AGENI Framework

Agile Generative AI Framework

A methodology for teams building products with Generative AI — where outputs are probabilistic, the model is an external dependency, and 'done' requires a new definition.

Developed by Daniel Salazar

Program Manager & AI Practitioner

Version 1.0 · 2026 · ageniframework.com · CC BY-SA 4.0

ABSTRACT

Traditional development methodologies — Agile, Scrum, PMBOK — were designed for deterministic software. Generative AI systems are fundamentally different: their outputs are probabilistic, their behavior changes with every model update, and their "code" includes prompt engineering as a first-class artifact. AGENI (Agile Generative AI Framework) is a practical methodology that extends Agile principles specifically for the GenAI era. It introduces five iterative phases, two feedback loops, decision gates, and rigor levels calibrated to project risk.

Maturity note: AGENI v1.0 is a conceptual framework derived from direct practice in AI automation and generative AI projects. It has not yet been formally validated with adoption metrics at scale. It is offered as a structured starting point for piloting and measurement — not as proven practice. The sections on limitations, when not to use AGENI, and defensible vs. indefensible claims reflect this honestly.

1. Introduction

Agile has had a good run. Two decades of sprints, retros, and user stories have genuinely made software teams faster and more responsive. For most software, that holds up — you write the function, test it, ship it, and the behavior is predictable. The input goes in, the same output comes out.

GenAI breaks that contract.

When your product's core functionality is powered by a large language model, you are not writing deterministic logic. You are orchestrating probabilistic behavior. A prompt that works perfectly today may produce different results after a model update. An evaluation that passes in development can fail in production because the distribution of real user inputs looks nothing like your test cases. The thing that shapes all of that behavior — the system prompt, the few-shot examples, the retrieval strategy — is written in plain English, which no version control system, CI (Continuous Integration) pipeline, or QA framework was ever designed to manage.

Teams that bolt Scrum onto a GenAI project without modification tend to hit the same walls: sprint reviews where nobody can agree whether the output was actually good, "done" criteria that fall apart the moment someone asks how you know, and production incidents that trace back to a model update from the provider that happened three weeks ago and nobody noticed.

AGENI came out of building these products, not theorizing about them. A common failure pattern is finding prompts with no owner — nobody can modify them with confidence because nobody knows why they work. That pattern is what led to the concept of Prompt Debt. The most disorienting moment is a silent model update from the provider that changes the product's tone without a single line of code changing — and nobody has a process for it. That gap is what the outer feedback loop addresses. AGENI keeps the parts of Agile that hold up and replaces the parts that assume your code will do the same thing twice.

> AGENI is not an academic framework. It is a practitioner's answer to a real problem: how do you ship reliable products when your core technology is inherently probabilistic?

2. Why Traditional Methodologies Fall Short

2.1 The Determinism Assumption

The core assumption of every major software methodology is so obvious it almost never gets stated: if your code is correct, it behaves the same way every time. A green build is a promise. That assumption is baked into everything — your CI pipeline, your definition of done, your acceptance criteria, your sprint review. Nobody questions it because, for traditional software, it is always true.

When you put a language model at the center of your product, four things change that no traditional methodology accounts for.

- Probabilistic outputs: the same input can produce different outputs across runs.
- External model dependency: core behavior lives in a third-party model that updates independently of your release cycle.
- Natural language as logic: prompts are instructions, not code — they are harder to test, version, and reason about.
- Emergent failure modes: the system can fail in ways that are contextual, subtle, and not caught by unit tests.

2.2 Where Teams Get Stuck

Challenge	Why traditional methods fall short
Defining 'done'	Acceptance criteria require objective pass/fail — impossible for generative outputs without an eval strategy
Sprint velocity	Prompt iteration does not map to story points; a single prompt change can take minutes or weeks
QA and testing	Unit tests cannot cover probabilistic behavior; traditional QA misses semantic failures
Technical debt	Prompt Debt accumulates silently — undocumented prompts, untested edge cases, no regression suite
Model updates	A model update from a provider can break production behavior without any code change on your side
Handoff to ops	Traditional runbooks do not cover model drift, hallucination monitoring, or prompt version rollback

3. AGENI Framework Overview

The framework has five phases and two feedback loops. Think of the first loop as the daily work: you design a prompt, test it, refine it, and do it again. That cycle runs in days because the cost of feedback is low. The second loop is slower — it kicks in when something outside the team's control changes. A provider updates the model. Users start using the product in ways nobody anticipated. The product scope expands. When that happens, the team goes back to the beginning and re-examines what 'good' looks like.

