aggregate:** Combine insights from multiple thoughts to form stronger conclusions.
- **Refine:** Improve or optimize individual thoughts based on feedback.
- **Generate:** Produce new candidate thoughts or solution paths.
- **Score:** Evaluate thought quality, relevance, or feasibility to guide exploration.



Key Characteristic: Flexible internal reasoning structure allowing multi-path, looped, and merged thought exploration beyond simple tree hierarchies.

Use Cases:

- **Document merging:** Integrate information from multiple sources with reasoning about conflicts or overlaps.

- **Sorting with rationale**: Organize items or ideas while considering context, dependencies, or priorities.

- **Complex reasoning tasks**: Solve multifaceted problems requiring iterative and interconnected thinking.

- **Multi-path exploration**: Explore multiple solution paths simultaneously, revisiting and merging them as needed.

- **Strategic decision planning**: Evaluate interconnected strategies or plans with multiple contingencies.
- **Creative brainstorming**: Develop ideas collaboratively within the internal thought graph.
- Knowledge synthesis: Combine fragmented knowledge from various domains into coherent solutions.
- **Mathematical or logical reasoning**: Track interdependent steps that may loop or require backtracking.

Code: Colab Notebook

---

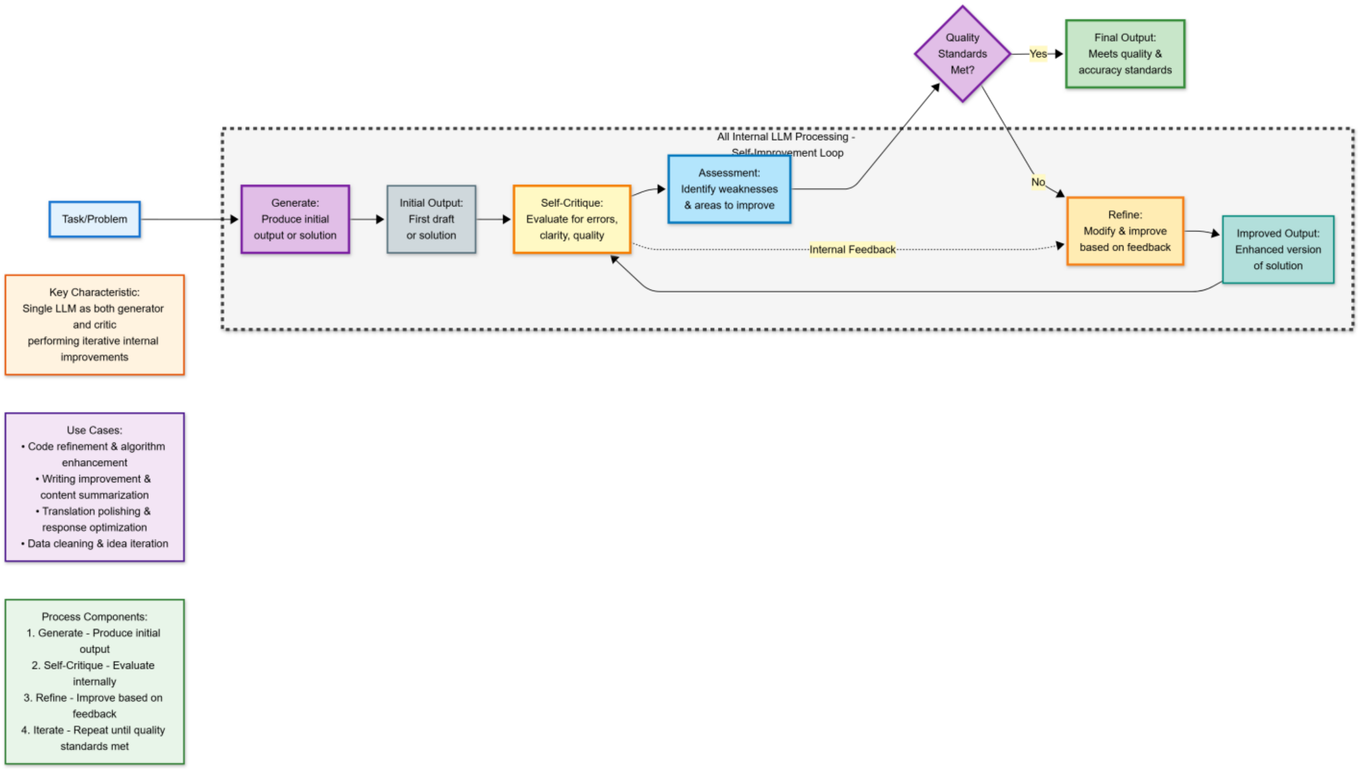
4. Self-Refine

Paper: Self-Refine: Iterative Refinement with Self-Feedback (Madaan et al., 2023)
Source: Carnegie Mellon University, AI2

Why Non-Agentic: Uses an internal self-improvement loop without interacting with the external environment.

Process:

1. **Generate Initial Output:** Produce a first draft or solution.
2. **Self-Critique:** Evaluate the output internally for errors, clarity, or quality.
3. **Refine:** Modify and improve the output based on self-feedback.
4. **Iterate:** Repeat the cycle until the output meets quality or accuracy standards.



Key Characteristic: A single LLM functions as both generator and critic, performing iterative internal improvements.

Use Cases:

- **Code refinement:** Debug, optimize, or improve generated code iteratively.
- **Writing improvement:** Enhance clarity, style, or grammar in essays, reports, or articles.
- **Translation polishing:** Refine translations for accuracy and fluency.
- **Response optimization:** Improve chatbot or AI responses for coherence and relevance.
- **Data cleaning suggestions:** Iteratively propose improvements for structured or unstructured data.
- **Algorithm enhancement:** Refine mathematical or procedural solutions internally.
- **Content summarization:** Gradually improve summaries to ensure conciseness and completeness.
- **Idea iteration:** Continuously enhance creative outputs like storylines, marketing copy, or design.

