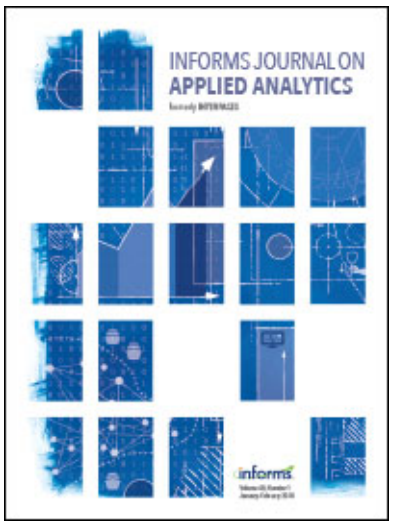

INFORMS Journal on Applied Analytics

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Fifth Column: On the Care and Feeding of an Analytical System

Michael F. Gorman

To cite this article:
Michael F. Gorman (2026) The Fifth Column: On the Care and Feeding of an Analytical System. INFORMS Journal on Applied Analytics

Published online in Articles in Advance 28 Jul 2026

. https://doi.org/10.1287/inte.2026.0338

https://pubsonline.informs.org/journal/ijaa

INFORMS JOURNAL ON APPLIED ANALYTICS
Articles in Advance, pp. 1–5
ISSN 2644-0865 (print), ISSN 2644-0873 (online)

The Fifth Column: On the Care and Feeding of an Analytical System

Michael F. Gorman[a]

[a] Department of MIS, Operations and Decision Sciences, University of Dayton School of Business Administration, Dayton, Ohio 45469
michael.gorman@udayton.edu (M. F. Gorman)
https://orcid.org/0000-0003-1295-8147 (M. F. Gorman)

Received: March 27, 2026
Revised: May 28, 2026
Accepted: July 7, 2026
Published Online in Articles in Advance: July 28, 2026

https://doi.org/10.1287/inte.2026.0338

Abstract. Analytical decision support systems in operations research (OR) are disproportionately represented in the literature by their successes, a publication bias that leaves practitioners ill-equipped to anticipate and prevent the conditions under which working systems fail. This paper presents a practitioner's retrospective account of a decision support system that achieved substantial organizational value before ultimately collapsing under a combination of assumption inaccuracy, parameter drift, accumulated complexity, eroded institutional knowledge, and insufficient maintenance infrastructure. Drawing on direct experience with both the system's success and its failure, the author articulates six operational lessons: the necessity of sustained system ownership and regular review cadence; the need for systematic monitoring of forecast accuracy and assumption validity; the hidden costs of incremental feature expansion on system transparency and user trust; the organizational importance of ensuring users can critically evaluate system recommendations; the dangers of implicit assumptions that fail silently under shifting conditions; and the fundamental misframing of analytical projects as having a terminal completion point rather than an ongoing maintenance lifecycle. Collectively, these lessons argue that the dominant challenge in applied OR is not system construction but system stewardship; a discipline the field's publication norms render largely invisible. The paper is offered as a corrective contribution to a literature that systematically underrepresents instructive failure.

History: This paper was refereed.

Keywords: profession • comments on • model maintenance

Introduction

I have had an instructive experience in my professional life that might have value to be shared.

In the first chapter, I built a forecasting and optimization system for a client organization that worked beautifully. The system was clean, purposeful, and fit the organization and its processes perfectly. The users trusted it. The recommendations it produced were intelligible and valuable. Management conducted a formal audit and found savings of more than 20 times the development cost in the first year of operation. I gave talks. People asked how I did it. I told them.

In the second chapter, the company was facing a recession with demand falling across the board. I had been released by the company months earlier for budget reasons brought on by the recession, and internally, no one knew why it was doing what it was doing. The same system, heavily enhanced by then, decorated with features added over the years of success, began producing recommendations that struck the people using it as wrong. In a difficult business environment, with pressure high and tolerance for error low, confidence in the system collapsed quickly. The system was shut down. The organization reverted to the rules of thumb it had used before I arrived.

The mathematics did not change between chapter 1 and chapter 2. The fundamental approach did not change. The problem the system was designed to solve did not change. What changed was the world around the system and the system's inability to respond to that world. What also changed, in ways I will describe, was the system itself: quietly, incrementally, in the direction of more features, more parameters, more sophistication, and less of the elegant simplicity that had made it work in the first place.

It may be useful, before proceeding, to observe that analytical decision support systems tend to be created for one of two broad reasons. The first is to plug a leak: to address a problem the organization has identified and is actively suffering from and whose solution the system quietly provides, day after day, in ways that gradually fade from institutional memory. The second is to realize an opportunity: to generate value from a capability the organization had not previously

possessed or envisioned, and that becomes part of the landscape so naturally that people forget it had to be invented. In this case, it was both; a change in the structure of asset ownership required a different way of thinking that led to a need for automation and optimization. In both cases, the same hazard arises as time passes and leadership turns over. New leaders, unfamiliar with the history, may remove the plug without knowing the leak exists. Or they may look at a system generating steady returns and see only a cost to be managed rather than a capability to be maintained. The lessons that follow are directed at technical practitioners, but the underlying argument is organizational: The value of an analytical system is inseparable from the institutional knowledge of why it exists.

I have spent considerable time thinking about what I should have done differently. What follows is the result of that thinking. I have stripped out the industry context deliberately. Context is what allows people to convince themselves that a lesson applies to someone else's situation. These lessons apply to any analytical system in any organization. They applied to mine. They will apply to yours.

