Agentic Coding × MOOS-IvP

Introducing the moos-ivp-skills plugin for MOOS-DAWG 2026

Agentic coding tools

Given a software task, the agent inspects the repository, makes changes, runs the relevant tools, and uses the results to decide what to do next.

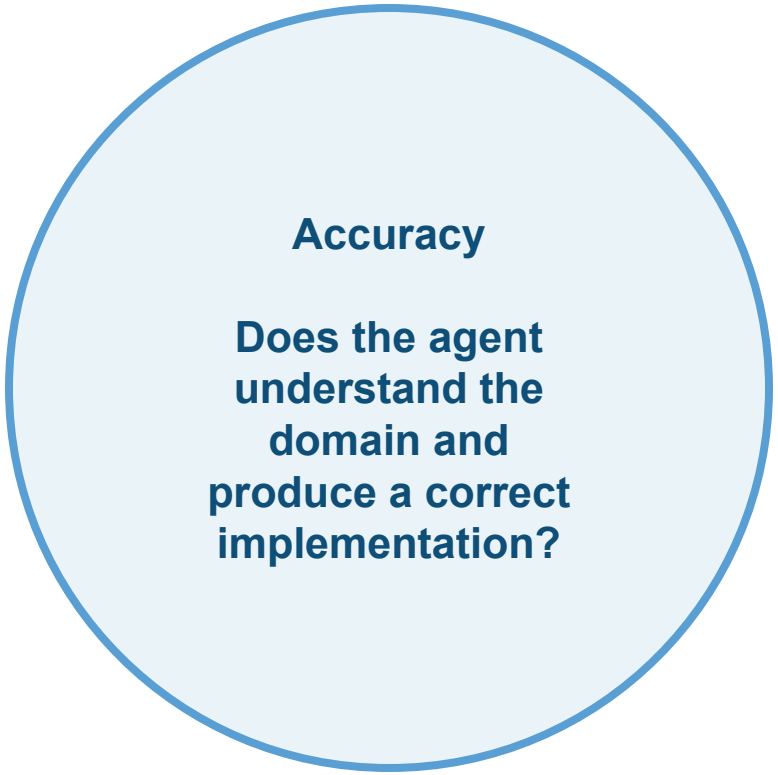
What guarantees do we need from agentic coding in MOOS-IvP?

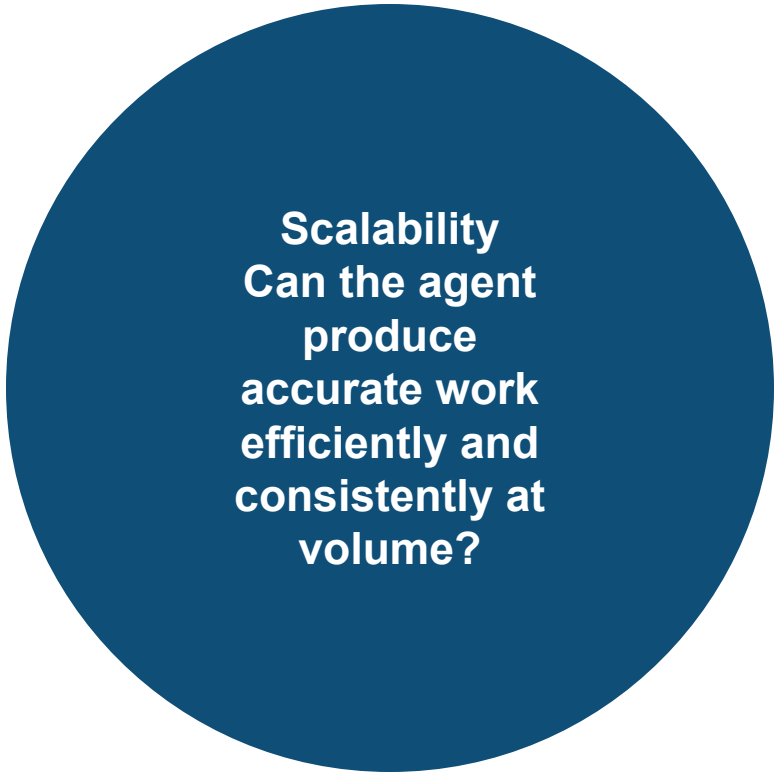
Agentic coding × MOOS-IvP

Accuracy

Scalability

Two goals


Accuracy
Does the agent understand the domain and produce a correct implementation?


