How MMS Company Shapes "Queue Intelligence" During Day One

1. Introduction

This paper documents the work MMS Company performed for Queue Intelligence (Q.I.), a startup developing a real-time adaptive traffic signal coordination system. The engagement was a product discovery and feasibility assessment — the kind of work that sits between a founder's technical specification and a fundable, buildable product.

The case is worth documenting because it illustrates a gap in how deep-tech startups typically get assessed. Strategy consulting firms evaluate markets. Engineering firms build software. Neither is well-suited to the intermediate question that founders of algorithm-driven products face first: "*is the math right, and if not, what specifically needs to change?*"

This paper describes what that intermediate work looks like in practice, using the Q.I. engagement as a concrete example.

2. Context: What Queue Intelligence Had Built

Q.I. is a patent pending, adaptive traffic signal coordination system designed to address a well-documented infrastructure problem. Many signalized intersections in the United States operate on static or time-of-day timing plans that degrade when real-world conditions diverge from the engineering assumptions that produced them. Adaptive signal control — adjusting phase durations in real time based on observed conditions — is not a new idea. Several incumbent systems have been deployed for decades.

Interestingly, what Q.I. brought to the problem was a different architectural approach. The founder had developed a mathematical pipeline that reads network conditions, processes them through a multi-layered computation and decision architecture, and produces bounded timing adjustments that layer on top of an existing signal plan. The system does not replace existing infrastructure but rather augments it. The approach is deterministic, explainable at every step, and designed to operate within conservative safety bounds.

At the time of engagement, Q.I. existed as a detailed technical specification spanning hundreds of pages, a provisional patent application, and a set of metric reference documents with formula structures and worked examples. No simulation had been run. No software had been built. No product had been defined for market entry. The founder needed to know whether the system's mathematical foundation was sound before investing in any of those next steps.

3. The Engagement Structure

The system existed as a mathematical specification. Before any investment in software development, hiring, or market entry, the founder needed to know whether the specification held up under scrutiny — and if it didn't, where exactly it broke and what would be required to fix it.

That question sounds simple. Answering it required work across four distinct disciplines: mathematical analysis of the algorithm's internal logic, simulation engineering to test it against real-world conditions, product definition to translate the findings into something a municipality would actually buy, and market assessment to determine whether the competitive and regulatory landscape would receive it. These workstreams ran concurrently, with findings from each informing the others.

The seven phases below describe each in turn.

Phase 1: System Internalization

The first phase of work was learning the system at the depth required to assess it. This meant reading and cross-referencing the full specification corpus — not summaries or abstracts, but the actual mathematical references serving as the backbone for the platform.

The system's architecture is multi-layered. Sensor data is ingested, processed through a mathematical pipeline that produces a suite of analytical metrics, and resolved into a single timing adjustment output. Multiple decision subsystems evaluate the metrics and a resolution mechanism reconciles their competing assessments into the final output. A safety layer constrains the system's behavior to ensure the algorithm operates within acceptable bounds.

During internalization, we performed a full inventory reconciliation across source documents. This revealed a parameter that was missing from the system's own planning table of contents despite being fully specified in the source material and actively referenced across multiple decision subsystems. We corrected the canonical inventory, updated downstream counts that had propagated through planning documents, and flagged a small delta of legacy parameters.

This reconciliation work is representative of a pattern we see in deep-tech engagements: founders working on complex systems accumulate specification drift across documents over time. Catching these inconsistencies before they propagate into product requirements or investor materials is unglamorous but consequential work.

Phase 2: Mathematical Analysis

The original plan was to assess Q.I.'s mathematical soundness by analyzing each metric in the pipeline independently. In addition, we also needed to analyze data dependency, implementation complexity, and scalability risk. We began this work and then pivoted.

The pivot came from a realization about where risk was concentrated. Our original hypothesis was that risk would be distributed across the pipeline — isolated issues in individual components that could be addressed one by one. Upon further discovery, the critical problem sat at the top of the pyramid: the resolution formula. If that formula is structurally flawed, the health of everything feeding into it is secondary.

