InfoQ Certified AI-Assisted Engineering Program Syllabus

Building production-grade software with AI coding agents

Week-by-week syllabus

Week 1: Comprehending the Codebase, Onboarding the Agent

- **Focus:** What developers actually do first: understand an unfamiliar brownfield codebase before changing it. Using agents for codebase archaeology, structure before depth, and capturing that understanding as durable context files (the context file as the artifact of comprehension, not a starting input). In parallel, onboarding the agent the way a team onboards any new member: least-privilege permissions, sandboxing, secrets hygiene, and prompt-injection risk, because an unvetted codebase is itself an attack surface. Participants also record a measurement baseline: one small task completed without a harness, with time, rework, and predicted-versus-actual effort logged.
- **Sample discussion task:** Given the capstone repository, use an agent to produce a structural map and a first context file, validated against the code rather than the stale README. Locate the seeded prompt-injection payload the agent encountered during comprehension, and configure a least-privilege permission profile that would have contained it.
- **Weekly homework:** Refine the context file against one deeper subsystem. Log the baseline task measurements.

Week 2: Making the Change Well

- **Focus:** Turning a thin ticket into an adequate requirement before generating anything: problem, intent, and acceptance criteria an agent and a reviewer can verify against. Working the way disciplined engineers work in legacy code: characterization tests before touching anything, preparatory refactoring so the change becomes easy, and consistency with the idioms of the codebase at hand. Building the sensor suite that makes all of this checkable: lint rules with agent-optimized error messages, architecture-fitness tests, and the generate-verify loop that lets the agent self-correct against them.
- **Sample discussion task:** Take a deliberately thin ticket for a capstone feature. Upgrade it into a verifiable requirement, add characterization tests around the affected area, perform one preparatory refactoring, then implement the change behind the sensor suite. Show one sensor firing and the agent self-correcting.
- **Weekly homework:** Add one sensor specific to the capstone domain and record how often it fires. Continue the weekly measurement log.

Week 3: Independent Verification Before a Human Sees It

- **Focus:** Conserving the most expensive resource in the workflow: another developer's attention. Separating code generation from code review by using an independent review harness rather than asking the generating agent to grade its own work. Pre-PR review to catch issues locally, risk-triaged findings (what blocks a merge, what merits discussion, what an agent can auto-fix), and interrogating findings conversationally instead of auto-accepting them. With an independent review gate now in place, the agent's permissions widen: trust granted in proportion to verification available.
- **Sample discussion task:** Submit the same capstone change to two reviews: one by the generating agent, one by an independent review harness with the full requirement as context. Compare findings, triage them by risk, challenge one finding through question-and-answer, and decide what genuinely needs a human before merge.
- **Weekly homework:** Establish a pre-PR review step in your personal workflow and log how many findings still reach human review. Continue the measurement log.

Week 4: Integrating: CI as the Team's Harness

- **Focus:** Scaling verification from one developer's loop to the change lifecycle the whole team depends on. Distributing checks by cost: fast sensors close to generation, expensive ones (mutation runs, deep review, cross-repository impact) in CI, keeping quality left. Continuous drift and health sensors, and background cleanup agents that fight entropy in agent-heavy codebases. The hardest governance case arrives here: unattended agents running in CI with the largest blast radius, so least privilege for pipeline agents, protected branches, and provenance of agent-authored commits. Closing with deployment of the verified system.
- **Sample discussion task:** Design the capstone's pipeline: assign every sensor from Weeks 2 and 3 a place (generation time, commit, CI) and justify each placement by cost and signal. Configure one scheduled background agent under least privilege and demonstrate one unsafe action the pipeline configuration blocks.
- **Weekly homework**: Add one drift or health sensor to CI. Continue the measurement log; this is the final week of data collection.

Week 5: Propagating and Proving It

- **Focus:** How the rest of the team comes to know, in both senses. First, knowledge: codifying recurring review findings into rules and agent skills (review, judge, codify, reuse), recording decisions and rationale so the team inherits the reasoning, and packaging the harness as a template and golden path a new team could adopt. Second, evidence: participants analyze their own five weeks of logged data against their week-one predictions, confronting the perception gap that randomized trials have documented in experienced developers, then the organizational view of measurement: throughput, instability, and trust. Closes with capstone presentations.
- **Sample discussion task:** From your recurring review findings, author one rule or skill and demonstrate a later review invoking it. Then produce a short measurement write-up: how your logged results compare against your baseline and your week-one predictions, where your perception diverged from your log, and what confounds prevent treating this comparison as proof. Close with the harness template you would hand to a new team, split into mandatory, default, and local choice.
- **Weekly homework**: Finalize and submit for certification: the harnessed capstone repository, the codified rule or skill, and the measurement write-up.