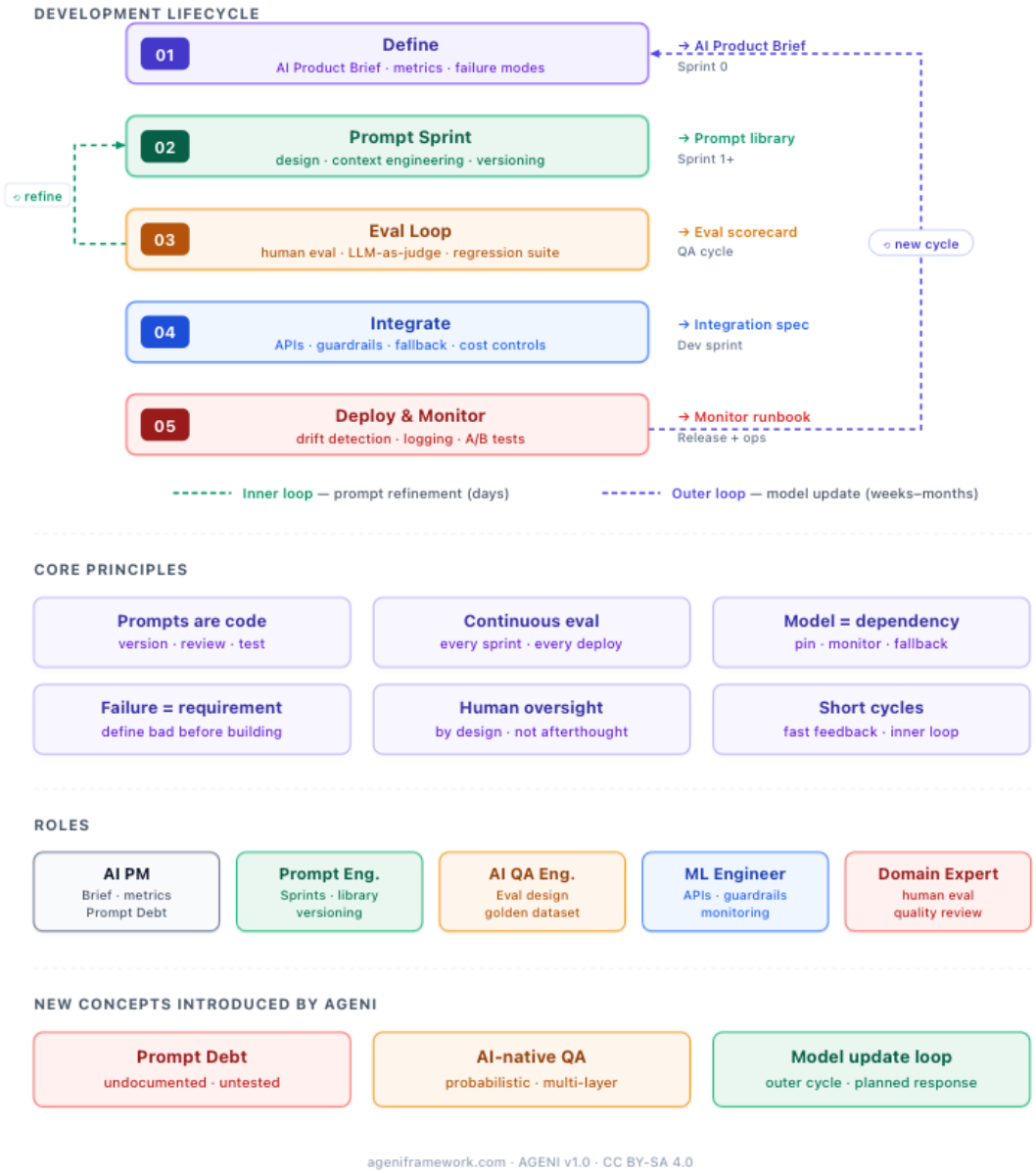

Figure 1 — Complete AGENI framework map: 5 phases, 2 feedback loops, 6 principles, 5 roles, and 3 new concepts.

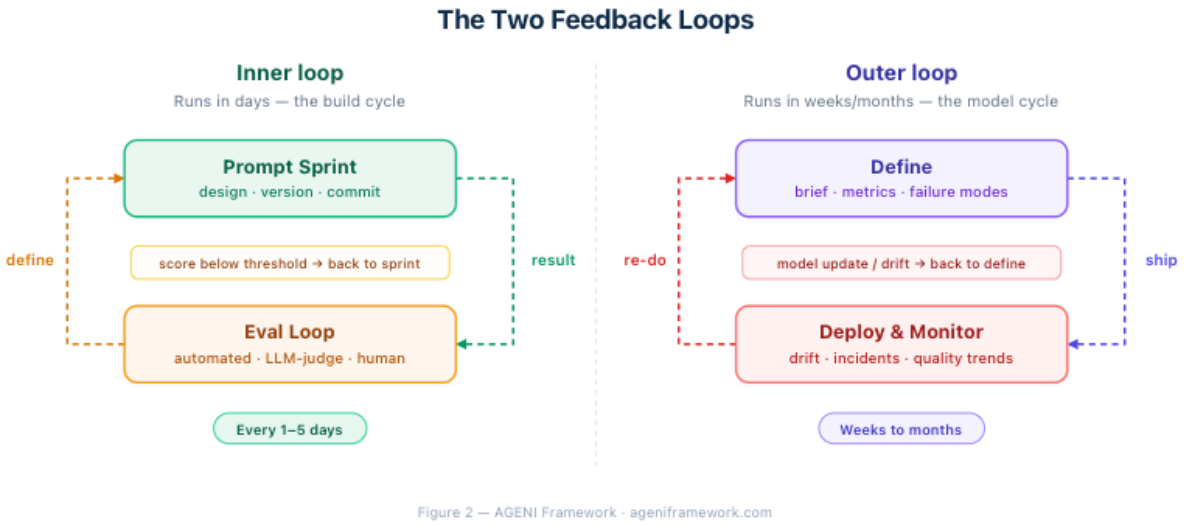

Figure 2 — The two feedback loops: inner loop (days) and outer loop (weeks–months).

Phase	Focus	Key Deliverable	Agile Analog
01 · Define	Scope, constraints, success metrics	AI Product Brief	Sprint 0 / Discovery
02 · Prompt Sprint	Rapid prompt design and iteration	Versioned prompt library	Sprint
03 · Eval Loop	Systematic output evaluation	Eval scorecard + regression suite	QA / Test cycle
04 · Integrate	Connect model to product systems	Integration spec + guardrails	Dev sprint
05 · Deploy & Monitor	Production observability and drift detection	Monitoring runbook	Release + ops

4. The Five Phases

01 Define

Most teams skip Define or treat it as a quick alignment meeting. That works fine when your product's behavior is determined by code you write. When it is determined by a model you do not control, skipping Define is where the trouble starts. The goal is not to write a perfect spec — it is to force the team to answer questions that will otherwise get answered badly, in production, under pressure.

The AI Product Brief does not need to be long. What it needs to cover:

- What model(s) are being considered, and what are their capability boundaries?
- What does 'good output' look like? How will we measure it?
- What are the acceptable failure modes, and which are unacceptable?
- Who is the human-in-the-loop, if any? Where does the model have autonomy?
- What happens when the model is updated by the provider?
- What data, context, or knowledge does the model need to do its job?

> Define is complete when the team can write a failing eval test. If you cannot describe what bad output looks like, you are not ready to build.