We worked through the resolution formula against the system's own reference scenarios and confirmed that it was producing outputs that ran counter to the system's intention. This is not an edge case, but a structural consequence of how the formula was designed.

This finding restructured the entire engagement. The companion report pivoted from an individual metrics audit to a system-level technical report led by the resolution mechanism and simulation evidence.

Following our diagnosis, we designed multiple alternative resolution formulas. Each addressed the identified flaw through a different mathematical approach, with different trade-off profiles. Each was specified to implementation-ready depth: full mathematical definitions, worked examples on the reference scenario, edge case analysis, invariant proofs, tuning parameters where applicable, and simulation implementation notes.

Phase 3: Simulation

Mathematical analysis can prove a formula is broken. However, it cannot prove a formula works well across the range of conditions the system will encounter in deployment.

In that spirit, we built a simulation engine that computed the full metric pipeline using real-world road network data sourced from open geospatial databases. The simulation covered multiple U.S. cities spanning the suburban-to-urban density spectrum, with varied intersection geometries and road segment lengths. Continuous scan intervals produced tens of thousands of computation cycles. All formula variants ran on identical inputs per cycle, enabling direct cross-formula comparison with full diagnostic output.

The simulation confirmed the mathematical findings empirically and produced a clear ranking among the candidates. It also surfaced patterns in how the system's internal subsystems interact — patterns that were invisible during mathematical analysis and that no formula change alone could address. These findings expanded the scope of what the next development iteration needs to tackle.

The simulation engine, the data processing pipeline, and the analysis methodology are all transferable deliverables. The founder's team can run the next iteration of testing independently.

Phase 4: Safety Architecture

A system that modifies traffic signal timing on live roads operates in a safety-critical context. Regardless of how well the optimization algorithm performs, a deployment-ready product requires a safety layer that can override, constrain, or flag the algorithm's output under elevated-risk conditions.

We designed a multi-tier safety architecture with graduated levels of intervention — from passive monitoring through active constraint to full override. Each tier has defined trigger conditions, a specified interaction with the system's computation pipeline, and clear boundaries for when it activates and when it steps back. The architecture is designed so that safety decisions are never downstream of optimization decisions — they sit above them.

Phase 5: Product Definition

The technical analysis established what the system can and cannot do. Product definition translates that into what should be built first.

We defined a minimum marketable product (MMP) — deliberately not a minimum viable product. The distinction is important in regulated infrastructure markets. The end buyer is not an early adopter willing to tolerate rough edges in exchange for innovation. It is a public agency with professional obligations, liability exposure, and zero tolerance for systems they cannot monitor, control, and reverse. An MVP that proves the technology works but lacks these operational assurances is not procurable.

The MMP definition included feature cards written as development-ready backlog items with acceptance criteria, the deployment scope, an explicit list of what is deferred to later phases, revenue model options, and measurable success criteria. We also mapped the procurement pathway as municipalities purchase signal technology through specific mechanisms.

Phase 6: Market and Competitive Assessment

We benchmarked Q.I. against every significant incumbent adaptive signal control system deployed globally. The assessment was structured to be sharp and useful in answering the following important question: "***Where does Q.I. have genuine architectural differentiation, and where are incumbents ahead?***"

The competitive landscape included market sizing (TAM/SAM/SOM), comparable company trajectories, and an assessment of the gaps between Q.I.'s current state and competitive parity on dimensions like deployment track record, procurement relationships, and integration ecosystems.

We also developed a market feedback strategy with specific target personas, core validation questions organized by persona, market feedback mechanisms, and a filtering framework for incorporating feedback without allowing scope creep. The strategy was designed to be executable immediately — not a theoretical plan for someday.

Phase 7: Gap Analysis and Roadmap

The final analytical phase synthesized findings from all preceding work into a single inventory of gaps between the system's current state and a deployable product.

