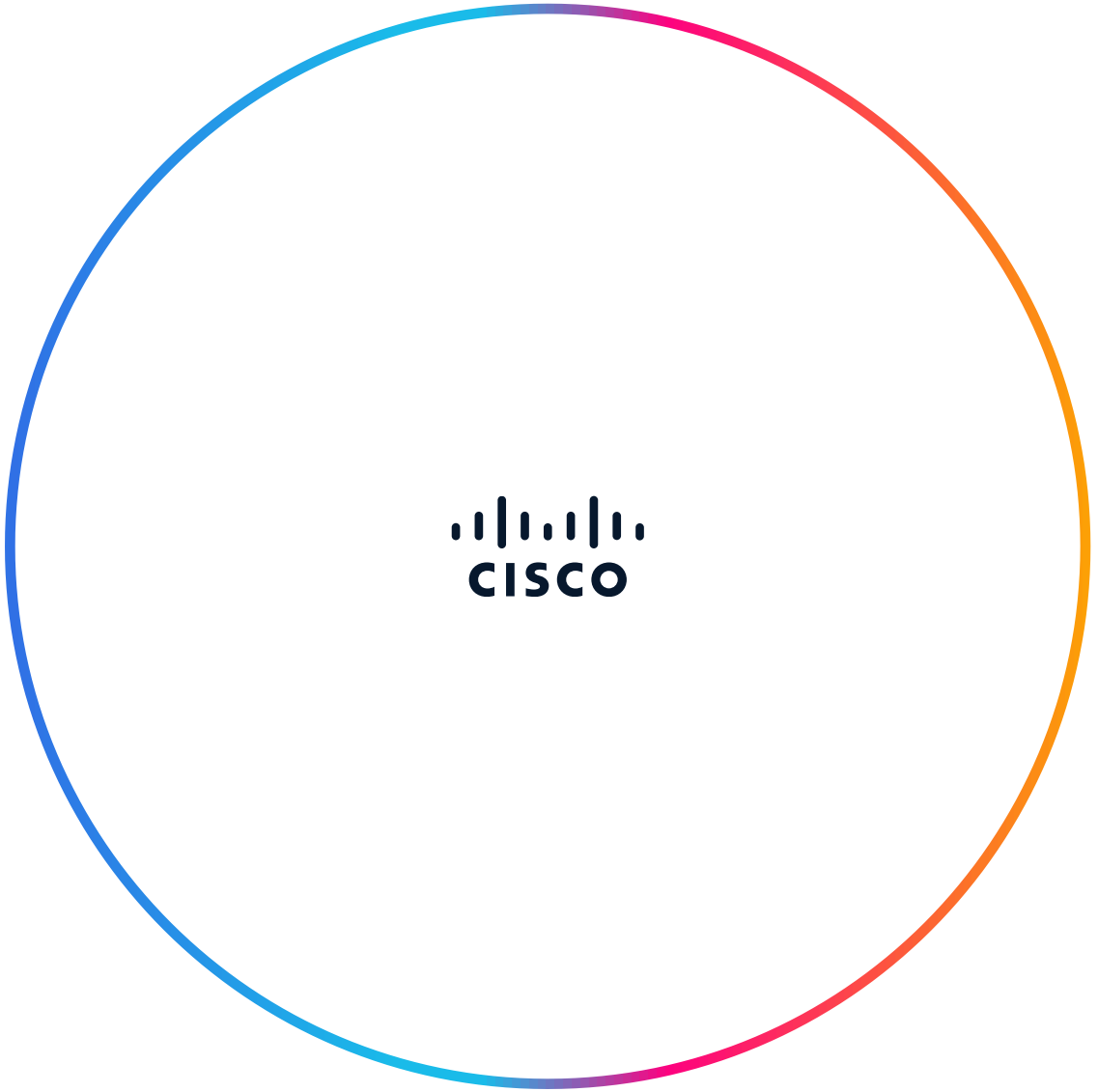
How to build AI agents for automating networking software development