02 Prompt Sprint

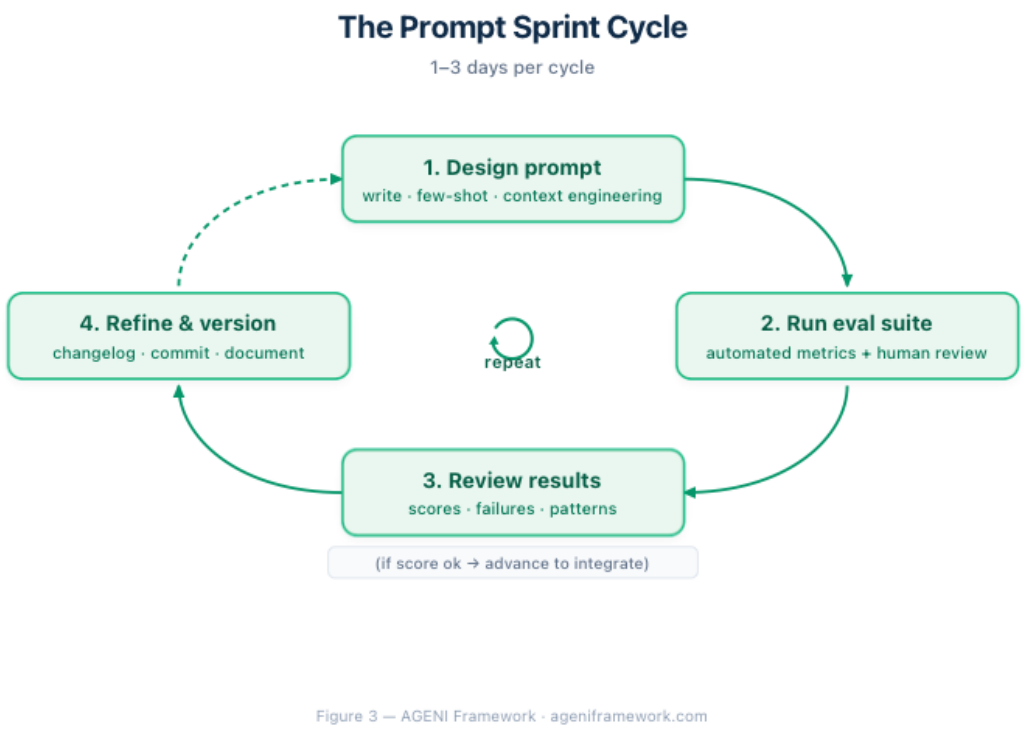

Figure 3 — The Prompt Sprint cycle: design, run, review, refine — repeated every 1–3 days.

The Prompt Sprint is the core development cycle of AGENI. It is shorter than a traditional sprint — one to three days — because the feedback loop is faster. You write a prompt, run it against your eval suite, see what breaks, adjust, and repeat. The deliverable is not a feature. It is a prompt or prompt chain that has been tested, versioned, and documented well enough that someone else on the team could understand why it works and change it without breaking something. Three of the four steps below require human judgment; running the eval suite is the only automated step.

What happens in a Prompt Sprint:

- Prompt design: write or revise system prompts, user-turn templates, and few-shot examples — *human, Prompt Engineer*
- Context engineering: define what information the model receives (RAG, tool outputs, memory) — *human, Prompt Engineer*
- Lightweight eval run: execute the prompt against your reference dataset and review the scores — *automated execution, human interpretation*
- Parameter exploration: test temperature, model variants, output format constraints — *human, Prompt Engineer*
- Edge case mapping: identify inputs likely to produce problematic outputs — *human, QA Engineer*
- Prompt versioning: commit prompts to version control with a changelog — *human, Prompt Engineer*

Note: The eval run inside a Prompt Sprint uses the same tools as Phase 03 — Eval Loop, but with a different purpose. Here it guides the work. In Phase 03, it validates it.

Los prompts sin documentar ni versionar se acumulan como Prompt Debt — ver Sección 6.

03 Eval Loop

The Eval Loop is where AGENI is most different from anything you have done before. Traditional QA is binary — the test passes or it fails. GenAI evaluation is not. An output can be partially correct, contextually appropriate but tonally off, factually accurate but formatted wrong, or perfectly reasonable for 80% of inputs and completely wrong for the other 20%. The Eval Loop is the practice of measuring all of that, continuously, so quality degradation shows up before users report it.

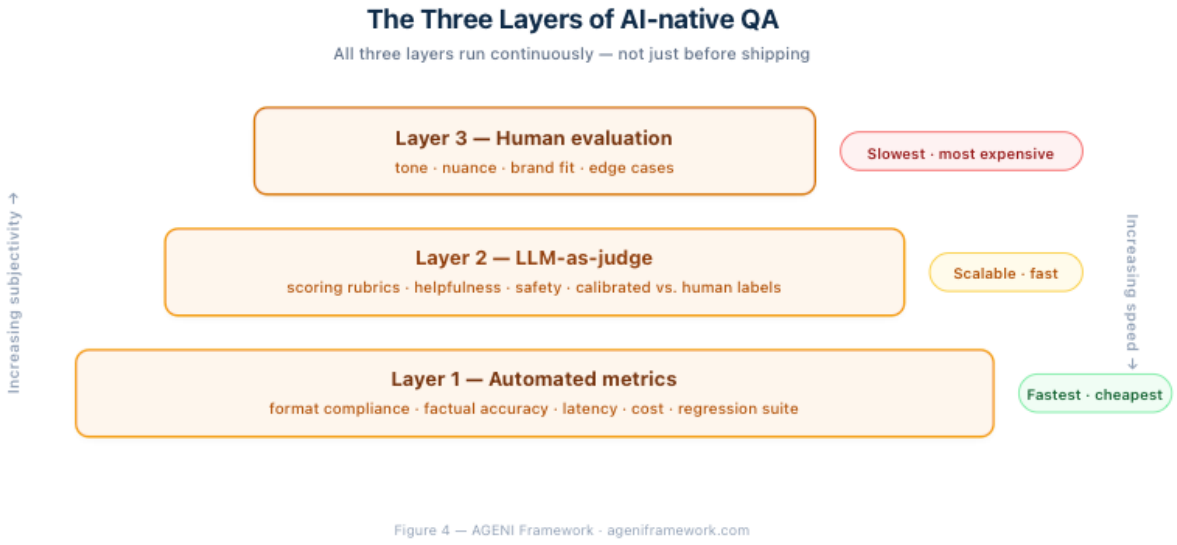

Figure 4 — The three layers of AI-native QA: automated metrics, LLM-as-judge, and human evaluation.

Layer 1 — Automated metrics

- Factual accuracy against ground truth (where available)
- Format compliance — does the output match the required structure?
- Response time and cost benchmarks
- Regression tests against a curated golden dataset

Layer 2 — LLM-as-judge

Use a separate model call to evaluate the output of the primary model. Define scoring rubrics (helpfulness: 1–5, safety: pass/fail) and run them systematically across the evaluation dataset. Calibrate LLM-as-judge scores against human labels before relying on them for gate decisions.

Layer 3 — Human evaluation

Domain experts, end users, or the product team periodically review samples of real outputs. This is where subjective quality — tone, nuance, brand alignment — gets assessed. Not every output can be machine-evaluated. In high-risk systems (A3), human evaluation is required, not optional.

> The Eval Loop never closes. In AGENI, 'done' means evaluated, documented, and monitored — never tested once and shipped.

04 Integrate

Integration is where a lot of GenAI projects fall apart. The prompt that worked perfectly in the playground stops working once it hits the real product — because the context window is now full of irrelevant data, because the output parser was written for a slightly different format, because there is no fallback when the model times out. Integration in AGENI is the phase where you close the gap between 'it works in isolation' and 'it works in production.'

