IOAN MARIAN

SENIOR .NET ENGINEER - PRINCIPAL-LEVEL PLATFORM OWNERSHIP

Bucharest, Romania
+40 747 761 739
marian_ioan2001@hotmail.com

I LOOK FOR THE BREAKING POINT BEFORE THE BUSINESS REACHES IT.

I BUILD MY FIRST MAP OF SYSTEM LIMITS BY HAND.

As a student working in VBA and Microsoft Access, I wanted to know not only whether an application worked, but where its design ceased to be useful. I ran the same SELECT with 1, 10, and 50 fields against 1, 10, 1,000, and 1 million rows, then increased the volume by factors of ten until response time crossed a practical limit.

I repeated the method with loops, adding one cost at a time: a condition, an addition, an object allocation, a function call, then reuse of the same instance - so I could isolate the cost of each decision.

At small volumes, different designs often looked equally fast. Near the limit, the curve changed abruptly: twice the work could create far more than twice the delay, and a harmless operation could become a bottleneck when repeated millions of times. I learned to measure thresholds and amplification, not only averages.

That was the beginning. I carried the same method through COBOL, VBA, VB6, ASP, C#, ASP.NET, ASP.NET MVC, Xamarin, SharePoint on-premises and Online, AD, Azure AD, Excel, PDF generation, PowerShell, external APIs, and applied AI. With every environment, I rebuild the benchmarks: isolate one variable, watch the cost grow, locate the practical breaking point, and design headroom before production reaches it.

MEASURE	ELIMINATE	ENDURE
Business scenario	Why calculate this?	Deliver quickly
Volumes and cardinalities	Reduce the data	Validate against real growth
Stages, time, errors	Remove repeated work	Design with 10-20 years of headroom

FOUR MILESTONES THAT EXPLAIN MY DIFFERENCE

2003 ID-FIRST PAGINATION

A business list loaded complete records to display only one page. I separated row identification from full loading, then generalized an adaptive multi-stage strategy.

From tens of seconds to under one second - approximately 100x

2004-2007 GIRAUD INTERNATIONAL - APPLICATION ENGINE

After testing the limits of Web Controls, I designed an ASP.NET engine with business objects, explicit actions, XML, XSLT, integrated pagination, and separation close to MVC principles. The notation cli;list;detail;save;5 expressed object, context, origin, action, and identifier.

Designed independently as an ASP.NET application engine - successively rehosted in SharePoint 2003 and SharePoint 2007, then expanded into a reusable component library

2008-2013 VENTE PRIVEE - PLATFORM CONTINUITY DURING HYPERGROWTH

Primary, highly hands-on technical ownership of the SharePoint environment used for live business operations, covering development, administration, deployment, user support, training, patching, production continuity, progressive modernization, and transfer to other specialists.

Critical path in third-party code accelerated by approximately 6x

2022-PRESENT CONFIDENTIAL CONSTRUCTION & INFRASTRUCTURE ENGAGEMENT

Primary technical ownership of an operational and financial platform during rapid business and operational growth. Scope includes architecture, C#, SQL, performance, observability, production evolution, and knowledge transfer.

Orders-of-magnitude gains in processing and data access through structural optimization

A "SINGLE-PERSON MULTIPLIER" DESIGNED TO BE TRANSFERABLE

At different stages of my career, I have held primary, highly hands-on technical ownership of operational platforms for years at a time: at Vente Privee, during a nine-year engagement with Europe Airpost, and in my current confidential construction-sector engagement. That concentration is sustainable only when architecture stays clear, observable, transferable, and simple to operate. I trained colleagues able to continue development and operations, so the system depends on its architecture rather than on my presence.

STRUCTURAL PERFORMANCE: REMOVE UNNECESSARY WORK BEFORE ADDING POWER

PUBLIC METHOD EXAMPLE - COMPLEX OPERATIONAL PLATFORM

- Measure each stage. Instrument data access, processing, external calls, output generation, repetitions, volumes, and cumulative time.
- Separate access from processing. A slow page becomes a sequence of measurable operations instead of one vague total.
- Remove repeated work. Reuse stable datasets and prepare direct lookups when the access pattern justifies them.
- Control combinatorial growth. Move invariant work outside nested traversals and avoid rebuilding the same intermediate result.
- Simplify before optimizing. Use a simple path for ordinary cases and preserve detailed processing only for exceptions that require it.
- Design for headroom. Optimization creates time to react when data volume, functionality, or external dependencies grow.

EVIDENCE WITH A CLEAR BOUNDARY

HISTORICAL BASELINE
A documented approximately 100x pagination case from 2003

INTERACTIVE
Response times restored to a practical range as volumes grow

STRUCTURAL
Fewer reads, traversals, combinations, and temporary allocations

OBSERVABLE
Stages, volumes, timings, errors, and trends remain explainable

TRANSFERABLE
Architecture operable by a team, not dependent on one person

WHAT I BRING TODAY

METHOD & ARCHITECTURE

- An empirical library of limits, refreshed in each language and application.
- Reproducible business models that can accelerate both development and execution.
- Hot-path discipline: follow successive causes without blocking business evolution.
- Two modes: start from a proven model when the problem is familiar; build quickly and then measure when the problem is new.

AI AS A MULTIPLIER, NOT AS THE DESIGNER

AI accelerates research, implementation, testing, documentation, and the replication of already validated models. It does not independently detect future breaking points or assume responsibility for the choice. The design, criteria, logging, and validation remain mine.

Across different systems and engagements, I delivered substantial performance gains, reaching orders of magnitude in selected cases. Depending on the bottleneck, I used parallelized data access, dictionary- and array-based lookup, Aho-Corasick multi-pattern matching, and distributed execution with Azure Batch. Sophistication is never the objective, only a measured means.

HOW I KEEP THE PUBLIC VERSION USEFUL AND SAFE

WHAT REMAINS	WHAT IS ANONYMIZED	WHAT CAN BE VALIDATED
Publicly named past employers, my roles, dates, method, technical breadth, ownership model, and the way I measure, simplify, and transfer systems.	The current organization's identity, commercial data, internal system names, detailed architecture, exact metrics, and sensitive implementation details.	The same methods can be demonstrated on synthetic data, historical work, or a controlled technical exercise without exposing a client's environment.

I DO NOT CLAIM TO HAVE CREATED THE GROWTH OF THE ORGANIZATIONS I SERVED.

I claim something more precise: I contributed platforms and working methods that could absorb change without becoming the next constraint, then transferred them so the organization depended on the architecture rather than on my presence.

I measure the limits before the business reaches them.

Public profile. The current engagement is anonymized: organization identity, commercial data, internal system names, exact metrics, and sensitive implementation details are omitted. Historical figures refer to earlier work outside that engagement.