Shamin Aggarwal
Software Engineer

Randy Zhang
Principal Architect

AI-1013

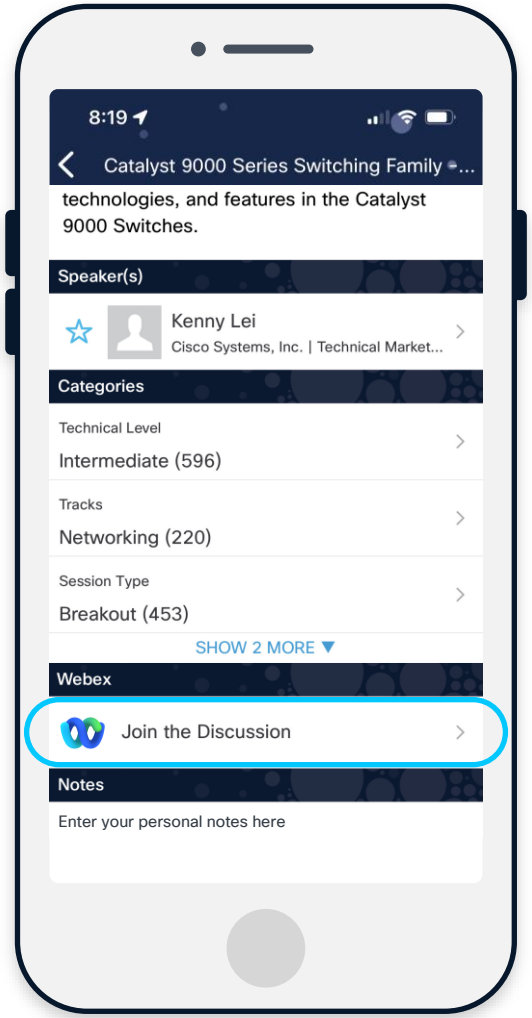
Webex App

Questions?

Use Webex app to chat with the speaker after the session

How

1. Find this session in the Cisco Events app
2. Click “Join the Discussion”
3. Install the Webex app or go directly to the Webex space
4. Enter messages/questions in the Webex space



Webex spaces will be moderated by the speaker until February 27, 2026.

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

Agenda

01 Introduction

02 Recommendations

03 Summary

© 2026 Cisco and/or its affiliates. All rights reserved.

Introduction

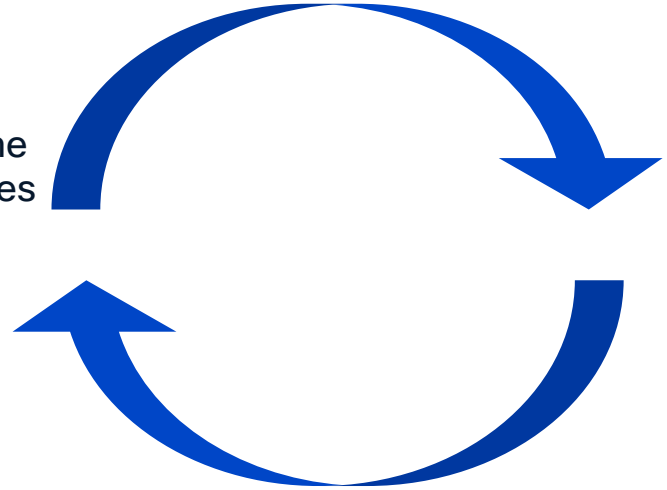
Software Development Lifecycle

2. Specifying functional components

Creating a plan for the overall software structure and architecture

1. Defining feature requirements

Clearly define the scope of the project with detailed objectives and requirements



3. Code development

Writing the code with a given language based on the specifications

4. Code testing and execution

Executing the code in a test environment

5. Bug fixes and enhancement

Fixing bugs and adding enhancements

6. Code validation

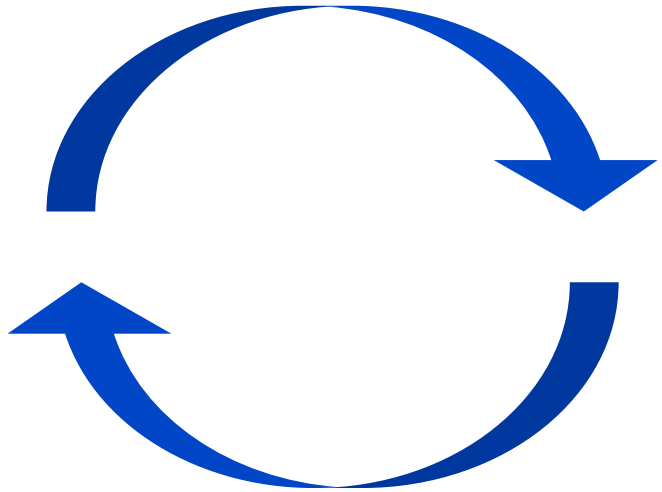
Validating code in an end to end development environment

© 2026 Cisco and/or its affiliates. All rights reserved.