Before shipping, the team needs to confirm:

- API and SDK integration with error handling for model unavailability
- Context window management: what fits, what gets truncated, what gets retrieved
- Skills and system prompts: the production prompt (Prompt v1) is the skill — the persistent instruction set that defines the model's role, tone, constraints, and expected output format. A skill does not provide current or official information on its own; it specializes behavior. To deliver current data, pair the skill with RAG (retrieval from updated documents or databases), live API calls, or context passed directly in each request.
- Output parsing and validation: never trust raw model output in production
- Guardrails: input filtering, output filtering, content policy enforcement
- Fallback logic: what happens when the model fails, times out, or produces unusable output?
- Cost controls: token budgets, rate limiting, caching strategies
- Human escalation paths: when should a human review or override the model?

05 Deploy & Monitor

Shipping is not the finish line. If anything, it is where the interesting problems start. Real users send inputs you never thought of. The model provider updates something quietly and your product starts behaving differently. Output quality drifts in ways that do not show up as errors — no 500, no crash, just subtly worse answers that users notice before your monitoring does.

What to watch in production:

- Output quality drift: are outputs getting worse over time? After model updates?
- Usage pattern analysis: which inputs are users actually sending?
- Cost and latency trends: is the system operating within budget?
- Failure mode tracking: what types of failures are occurring? How often?
- User feedback signals: explicit ratings, implicit signals (retry rate, abandonment)

The outer feedback loop

When monitoring reveals significant drift, new failure modes, or a model update from the provider, the outer loop triggers: go back to Define, review the AI Product Brief, and start a new set of Prompt Sprints. This is expected, planned behavior in AGENI — not a crisis.

5. Core Principles

Six ideas show up throughout the framework. They are worth stating upfront because they are the places where AGENI asks teams to work differently than they are used to.

- Prompts are code. Not metaphorically — in a GenAI product, the system prompt is production code. It needs a version history, a review process, and a way to test that changing it does not break something else. A prompt with no documentation is Prompt Debt waiting to surface.
- Evaluation is continuous. There is no 'done with QA' in a probabilistic system. You run evals during development, after deployment, and every time the underlying model changes. The moment you stop measuring is the moment quality starts drifting invisibly.
- The model is someone else's code. OpenAI, Anthropic, Google — whoever provides the model can update it without asking. That update can change your product's behavior in production without you touching a single file. Treat it the way you would treat any dependency you do not control: pin versions where possible, monitor for changes, and have a plan for when something breaks.
- Failure modes are requirements. Before writing the first prompt, the team should be able to answer: what does a bad output look like? What is unacceptable under any circumstances? If you cannot answer that, you cannot write a meaningful eval — and without a meaningful eval, you are guessing.
- Human oversight is a design decision. At some point, someone on the team needs to decide: where does a human review the output before it reaches the user? Where does the model act autonomously? That decision should happen in the design phase, not after the first production incident.
- Keep cycles short. One of the advantages of building with GenAI is that the feedback loop can be fast — running a prompt against an eval suite takes minutes, not weeks. Use that. Design the work in short cycles so the team learns quickly and does not spend three weeks building in the wrong direction.

6. New Concepts Introduced by AGENI

Three terms come up throughout the framework that do not have direct equivalents in Scrum or PMBOK. They are worth defining clearly because if people use them loosely, the framework stops being useful.

Prompt Debt — the GenAI equivalent of technical debt

Accumulates when prompts are undocumented or not versioned, copied from examples without understanding why they work, modified in production without a corresponding eval update, or optimized for the demo rather than the full distribution of real user inputs. Like technical debt, Prompt Debt is invisible until it causes a production incident. It manifests as degraded output quality after a model update, inconsistent behavior across similar inputs, or inability to explain why the product behaves differently than expected. AGENI teams track Prompt Debt explicitly as a backlog item and allocate sprint capacity to reduce it.

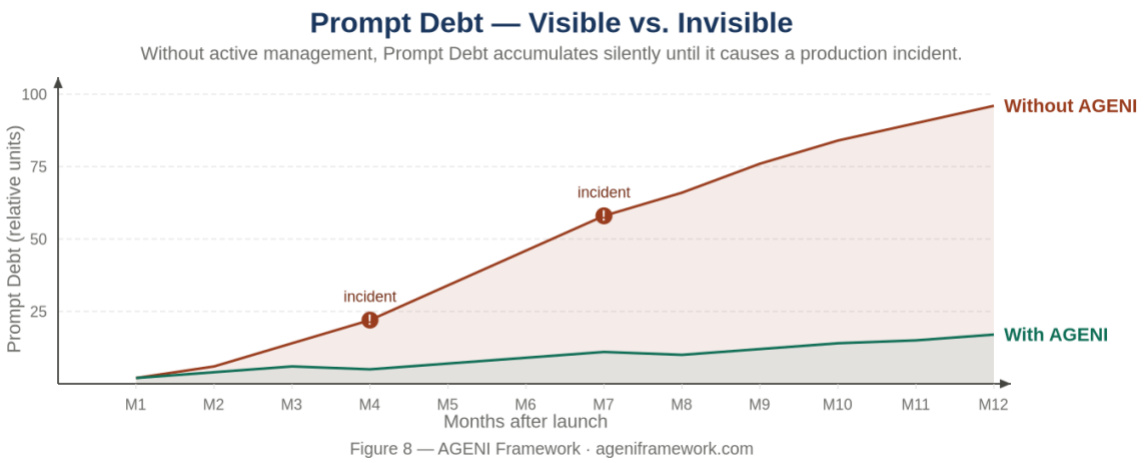

Figure 8 — Prompt Debt accumulation: without active management, debt grows until a production incident.

How does AGENI prevent Prompt Debt from growing unchecked? It does not prevent it — it makes it visible and manageable. Just as Scrum does not eliminate technical debt but puts it on the backlog so the team can manage it deliberately, AGENI makes Prompt Debt visible in the Eval Loop, registers it as a backlog item, and allocates sprint capacity to address it. Without a framework, Prompt Debt is invisible until it causes a production incident.

AI-native QA — quality assurance redesigned for probabilistic systems

AI-native QA is the practice of evaluating generative AI outputs systematically, accepting that there is no binary pass/fail. It operates across three layers: automated metrics (format compliance, factual accuracy, cost and latency benchmarks, regression tests on a golden dataset), LLM-as-judge (a second model evaluates the output of the primary model — for example, scoring whether a response was helpful, accurate, or appropriate on a 1–5 scale), and human evaluation (domain experts review samples for subjective quality). AI-native QA is not a phase; it is a continuous practice that runs in every Prompt Sprint and continues in production via Deploy & Monitor.