Code: [Colab Notebook]

---

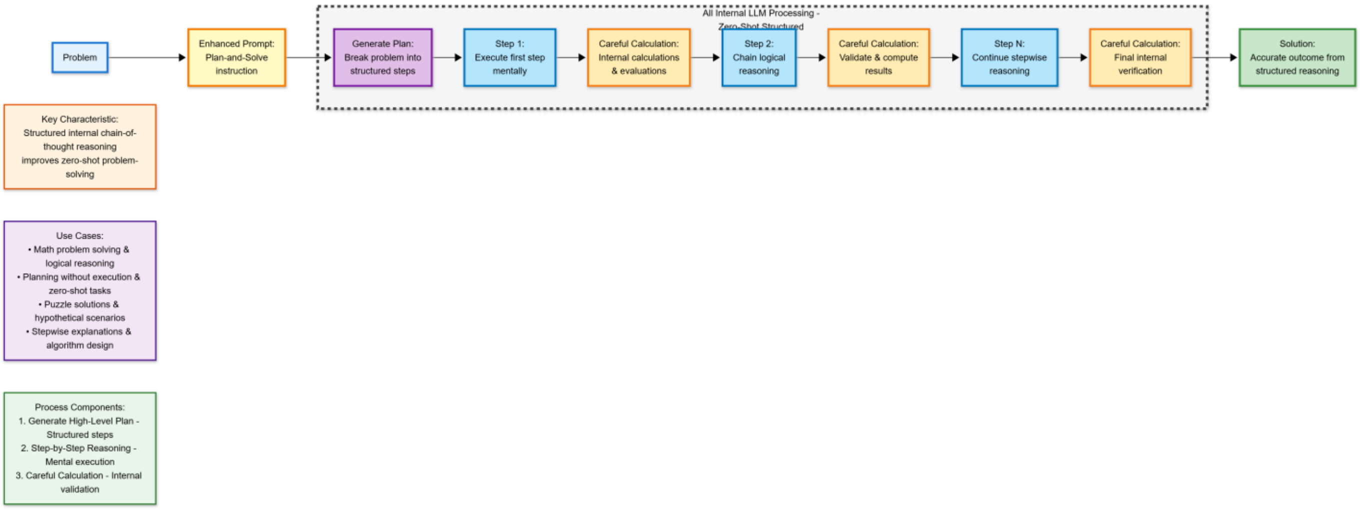
5. Plan-and-Solve

Paper: [Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning (Wang et al., 2023)]
Source: University of Tokyo, Microsoft

Why Non-Agentic: Uses enhanced prompting to guide reasoning internally, without executing actions or interacting with external tools.

Process:

1. **Generate High-Level Plan:** Break the problem into structured steps.
2. **Step-by-Step Reasoning:** Execute each step mentally within the model, chaining thoughts logically.
3. **Careful Calculation:** Perform internal calculations and evaluations for accurate outcomes.



Key Characteristic: Structured, internal chain-of-thought reasoning that improves zero-shot problem-solving without external validation.

Use Cases:

- **Math problem solving:** Solve arithmetic, algebra, or multi-step word problems internally.
- **Logical reasoning:** Deduce conclusions from given premises or scenarios.
- **Planning without execution:** Outline strategies or project steps mentally before acting.
- **Zero-shot improvements:** Enhance reasoning performance without prior examples or fine-tuning.
- **Puzzle and brainteaser solutions:** Solve Sudoku, logic grids, or pattern recognition problems.
- **Hypothetical scenario analysis:** Explore "what-if" situations internally to anticipate outcomes.
- **Stepwise explanation generation:** Produce detailed reasoning explanations for educational or explanatory purposes.
- **Algorithm design:** Mentally plan algorithms and workflows before coding or implementation.

Code: [Colab Notebook]

---

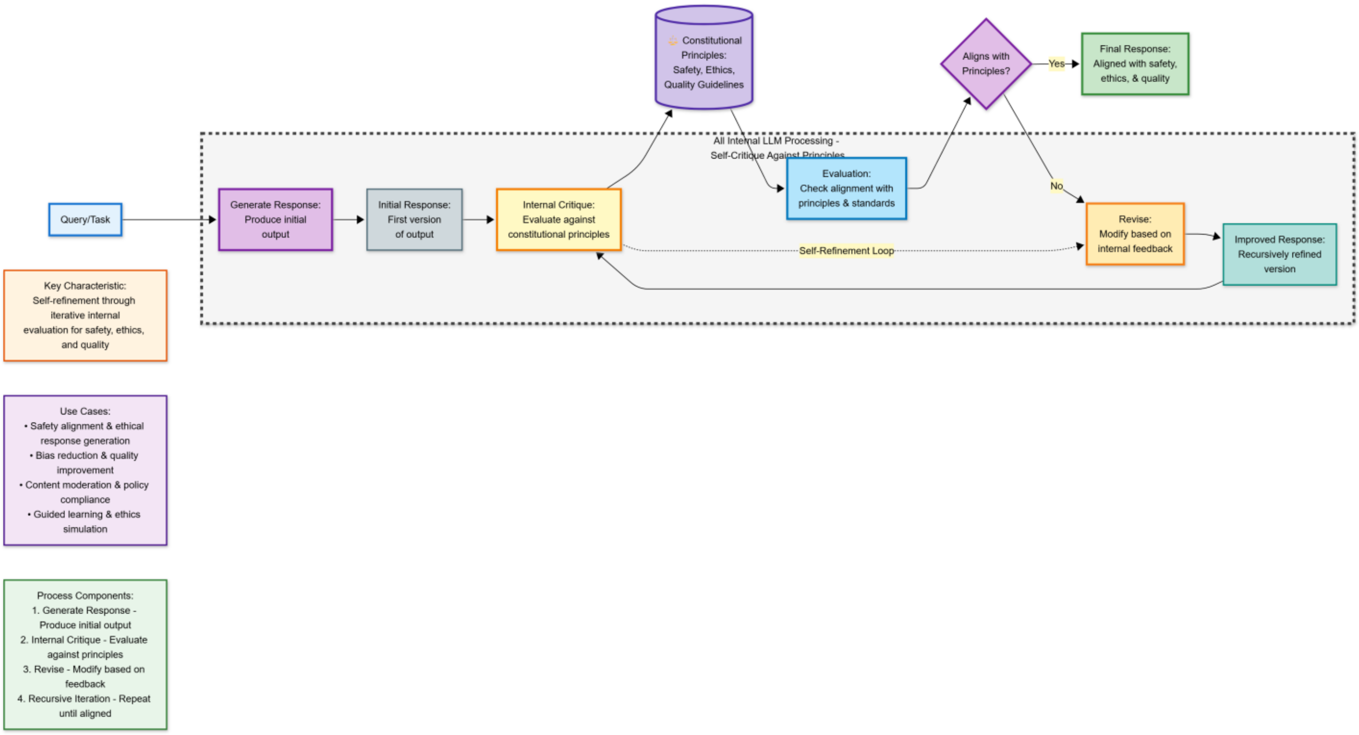
6. Recursive Criticism and Improvement (RCI)

