DARIUSZ WALAT

Senior Software Engineer | Trustworthy Output from Untrustworthy Data

Email: dariusz@walat.eu
LinkedIn: linkedin.com/in/dariusz-walat | **GitHub:** github.com/stakent | **Articles:** walat.eu
Location: Rzeszów, Poland (EU Timezone) | **Remote only** (EU hours or async)

PROFESSIONAL SUMMARY

Senior Software Engineer with 20+ years building systems where silent failure is not an option — from industrial control where downtime cost thousands per minute, to data pipelines that must stay correct when their inputs aren't.

I build systems that catch what's silently wrong — a dead sensor nobody noticed, contradictory records, a garbled scan, a confidently-wrong model — so it surfaces as a flag today, not a six-month catastrophe.

I use LLMs where they earn their place: extracting meaning, embeddings, plain-language access to jargon-locked data. I keep them out of decisions and out of the data flow that has to be correct, which stays deterministic and verifiable. An LLM can help you find the answer; it can't be the answer.

TECHNICAL EXPERTISE

Core Engineering

- **Minimal sufficient engineering:** the simplest system and the least resource that meets the goal — fewest dependencies, deliberately boring technology. I evaluate the bleeding edge early (PyPy, pre-1.0 React, Elixir/OTP at launch, Go at release, LLM agents now) and commit only when the ecosystem carries the mundane parts and the culture doesn't manufacture complexity. The domain problem must be the only hard problem.
- **Languages:** Python, Go, SQL
- **Backend:** FastAPI, Django, DRF; server-rendered dynamic UIs — pattern shipped in production with Unpoly, HTMX today
- **Data:** PostgreSQL-first — pgvector, full-text search, JSONB; dedicated systems only where PostgreSQL genuinely isn't enough
- **Infrastructure:** Linux, systemd, Incus system containers (sandboxing autonomous AI agents); right-sized VPS + tiered bulk storage (European hosting)
- **Concurrency by workload:** asyncio fan-out where I/O dominates, long-lived worker processes for expensive tools, plain sequential code where that is enough
- **Design before code:** the level where systems succeed or fail — concepts and entities, data flow, and which guarantees hold (and which deliberately don't); written down before implementation, so failure modes are design decisions, not discoveries
- **Verification against the territory:** documentation — vendor docs, design documents, model output — is a map, and maps drift; correctness is checked against the live system with the method chosen by stakes and context: reading the code, running it, probing the real service, reading the interaction logs
- **Code-level gates:** tests-first, CI/CD, linters and static analysis — enforced on all code, whoever or whatever wrote it

AI & Data Reliability

- **LLM integration:** meaning extraction, embeddings, semantic search — access layer only, never the decision path
- **Hybrid search:** PostgreSQL full-text + vector; multiple embedding spaces versioned side by side with atomic switchover and sticky per-user assignment; embeddings served by a custom dual-mode inference daemon (batch throughput / low-latency queries)
- **Directing AI coding agents (AI-assisted development):** design decisions and verification stay

mine; implementation is delegated and checked by measured behavior
- **Document processing:** PDF extraction, OCR, layout analysis, multi-format parsing
- **Data quality & verification:** correctness gates, cross-method validation, silent-failure detection — deterministic, verifiable data flows over unreliable inputs
- **Background:** neural networks and evolutionary computation since the mid-1990s (see below) — multiple AI cycles of judgment on what survives the hype

PROFESSIONAL EXPERIENCE

SENIOR SOFTWARE ENGINEER

R Systems Computaris Poland Sp. z o.o. | Remote | *January 2023 - Present*

PRAWOPROGNOZA — POLISH LAW AS VERIFIED, QUERYABLE DATA

Personal Project | In active development | *July 2025 - Present*

Building a system that turns the full corpus of Polish national law (~80,000 PDFs, decades of inconsistent formats, OCR-era scans) into a faithful text representation and a document graph — with automated correctness verification as the core design problem.

- **The hard part is not the extraction — it's knowing, automatically, whether the extraction is right:** format-specific correctness gates, cross-method agreement checks, structural invariants from the formal grammar of Polish legislation
- **LLM-assisted development, deterministic production:** LLMs help build and accelerate; the production data flow stays deterministic and verifiable
- **Distributed by design:** PostgreSQL-coordinated batch pipeline (lease-based claiming, timeout recovery), tiered VPS + bulk-storage lifecycle — runs on one host today, scales out when the corpus demands it

POLISH LAW MONITORING SYSTEM

unkot.pl | Personal Project | *Built and operated for 3+ years*

Built a legislative-change notification service and **operated it autonomously in production for over three years** before retiring it — automated document ingestion, processing, search, and subscriber notifications, running unattended.

- **Technical stack:** Django, PostgreSQL, Celery, Docker, Unpoly (server-rendered dynamic UI)
- **Production operations, single-operator:** automated document pipeline, full-text search, subscription management — years of unattended operation against a live, changing, government-published document stream

INDEPENDENT SOFTWARE DEVELOPMENT

Various Personal Projects | *1995 - 2022*

Evolved Recurrent Networks for Market Prediction (1995-1997)

- Rejected published technical-analysis methods on efficient-market grounds; reasoned the model needed memory → recurrent neural networks, with architecture and weights evolved by genetic programming
- Ran the evolution continuously on a single Pentium-class machine; by 1997, daily paper-tested predictions gave usable direction and magnitude for next-day prices — never deployed with real money

Production Supervision System (2012-2013)

- Affordable industrial monitoring — designed to cost an order of magnitude less than the incumbent class of systems
- Prototyped in Python, evaluated Elixir, settled on Go (goroutines, channel-based message pass-

ing), with NSQ isolating the system from unreliable networks (buffering, retry, back-pressure)

E-commerce Data Processing (2008-2018)

- Built and ran a system processing 5-7 million products from multiple, inconsistent sources
- Input-side processing hit Python's performance ceiling despite PyPy and heavy optimization — migrated it piece by piece to Go, after which the performance problems disappeared

INDUSTRIAL SYSTEMS ENGINEER

Various Manufacturing Clients | *1995 - 2015*

20 years maintaining and extending mission-critical production control systems:

- **Re-implemented an estimated 20-50% of the code running a live factory** over the years — largely without a staging environment; a spare controller to test parts of the code against was a luxury, and the inputs could not be emulated. Changes went into a running plant and had to be right.
- **Safety-critical process design:** designed and maintained industrial processes where a wrong process harms people or the environment — long before it costs money
- **Reliability:** zero-downtime systems where failures cost thousands per minute; an unnoticed bad sensor is the canonical catastrophe — detecting it early was the job
- **Technologies:** PLC programming, real-time systems, industrial protocols

EDUCATION & COMMUNICATION

Master's Degree in Agricultural Engineering
Uniwersytet Rolniczy, Kraków, Poland
Thesis: Resource Allocation Optimization System (Pascal)

Communication: 8 years of sustained public-speaking and facilitation practice (Toastmasters); delivered a workshop at an international conference (2026)

ADDITIONAL INFORMATION

Languages: Polish (native), English (daily working language; public speeches delivered in English)
Work arrangement: remote only — EU hours or async; occasional travel (a few times a year) is fine
Work authorization: EU citizen, no visa requirements for EU positions
Personal interests: swimming, public speaking, permaculture, technical reading

Best fit: correctness matters, the data is real-world-messy, and you want someone who names what doesn't work and why.