Model update loop — the outer feedback cycle triggered by model changes

When the LLM provider publishes a new model version, the model update loop activates: re-run the full eval suite against the new model, analyze impact on prompts and guardrails, re-prompt if quality degraded, re-validate adversarially, then decide whether to migrate, migrate with fallback, or hold the prior version. This loop runs on a cycle of weeks to months — much slower than the inner loop — but must be planned for in every AGENI project. It is the behavior that most distinguishes GenAI product operations from traditional software operations.

7. Is AGENI a Methodology, a Tool, or a Project Management Framework?

Worth answering directly.

7.1 The AI-as-tool argument

The objection goes like this: AI is a tool. Teams have always used tools — compilers, cloud services, IDEs — and those tools never required new methodologies. If you use Claude to write code or ChatGPT to draft a spec, you are right. The output of those workflows is still deterministic code or a document that a human reviews before it matters. Your existing process handles that fine.

> The critical distinction: AGENI applies when the model's output IS the product delivered to the user — not when AI is used internally to build that product. A team using Copilot to write deterministic code does not need AGENI. A team building a virtual assistant, a content generation tool, or any system where the end user interacts directly with LLM responses does. Importantly: that same product may have other parts — a UI, a database, business logic — that continue to be managed with Scrum, PMBOK, or any other methodology. AGENI complements those methodologies for the specific part where the LLM generates the user-facing output.

7.2 Implementation methodology vs. project management framework

AGENI spans both categories intentionally. The problem it addresses exists at both layers simultaneously. A team with excellent engineering practices but no PM governance for model updates will ship products that break silently. A team with excellent governance but no eval discipline will deliver degrading quality with no mechanism to detect it.

Phase	Implementation methodology	Project management
01 · Define	—	AI Product Brief, success metrics, failure mode inventory
02 · Prompt Sprint	Prompt design, context engineering, versioning	Sprint planning, backlog management
03 · Eval Loop	Eval suite design, LLM-as-judge implementation	Quality gates, acceptance criteria, release decisions
04 · Integrate	API integration, guardrails, fallback logic	Dependency management, risk mitigation
05 · Deploy & Monitor	Logging, drift detection, A/B testing	Governance, stakeholder reporting, model change management

> AGENI's formal category is a Product Development Framework — it sits at the intersection of implementation methodology and project management, because the problem it solves lives at that intersection.

8. AGENI vs. Existing Methodologies

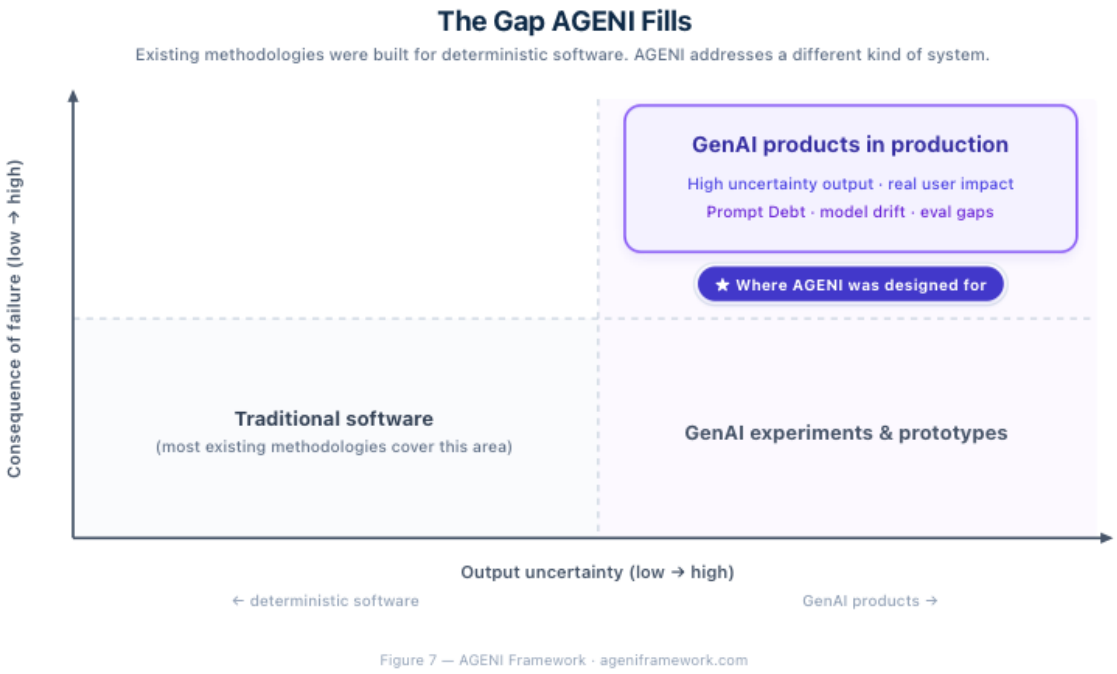

Figure 7 — The gap AGENI fills: GenAI products in production combine high uncertainty with real user impact.

AGENI is not trying to replace Scrum or PMBOK. Most teams building GenAI products already have a methodology that handles the non-AI parts of their work well. The comparison below is not a claim that AGENI outperforms established frameworks — it maps where each was designed to operate, and where the gap appears when a language model is at the core of the product. AGENI's position in these comparisons reflects design intent, not validated performance.

8.1 AGENI vs. Traditional Agile / Scrum

Concept	Traditional Agile / Scrum	AGENI
Unit of work	User story / feature	Prompt experiment / eval result
Definition of 'done'	Tests pass, feature ships	Eval score meets threshold, documented, monitored
QA approach	Unit tests, integration tests	Eval suites, LLM-as-judge, human review
Code artifact	Source code	Source code + prompt library
Technical debt	Code debt	Code debt + Prompt Debt
External dependency	Libraries, APIs (versioned)	Model providers (probabilistic, external updates)

Sprint output	Shippable increment	Evaluated prompt version or integration milestone
Production monitoring	Uptime, errors, latency	+ Output quality drift, semantic failures, model changes

Note: User stories can coexist with AGENI, particularly in the Define phase to establish user context and goals. The difference is that the measurable unit of progress is the evaluation result — not the delivered feature.

8.2 AGENI vs. PMBOK / Waterfall

PMBOK was built for projects with a clear start, middle, and end — fixed scope, defined requirements, predictable delivery. GenAI projects rarely work that way. That said, PMBOK still has useful pieces: stakeholder communication, risk frameworks (which can be adapted to cover model risk), budget planning, and formal documentation for regulated environments. AGENI and PMBOK can coexist — they just handle different concerns.