Paper: Constitutional AI: Harmlessness from AI Feedback (Bai et al., 2022)
Source: Anthropic

Why Non-Agentic: Relies on internal self-critique against predefined principles without interacting with the external environment.

Process:

1. **Generate Response:** Produce an initial output.
2. **Internal Critique:** Evaluate the response against constitutional or ethical principles.
3. **Revise:** Modify the response based on internal feedback.
4. **Recursive Iteration:** Repeat critique and revision until the response aligns with principles.



Key Characteristic: Self-refinement through iterative internal evaluation to improve safety, ethics, and quality.

Use Cases:

- **Safety alignment:** Ensure outputs adhere to safety guidelines and avoid harmful content.
- **Ethical response generation:** Produce morally sound and socially responsible answers.
- **Bias reduction:** Detect and mitigate biases in responses across sensitive topics.
- **Quality improvement:** Enhance clarity, correctness, and relevance of generated outputs.
- **Content moderation assistance:** Automatically flag and refine inappropriate or unsafe content.
- **Guided learning for LLMs:** Train models to self-correct based on internal principles.
- **Policy compliance:** Ensure outputs meet regulatory or organizational standards.
- **Scenario simulation for ethics:** Evaluate hypothetical responses to ensure consistency with ethical frameworks.

Code: [Colab Notebook]

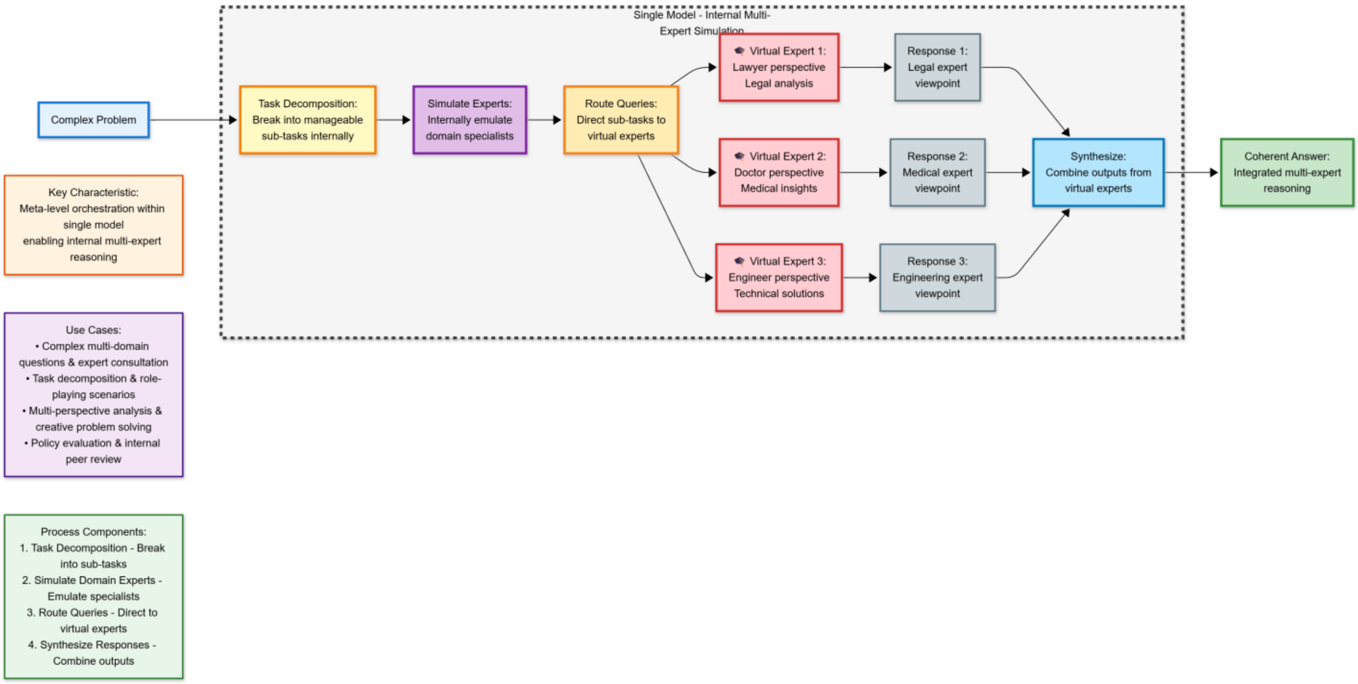
7. Meta-Prompting

Paper: [Meta-Prompting: Enhancing Language Models with Task-Agnostic Scaffolding (Suzgun & Kalai, 2024)]
Source: Stanford, Microsoft Research

Why Non-Agentic: Uses a single model to internally simulate multiple expert roles without external interaction.