Scalability
Can the agent produce accurate work efficiently and consistently at volume?

What is a skill?

Purpose

- Specialized guidance for a recurring class of work.

Selection

- Chosen automatically from the task or invoked explicitly by name.

Examples

- Build an app. Assemble a mission. Analyze a log.

How a skill is packaged

Each skill is a folder built around SKILL.md.

skill-name/

SKILL.md	when to use the skill and how to do the work
references/	detailed context and examples
scripts/	repeatable checks and operations
assets/	templates and baselines

Long-form article

Read the companion article →

Open the MOOS-DAWG talk page →

App Builder: The Canonical Example

app-builder

What it owns

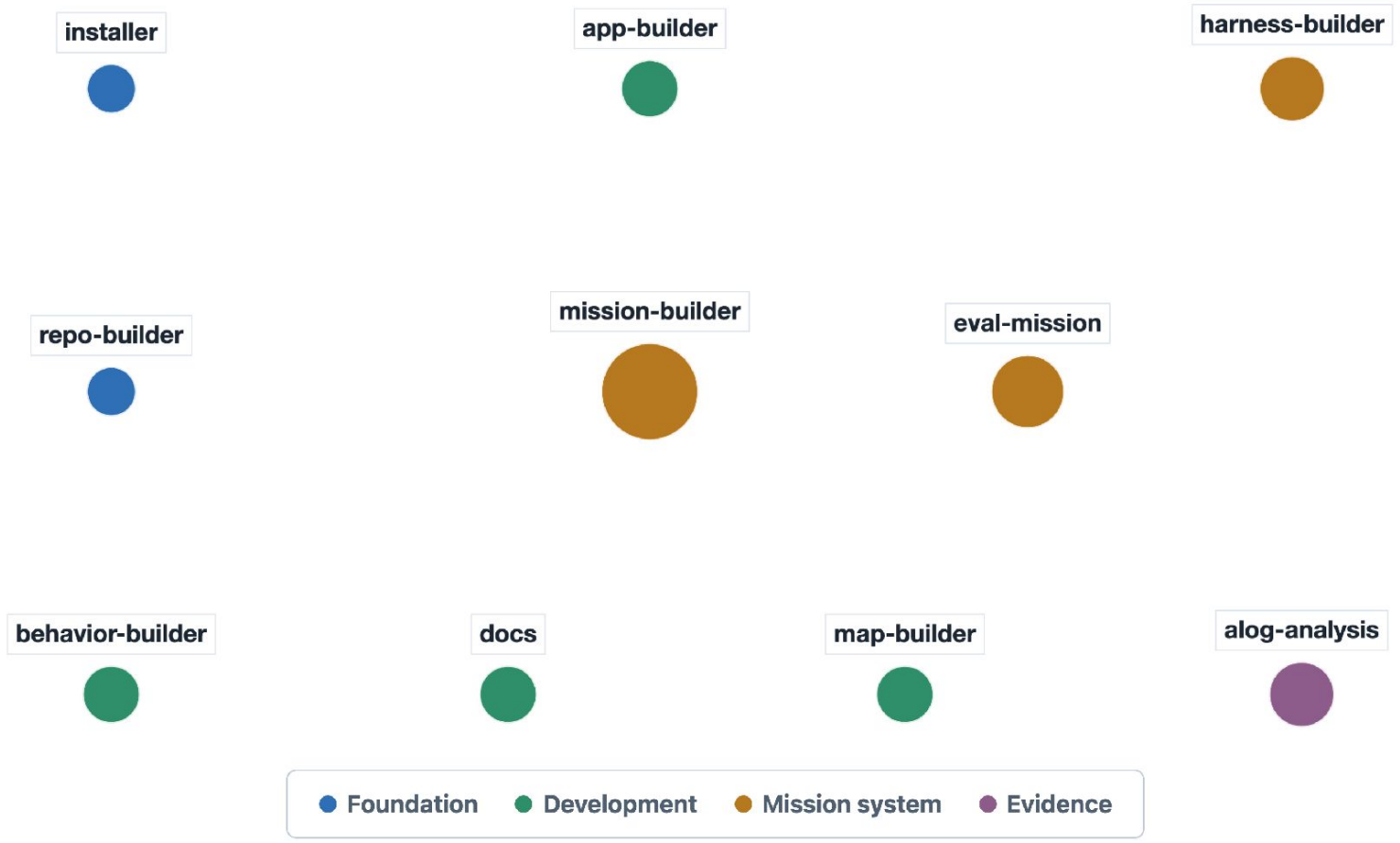
- Builds custom MOOS apps
- Project CMake wiring, AppCasting, accurate documentation, usable ProcessConfig example.
- Uses Mission Builder and Alog Analysis to validate in live launch

