My Claude Code Workflow Today

Discovery

What are we changing?

→

Planning

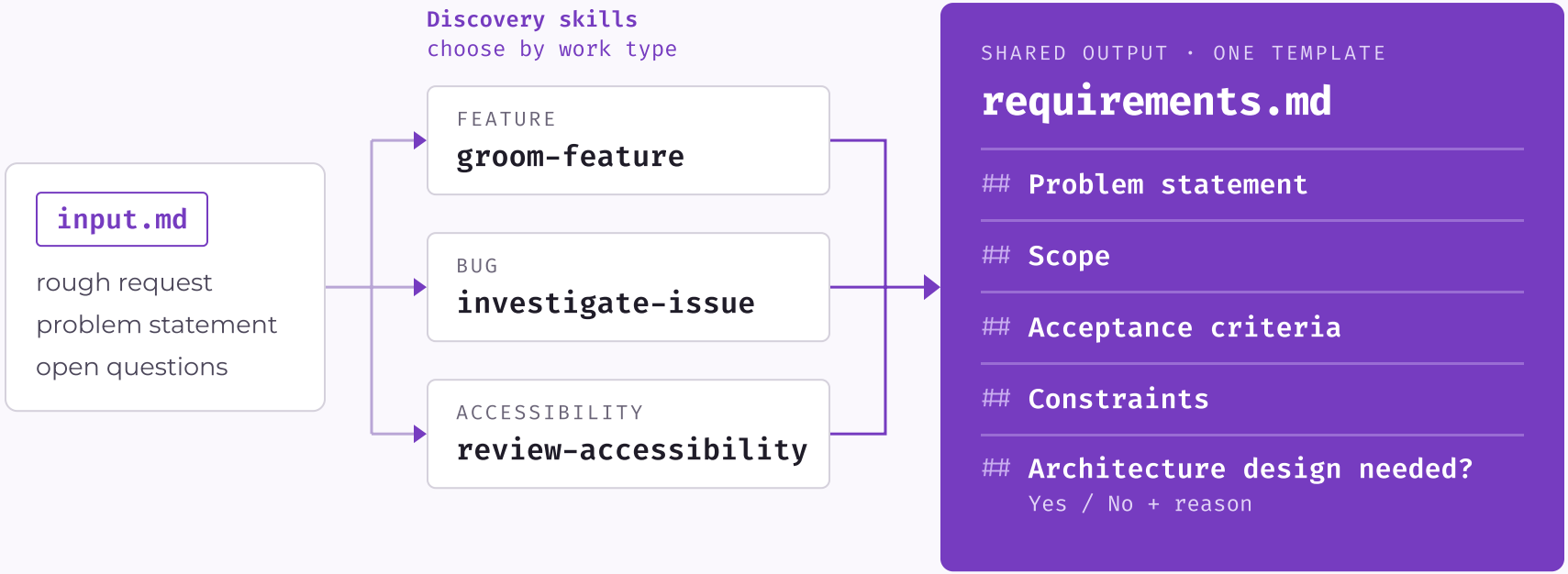
How will we deliver it?

→

Implementation

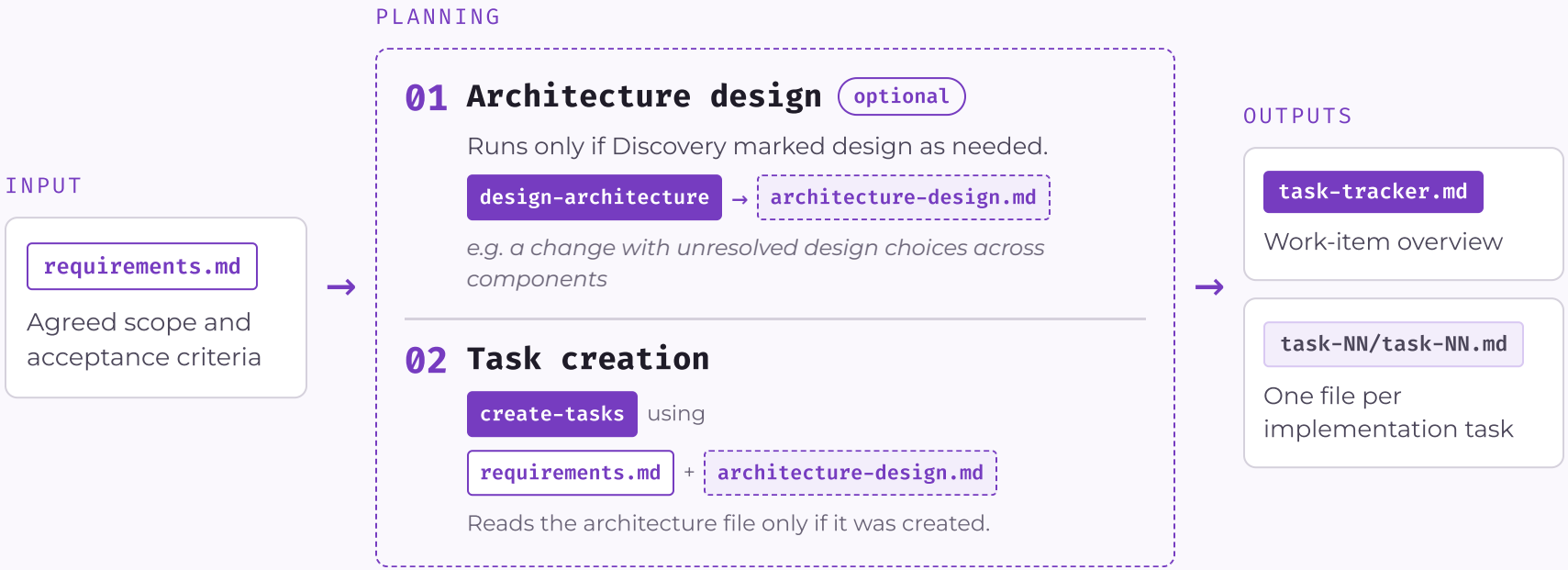
Build and verify.