Process:

1. **Task Decomposition (Internal):** Break complex problems into manageable sub-tasks.
2. **Simulate Domain Experts:** Internally emulate specialists for each sub-task or domain.
3. **Route Queries:** Direct different parts of the problem to the simulated virtual experts.
4. **Synthesize Responses:** Combine outputs from virtual experts into a coherent answer.



Key Characteristic: Meta-level orchestration within a single model, enabling internal multi-expert reasoning.

Use Cases:

- **Complex multi-domain questions:** Answer queries requiring knowledge from multiple fields.
- **Expert consultation simulation:** Emulate advice from lawyers, doctors, or engineers internally.
- **Task decomposition:** Break large problems into smaller, expert-driven solutions.
- **Role-playing scenarios:** Simulate interactions among multiple internal personas or stakeholders.

- **Multi-perspective analysis:** Generate insights from different internal "viewpoints" for decision-making.
- **Creative problem solving:** Combine expert reasoning to produce novel solutions.
- **Policy or strategy evaluation:** Simulate expert review to assess options and outcomes.
- **Internal peer review:** Critically assess and refine outputs by leveraging multiple internal expert roles.

Code: [Colab Notebook]

---

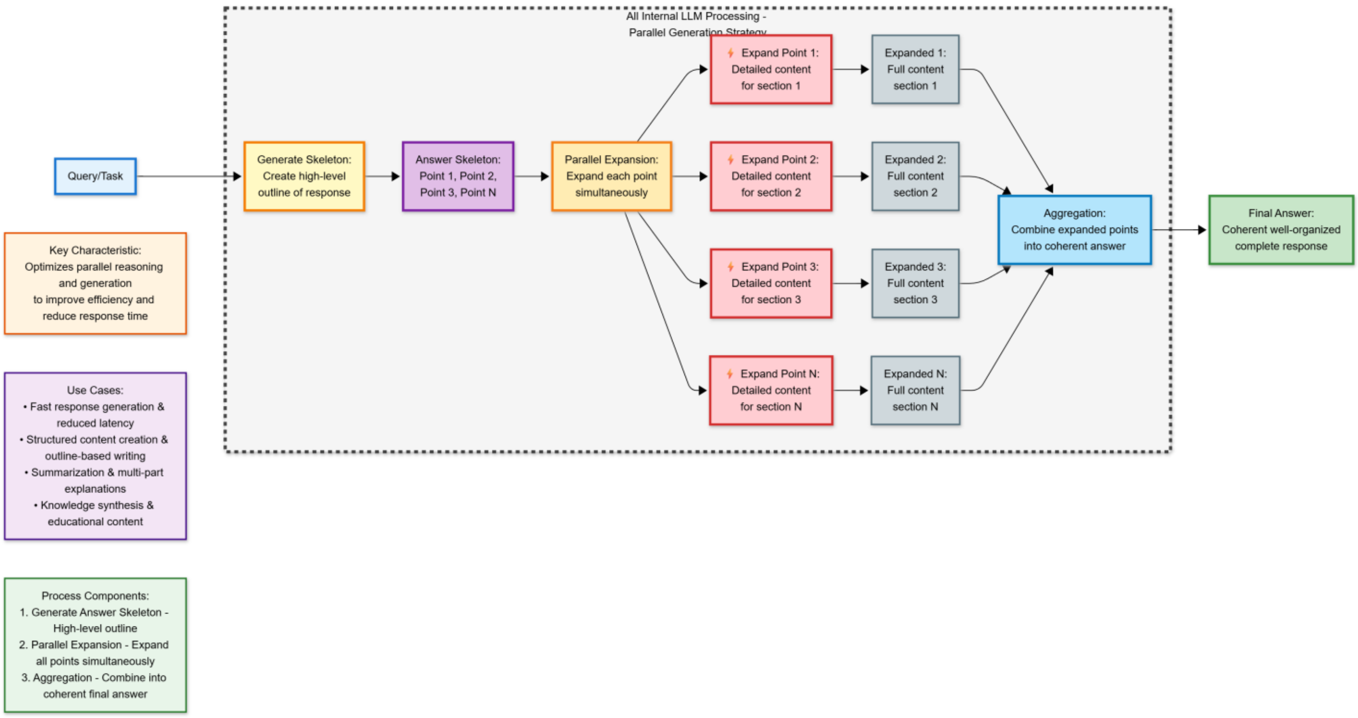
8. Skeleton-of-Thought (SoT)

Paper: [Skeleton-of-Thought: Prompting LLMs for Efficient Parallel Generation (Ning et al., 2023)]
Source: Microsoft Research

Why Non-Agentic: Focuses on internal generation strategies without interacting with external tools or environments..

Process:

1. **Generate Answer Skeleton:** Create a high-level outline of the response.
2. **Parallel Expansion:** Expand each point of the skeleton simultaneously.
3. **Aggregation:** Combine expanded points into a coherent final answer.



Key Characteristic: Optimizes parallel reasoning and generation to improve efficiency and reduce response time.

Use Cases:

- **Fast response generation:** Produce answers quickly by parallelizing content creation.
- **Structured content creation:** Ensure well-organized and coherent outputs.
- **Reduced latency:** Lower time needed for long or multi-step responses.
- **Outline-based writing:** Create essays, reports, or documentation using skeleton-first approach.
- **Summarization tasks:** Generate structured summaries of long documents efficiently.
- **Multi-part explanations:** Produce detailed explanations broken into clear, parallel sections.
- **Knowledge synthesis:** Combine information from multiple internal chains concurrently.
- **Educational content generation:** Build lesson plans, guides, or tutorials with structured flow.