AI Can Help with an Agentic Workflow

2. Convert requirements into an action plan for coding

1. Understand my networking requirements



3. Write code based on the plan

4. Execute tests and review the code

5. Fix errors and improve the code

6. Validate the code

© 2026 Cisco and/or its affiliates. All rights reserved.

Why Agentic AI?

89% of executives expect agentic AI to become the industry standard for software development within 3 years.

The Harris Poll: The Economics of Software Innovation
https://about.gitlab.com/software-innovation-report/

© 2026 Cisco and/or its affiliates. All rights reserved.

AI Tools for Software Development

AI Chatbots

- Reactive AI to user prompts
- Interactive: request and response
- Separate tools for different functions
- Useful for isolated coding needs or a small project

Context Aware AI Assistants

- Proactive AI solutions with limited context
- Monitors ongoing activities, offers relevant suggestions
- Requires human approval or intervention for actions

Agentic AI

- Autonomously pursue complex goals
- Independently initiate actions, make decisions within defined boundaries,
- Coordinate complex workflows without requiring human approval for each step.
- Use multi-agents for long running tasks

- Enables "lights-out" development operations for routine tasks, optimizations, and even complex problem-solving automatically
- Allows human developers to focus on strategic architecture, business logic, and innovation rather than operational overhead

© 2026 Cisco and/or its affiliates. All rights reserved.

Agentic Functions

Planning

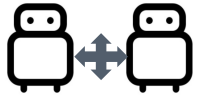
Breaking down complex tasks into manageable steps, determining sequence of actions, self prompting

