# **Technical Research Report: A Reflective Coding Agent with a Read-Think-Write Micro-cycle**

Version: 23.0

Date: 18 June 2025

Status: Approved

1. Abstract

This document details a critical architectural enhancement for the Coding Agent, evolving it from a static plan-executor to a reflective agent with a dynamic execution loop. The previous "scaffold" implementation revealed a key failure mode: attempting to generate the complete content of a modified file within the initial planning phase is a "one-shot" task that is too complex and brittle. This resulted in the agent producing placeholders instead of code. The v7 architecture solves this by introducing a Read-Think-Write micro-cycle within the agent's execution node. For each high-level task in its plan (e.g., "Add a function to supervisor.py"), the agent will now 1) Read the current file content, 2) Think, using a dedicated LLM call to generate the new, modified content, and 3) Write the result back to the file. This decomposition of the task dramatically reduces the cognitive load of each LLM call, enhances reliability, and moves the agent from a simple tool-caller to a more robust, context-aware programmer.

2. Research: The Failure of One-Shot Code Generation

The v6 implementation attempted to have the Coding Agent generate its entire execution plan, including the final code for each file, in a single LLM call. This failed. The agent correctly identified the write\_file tool was needed but used a placeholder like "<Complete content...>".

This reveals a fundamental principle of agentic design: complex generative tasks must be decomposed. Asking an LLM to simultaneously hold the logic of multiple file changes and generate the full text for all of them leads to cognitive overload. The model then defaults to a high-level abstraction (the placeholder) rather than producing an incorrect, low-level implementation.

The solution is to mimic a human developer's workflow more closely. A developer does not write a plan that includes the final code. They write a plan of *actions*, and then execute each action individually, focusing on one file at a time.

3. The Read-Think-Write Micro-cycle

The v7 architecture implements this human-like workflow within the code\_execution\_node. The agent is no longer a simple loop that executes pre-determined tool calls. Instead, for each high-level step in its plan, it performs a new, more intelligent sub-process:

1. Read: The agent uses a read\_file tool to load the full, current content of the target file into its context.
2. Think (Generate): This is the core enhancement. The agent makes a new, highly-focused LLM call. It uses a dedicated prompt (generate\_file\_content\_prompt) that combines the original file content with the specific instruction from its plan (e.g., "Add a dashboard function"). The LLM's sole task is to return the complete, new version of the file.
3. Write: The agent takes the generated content from the "Think" step and uses the write\_file tool to save the changes.

This architecture ensures that each LLM call has a narrow, manageable context, leading to higher-quality code generation and a much more robust and reliable Coding Agent.