Inside the skill package

- **references/** build and CMake wiring upstream examples and app patterns
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

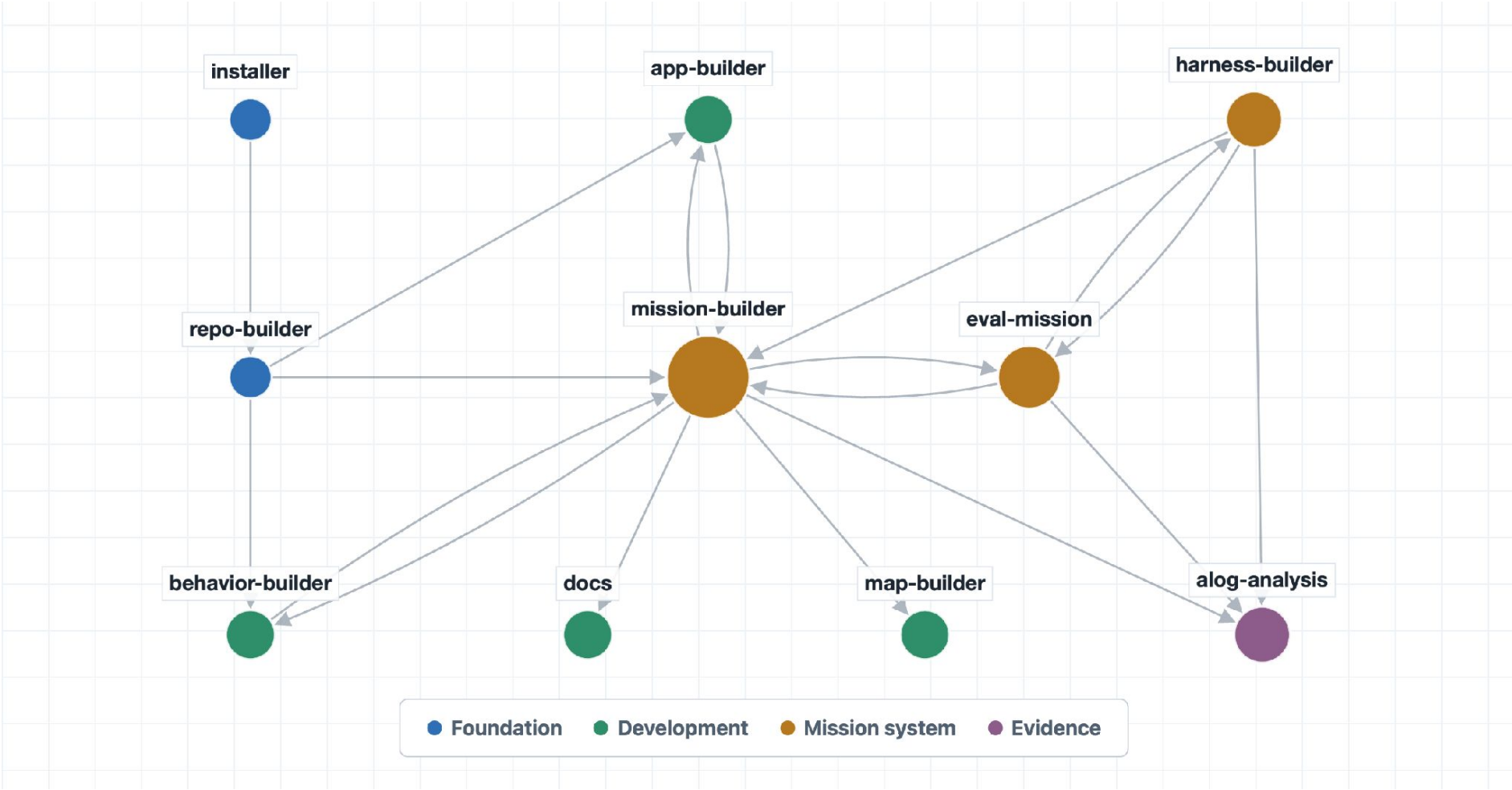
Ten skills so far

The plugin currently contains ten skills, with more expected as the project matures. Each owns a distinct part of the developer experience. No calls are shown yet.