Discovery turns rough input into requirements



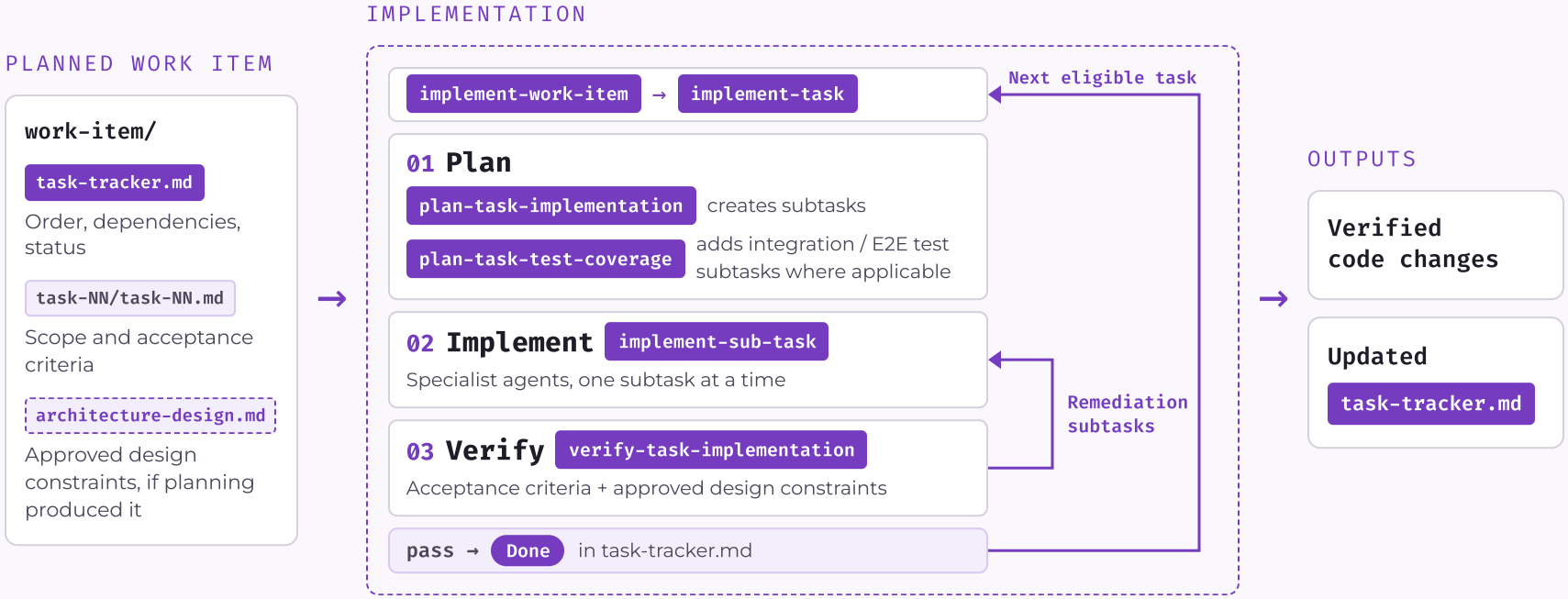
Discovery varies by work type. Planning receives the same contract.