Code: [Colab Notebook]

---

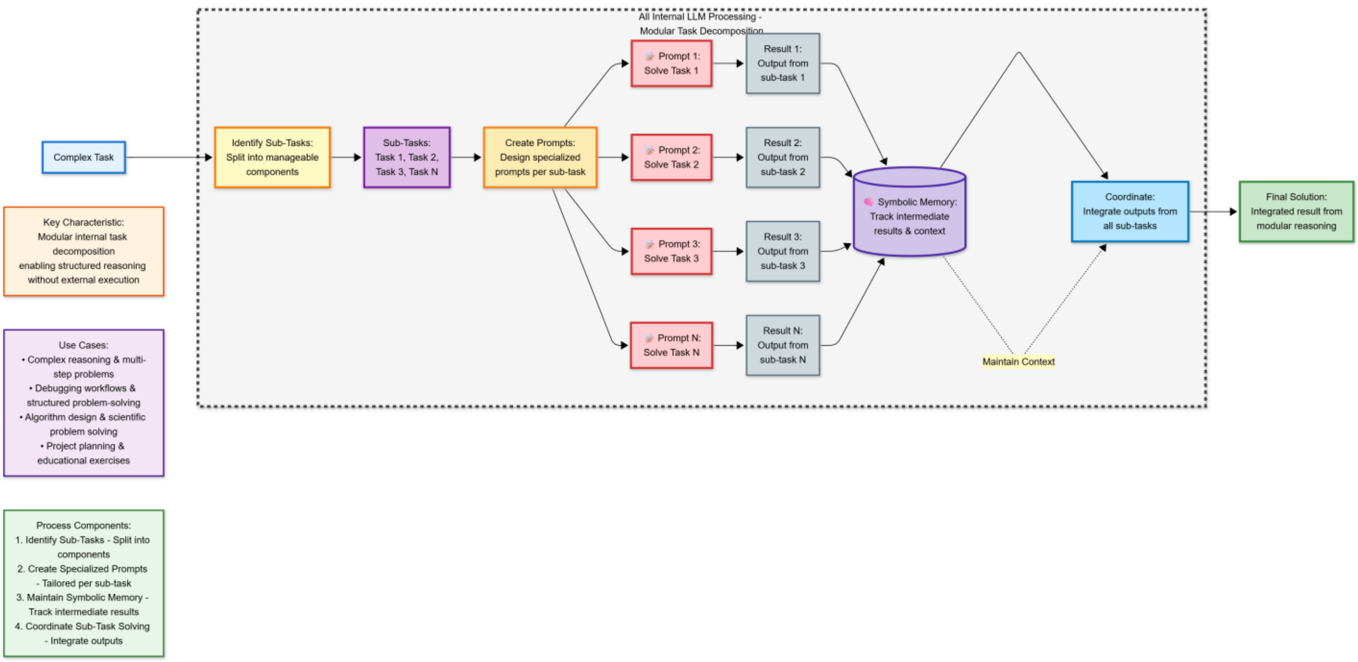
9. Decomposed Prompting

Paper: [Decomposed Prompting: A Modular Approach for Solving Complex Tasks (Khot et al., 2023)]
Source: AI2, University of Washington

Why Non-Agentic: Breaks down tasks internally using symbolic memory and modular prompts, without interacting with external systems.

Process:

1. **Identify Sub-Tasks:** Split complex tasks into smaller, manageable components.
2. **Create Specialized Prompts:** Design prompts tailored to each sub-task.
3. **Maintain Symbolic Memory:** Track intermediate results and context internally.
4. **Coordinate Sub-Task Solving:** Integrate outputs from all sub-tasks to form the final solution.



Key Characteristic: Modular, internal task decomposition that enables structured reasoning without external execution.

Use Cases:

- **Complex reasoning:** Solve multi-faceted problems by addressing each part systematically.
- **Multi-step problems:** Tackle challenges requiring sequential and interdependent steps.
- **Debugging workflows:** Identify and resolve issues step-by-step within code or processes.
- **Structured problem-solving:** Approach tasks with clear modular stages for clarity and accuracy.
- **Algorithm design:** Break algorithms into discrete modules for stepwise reasoning.
- **Scientific problem solving:** Analyze experiments or data using sequential internal steps.
- **Project planning:** Decompose large projects into smaller tasks and plan internally.
- **Educational exercises:** Stepwise problem decomposition for tutoring or learning applications.