Direct handoffs

An arrow marks a deliberate handoff of work or evidence. Mission Builder can consult Docs, use App or Behavior Builder when new software is needed, and pass runtime evidence to ALog Analysis.



Ten skills, one system

Each skill owns one part of the developer experience—and can call adjacent expertise when needed.

What's ahead

Skills

- What does each specialized workflow **own**?

System design

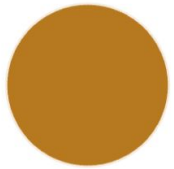
- How do boundaries, layers, handoffs, and **local customization** keep the system coherent?

Validation + practice

- How are skills tested with fresh agents—and improved through **real projects**?

Mission Builder

mission-builder



What it owns

- Builds complete MOOS-IvP missions that people can run and modify
- Sets up vehicle(s), shoreside, behaviors, networking, and visualization
- Checks both the generated files and the live mission

Inside the skill package

- **references/** bundled single- and two-vehicle baselines mission style and layered validation
- **scripts/** networking, port, and structural checks
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

MOOS-IvP Docs

What it owns

- Documentation-backed answers about MOOS-IvP apps, behaviors, and architecture.
- Prefers oceanai documentation first then local ivp/src source by default
- Can flag differences between documentation and local implementation

Inside the skill package

- **references/** manual selection and source fallback
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

App Builder

app-builder

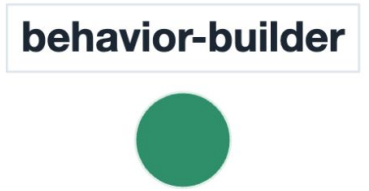
What it owns

- Builds custom MOOS apps
- Project CMake wiring, AppCasting, accurate documentation, usable ProcessConfig example.
- Uses Mission Builder and Alog Analysis to validate in live launch

Inside the skill package

- **references/** build and CMake wiring upstream examples and app patterns
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Behavior Builder



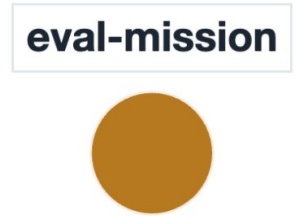
What it owns

- Builds custom IvP behaviors
- Prefers the simplest behavior form that matches the effect, from posting-only logic through coupled objective functions
- Verifies the full path: build, export, and live pHelmIvP loading

Inside the skill package

- **references/** build, lifecycle, and examples IvP-function selection patterns
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Eval Mission Builder



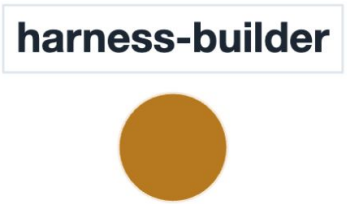
What it owns

- Adds single-run functionality to an ordinary mission while preserving manual operation.
- Runs headlessly and writes a parseable results.txt.
- pAutoPoke, pMissionEval, uMayFinish

Inside the skill package

- **references/** grading, style, automation, validation
- **scripts/** static and isolated live checks
- **assets/** project-scoped teardown helper
- **agents/** client presentation metadata

Harness Builder



What it owns

- Runs a self-evaluating stem mission across a documented matrix of named cases
- Rolling parallel execution
- –case –jobs –port_base –port_stride –keep_workdirs –gui default optionality per harness

Inside the skill package

- **references/** case matrices, patches, ports, parallelism
- **scripts/** harness contract and static checks
- **assets/** project-scoped teardown helper
- **agents/** client presentation metadata

ALog Analysis

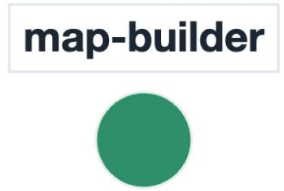
What it owns

- Reconstructs missions and investigates incidents directly from original MOOS .alog files.
- Decides variables worth inspecting, uses aloggrep, aloghelm, alogscan
- Produces timestamped evidence to supplement broader conclusions