Reflection

Reviewing and critiquing their own outputs, iterate based on self feedbacks

Tool Use

Using specialized tools to finish specific tasks



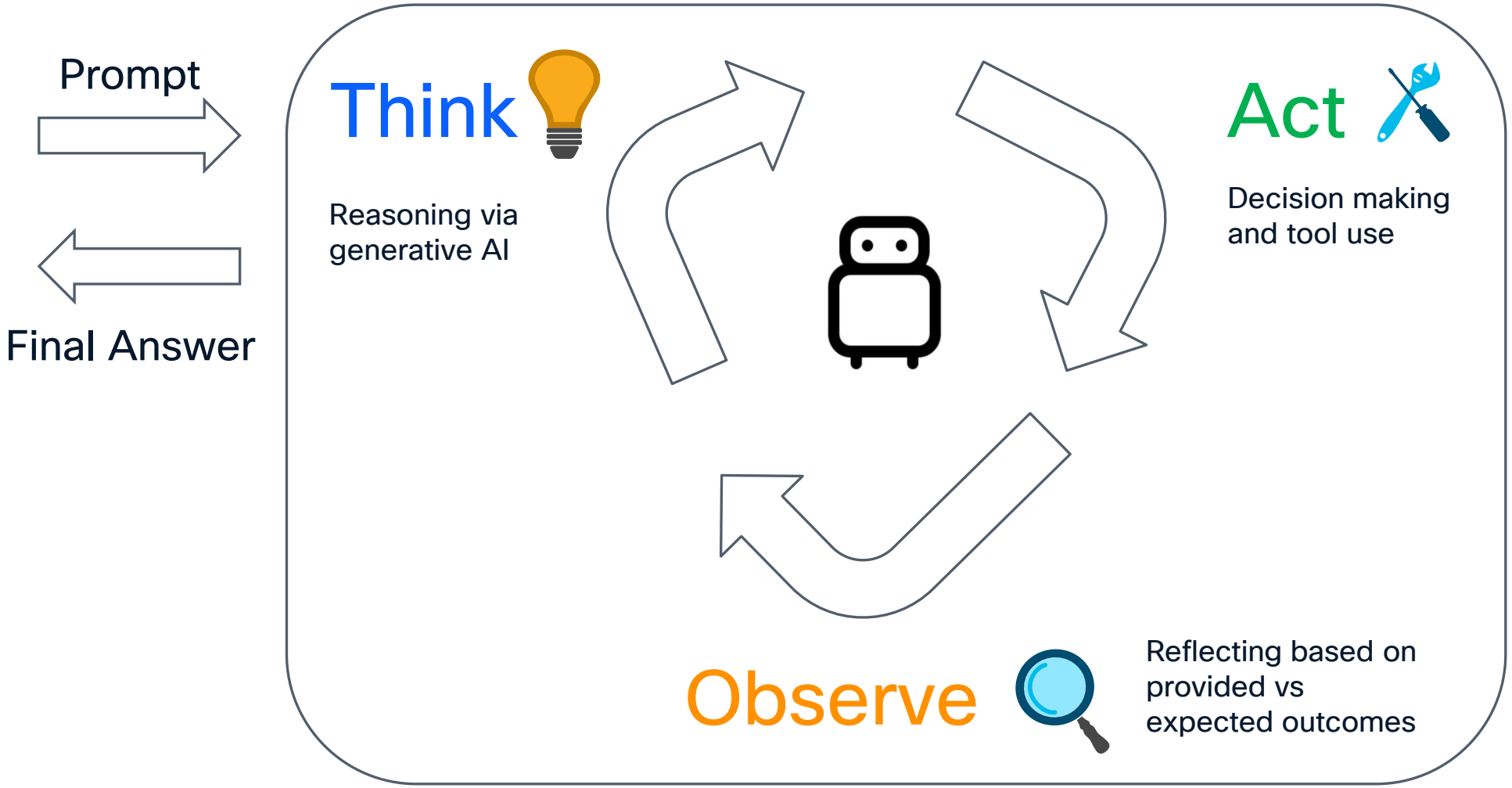
Collaboration

Delegating work to other agents

© 2026 Cisco and/or its affiliates. All rights reserved.

Agentic Loop

A continual loop until objective is achieved



© 2026 Cisco and/or its affiliates. All rights reserved.

A Multi-Agentic Example for Software Development

Agent	Functions
Planner	Kicks off the project with a detailed project plan
Coder	Writes and debugs code per requirements
Critic	Reviews code, points out issues and makes suggestions for enhancement
Unit Tester	Writes and runs unit tests
System Tester	Writes and runs system tests
Reporter	Summarizes the project and writes code documentation

© 2026 Cisco and/or its affiliates. All rights reserved.

A BGP Script Development Example

Objective: Create a script to analyze BGP configurations against best practices. Validate the script through live routers.

Artifacts

1. Describe the project to agents
2. Provide BGP best practices
3. Provide a set of routers for live validation
4. Provide a sandboxed workspace for agents to execute the script

Task Planning

1. Login to Cisco routers
2. Collect the BGP running configuration
3. Compare it with BGP best practices
4. Generate a recommendations report

GitHub https://github.com/ranzhang/ai-codegen-routers

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

Agentic Use Cases for Software Development

- New feature development
- Legacy code modernization (via codebase analysis)
- Bug fixes
- Unit testing
- Codebase standardization
- Validation & Documentation generation

These use-cases show where agentic AI can deliver rapid ROI, especially in organizations with complex, evolving codebases and compliance needs.

© 2026 Cisco and/or its affiliates. All rights reserved.

Recommendations

Summary of Recommendations

Foundation & Configuration

1. **Agentic Library Selection:** Selection of an agentic framework for software development
2. **Large Language Model Selection:** Selection of an LLM that is optimized for code writing and debugging

Operation & Intelligence

1. **Grounding Agents for Developing Quality Code:** Providing AI agents with domain knowledge and environment to reduce hallucination
2. **Handling Context between Agents:** Sharing agent decisions and outputs for common knowledge to reduce noise and token usage
3. **Optimizing Token Consumption:** Getting the most amount of work done in least number of words for LLMs
4. **Tool Calling**: Enhancing agents' abilities to interact with workspace and/or external environment through custom functions
5. **Dependency Analysis**: Allowing Agents to lookup definitions of upstream code dependencies

Quality Assurance & Feedback

Testing & Validation: Automated testing and review with iterative enhancement

Building AI Agents to Automate Enterprise-Level Software Development: A Practical Perspective

© 2026 Cisco and/or its affiliates. All rights reserved.