Code: Colab Notebook

---

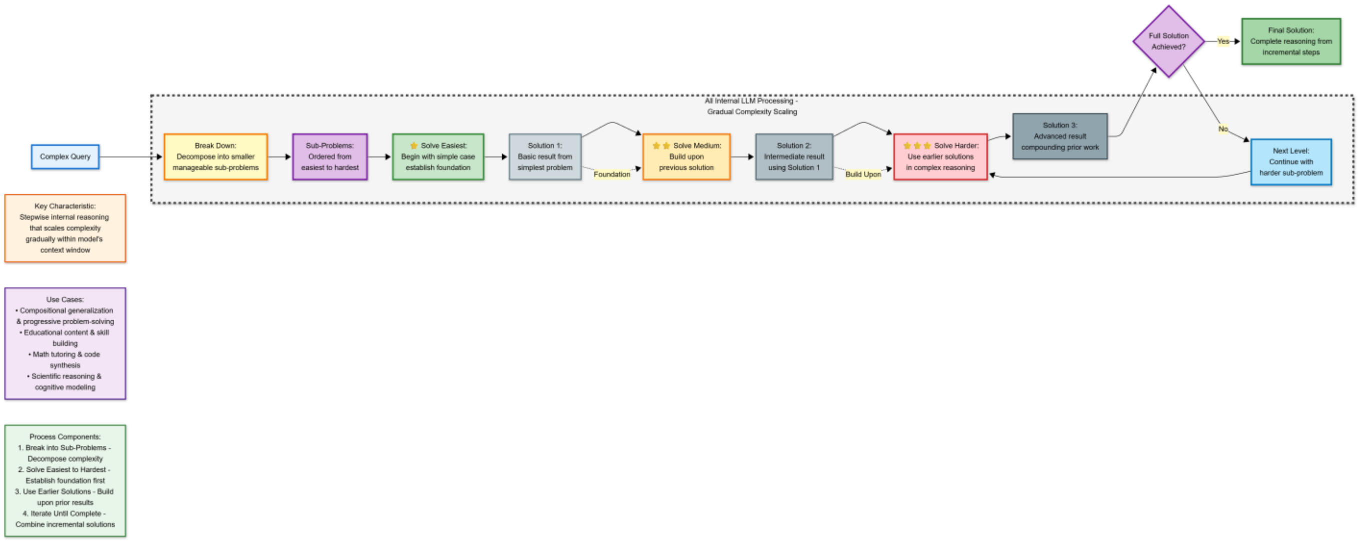
10. Least-to-Most Prompting

Paper: Least-to-Most Prompting Enables Complex Reasoning in Large Language Models (Zhou et al., 2022)
Source: Google Research

Why Non-Agentic: The model performs gradual internal reasoning by solving simpler sub-problems first, then compounding results, no tool use or real-world action.

Process:

1. **Break into Sub-Problems:** Decompose a complex query into smaller, manageable parts.
2. **Solve from Easiest to Hardest:** Begin with simple cases to establish foundational reasoning.
3. **Use Earlier Solutions in Later Problems:** Build upon previous results to handle higher complexity.
4. **Iterate Until Full Solution Emerges:** Combine incremental solutions to achieve final reasoning.



Key Characteristic: Stepwise internal reasoning that scales complexity gradually within the model's context window.

Use Cases:

- **Compositional generalization:** Solve tasks requiring layered or hierarchical logic.
- **Educational content:** Teach reasoning through incremental problem progression.
- **Progressive problem-solving:** Approach challenges where prior context informs next steps.
- **Skill building:** Train models or learners on gradual cognitive development.
- **Math tutoring:** Solve multi-stage equations by reasoning from simple to complex steps.
- **Code synthesis:** Build and debug functions progressively from basic logic to advanced structures.
- **Scientific reasoning:** Derive conclusions by iteratively applying principles to sub-parts.
- **Cognitive modeling:** Mimic human-like stepwise understanding and reasoning growth.

Code: [Colab Notebook]

---

11. Multi-Agent Debate

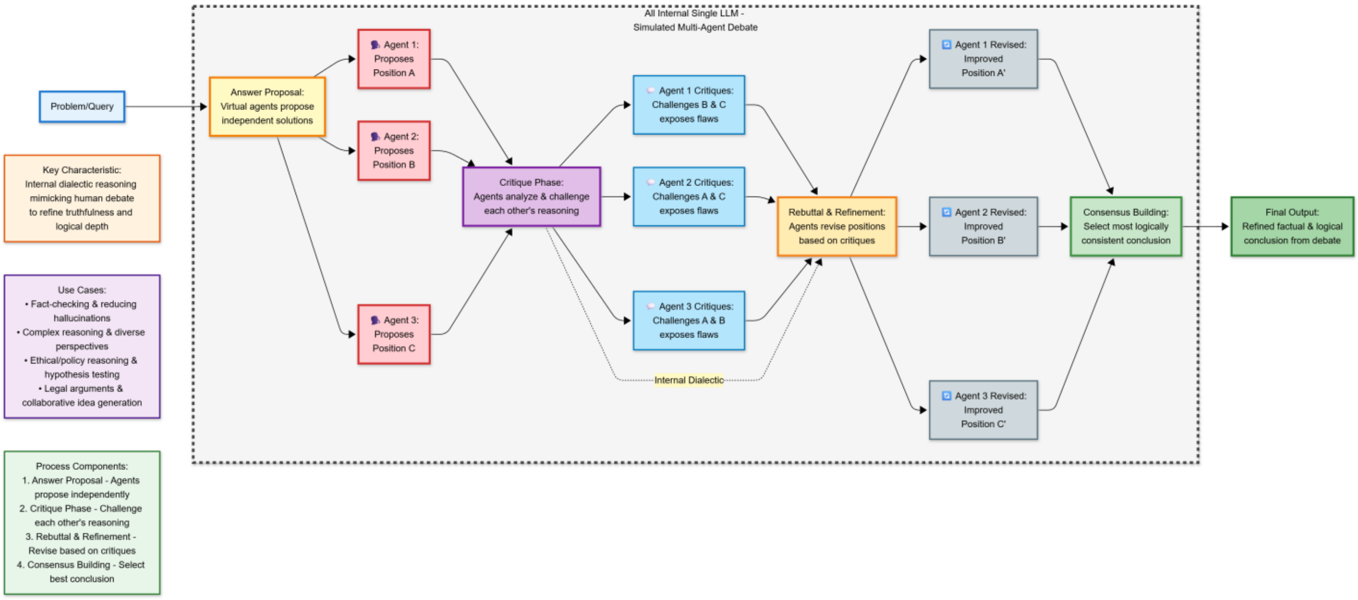
Paper: [Improving Factuality and Reasoning in Language Models through Multiagent Debate (Du et al., 2023)]
Source: MIT

Why Non-Agentic: Multiple simulated agents debate internally within the same LLM environment, no real-world feedback or tool use involved.

Process:

1. **Answer Proposal:** Multiple virtual agents independently propose solutions or arguments.
2. **Critique Phase:** Each agent analyzes and challenges others' reasoning, exposing flaws or biases.

3. **Rebuttal and Refinement:** Agents revise their positions based on critiques to improve factual soundness.
4. **Consensus Building:** Final output is selected based on the most logically consistent or factual conclusion.



Key Characteristic: Internal dialectic reasoning that mimics human debate to refine truthfulness and logical depth.

