# Gila (Geetha) Nehemia

gnehemia@gmail.com • linkedin.com/in/gnehemia

Portfolio case study

# Enterprise release notes automation pipeline

A full project lifecycle: from pain point discovery to a working AI-powered content operations system, for complex multi-phase software and hardware releases.

| **Active**<br>pilot in progress | A documentation pipeline that previously required significant manual effort across every release cycle now runs automatically, significantly reducing manual production time. Formal metrics collection is underway as part of an active structured pilot. |
|---|---|

HOW IT STARTED

This wasn't a top-down AI initiative. It started with one person solving her own problem.

> *"I wasn't trying to build an AI pilot. I was trying to solve my own problem. The work opened conversations with senior leadership, generated demo requests across teams, and connected me to the organization's broader AI initiatives. The solution spoke for itself."*

The pilot created organic momentum that no formal program produced. It was built from real frustration, real expertise, and a genuine need to do better work.

| **Senior leadership conversations** | **Cross-team AI pilot connection** | **Demo requests** |
|---|---|---|
| Work surfaced to senior manager organically through results | Linked to broader organizational AI initiatives led by another manager | Colleagues and stakeholders asked to see the work firsthand |

ORIGIN

A year before the first line of automation was built, a colleague and I identified that the release notes process was fundamentally broken. We didn't jump to a solution. We documented the pain points, produced design documentation, a phased pilot plan, and a testing plan, then built.

THE PROBLEM

Enterprise release notes for large-scale software products are manually intensive, inconsistent across product lines, and difficult to scale across service pack releases and hardware integrations. Each

release cycle required significant human effort across multiple enterprise systems with no automated handoffs between them.

---

THE INSIGHT

My own release note workflow, built over years of enterprise documentation experience, could be deconstructed into discrete, testable steps and encoded into an AI pipeline. Rather than starting from scratch, I digitized my own expertise into a replicable, automated system that applies institutional knowledge as a guardrail at every step.

---

ARCHITECTURE

The pipeline is built as 6 modular, independently testable skills, each consuming the output of the one before it. A shared content generation engine is called by multiple skills, applying institutional style guide and terminology guardrails consistently. Example XML structures are embedded directly within each skill, enabling the system to generate correctly structured output files from scratch.

> *Designing each skill as a small, discrete unit means errors can be isolated at the module level without disrupting the rest of the pipeline. The same principle used in software engineering. Build once, reuse everywhere.*

---

ENGINEERING PROGRESSION

It follows a deliberate engineering arc, the same progression used by professional AI engineering teams:

### → Deterministic model

Built first. Rule-based, predictable, controlled outputs. Every input produces the same output. The foundation was made reliable before any intelligence was layered on top.

### → JSON logging and observability

The pipeline is fully instrumented. Every action is logged, traceable, and auditable, enabling debugging, improvement, and future metrics collection.

### → Python processing

Connective tissue that moves structured data intelligently between enterprise systems at each stage of the pipeline.

### → Guardrails

AI behavior is constrained by institutional knowledge: style guide, terminology file, and embedded XML templates, keeping quality consistent and outputs compliant.

### → Agentic reasoning

During testing, the model demonstrated agentic behavior: it identified a version mismatch unprompted and asked whether to extend logic to the next release, anticipating the next step without being instructed to. Full agentic deployment is in progress.

### → Programmatic CCMS integration

In progress. Direct API access to the enterprise content management system will eliminate the final manual handoff, completing a fully continuous pipeline from source data to published documentation.

---

ORCHESTRATOR LAYER

Three of the pipeline's core data-processing skills are linked and sequenced by a Windsurf-native orchestrator, a coordination layer that sits above the modular skills and manages execution, state handoff, and routing programmatically. Rather than triggering each skill manually, the orchestrator fires all three in a single automated run, passing structured output from each skill directly to the next.

**WINDSURF ORCHESTRATOR**

Single invocation: routes and sequences all three skills programmatically

| 01 **Deferred Issue Validator** ↓ | 02 **Resolution Detector** ↓ | 03 **File Transform Engine** ↓ 3 |
|---|---|---|
| annotated XML | status-annotated XML | output files |

Windsurf's MCP server integration gives the orchestrator direct, programmatic access to the engineering issue tracking system and the file system. Each skill runs as an instrumented unit and the orchestrator reads its JSON-logged output to determine routing for the next skill. No manual handoff, no context switch between tools.

---

THE SIX SKILLS

01

#### Deferred issue validator

Queries the engineering issue tracking system using standard engineering queries to identify deferred issues. Cross-references the master XML file, flags new deferred issues, and generates compliant first-draft documentation using the shared content generation engine.

■ uses content generation engine ↓ annotated XML ↓ first-draft content

02

#### Resolution detector

Scans the engineering issue tracking system and enterprise knowledge base for resolved issues. Annotates the XML file with fixed-status comments, transforming it into a live document that reflects current release state at all times.

↓ status-annotated XML

03

### File transformation engine

Rebuilds the known issues file so only current deferred issues remain. Simultaneously generates two new XML files from scratch using embedded template structures: one for resolved issues, one for new features processing.

↓ rebuilt known issues XML ↓ resolved issues XML ↓ new features XML

04

### New features extractor

Simultaneously queries the engineering issue tracking system and design platform to pull feature data, bridging engineering tickets and design artifacts automatically. Passes structured data to the content generation engine for compliant first-draft content.

■ uses content generation engine ↓ completed new features XML ↓ first-draft content

05

### Content generation engine

Shared service used across the pipeline. Takes structured data and generates compliant first-draft content using the institutional style guide and terminology file as guardrails, producing on-brand, accurate documentation before a human ever touches it.

■ shared service · called by skills 01 and 04

06

### SME notification engine

Automatically posts a comment in the engineering issue tracking system to the responsible subject matter expert when a new deferred issue is documented, closing the loop between automation and human review without manual tracking.

↓ automated SME notification

#### WHAT THE PIPELINE PRODUCES

Five outputs from one pipeline run, all compliant, correctly structured, and routed for review.

| Output 01 | Output 02 | Output 03 | Output 04 | Output 05 |
|---|---|---|---|---|
| **Rebuilt known issues XML** | **Resolved issues XML** | **New features XML** | **First-draft content for all sections** | **Automated SME review notification** |

#### TOOLS AND INTEGRATION

| | | | |
|---|---|---|---|
| Python | JSON processing | XML manipulation | Template-driven generation |
| Windsurf | MCP server integration | Orchestrator layer | Engineering issue tracking |
| Enterprise knowledge base | Design platform integration | Enterprise CCMS | Agentic workflow design |
| Modular pipeline architecture | | | |

**OUTCOME**

A working, enterprise-scale AI content operations pipeline built from first principles: originated through collaborative pain point discovery, governed by formal project artifacts, and engineered through a deliberate progression from deterministic model to instrumented, guardrailed system with agentic behavior demonstrated in testing and full agentic deployment in progress. A Windsurf-native orchestrator links three automation skills (Deferred issue validator, Resolution detector, and File transformation engine), executing them in sequence and completing the full data ingestion and transformation phase in seconds from a single invocation. Integrated across five enterprise systems with programmatic CCMS integration in progress. Significantly reducing manual release note production time with formal metrics collection underway as part of an active structured pilot. Generated organic senior leadership conversations, cross-team demo requests, and connection to broader organizational AI initiatives, all without a formal mandate. Actively advocating for formal engineering partnership to scale the model organization-wide and connect documentation quality to measurable reductions in service calls. Designed, built, and operated by one person.