Gaps were categorized across four dimensions: technical, product, market and commercial, and organizational. Each gap was classified by severity, assigned a path to close, and plotted on a priority matrix. The matrix serves as the organizing structure for everything downstream.

4. The Development Methodology

During the engagement, we designed and operated a structured iterative development loop: specification, simulation, data collection, human analysis and diagnosis, AI-assisted improvement, and human review and approval. The loop includes two mandatory decision gates. The first **diagnosis gate** ensures the right problem is being framed before solutions are proposed. The second **approval gate** ensures proposed solutions are sound and scoped before entering simulation. Internally, we regard the process as "**Whitebox AI**" — preserving every decision's explainability that Q.I. requires.

We ran this loop through multiple complete iterations during the engagement, producing progressively refined candidates at each pass. The methodology is designed so that each iteration builds on the precedent of previous ones — every documented decision becomes input to the next cycle. The cost per iteration decreases as the precedent base grows.

The founder's team inherits this methodology along with the simulation engine, and the formula specifications.

5. What This Type of Work Requires

The Q.I. engagement required simultaneous competence in **mathematical analysis** (tracing formulas, proving flaws, designing alternatives), **simulation engineering** (building computation engines against real-world data), **product definition** (feature cards, deployment scoping, procurement pathway mapping, compliance with regulations), and **market assessment** (competitive benchmarking, persona development, feedback strategy).

This combination is unusual. Strategy consulting firms operate at the market and competitive assessment level but do not go inside the mathematics. Engineering firms build software but do not define products or assess markets. Academic researchers analyze algorithms but do not produce development-ready specifications or map procurement pathways.

The work sits in between — and that space is where founders of algorithm-driven products most need support during the earliest phase of product development, before they have a team, before they have funding, and before the system has been tested against anything other than its own specification.

6. Outcome

The engagement produced two primary deliverables totaling over 150 pages:

1. A **main report** covering system understanding, development workflow, technical feasibility, competitive landscape with market sizing, regulatory and IP considerations, gap analysis, MMP definition, market feedback strategy, risk scenarios, and a go/no-go recommendation.
2. A **technical report** providing simulation evidence, formula-by-formula mathematical analysis, cross-formula comparison across real-world data, input layer structural findings, and safety constraint layer design.

The engagement gave the founder confidence in the system's architecture — and equally important, a precise map of where the critical flaws sit, why they exist, and what it takes to resolve them. Both are necessary. Confidence without specificity produces false starts. Specificity without confidence produces paralysis. The founder received both: a clear verdict on what is sound, a bounded diagnosis of what is not, and implementation-ready alternatives for closing the gap.

The founder entered the engagement with a mathematical specification. They exited with a validated architecture, a diagnosed flaw with alternatives in hand, a simulation engine, a defined product, a mapped market, a gap-prioritized roadmap, and a transferable development methodology.

7. About MMS

MMS Company is a product discovery, technical advisory, and software development firm. We work with founders building technically complex products where the core intellectual property is mathematical, algorithmic, or architectural.

Our team is senior and spans the end-to-end software development lifecycle. Strategy and engineering sit under the same roof, with domain specialists brought in per engagement. There is no junior staff rotation. The people who assess the system are the same people who design the alternatives and build the software. Over the past year, MMS has evolved into an AI-native software studio — we use AI as a core part of how we understand problems and develop solutions.

Most of our client relationships begin at discovery and continue through the build. Over the past six years, we have become the long-term technical partner for clients across multiple verticals including Artificial Intelligence, Traditional and Modern Finance, Food and Beverage, Human Capital Management, and most recently Road Traffic Management — working with organizations ranging from startups to enterprises.

The Q.I. engagement is representative of how these relationships start: we enter at the point where a founder has built something sophisticated and needs to know whether it holds up, what needs to change, and what it takes to get to market. We tend to stay and build things that last.