Use Cases:

- **Fact-checking:** Cross-verify claims using internal adversarial reasoning.
- **Complex reasoning:** Handle multi-dimensional problems requiring multiple viewpoints.
- **Reducing hallucinations:** Mitigate false outputs through internal critique and consensus.
- **Diverse perspective generation:** Explore different interpretations before forming a conclusion.
- **Ethical and policy reasoning:** Debate moral trade-offs or governance scenarios for balanced decisions.
- **Scientific hypothesis testing:** Contrast competing theories to identify the most evidence-aligned view.
- **Legal argument simulation:** Evaluate multiple legal interpretations and counterarguments internally.
- **Collaborative idea generation:** Simulate expert panel discussions to produce richer insights.

Code: [Colab Notebook]

---

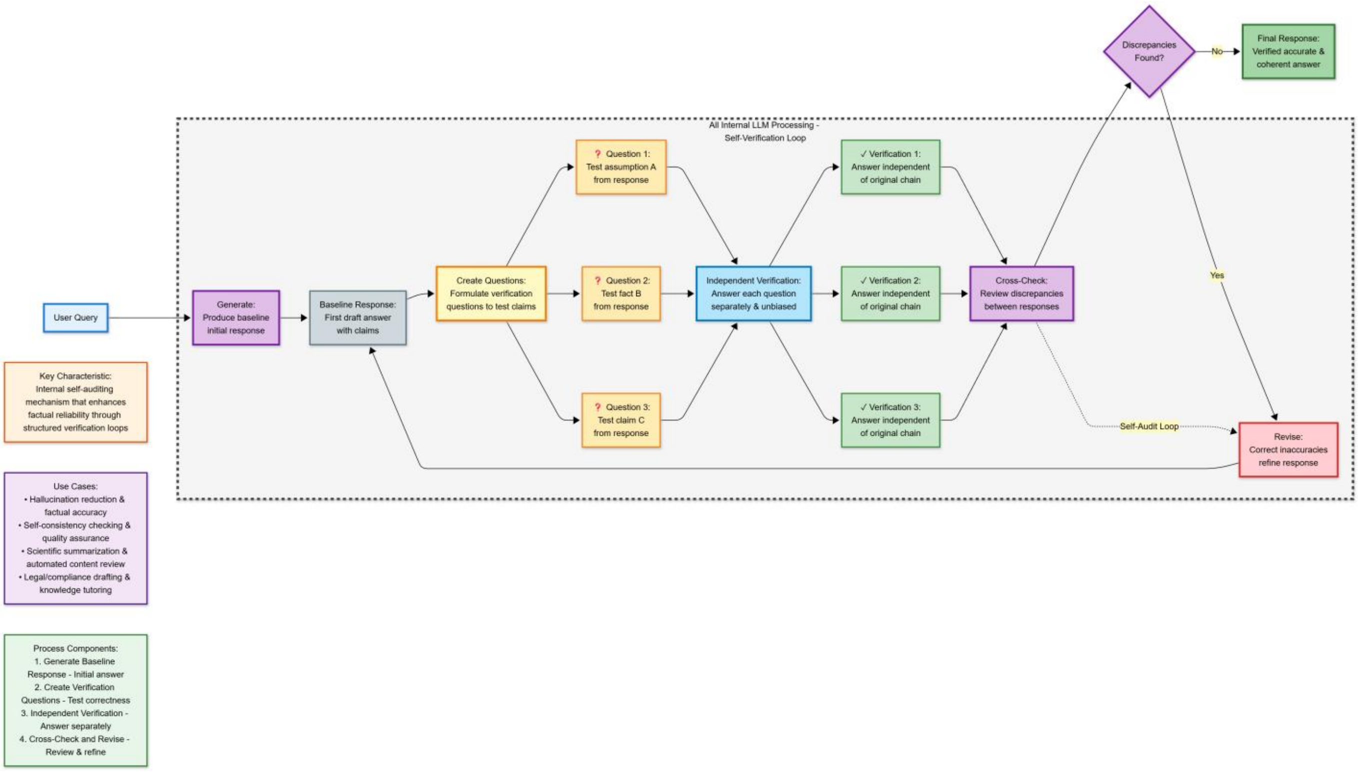
12. Chain-of-Verification (CoVe)

Paper: Chain-of-Verification Reduces Hallucination in Large Language Models (Dhuliawala et al., 2023)
Source: Meta AI

Why Non-Agentic: The model validates its own outputs through internal verification steps, without relying on external databases or real-world interaction.

Process:

1. **Generate Baseline Response:** The model first produces an initial answer to the user's query.
2. **Create Verification Questions:** It then formulates follow-up questions that test the correctness or assumptions in its own response.
3. **Independent Verification:** Each verification question is answered separately to avoid bias from the original reasoning chain.
4. **Cross-Check and Revise:** The model reviews discrepancies and refines its final response for factual accuracy and coherence. s



Key Characteristic: Internal self-auditing mechanism that enhances factual reliability through structured verification loops.

Use Cases:

- **Hallucination reduction:** Detects and corrects unsupported or fabricated claims internally.
- **Factual accuracy:** Improves the trustworthiness of generated information in knowledge-heavy domains.
- **Self-consistency checking:** Ensures logical alignment between intermediate reasoning steps and final answers.
- **Quality assurance:** Enhances precision in long-form or high-stakes text generation.
- **Scientific summarization:** Validates data interpretation within research-focused outputs.

- **Automated content review:** Performs internal QA for generated documents or reports.
- **Legal or compliance drafting:** Ensures statements align with defined rules or known facts.
- **Knowledge-intensive tutoring:** Provides accurate educational content by verifying key explanations internally.

