AI: From Theory to Practice

The Venom v1.5 Case Study

Warsaw, February 16, 2026
Author: Maciej Pieniak
Proofreading and diagrams: Gemini 3 and ChatGPT 5.2

Introduction

AI assistants are very good at answering questions. They are far less capable when it comes to responsibility for decisions.

Venom was created to monitor the decision-making process and to control it where decisions have real consequences.

In my previous series of popular-science articles, I focused on theoretical considerations concerning artificial intelligence, based on a working definition of an AI Assistant that I developed earlier. The time has now come to verify those assumptions in practice.

This article represents the next step — a transition from theory to the presentation of a tool that allows those assumptions to be tested. In other words: from theory to experiment.

Over the past few months, I have created several GitHub repositories as the result of experiments in process automation, based on architectures designed for specific, clearly defined goals. One of those projects is Venom.

Venom was designed to verify two key assumptions:

- Whether the concept of an AI Assistant — with clearly defined norms and an explicitly established level of autonomy — can be effectively implemented in a working system that supports decision-making processes.
- Whether, based on open-source technologies, it is possible to architecturally and functionally approach commercial solutions developed by the largest technology companies in the AI domain.

I. Business Context

Most users have already become accustomed to interacting with language models in the form of a chat — such as Gemini, GPT, or Grok. This interaction is becoming increasingly natural, and the impression that the system “understands the user’s needs” is steadily growing. Rightly so — from the perspective of user experience, chat has proven to be an exceptionally effective interface.

However, this is not the natural domain of operation for most language models.

At their core, LLMs generate a response to the most recent textual input. They do not “remember” the conversation in a continuous way, nor do they track its meaning over time. What users perceive as conversational context is, in practice, the result of additional mechanisms: memory systems, auxiliary tools, control rules, and technical prompts appended to the user’s visible input.

Commercial solutions do not expose the technical layer of the conversation. As a result, the user has the impression of interacting with an “intelligent entity” operating as a black box — without insight into the rules according to which reasoning was performed and a specific response was generated.

II. Business Assumptions of the Venom Project

Venom is an experimental system designed to analyze and control decision-making processes in solutions based on language models. It is not a finished product, but rather a system that allows observation, testing, and modification of how an AI assistant operates beyond the LLM itself.

The core assumption of the project is local processing. The system can be launched and configured on a standard personal computer. Once a minimal version is configured, it can operate without Internet access.

By design, Venom assumes a single-user mode, where the user acts as the system supervisor. The user determines the scope of autonomy, responsibility, and access to external resources. The use of external services or commercial APIs is either referential or serves to acquire auxiliary knowledge in cases where local system results are unsatisfactory.

An integral part of Venom is a web interface designed as an operational tool. Its purpose is to provide insight into:

- the course of decision-making processes,
- the current state of the system,
- dependencies between components.

Key elements of the interface include:

- the **Inspector** — enabling tracing of the decision path for individual interactions,
- the **Knowledge Graph** — presenting a dynamic model of context and relationships between information.

In designing the interface, the “data close to the user” strategy was applied, meaning that information is presented according to priority, with the ability to drill down into details using a one-click principle.

In Venom, the level of instrumentation may appear excessive. This follows directly from the nature of the project — both the implementation of Venom itself and its core utility rely on multi-agent AI systems.

In such architectures, accountability of individual components and proper documentation are not optional — they are a prerequisite for controlled development and further system evolution.

Without explicit instrumentation and accountability of decision-making processes, an AI-based project is not a system — it is a collection of random behaviors.