Inside the skill package

- **references/** tool selection and evidence workflow
- **scripts/** compact variable discovery
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Map Builder



What it owns

- Creates verified MOOS-IvP TIFF background-map bundles through a visual GUI or coordinate-driven CLI
- Preserves bounds, imagery source, zoom, mission origin, output paths, and verification evidence
- Uses one public implementation for both interfaces: moos-map on PyPI.
- Inspired by Conlan Cesar's AnaxiMap and Raymond Turrisi's non-public prototype

Inside the skill package

- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Repo Builder

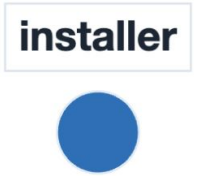
What it owns

- Creates an independent, user-owned extension repository from the official moos-ivp-extend template
- Wires the local dependency, build, executable path, behavior-library path, project identity, and fresh Git history
- Consolidates earlier organizational drift around one universal template in the moos-ivp GitHub organization

Inside the skill package

- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Installer



What it owns

- Locates or installs the upstream MOOS-IvP checkout and follows its platform-specific setup instructions
- Creates a small environment file that exposes the core binaries, scripts, and checkout root
- Expected to be the least used skill, but still addresses a common friction point

Inside the skill package

- **scripts/** installation validator
- **assets/** shared plugin artwork
- **agents/** client presentation metadata

Customizing a skill

Bundled skills suggest useful defaults and can follow references or style requested in a prompt. When a repository needs persistent rules that differ from those defaults, it can provide a project-local skill without forking the complete plugin.

1 Use the bundled defaults or name a reference and style

2 Add a same-name local skill for persistent project rules

3 The bundled workflow stays available by its qualified name

Sandboxed agent evaluations

Test the guidance

- A main agent creates a clean sandbox and gives a fresh subagent only the skill and a realistic user task.
- The task describes the goal—not a checklist that coaches the agent through the workflow.
- The report includes the artifact, supporting evidence, and the agent's stated approach and key decisions.

Revise from evidence

- The grader asks whether the result works and whether the skill enabled the agent to reach it independently.
- The author decides whether a failure came from execution or from a reasonable but wrong reading of the guidance.
- If the guidance is at fault, revise it and rerun the task with a fresh agent.

Skills in practice

Developer tooling

- CI/CD pipeline
- MOOS-IvP VS Code extension
- Headless missions-auto debugging

Research + field work

- Master's thesis harnesses
- **PEARL development**
- **KONGSBERG Coastal Monitoring**

Education

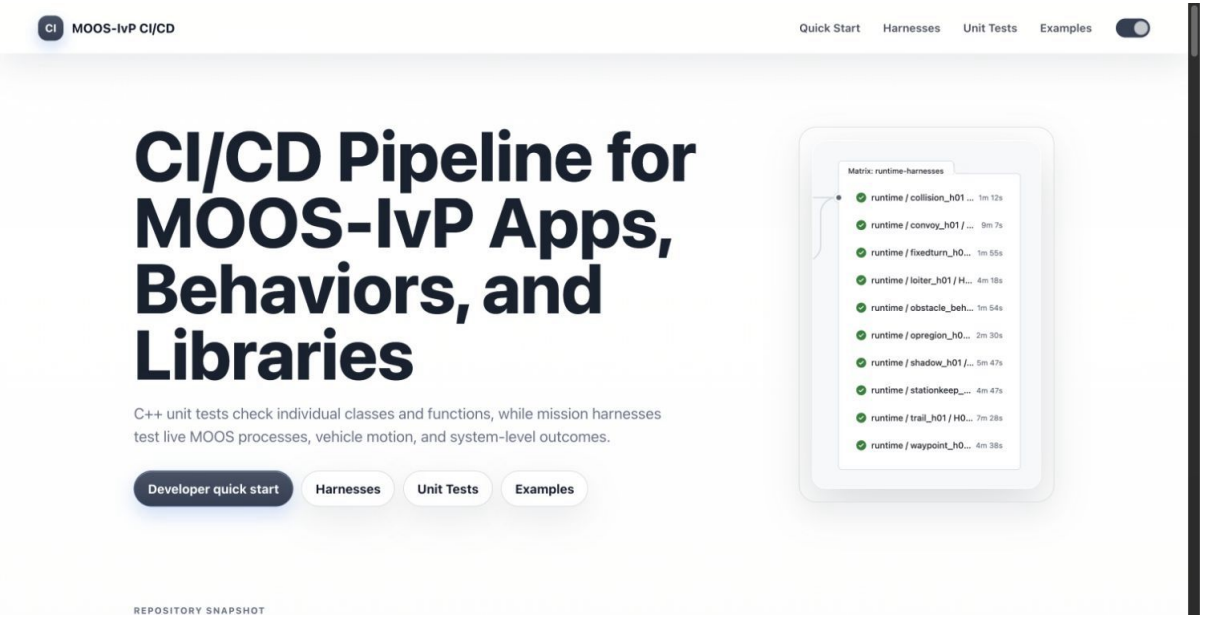
- MOOS-IvP NotebookLM TA
- Greece minicourse