Agentic Library Selection

Multi-Agent Conversation Patterns

- Supports diverse chat models (e.g., two-agent, group chat, hierarchical).
- Allowed us to model a dev team for collaborative task completion.

Tip: *Clearly define agent roles and communication protocols.*

Human-in-the-Loop (HITL) Workflows

- Enables human participation for input, decisions, approvals, or corrections.
- Crucial for validating critical code or guiding agents in ambiguous situations.

Tip: *Implement HITL at key checkpoints (e.g., code merges, complex bug fixes).*

Tool Use Integration

- Allows agents to execute code (sandboxed) and manage files.
- Agents can write, save, execute, and analyze code files.

Fast Prototyping & Complex Task Orchestration

- High-level abstractions facilitate quick setup of multi-agent systems.
- Allowed us to rapidly test automation ideas for various software development lifecycle stages.

Tip: *For production, evaluate operational aspects (monitoring, logging, error handling, security, scalability).*

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

AutoGen Example: Setting up Agents

Configure an LLM

```
openai_config_list = [
  {
    "model": "gpt-4o",
    "api_key": OPENAI_API_KEY,
    "base_url": azure_endpoint,
  },
  {
    "model": "gpt-4-turbo",
    "api_key": OPENAI_API_KEY,
    "base_url": azure_endpoint,
  }
]
```

```
llm_config_openai = {
        "config_list": openai_config_list,
    }

llm_config = llm_config_openai
```

Set up a Planner agent

```
Planner = ConversableAgent(
    name="Planner",
    system_message="""You are a software architect. You generate clear and detail
coding instructions per requirements for Coder.
     You can generate low level logic to help Coder but you do not write the code""",
    llm_config= llm_config,
)
```

Set up the User Proxy agent for user interactions

```
user_proxy = UserProxyAgent(
    name = "user_proxy",
    human_input_mode = "ALWAYS",
    code_execution_config={
        # the executor to run the generated code
        "executor":LocalCommandLineCodeExecutor(work_dir="tmp_dir"),
    },
```

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

AutoGen Example: Launching the Workflow

Set up the group chat for agent messaging

```
group_chat = GroupChat(
    agents = [user_proxy, Planner, Coder, Critic,
Reporter],
    messages = [],
    send_introductions= True,
    speaker_selection_method = “auto”,)

group_chat_manager = GroupChatManager(
    groupchat = group_chat,
    llm_config = llm_config,)
```

Initiate the agentic workflow

```
chat_result = user_proxy.initiate_chat(
    group_chat_manager,
    message=""" You lead a team to collaborate on a networking
feature development project using Python.
    The Python code should perform the following 3 tasks:
    <snip>    """,
    summary_method="reflection_with_llm",
    max_turns=5,
)
```

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

Grounding Agents for Developing Quality Code

To ensure AI agents generate accurate, relevant, and high-quality code, they must be "grounded." This involves providing them with the **necessary context, constraints, and feedback mechanisms** related to the specific software development task.

Narrows the solution space, ensures adherence to project conventions, and reduces irrelevant outputs.

Tip: *Curate and structure your knowledge base for efficient retrieval by the agent.*

Minimizes misinterpretations and guides the agent towards the desired solution.

Tip*: Break down complex tasks into smaller, well-defined sub-tasks with clear objectives for each agent or step.*

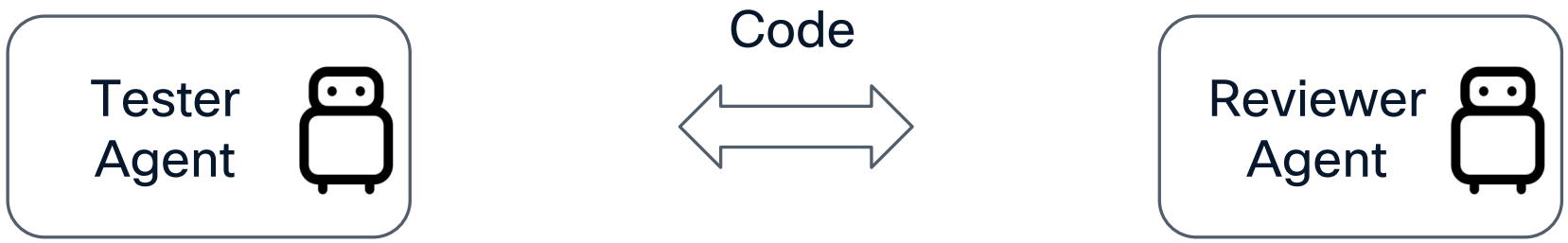
- Simulates a peer review process, catching errors, improving code quality, and ensuring adherence to requirements iteratively.
- Integrated testing provides continuous feedback on code correctness and functionality, enabling agents to iteratively improve their output. Catches regressions early.