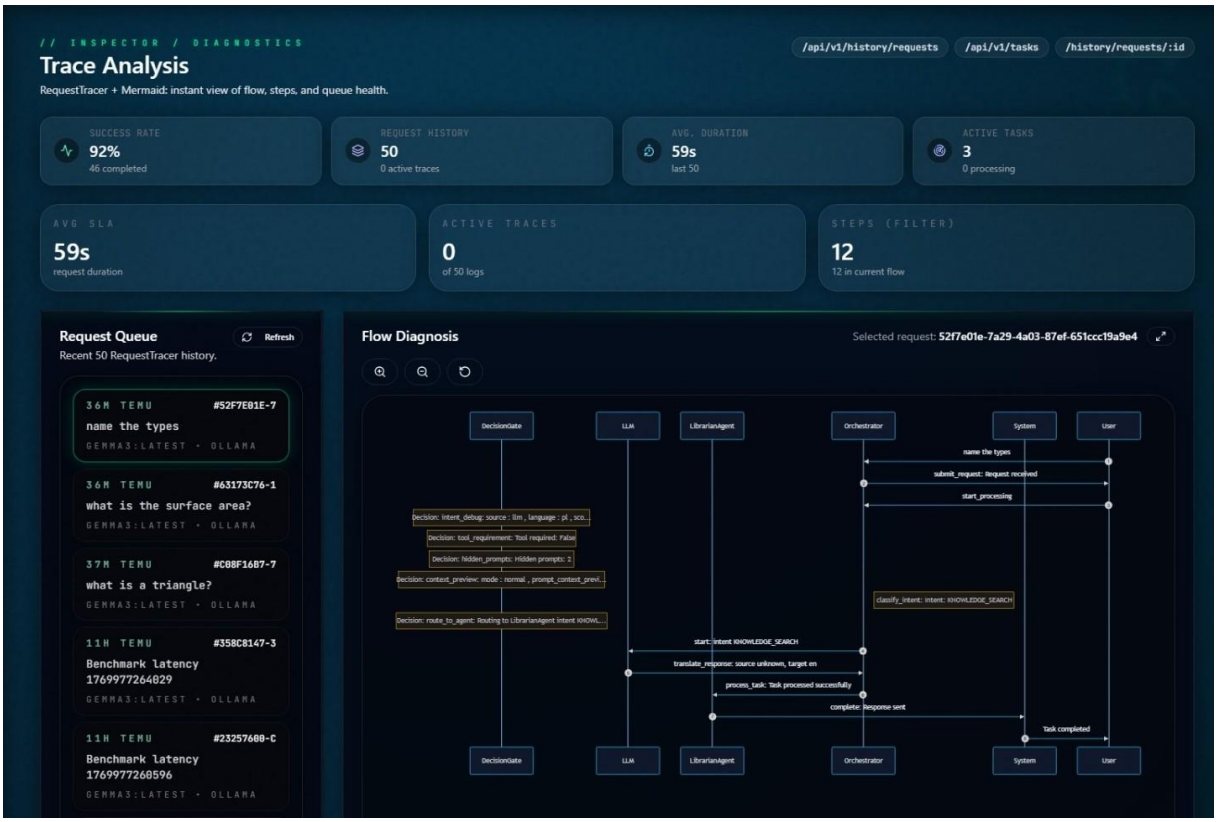

Figure 1 - Trace Analysis — request flow & orchestration diagnostics

III. Venom System Architecture

Discussions about architecture are best started with a diagram. In the case of Venom, a dedicated ecosystem was designed.

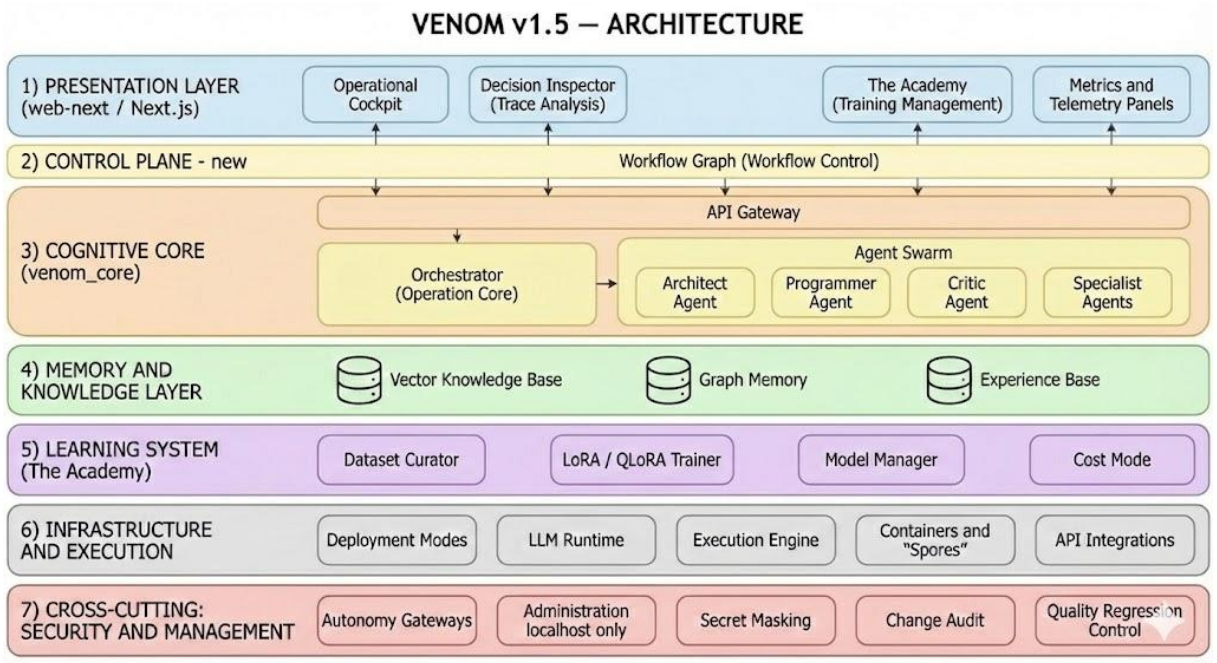

Figure 2 - High-Level Logical Architecture of Venom v1.5

In my projects, I prefer real-world analogies instead of internal code names. A simple conceptual table — without implementation detail — helps maintain a systemic perspective.