MOOS-IvP CI/CD

Local and automated regression testing for MOOS-IvP applications, behaviors, and missions. Focused CTests are paired with complete headless mission harnesses.

[GitHub repository]

[Project site]



MOOS-IvP VS Code

The editor extension goes beyond syntax highlighting with hover documentation, folding, formatting, definitions, and conservative diagnostics for mission, behavior, and patch files.

[GitHub repository]

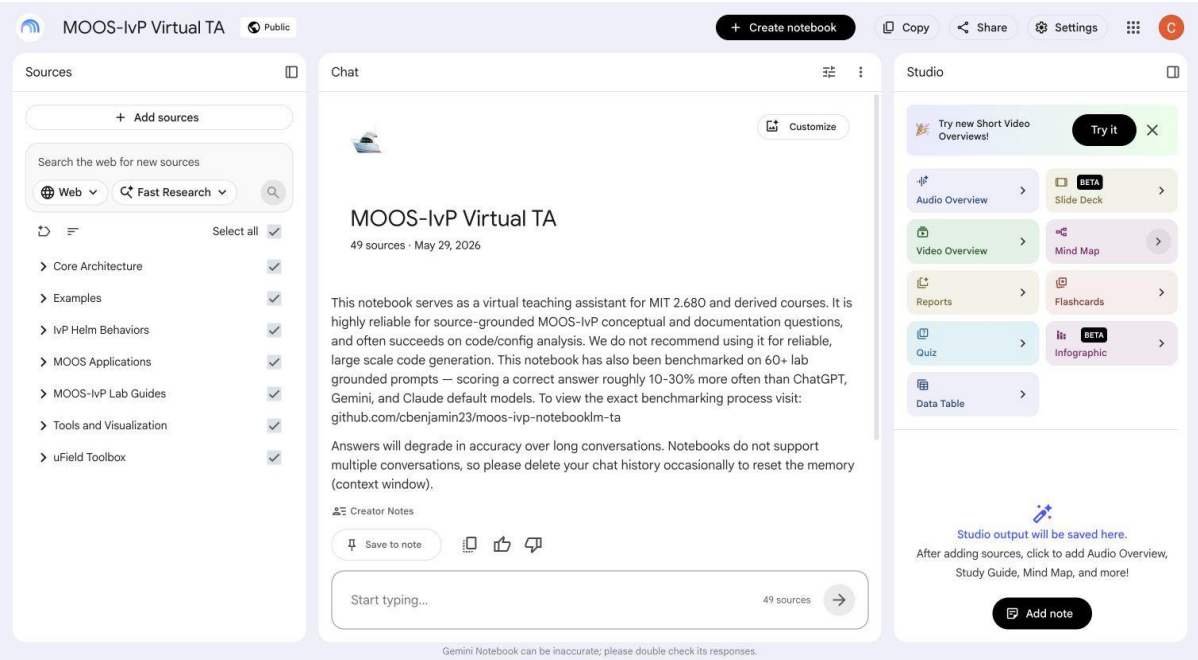
```
ProcessConfig = pHelmIvP
{
  AppTick   = 4 // Quick Fix should preserve this comment
  CommsTick = 4

  verbose   = true
  behaviors = meta_vehicle.bhv
}

ProcessConfig = pMarineViewer
{
  AppTick      = 4
  CommsTick    = 4
  tiff_file    = viewer_back.tif

  // Full-line comments keep their text, but indentation is normalized.
  left_context = x=0,y=0,scale=1.0
}
```

MOOS-IvP-TA

A documentation-grounded NotebookLM teaching assistant for MIT 2.680 labs, backed by 49 curated sources and benchmarking checked against MOOS-IvP source.