8.3 AGENI vs. MLOps / LLMOps

MLOps and LLMOps handle the infrastructure side — how to deploy, monitor, and update models technically. AGENI handles the process side — how the team decides what to build, how to evaluate it, who approves what, and how to respond when something goes wrong. They solve different problems and work well together.

8.4 Methodology comparison

Dimension	Scrum	PMBOK	AGENI
Handles uncertainty	High	Low	**High**
Probabilistic outputs	None	None	**Native**
Short cycle speed	High	Low	**High**
Enterprise scalability	Medium	High	**Medium**
GenAI native	No	No	**Yes**
Prompt management	No	No	**Yes**
Planned response to model updates	No	Partial	**Yes**

** Planned response to model updates: whether the framework has an explicit process when the model provider changes the underlying model without the team's intervention.*

9. Strengths and Limitations

To be direct: a document that only lists strengths is marketing material. Here is an honest look at both sides.

9.1 Strengths

Built for uncertainty. AGENI accepts probabilistic outputs as the norm. Every phase is designed around the assumption that model behavior will vary, drift, and change.

Prompt versioning and governance. By treating prompts as first-class artifacts, AGENI prevents the most common form of invisible technical debt in GenAI projects.

Eval-first culture. Quality gates are defined before development begins. This is the single biggest process failure in GenAI projects today.

Model change resilience. The outer feedback loop explicitly plans for provider model updates — a scenario that no existing PM methodology addresses.

Composable adoption. You do not have to adopt all of it at once. A team dealing with prompt chaos can start with just the Prompt Sprint discipline. A team with production incidents can start with Deploy & Monitor. The framework is designed to be picked up in pieces.

Cross-functional bridge. AGENI gives PMs, engineers, domain experts, and QA teams a shared vocabulary for AI product quality.

9.2 Limitations

No track record yet. This is the most important limitation and the one worth stating first. AGENI v1.0 has not been formally validated. It comes from practice — real projects, real failures — but it does not have the evidence base that Scrum or PMBOK built over twenty years. That takes time and pilots.

Learning curve. Teams trained in deterministic QA need to fundamentally rethink what 'done' means. This is a cultural change, not just a process change.

Eval infrastructure investment. Building golden datasets, LLM-as-judge pipelines, and regression suites requires real engineering effort before it pays off.

Tooling immaturity. The ecosystem for prompt versioning, eval management, and GenAI-specific monitoring is still early.

Velocity measurement is harder. Story points do not translate cleanly to prompt experiments. Teams will need to develop new proxies for progress.

Not for non-AI-core products. If GenAI is a minor feature rather than the product's core, AGENI introduces overhead without proportional benefit.

10. When NOT to Use AGENI

A framework that claims to work for every situation works for none of them.

Scenario	Better approach
AI used internally by the team (Copilot, Claude for drafting) but the product delivered to customers is deterministic — it does the same thing every time	Use your existing methodology unchanged. AI tools do not change your QA requirements.
GenAI is a minor, non-core feature	Adopt only the Eval Loop from AGENI within your existing Scrum or Kanban workflow.
Fixed scope, regulatory mandate, hard deadline	Use PMBOK for governance and embed AGENI's Define and Eval Loop phases for the AI-specific components.
Internal tooling with constrained, well-defined prompts	Simplified eval + prompt versioning is sufficient. The full framework is overkill.
Solo builder, pre-product-market-fit	Use AGENI Lite: Define + Prompt Sprint + Eval Loop only.

11. Common Objections — and Honest Answers

These are the questions that come up every time someone presents AGENI to an experienced PM or engineering team. Stated as fairly as possible, with the most honest answers available.

1. "This is just Agile with a GenAI checklist."

Partially true. AGENI deliberately preserves Agile's core — short cycles, continuous feedback, working software over documentation. What it adds is not cosmetic: the Eval Loop has no analog in Scrum, Prompt Debt is a genuinely new category of technical debt, and the outer feedback loop addresses a problem that did not exist before LLM APIs became infrastructure.

2. "AI is a tool of execution, not a methodology driver."

Correct — when AI is used internally. AGENI's scope is narrower than it appears: it applies only when the model's outputs are delivered directly to end users as the product's core value. In that scenario, the model is not a tool — it is the system.

3. "MLOps and LLMOps already solve this problem."

MLOps and LLMOps (infrastructure practices for operating AI models in production) solve the infrastructure problem: how to serve, monitor, and retrain models at scale. AGENI solves the product process problem: how a cross-functional team organizes its work. These are different layers.

4. "Without a certification body, this is just a blog post with a name."

Fair. AGENI v1.0 is the beginning of a body of knowledge. Every established framework started as a document. The measure of legitimacy is adoption and refinement over time — not the existence of a certification on day one.

5. "Is this methodology or PM framework — it seems to be both."

Both, deliberately. The problem exists at both layers simultaneously. Separating them creates the exact gap that causes most AI product failures.

6. "Story points and velocity don't work here."

Correctly identified as a limitation. AGENI proposes eval score progression, prompt experiment throughput, and Prompt Debt reduction as alternative proxies. These are less precise than story point velocity — an honest limitation at this stage.

12. Roles in an AGENI Team

No new job titles needed. The roles below map existing titles to what they own in an AGENI project — the responsibilities shift, but the org chart does not.

Role	Primary responsibility in AGENI	New skill required
AI Product Manager	Owns the AI Product Brief, defines eval success criteria, manages Prompt Debt backlog	Eval design, LLM capability awareness
Prompt Engineer / Developer	Runs Prompt Sprints, maintains prompt library, owns versioning and regression suite	Prompt design, context engineering
AI QA Engineer	Designs and runs Eval Loops, maintains golden datasets, interprets quality metrics	LLM-as-judge techniques, semantic evaluation
ML / AI Engineer	Integration, guardrails, model infrastructure, monitoring pipeline	Model APIs, RAG systems, output validation
Domain Expert / Reviewer	Human evaluation of outputs, defines what 'good' looks like in context	Structured feedback for AI outputs

As with other PM Methodologies, one person could play multiple roles in smaller teams.

13. Getting Started with AGENI

Do not try to implement everything at once. Pick the problem that is causing the most pain right now and start there.

If your biggest problem is output quality:

- Start with the Eval Loop. Build a golden dataset of 50–100 representative inputs and expected outputs. Run it after every prompt change.

If your biggest problem is prompt chaos:

- Start with Prompt Sprint discipline. Version your prompts in Git, write a changelog for every change, and require a regression run before merging.

If your biggest problem is production incidents:

- Start with Deploy & Monitor. Add output quality logging, set up alerts for model API changes, and define a rollback procedure for prompt versions.

