Software Analysis Guide

AXIOM Analysis

Software Analysis Guide

Version 1.0 | April 12, 2026

Purpose

Use the Software domain when you want AXIOM to review technical material for likely defects, regressions, architecture stress points, rollout risk, and validation gaps.

This workflow is strongest when the input contains real technical evidence:

- code or diffs
- architecture notes
- logs or traces
- expected behavior
- the reason the review matters now

Best Starting Settings

Code / change review

- Domain: `Software`
- Modules: `review`
- Report depth: `Standard`
- Engine: `AXIOM Gateway` or `Auto`
- Live sources: `off`

Architecture review

- Domain: `Software`
- Modules: `architecture`
- Report depth: `Standard`
- Engine: `AXIOM Gateway` or `Auto`
- Live sources: `off`

Broad technical review

- Domain: `Software`
- Modules: `review`, `architecture`
- Report depth: `Standard`
- Engine: `AXIOM Gateway` or `Auto`
- Live sources: `off`

What To Prepare

Before you run the analysis, collect:

- the scope being reviewed
- what changed
- the expected behavior
- the current observed behavior
- the main risk areas
- code or config evidence
- logs, tests, and open questions

If you only provide a vague summary, the review will stay vague.

Best Input Pattern

The cleanest current pattern is:

1. fill out the intake template
2. upload the intake plus the technical material
3. use the short prompt file in the portal

This works better than turning the request box into a long, improvised technical memo.

Common Mistakes

- asking for a serious review without uploading the real code or logs
- not naming the files or modules in scope
- not saying what behavior is expected
- mixing architecture concerns and bug symptoms without labeling them
- using `fast` when the task is actually a deep review

Recommended Run Order

1. Start with `review` or `architecture` and `Standard`
2. Improve the evidence pack if the output feels generic
3. Use `review + architecture` when both code and structure matter
4. Only then try `Deep`

What A Strong Result Should Include

A strong software report should clearly show:

- the highest-risk issues
- the evidence behind them
- likely consequences
- architecture stress points
- missing tests or validation
- the most important information gaps

Related Files

- `SOFTWARE_ANALYSIS_INTAKE_TEMPLATE_2026-04-12.md`
- `SOFTWARE_ANALYSIS_PROMPT_2026-04-12.txt`
- `SOFTWARE_ANALYSIS_EXAMPLE_REVIEW_2026-04-12.txt`

End of Software Analysis Guide v1.0