[Open the notebook]

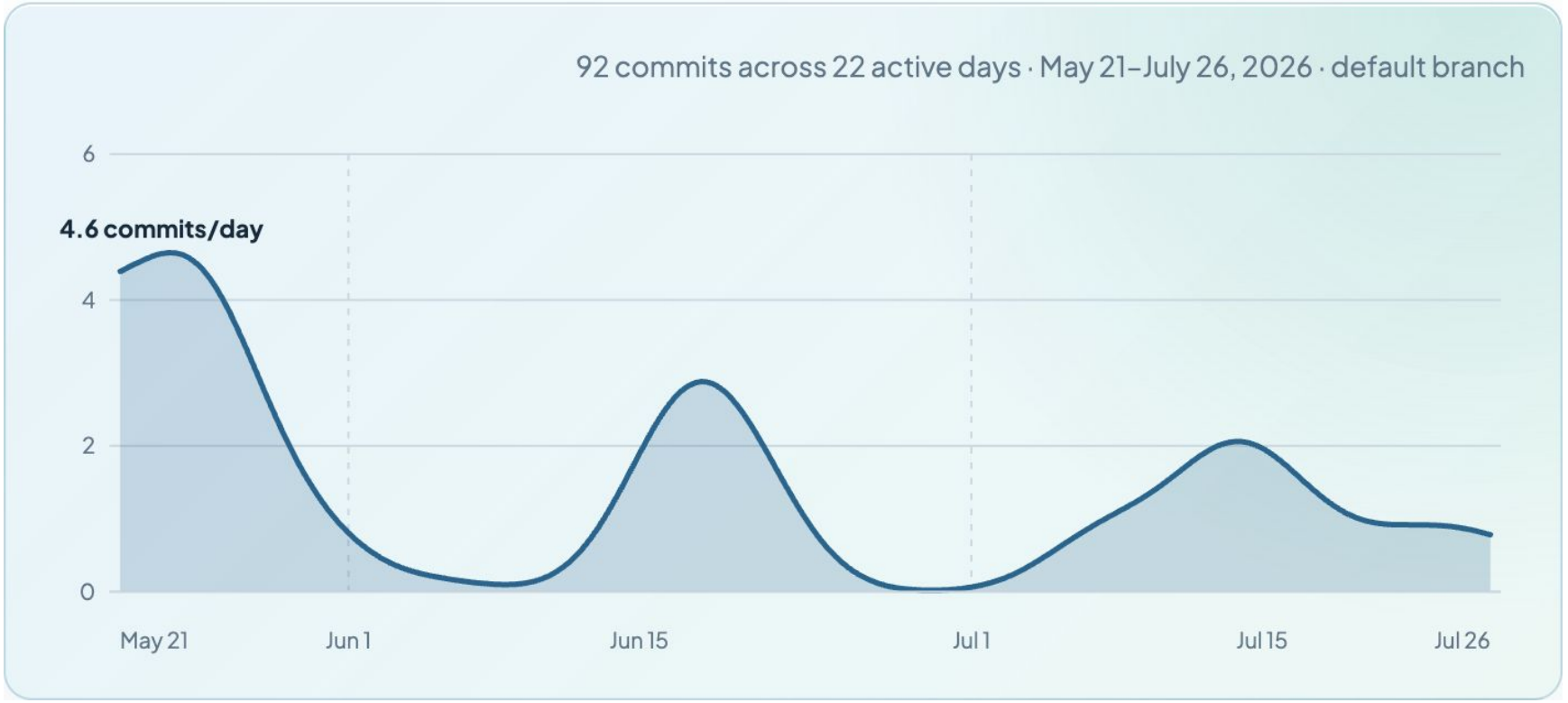
[GitHub repository]



Commit activity

92 commits across 22 active days.