If your biggest problem is planning and alignment:

- Start with Define. Use the AI Product Brief to force explicit decisions about success metrics, failure modes, and human oversight before development begins.

If you are a solo builder or early-stage team:

- Use AGENI Lite: Define + Prompt Sprint + Eval Loop. Skip the formal integration spec and monitoring runbook until scale demands it.

> If you apply AGENI to a real project, the most useful thing you can do for the framework is measure what happens. Before you start: define what you will track (time from prompt change to release decision, number of Prompt Debt items found, time to detect drift). After you finish: share what worked and what did not. That evidence is what turns a conceptual framework into a validated one. AGENI v1.0 needs pilots more than it needs advocates.

14. Decision Gates — G0 through G3

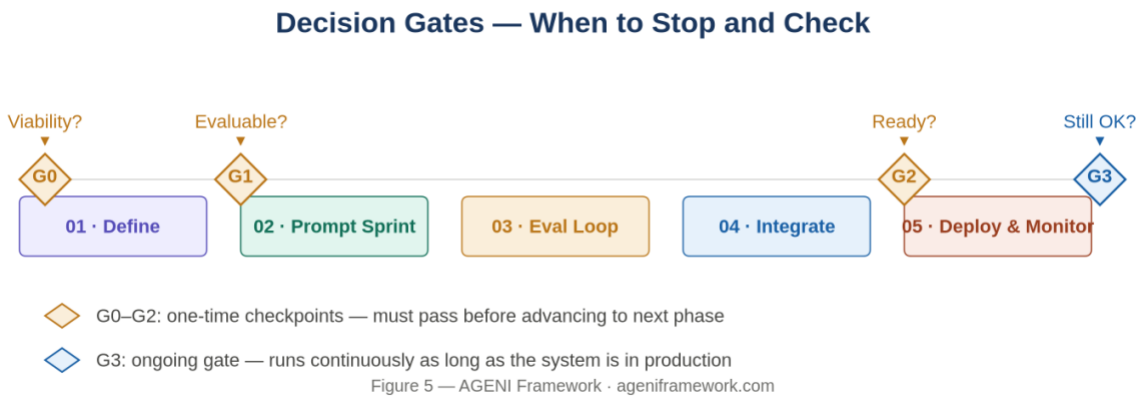

Figure 5 — Decision gates G0–G3 mapped to the five phases of AGENI.

Without clear decision points, a framework is just advice. The four gates below define when to stop and check before moving forward. Each one has a simple question, a minimum set of evidence required to answer it, who makes the call, and what causes a hard stop.

> The gates below are proposed governance checkpoints, not battle-tested procedures. They represent the structure a pilot team would adopt and refine through actual use.

G0 — Viability — Should we use GenAI for this?

Decision question	Should we use GenAI for this problem at all?
Minimum evidence	Use case defined with user and expected impact. Non-AI alternative evaluated. Initial constraints identified (cost, privacy, compliance). Key risks documented.
Approver	Business Sponsor with PM consultation
Blocking condition	No non-AI alternative was evaluated. The use case does not justify the variability inherent in generative outputs.

G1 — Evaluable Design — Can we build and measure this?

Decision question	Can we build and measure this solution?
Minimum evidence	Quality metrics defined and quantifiable. Initial evaluation dataset created. Failure modes identified. Model, RAG, and tooling architecture proposed.
Approver	PM with QA Engineer consultation
Blocking condition	No way to measure output quality. No evaluation dataset. Failure modes are unknown.

G2 — Production Ready — Is the risk acceptable to release?

Decision question	Is the risk acceptable to release to production?
Minimum evidence	Eval Scorecard results within thresholds. Guardrails tested adversarially. Fallback configured and tested. Cost within budget. Observability active. Kill-switch tested. Residual risk accepted by Sponsor.
Approver	PM (go/no-go). Sponsor (residual risk). Legal/Compliance veto if regulated.
Blocking condition	Eval Scorecard fails. Guardrails do not cover critical risks. Fallback untested. Residual risk not accepted.

G3 — Controlled Operation — Is this still acceptable in production?

Decision question	Is the system still acceptable in production?
Minimum evidence	Quality trends stable or improving. No unresolved Sev1 incidents. User feedback within tolerance. Model provider changes evaluated. Drift monitor active.
Approver	QA Engineer (quality metrics). PM (maintain/revert decision).
Blocking condition	Critical drift detected. Active Sev1 incident. Model update without re-evaluation. Sustained quality degradation.

15. Rigor Levels — A1, A2, and A3

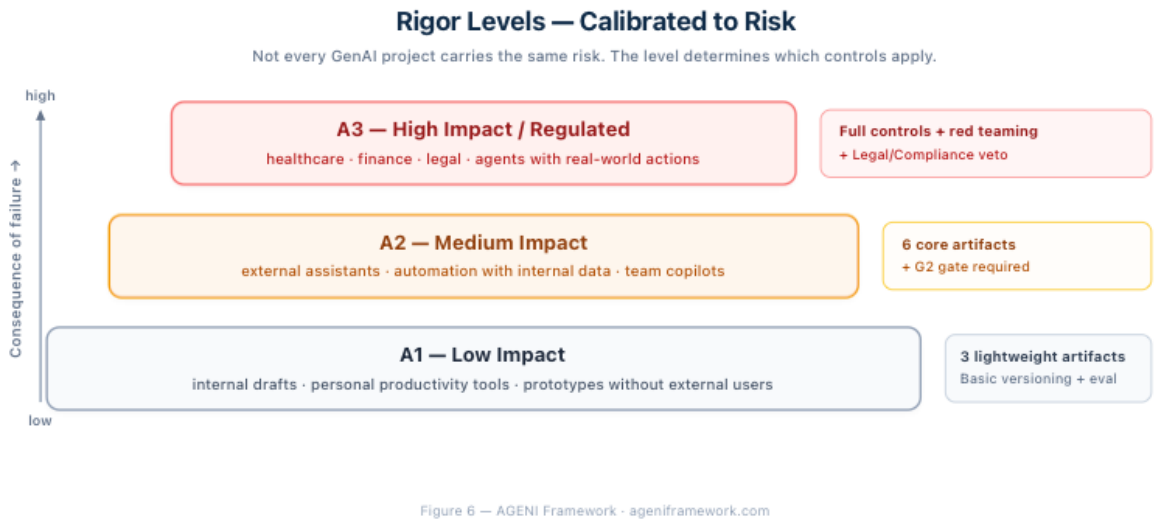

Figure 6 — Rigor levels A1, A2, A3: controls calibrated to the consequence of failure.

