

Generative Specification: A Two-Page Introduction

Generative Specification: A Two-Page Introduction

For people who build things, lead teams, or work in any domain where AI is changing how work gets done — and who don't have time to read a forty-page paper.

The Problem

Every organization using AI coding tools is discovering the same thing: AI can write code faster than any human, but the results drift. One session produces something good; the next session, on the same project, produces something incompatible. The team gets more output but also more chaos. The code is abundant; quality is not.

The reason is structural. An AI assistant has no memory of what your system is supposed to be. Every session starts from scratch. Without a persistent description of what correctness means for your specific system, the AI fills the gap with its best guess — which changes every time.

This is not a model quality problem. It is an architecture problem. And it has a solution.

The Idea

Generative Specification (GS) is a programming discipline built around a single inversion: the specification is the primary artifact, and code is derived from it.

In conventional software development, code is primary. Documentation, tests, and architecture diagrams try to describe the code after it exists — and immediately begin to go stale. The AI, reading stale documentation, produces stale code.

In GS, you write a complete, precise description of what your system must be — what it does, what it must never do, how it handles failures, what compliance obligations it carries, how it should be tested — before any code exists. That description is machine-readable. The AI reads it at the start of every session and derives the code from it, every time, without drift.

The specification does not go stale because the specification *is* the source. If the system changes, the specification changes first. Code is always derived. The artifact that governs the system and the artifact the AI reads are the same thing.

What It Removes

GS does not improve how software is written. Applied fully, it removes entire categories of work from the practitioner's responsibility:

What gets removed	How
Writing code	The AI derives it from the specification
Reading and reviewing generated code	Automated behavioral contracts verify it

What gets removed

Managing infrastructure
 Monitoring and diagnosing bugs
 Evolving and improving the system

How

The same specification governs deployment
 Runtime signals are evaluated against the spec automatically
 The specification governs its own mutation

The first four of these are demonstrated across production deployments. The fifth is running in the Loom research project.

The Evidence

In April 2026, 83 software developers participated in a controlled study at Mitikah, Mexico City. Half worked with standard AI assistance; half worked under GS discipline using the same AI tools.

The GS group produced three times the rate of fully executable, deployable implementations — systems that passed live integration tests without modification. The effect was strongest for the most demanding output level: perfect implementations.

This was not a study of AI capability. The AI tools were identical. The difference was whether the practitioners had a complete specification or not.

Who It Is For

GS is not only for software engineers. The specification discipline applies to any domain where:

- A capable AI executor exists (it does, for software, legal, medical, financial, and scientific domains)
- Outcomes can be observed and verified
- The cost of drift — inconsistent outputs across sessions, teams, or time — is significant

The practitioner GS most rewards is not the fastest coder. It is the practitioner who can say precisely what a system must be — who holds the domain knowledge, the compliance requirements, the architectural judgment — and translate that knowledge into a specification the AI can execute indefinitely.

How to Evaluate This Work Without Reading the Papers

A guide for using an AI assistant as your reading proxy.

The research behind GS spans a white paper, two companion essays, a formal language specification, and an ongoing biological isomorphisms study. Combined, they exceed 200 pages. You should not read all of them. You should ask an AI to read them for you — and then ask the AI about your domain.

What to do

1. **Get the files.** Ask Juan Carlos for the document package: the white paper, the *Onwards!* essay, and the Loom language overview. Load them into your AI assistant of choice (Claude, ChatGPT, Gemini — any that accepts document uploads).
 2. **Add your domain context.** The AI's response quality scales with the specificity of your question. Tell it where you work and what you care about before you ask anything.
 3. **Ask probing questions, not soft ones.** The papers are designed to hold up under adversarial questioning. A soft question ("is this good work?") gets a soft answer. A hard question gets a useful one.
-

Questions that work well

On quality and credibility: - "What is the central claim of this paper and what evidence supports it? Where is the evidence weakest?" - "Is the statistical methodology in the DX1 study sound? What are its limitations?" - "What would a skeptical peer reviewer object to, and how does the paper address those objections?"

On your domain: - "I work in [healthcare / legal / financial services / defense / education]. What does GS mean for my domain specifically?" - "We have significant compliance obligations [HIPAA / SOC 2 / GDPR / FedRAMP]. Does this methodology address those, and how?" - "Our AI adoption is stuck at the 'chat interface for discrete tasks' level. What would moving to GS discipline actually require from our team?"

On the research program: - "What is the relationship between the white paper, the Loom language, and the biological isomorphisms work? Do they hold together as a coherent program?" - "The author claims five of the seven obligation tiers are proven. Is that claim defensible given the evidence presented?" - "What comes next in this research program and does it follow logically from what's been demonstrated?"

On working with the author: - "Based on these documents, what kind of problems is this person well-suited to work on?" - "What domains or problem types would benefit most from this methodology?" - "If I brought this person into a project in [your domain], what would the collaboration look like?"

What to expect

A well-formed question to an AI that has read these documents will produce a substantive, domain-specific answer — not because the AI is being flattering, but because the papers are designed to be queryable. They contain evidence, named mechanisms, concrete examples, and honest statements of what is not yet proved. The AI will surface all of it.

If the AI finds gaps or weak claims in response to your questions, that is useful information. The papers are designed to hold up under that test. If they don't hold up in your domain, that matters — and it is better to find out through a ten-minute AI conversation than through a project engagement.

Juan Carlos Ghiringhelli — jcghiri@gmail.com — pragmaworks.dev

White paper: "Generative Specification: A Pragmatic Programming Paradigm for the Stateless Reader" (arXiv, 2026) *Companion essay:* "Onwards! The Formal Tradition Was Waiting for Its Executor" (submitted to ACM SIGPLAN Onward! 2026) *Language:* Loom — github.com/jghiringhelli/loom