Tip*: Define clear review and testing criteria.*

© 2026 Cisco and/or its affiliates. All rights reserved.

Code Testing and Review

Rigorous code validation is essential to ensure that AI-generated or modified code is correct, reliable, adheres to standards, and meets all specified requirements before integration. This involves a collaborative effort, often mimicked by specialized AI agents.



Role

Focuses on functional correctness and robustness by executing automated tests.

Actions

- Runs unit tests against new/modified code;
- Executes integration tests to check interactions with other modules.

Output

Detailed test results (pass/fail status, error messages, code coverage reports).

Role

Evaluates the code against a broader set of criteria, including requirements, quality standards, and best practices.

Actions

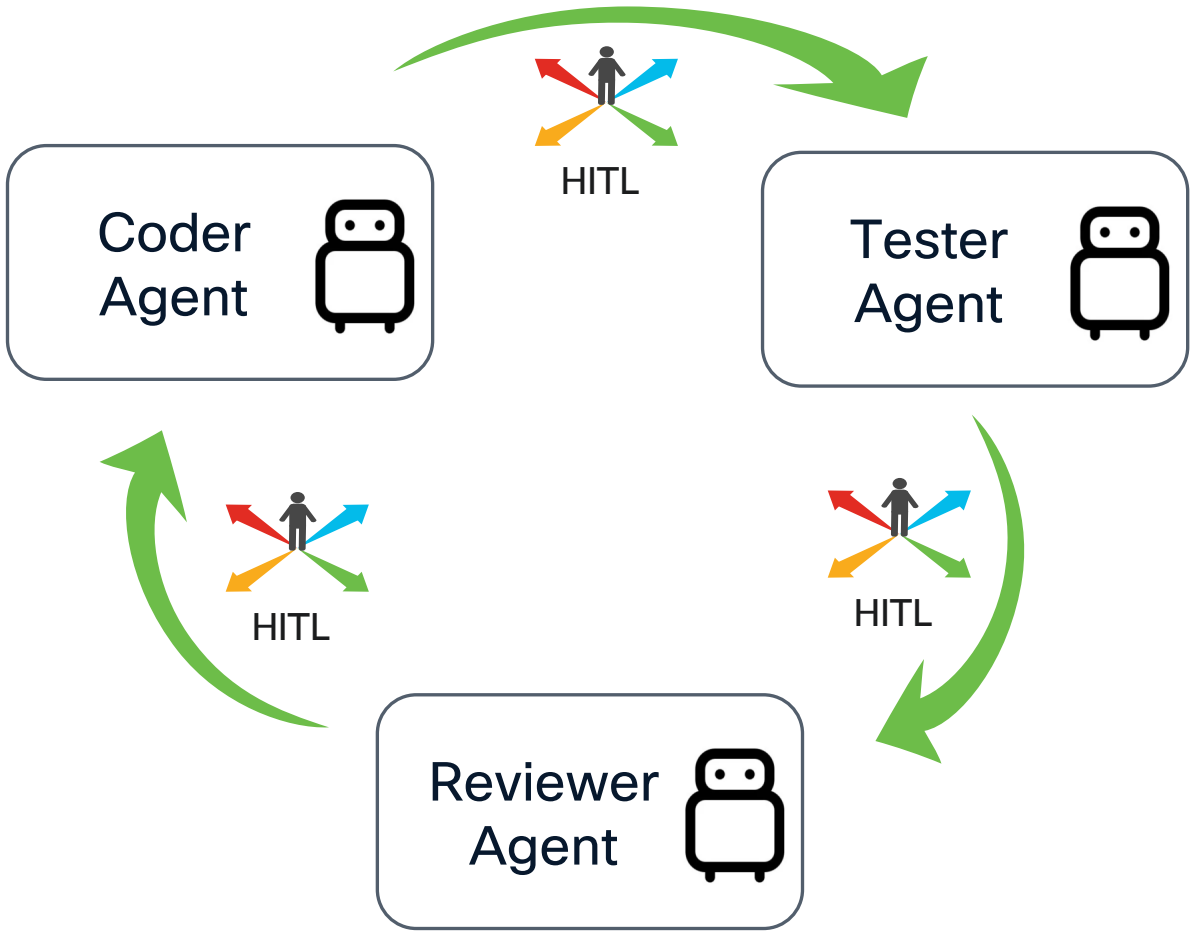
- Reviews code against functional requirements
- Assesses non-functional requirements (e.g., performance, security vulnerabilities).
- Flags discrepancies or areas for improvement with specific, actionable feedback