Planning turns requirements into tasks



Architecture design runs only when needed. Task creation always starts from requirements.md.

Implementation works task by task



Actionable findings become subtasks. A task is marked done only after verification passes.

Put each instruction in the right place

Rules `project constraints`	**How the project should be built** Conventions, architecture rules, testing expectations, framework patterns
Hooks `execution checks`	**Enforce checks when Claude acts** PreToolUse: block a tool call before execution and return corrective feedback
Agents `responsibilities / roles`	**Who should do the work** Frontend, backend, QA, DevOps
Skills `repeatable workflow units`	**How the work moves forward** Investigate issue, design architecture, create tasks, implement task, verify
Artifacts `stage contracts`	**What carries decisions forward** requirements.md, architecture docs, task definitions, implementation plans

How I organize my configuration

PERSONAL / GLOBAL

Personal defaults

- General CLAUDE.md guidance
- Agents: frontend, backend, DevOps, QA
- Reusable rules: TypeScript, Jest, Cypress
- Reusable skills: PR workflows, feature grooming

PROJECT

Repository requirements

- Project CLAUDE.md guidance
- Repository-specific rules
- Project hooks: require PNPM; call Nx directly
- Project skills and commands

APP / LIBRARY

Local guidance

- Local architecture guidance
- App-specific conventions
- Library-specific constraints

My reusable toolkit? Keep it personal.
Required by the codebase? Keep it with the code.

Scope determines where guidance belongs.
Loading and precedence vary by primitive.

Not every automation needs an agent

Does this step require judgment, or follow a known procedure?

	REQUIRES JUDGMENT **Claude decides** Interprets context and makes the call.	KNOWN PROCEDURE **Script executes** Claude helps build it; the script or generator handles repeatable steps.
PR workflow	Claude reads the changes and writes the PR title and description.	A script runs the Git operations and creates the PR.
Nx setup	During setup, Claude helps design and build the Nx generators.	For each new app, generators apply Jest, Cypress, and internationalization configuration.

Spend review attention where mistakes matter

Review high-risk code closely

Business logic, security, infrastructure, data access, and integrations.

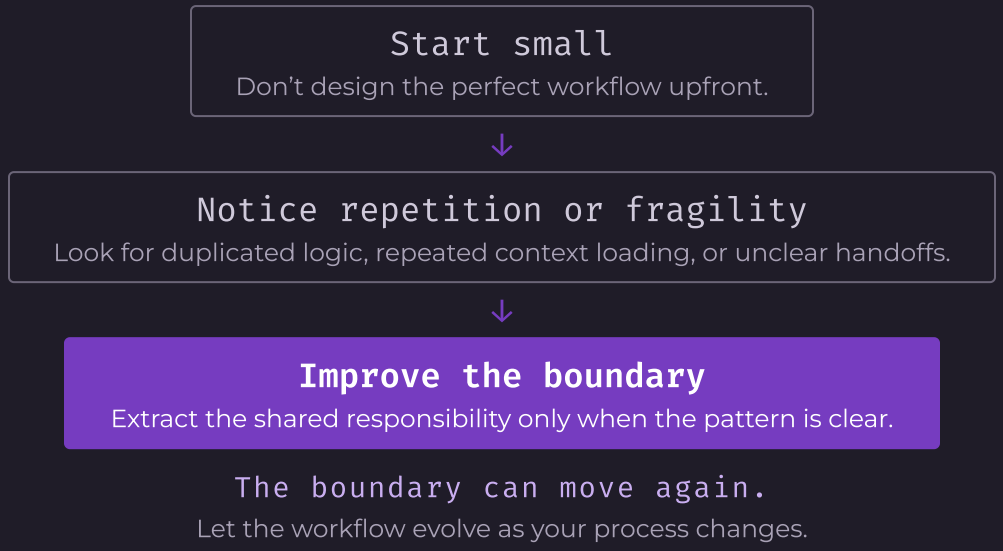
Review AI-generated skills and rules carefully

Check that instructions match your intent. Remove repetition and unnecessary process: these instructions guide future runs.

Match verification to risk

For lower-risk changes, focus on tests and observed behavior. Assess risk by impact, not file type.

Fix the painful boundary



Don't design the perfect AI workflow upfront.
Fix what becomes painful enough to deserve architecture.



Sergii Luchko

Full-Stack JavaScript Engineer
Tech Lead · Software Architect

Thanks — let's stay in touch.

WEBSITE

sergiiluchko.com

LINKEDIN

linkedin.com/in/sergiiluchko