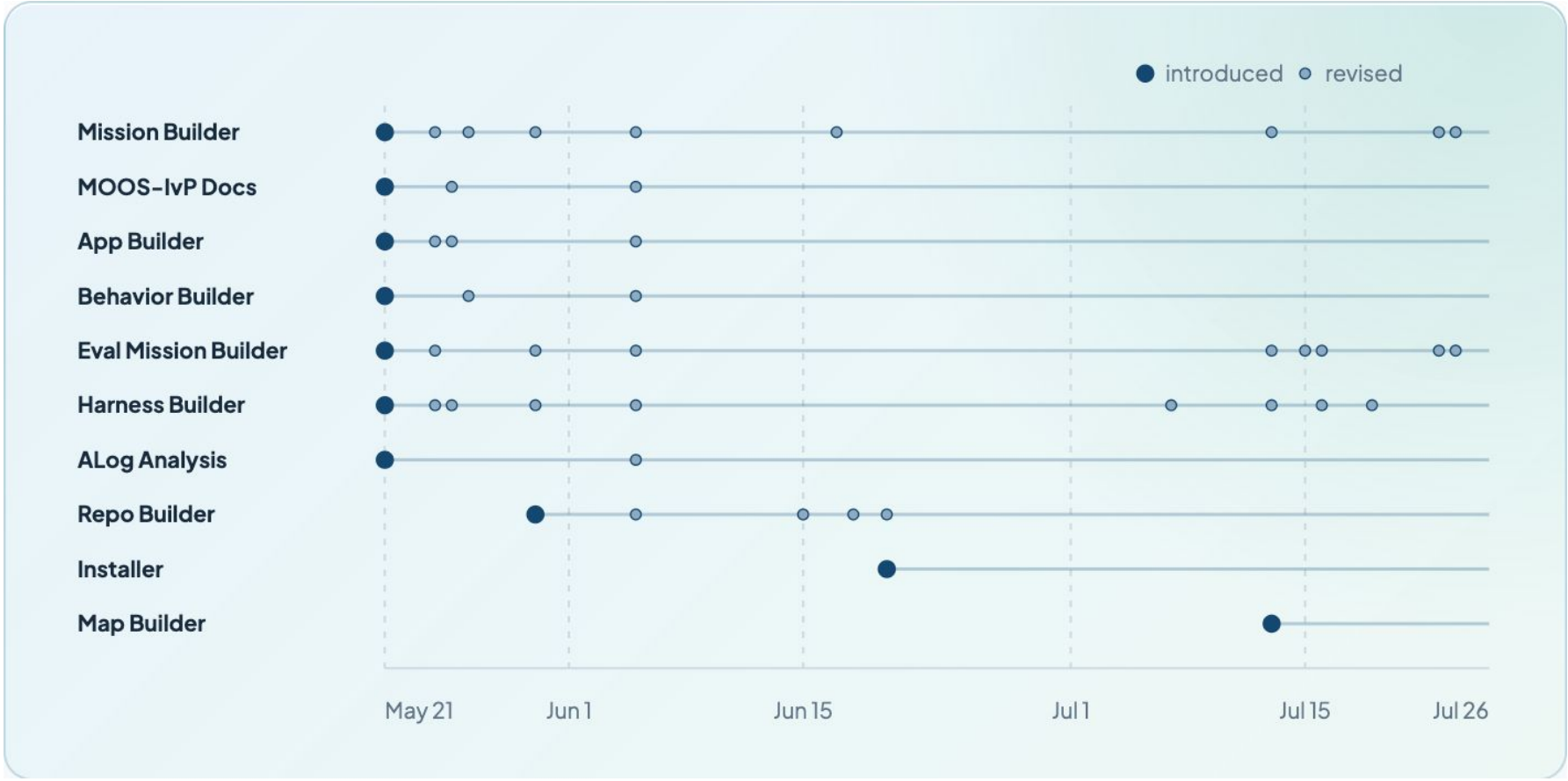
The wave shows the concentrated periods in which the skills, supporting material, and validation workflows were assembled and revised.



Skill evolution

Large markers show when a skill entered the plugin. Smaller markers show later revisions to its guidance or supporting material.

Branding-only changes are excluded.



Takeaways

- Accuracy + Scalability
- Customizable
- Validation MOAT
- Encourage Use

Questions?

MOOS-IvP-Skills

github.com/cbenjamin23/moos-ivp-skills

MOOS-DAWG 2026 • Talk 19