Output

A structured review report, list of issues, suggestions for refinement, or approval

© 2026 Cisco and/or its affiliates. All rights reserved.

Code Validation Loop



Repetition: This loop repeats until the code passes all tests and meets the Reviewer agent's approval criteria.

Human Oversight: Human-in-the-loop (HITL) can be integrated at any stage, especially for complex issues or final approval.

Benefits of Multi-Agent Validation

- **Ensured Quality:** Systematically improves code quality by enforcing functional correctness and adherence to non-functional standards.
- **Increased Confidence:** Builds trust that the AI-generated code meets project objectives and quality bars before it's integrated into the main codebase.
- **Efficient Refinement:** Provides targeted feedback, allowing the Coder agent to make more effective revisions.

© 2026 Cisco and/or its affiliates. All rights reserved.

Summary

Summary and Additional Information

Agentic AI presents a new low code alternative to networking software development

LABAI-1402: Exploring Agentic AI for Network Operations

Building AI Agents to Automate Enterprise-Level Software Development: A Practical Perspective

Automated networking script development with AI agents

Project Neo: an AI-powered automated event-routing and response system

Demystifying AI Networking Protocols

© 2026 Cisco and/or its affiliates. All rights reserved.

Complete your session surveys

Complete your surveys in the Cisco Events app.

Complete a minimum of 4 session surveys and the overall event survey to receive a unique Cisco Live t-shirt.

(from 11:30 on Thursday, while supplies last)

© 2026 Cisco and/or its affiliates. All rights reserved.

Continue your education

Visit the Cisco Showcase for related demos

Book your one-on-one Meet the Engineer meeting

Visit the Technical Solutions Clinics to discuss your technical questions

Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs

Visit the On-Demand Library for more sessions at CiscoLive.com/On-Demand

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

Thank you


CISCO

Large Language Model Selection

REFERENCE

The choice of LLM significantly impacts your AI agent's performance, cost, and capabilities in software development automation. A diverse range of models exists, varying in size, specialization, access (open-source vs. proprietary), and cost.

For most software development tasks, a larger context window is highly beneficial, if not essential.

LLM Selection Considerations

1. Model Size & Capabilities vs. Use-Case Performance: Match model capability to task complexity. Don't overpay for a powerful model on a simple task.
2. Context Window vs. API Cost & Latency: A larger context window allows more information to be processed, leading to better understanding but typically incurs higher costs and potentially higher latency.
3. Essential Reasoning & Coding Abilities:
- **Tool/Function Calling:** Critical for agents to interact with external tools (compilers, linters, APIs, file systems). *Non-negotiable for most software agents.*
- **Chain-of-Thought (CoT) / Step-by-Step Reasoning:** Enables models to break down complex problems, show their work, and arrive at more accurate solutions. Look for models strong in this.
- **Code Understanding & Generation Quality:** Evaluate proficiency in target programming languages, ability to understand existing code, generate new code, debug, and explain code.
- **Tip:** Test models on representative coding tasks specific to your domain.

4. And Many More: Open-Source vs. Proprietary Models, Fine-Tuning & Customization, Cost, Latency, and Rate Limits

© 2026 Cisco and/or its affiliates. All rights reserved.

Handling Context between Agents

REFERENCE

For AI agents to collaborate effectively on software development tasks, mechanisms for managing shared understanding (context) is essential. Poor context handling leads to errors, rework, and inefficient collaboration.