Not every GenAI project carries the same risk. A tool that helps an employee write internal drafts is different from a system that answers medical questions. The three levels below let teams calibrate how much process to apply based on what happens if something goes wrong.

> Applying A3 controls to an A1 project wastes resources. Applying A1 controls to an A3 project introduces unmanaged risk.

A1 — Low Impact

Typical cases: internal draft generation, personal productivity tools, prototypes with no external users.

Minimum requirements:

- AI Product Brief — lightweight (one page)
- Prompt Library — basic versioning
- Eval Scorecard — basic metrics, user feedback as primary signal
- Fallback — simple default response
- Cost monitoring — basic usage tracking

A2 — Medium Impact

Typical cases: external-facing assistants (non-regulated), automation with internal data, productivity copilots.

Minimum requirements:

- AI Product Brief — complete, with quantifiable metrics
- Prompt Library — versioned with change log and QA review for major changes
- Eval Scorecard — defined thresholds, LLM-as-judge calibrated
- Guardrails Spec — adversarial testing required
- HITL Protocol — defined escalation thresholds
- Drift monitoring active
- Gate G2 approval before production

A3 — High Impact / Regulated

Typical cases: healthcare, finance, legal, agents with permissions to act in external systems, mass public exposure.

All A2 requirements, plus:

- Formal human review in the evaluation loop
- Adversarial red teaming before production
- Kill-switch tested under failure conditions
- Incident response plan with assigned roles
- Legal/Compliance review and veto authority
- Post-mortem required after any Sev2+ incident

16. Roles and Responsibilities

16.1 The five roles

In small teams, one person will wear multiple hats. What matters is that the responsibilities are assigned.

Role	Primary responsibility	New skill required
AI Product Manager (PM)	Owns the AI Product Brief, defines success criteria, manages Prompt Debt backlog, signs go/no-go at G2	Eval design, GenAI capability awareness
AI Engineer (AIE)	Runs Prompt Sprints, maintains Prompt Library, builds guardrails and integration	Prompt design, context engineering, RAG
QA / Eval Engineer (QAE)	Designs and runs Eval Loops, owns golden dataset, defines quality thresholds — cannot be overridden by PM or AIE on eval results	LLM-as-judge calibration, semantic evaluation
ML / Platform Engineer (MLO)	Deployment, infrastructure, observability, incident response — pre-authorized for kill-switch without PM approval in Sev1/Sev2	LLMOps, monitoring pipelines
Business Sponsor (SPN)	Funds the project, accepts residual risk at G2, final escalation point	AI risk literacy

Legal/Compliance is a conditional role — not a standing participant. It activates when the system processes PII, health, financial, or legal data; when outputs are exposed publicly; or when a regulated domain is involved.

16.2 Simplified RACI

R = Responsible. A = Accountable (one per row). C = Consulted. I = Informed.

Artifact / Decision	PM	AIE	QAE	MLO	SPN
AI Product Brief	**A/R**	C	C	I	C
Prompt Library	I	**A/R**	C	—	—
Eval Scorecard	C	C	**A/R**	I	I
Guardrails Spec	I	**A/R**	C	I	—
Gate G0 — Viability	C	C	—	—	**A**
Gate G1 — Evaluable	**A**	C	C	—	I
Gate G2 — Production	**A**	C	C	C	**A***
Gate G3 — Ongoing ops	C	—	**A**	R	I
Kill-switch (Sev1/Sev2)	I**	R	—	**A**	I
Prompt Debt backlog	**A**	R	C	—	—

* SPN signs residual risk acceptance at G2 — not the go/no-go decision itself.
** PM is informed post-action within 15 minutes — not pre-approval.

17. Definition of Done — Per Phase

In traditional Agile, 'done' means the code works and the tests pass. In GenAI development, done means something more: the output has been evaluated, the configuration is traced, and the quality is documented.

Phase	Done means...	Gate enabled
01 · Define	Success metrics are quantifiable. Failure modes are documented. Sponsor is aligned. Non-AI alternative was considered.	G0 — Viability
02 · Prompt Sprint	Prompts are versioned in version control with a change log entry. Parameters are documented. At least one evaluation run has been executed.	G1 — Evaluable Design
03 · Eval Loop	Eval Scorecard results are reproducible. Metrics within defined thresholds or decision to iterate is documented. No critical failures unresolved.	G2 input

04 · Integrate	Guardrails are tested adversarially. Fallback is configured and tested. Observability is active. Kill-switch is tested. Configuration is fully traced.	G2 — Production Ready
05 · Deploy & Monitor	Drift thresholds are defined and monitored. Incident response is in place. Baseline eval results documented for future comparison.	G3 — Controlled Operation

18. What to Claim — and What to Avoid

Credibility comes from saying what you actually know — not what sounds good. The tables below separate the claims AGENI can back up from the ones that would fall apart under scrutiny.

18.1 Claims to avoid

Claim	Why to avoid it
"AGENI is the first framework for GenAI"	Not provable. MLOps, LLMOps, and AI governance practices already exist.
"These concepts don't exist anywhere else"	Prompt versioning, LLM-as-judge, and drift monitoring are known in LLMOps. The value is integrating them into a PM-accessible workflow.
"AGENI replaces Scrum or PMBOK"	AGENI complements existing methodologies. It fills the gap they leave for probabilistic systems.
"AGENI guarantees quality in GenAI systems"	No framework guarantees quality in probabilistic systems. AGENI manages risk and makes degradation visible earlier.
"AGENI is irrefutable"	Nothing in project management is irrefutable. Practices are validated by evidence from real use.

18.2 Defensible claims

Claim	Why it holds
"A framework for teams where the model's output is what the user sees"	Precisely scoped. Excludes teams using AI as an internal development tool.
"Integrates evaluation, governance, and operations for GenAI"	The individual practices exist elsewhere. The integration in a PM-accessible structure is the contribution.
"Complements Scrum, Kanban, PMBOK, and LLMOps"	Accurate and makes adoption easier. Practitioners do not have to abandon what they know.
"Adapts rigor to the risk level of the project (A1–A3)"	A differentiating and verifiable characteristic.
"Treats model updates as a planned event, not a production incident"	A genuinely underserved problem no existing PM methodology addresses explicitly.

The strongest position AGENI can take is intellectual honesty. A framework that acknowledges its own limits, names what it does not cover, and refuses to over-promise will earn more trust from experienced practitioners than one that claims to solve everything. That trust is what drives adoption.

AGENI v1.0 is a starting point. ageniframework.com — CC BY-SA 4.0

I invite you to use it and send me any feedback - ageni.framework@gmail.com