Code: [Colab Notebook]

---

13. Self-Evolving Agents (Borderline)

Papers: Multiple surveys (2024-2025)
Source: International research

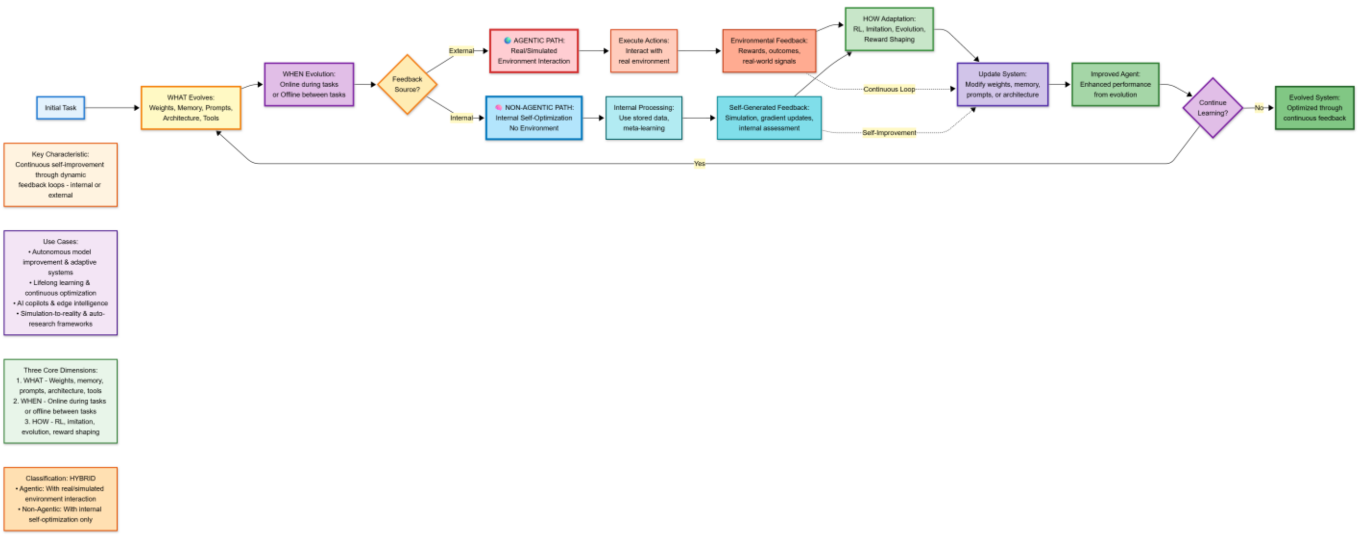
Classification: **Hybrid →** can operate as either Agentic or Non-Agentic depending on feedback source.

Why Hybrid:

- **Agentic** when the system interacts with real or simulated environments, executing actions and learning from feedback.
- **Non-Agentic** when it self-optimizes internally using stored data, meta-learning, or gradient-based updates without environmental input.

Three Core Dimensions:

1. **What:** Defines *what* evolves, model weights, memory, prompts, architecture, or tool usage strategy.
2. **When:** Specifies *when* evolution occurs, dynamically during tasks (online learning) or between tasks (offline retraining).
3. **How:** Describes *how* adaptation happens, via reinforcement learning, imitation of expert data, evolutionary algorithms, or reward shaping.



Key Characteristic: Continuous self-improvement through dynamic feedback loops, internal (simulation, meta-learning) or external (environmental reinforcement).

Use Cases:

- **Autonomous model improvement:** LLMs that refine reasoning or performance based on user interactions.
- **Adaptive systems:** Agents that modify behaviour across changing contexts or domains.
- **Lifelong learning:** Models that accumulate knowledge over multiple tasks without catastrophic forgetting.
- **Continuous optimization:** Systems that tune performance using feedback or self-assessment signals.
- **AI copilots:** Adaptive assistants that evolve through long-term user collaboration.
- **Edge intelligence:** On-device models that self-learn from real-world signals.
- **Simulation-to-reality adaptation**: Agents improving virtual learning outcomes for real-world execution.
- **Auto-research frameworks**: Meta-agents that experiment, analyze results, and self-upgrade prompts or architectures.

Code: Colab Notebook

Quick Decision Guide

Use Agentic Patterns when you need:

- Real-time information gathering
- Tool and API integration
- Environment interaction
- Execution feedback
- Persistent state changes
- Multi-turn exploration

Use Non-Agentic Patterns when you need:

- Better reasoning quality
- Complex problem decomposition
- Self-improvement loops
- Multiple perspective exploration
- Structured thinking
- No external dependencies

Combine Both when:

- Planning (non-agentic) + Execution (agentic)
- Reasoning (non-agentic) + Tool use (agentic)
- Internal debate (non-agentic) + Action taking (agentic)

Summary

Agentic Patterns

Pattern	Key Action	External Interaction
ReAct	Tool execution	APIs, tools, databases
Reflexion	Task execution + learning	Environment feedback
Toolformer	API calls	Calculator, search, calendar
ART	Multi-tool chains	Tool library execution
Generative Agents	Environment actions	Simulated world
LATS	Action search	Environment state changes
RAG	Information retrieval	Database queries
RAP	Action planning	World model execution
Deep Research	Multi-turn research	Web, APIs, browsers
ACI	Computer control	GUI, system operations
Orchestrator-Worker	Distributed execution	Multi-agent tool use
MCP Agents	Protocol-based tools	Universal tool access

Non-Agentic Patterns

Pattern	Key Feature	Interaction Type
Chain-of-Thought	Step reasoning	Internal only
Tree of Thoughts	Path exploration	Internal only
Graph of Thoughts	Graph reasoning	Internal only
Self-Refine	Self-improvement	Internal only
Plan-and-Solve	Structured planning	Internal only
RCI	Recursive critique	Internal only
Meta-Prompting	Expert simulation	Internal only
Skeleton-of-Thought	Parallel generation	Internal only
Decomposed Prompting	Task breakdown	Internal only

Least-to-Most	Progressive solving	Internal only
Multi-Agent Debate	Virtual debate	Internal only
Chain-of-Verification	Self-verification	Internal only

Happy Learning 😊