Data & Conversational Context Management Techniques

- **Explicit Context Passing & Summarization:**
  - **Action:** At the end of an agent's task or phase, explicitly summarize key findings, decisions, outputs, and unresolved issues.
  - **Mechanism:** This summary (structured or natural language) is then passed as focused input to the next agent(s) in the workflow.
  - **Benefit:** Reduces noise, manages token limits for LLMs, and ensures the next agent receives relevant information.

- **Threaded Conversations & Message History:**
  - **Action:** Maintain a clear history of interactions, decisions, and rationales within a specific task or sub-task.
  - **Mechanism:** Most agent frameworks (like AutoGen) manage this inherently within agent conversations.
  - **Benefit:** Allows agents (and humans) to trace back decisions and understand the evolution of a solution.
  - **Tip:** Ensure that the full relevant history is accessible or adequately summarized when an agent needs to make a context-dependent decision.

© 2026 Cisco and/or its affiliates. All rights reserved.

Optimizing Token Consumption

REFERENCE

- **Cost Efficiency:** LLM usage costs are typically proportional to the number of tokens processed (both input and output). Optimizing "work done per token" is vital.
- **Performance & Scalability:** Effective token management is essential for handling complex tasks, scaling agentic workflows, and preventing errors due to exceeding context window limits (e.g., context loss, truncation).

Challenge	Mitigation
Finite LLM Context Windows	**Context Summarization**: Use prompts that ask for concise summaries or key takeaways necessary for next Agent
Long tasks or large codebases can exceed limits	**Focused Processing**: Use tools like Abstract Syntax Tree parsers and dependency analyzers, to allow Agents to request 'focused' code blocks
Verbose/Redundant Prompts & LLM Outputs	· **Precise Prompt Engineering** · **Instruction Re-use**: Utilize system prompts for standing instructions or roles, rather than repeating them in every user message · **Requesting Structured/Concise Output**
Inefficient Agent Communication Loops	· **Purposeful Agent Interactions:** Restrict Agent interactions with an FSM-like graph · **Caching LLM Responses**

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco

Tool Calling

REFERENCE

Well-designed tool calling (also known as function calling) transforms AI agents from mere text generators into active participants that can interact with real-world systems, access information, and perform concrete actions, enabling true automation.

General Considerations for Designing & Integrating Tools

1. Robustness & Reliability:
Tools must handle a variety of inputs gracefully and recover from or report errors clearly.

2. Flexibility & Adaptability:
Tools should be configurable and adaptable to different scenarios or parameters provided by the agent.

3. Security & Safety:
- **Sandboxing:** Execute code or commands in isolated environments.
- **Permissions & Access Control:** Limit what tools can do based on agent roles or task requirements.
- **Human-in-the-Loop (HITL):** Require human approval for high-risk tool operations.

4. Clear Interface & Discoverability (for the Agent):
Provide clear, concise, and accurate descriptions (function signatures, docstrings) of tools to the LLM. This enables the LLM to correctly decide when and how to use a tool.

5. Observability & Debuggability:
Log tool invocations, inputs, outputs, and any errors. It's essential for understanding agent behavior, debugging issues, and monitoring tool usage.

6. Relevant Tools for Software Development AI Agents:
- File System Operations
- Code Execution & Analysis
- Testing & Validation
- Dependency & Version Control
- Information Retrieval & External APIs

© 2026 Cisco and/or its affiliates. All rights reserved.

Dependency Analysis

REFERENCE

Challenge: Ensuring that code generated or modified by one agent correctly integrates with code from other parts of the project or external libraries.

Tool-Assisted Dependency Verification

- **Action:** Equip agents with tools to inspect and verify code dependencies.
- **Examples:**
  - **Static Analysis Tools:** To parse import statements, check for undefined functions/classes, and analyze call graphs.
  - **Build System Integration:** To verify that dependencies are correctly declared and resolvable.
  - **IDE-like Functionality (via tools):** To perform "go to definition" or look up function signatures from imported modules.
- **Benefit:** Proactively identifies potential integration issues before full compilation or runtime.
- **Tip:** Provide agents with access to the project's dependency manifest files (e.g., requirements.txt, pom.xml, package.json).

© 2026 Cisco and/or its affiliates. All rights reserved.

cisco


CISCO