Source: authored in b3u_use_cases as this use case's structure note.

b3ubot — what the steps built

A companion to the published step specifications. This is the shape of the program those steps produced, so a reader can weigh the input against the output.

The application itself is not published here; subscribers download it from b3u.dev.

Subsystems

subsystem	files	lines	what it is
app	44	10 264	orchestrator, tool surface, ledger engine, ratify gate
adapters	9	731	provider bindings (BYO key)
dag	127	15 192	⚠ FORKED workflow substrate — not covered by the claim
scripts	9	1 194	operator entry points
tests	52	10 603	the suite (688 tests)
docs	6	866	design + ledger
total	247	38 850	

Reading the table

`app/` is the assistant. The orchestrator that turns a request into a plan, the tool surface (allowlisted shell, git operations, refusal stubs for destructive actions), the ledger engine, and the ratify gate that makes a proposed step wait for approval.

`adapters/` is deliberately thin. Provider bindings only — the program is provider-agnostic and brings no key of its own. 731 lines is the whole of it, which is the point: swapping a provider is not an architectural act.

⚠ **`dag/` is forked, and is not covered by the generation claim.** It is a multi-target workflow substrate cloned from a different project at a recorded commit, and carries its own provenance file. Within it, much is itself CCS-generated — from two SGDL grammars into four language targets — but it was generated *there*, by a different arc of steps, not by the 29 published here.

`tests/` is nearly as large as `app/`. 10603 lines against 10264. That ratio is not incidental to the method: a specification that cannot be checked is a wish, and the retros show tests refusing their own step's skeleton more than once.

What this does and does not show

Excluding the forked subtree, roughly 22,792 lines correspond to 29 prose specifications.

That ratio is the interesting number, and it is also where a careful reader should be sceptical: lines of code is a poor measure of anything, generated code is often more verbose than hand-written code, and a specification that produced working software is not the same as one that produced *good* software. The step files and their retros are published so that judgement can be made from the evidence rather than from a ratio.