In systems of this type, it is easy to fall into ambition-driven design and aim for "the system should do everything." Clear versioning and execution order prevent architectural chaos.

Below is a conceptual overview of Venom components.

Organ	Function	Role in the organism	Technology	Version
Habitat	Environment	Sandbox	WSL2 + Dev Containers	v1.0
Communication	Thought exchange	Inference engine	Ollama / vLLM, FastAPI + WebSocket, Next.js	v1.0
Nervous system	Orchestration	Dialogue, decision loops	AutoGen + Orchestrator (FastAPI)	v1.0
Metabolism	Performance	Model execution	ONNX / GGUF	v1.0
Circulatory system (Hive)	Queues & distribution	Task routing & statuses	Redis + ARQ	v1.0
Frontal lobe	Fast thinking	Generates ~90% of code	Phi-3 (ONNX/GGUF), Ollama/vLLM	v2.0
Oracle	Deep thinking	Hard problems	OpenAI GPT-4o, Gemini, Claude	v1.0
Extended intelligence	External sense	Internet knowledge	Researcher Agent + DDG/Tavily	v2.0
Hippocampus	Memory	Knowledge map	GraphRAG + LanceDB	v1.0
Cerebellum	Learning (Fine-tuning)	Muscle memory, reflexes	The Academy (LoRA/QLoRA)	v1.5
Prefrontal Cortex	Control	Conscious planning	Workflow Control Plane	v1.5
Hands	Action	Files, shell, git	Semantic Kernel + Skills	v1.0
Eyes (digital)	UI perception	Screenshot analysis (eyes.py)	Ollama (vision) / OpenAI GPT-4o	v1.6
Eyes (digital)	UI perception	Target local engine	Florence-2 ONNX	v2.0
Ears	Hearing (STT)	Audio transcription	faster-whisper (CTranslate2)	v1.6
Mouth	Speech (TTS)	Voice synthesis	Piper TTS (ONNX)	v1.6
Eyes (physical)	World perception	Objects, obstacles	YOLO ONNX	v2.0
Legs	Movement	Mobility	Rider-Pi integration	v2.0

IV. Architecture Evolution

Within 50 days, Venom evolved from a proof-of-concept script into a modular ecosystem. The repository structure now reflects clear responsibility domains.

The system consists of five cooperating layers:

1. **Venom Core (/venom_core) — The Operations Brain**
   - The decision-making engine of the Assistant.
   - It implements orchestration, agent coordination, and memory management.
   - The Core performs the "thinking" process and exposes an API, but has no graphical interface. It is pure business logic.
2. **Web Next (/web-next) — Control Plane**
   - The operational layer separated from the logic.
   - It provides real-time visibility into agent states, decision traces, and system metrics.
   - Separating presentation from execution enables asynchronous interaction and observability.
3. **Venom Spore (/venom_spore) — Execution Units**
   - A lightweight execution component enabling distributed task handling.
   - It allows heavy or isolated operations to run outside the main Core, preparing the system for multi-node environments.
4. **The Academy — Learning Layer**
   - A mechanism that transforms system history into development material.
   - Venom can distill operational experience and use it to refine models through controlled fine-tuning, preserving versioning and change traceability.
   - This marks the transition from a prompt executor to a system capable of structured learning.
5. **Workflow Control — Plan Before Apply**
   - A governance layer for configuration and structural changes.
   - It introduces a controlled Plan → Apply flow, ensuring validation before modification.
   - Changes are audited and predictable, reducing configuration risk in agent-based systems.
6. **Security Policy — Dual Trust Model**
   - As autonomy increases, boundaries become critical.
   - Venom is designed to be secure both against uncontrolled agent actions and against unintended operator interference.
   - System areas and workspace are clearly separated, and mutating operations require explicit authorization and traceability.
7. **Metrics and Observability**
   - Venom treats AI as an engineering system, not a black box.
   - Token economy, latency, and decision traces are monitored.
   - This enables debugging not only of code, but of reasoning paths themselves.

Summary

The Venom project serves as a practical verification of the thesis that the primary challenge in LLM-based systems is not response generation, but decision control and responsibility management.

The architecture demonstrates that an AI Assistant — defined by explicit norms, roles, and autonomy levels — can be implemented as a transparent and analyzable system.

At the same time, the project confirms that open-source technologies can architecturally approach commercial-grade solutions, provided that orchestration, memory, governance, and security are treated as first-class system components.

Venom is not a chatbot experiment.

It is a decision-analysis environment for AI systems.

In the next article, I will focus on another dimension of the project — AI-Augmented Development and its impact on the architect's role and software engineering workflows.

Project Repository

Venom is developed as an open engineering experiment.

The complete source code, architecture diagrams, and documentation are publicly available:

👉 https://github.com/mpieniak01/Venom

The repository reflects the current v1.5 architecture and its evolutionary path from a proof-of-concept into a decision-analysis environment.