This kind of paper is not the norm. The operations research (OR) literature is, by and large, a literature of success. The journals are full of systems that worked, competitions that celebrate implementations that delivered, and case studies in which the protagonist identified an opportunity, built a system, and collected a return on investment that would embarrass a venture capitalist.

There is a well-known phenomenon in academic publishing called the file drawer problem. Studies that find something interesting get submitted; studies that find nothing or fail get put in the file drawer. The result is a published literature that dramatically overstates the success rate of anything: pharmaceuticals, educational interventions, management techniques, and yes, OR models and analytical systems.

The OR literature is, in this respect, somewhat worse than average. It is not merely that failures go unpublished. It is that failures go *unpublished and unexamined*. The practitioners who built the failed systems move on. The organizations that funded them do not advertise the waste. Everyone has good reasons to stay quiet, and therefore, the collective wisdom of the field is assembled almost entirely from its best days while learning nothing from its mistakes.

Failed systems do not get written up. The reasons are not mysterious. Practitioners do not volunteer their failures. Organizations do not publicize wasted investments. Developers move on and attribute the failure to circumstances beyond their control, which is usually partly true and rarely the whole story.

Gene Woolsey was one of the few writers in these pages who consistently refused to pretend that everything worked. He was honest. What follows is an attempt to be honest about a project that I am proud of and that I also, eventually, failed to keep alive. Both things are true.

The Six Lessons

Lesson 1: Keep an Eye (or Two) on the System—And Keep Someone Accountable for It

A well-functioning system produces good recommendations. Good recommendations build confidence. Confidence, however, if taken too far, produces complacency. And complacency is the condition in which systems fail.

When a system is running well and requiring little attention, the people assigned to manage it tend to be reassigned to things that are requiring more attention. This is rational behavior at the individual and organizational level. It is catastrophic at the systems level. The system that has been working fine for two years has simply had two years to accumulate the conditions for an unanticipated failure. The absence of recent problems is not evidence that problems are not lurking.

In this case, the system ran without interruption for more than two years. During the early period of active development, ownership was clear; there was close involvement with both the system and users, internal resources were engaged and knowledgeable, and problems were caught and addressed quickly. Although fledgling, the system was alive in an organizational sense. As it matured and stabilized, that attention dispersed. Resources follow problems; the system had "no" problems. The people responsible for it grew confident in it, and reasonably so, based on its track record. Monitoring became less frequent.

When the business environment shifted and the system began producing questionable recommendations, the organization discovered that the institutional knowledge needed to diagnose and fix the problem had never been fully developed, and what had existed was gone. It was like a homeowner discovering structural damage to their home: years of slow deterioration, now requiring immediate and expensive repair, and no time to fix it. The system was an orphan. Orphaned systems do not get fixed. They are left running past the point at which their recommendations are trustworthy then get replaced or abandoned.

The organizational consequence of an orphaned system is more serious than the technical one. When a system is abandoned, the organization does not simply lose a tool; it loses the institutional memory of the problem the tool was solving or the opportunity it was capturing. New leadership, encountering a line item for system maintenance with no visible corresponding output, may reasonably conclude that the

expenditure is unnecessary. Plugs are like that; the boat is not leaking. Maintaining institutional memory of the system's purpose and value is essential to keeping it healthy and operational.

No system, regardless of how well it has performed, can be put on autopilot indefinitely. Pilots who have not had an incident in five years still fly with checklists. Surgeons who have not had a complication in a year still follow preoperative protocols. The frequency of past problems does not determine the appropriate level of ongoing vigilance. The cost of future failure does.

The rule: Vigilance is not a response to problems, it is the reason there are fewer problems. Establish a regular system review cadence and hold to it even when everything seems fine. *Especially* when everything seems fine. Assign a system owner before the system goes into production. This is an accountable assignment, not a nominal one. The system owner monitors performance regularly, maintains familiarity with the system's internals, serves as the first point of contact when users raise concerns, and has a standing relationship with the developer for issues that exceed internal capacity. Selfishly, a retainer with the original developer (me) costs a fraction of what it costs to rebuild trust after a visible failure, and it is far less expensive than rebuilding the system from scratch, or worse, reverting back to old, ineffective decision rules and intuition.

Lesson 2: A System Trained on Yesterday's World Will Be Challenged Today

Every statistical model is a bet that the past is a useful guide to the future; every optimization model parameter is based on historical experience. This was the case in this system. This bet is usually reasonable but not always. And there is no mechanism built into most systems that causes them to announce when the bet has stopped paying off.

The system I built was calibrated on several years of relatively stable historical demand and cost data. When the business environment abruptly shifted, the system's forecasts continued to reflect a world that was no longer there. Demand patterns that had been consistent for years changed. The historical error distributions that the optimization model used to allocate capacity uncertainty were no longer symmetric; we were now consistently over forecasting, but no one was monitoring the forecast. The model was based on an implicit assumption that capacity was scarce, and that was no longer the case. Contracts were renegotiated, changing cost and incentive structures. The system was optimizing a profit function that no longer existed and overvaluing production capacity. The same can occur when upstream data collection mechanisms change; the interpretation of the data must change. The model's underpinnings were wrong and so were its recommendations.

The rule: Build monitoring into the system from the beginning. Track forecast accuracy over time. Set thresholds at which automated alerts fire when systematic bias is detected. If the statistical model has been consistently overforecasting for three weeks running, someone should know that before the end of week 3. Metrics on optimization model allocation should reveal failed recommendations immediately. The failure to avoid is not the system that breaks visibly. It is the system that degrades invisibly while everyone assumes it is still working because nobody has checked.

Lesson 3: What Gets Added in the Name of Improvement Often Diminishes What Was Working

This is the lesson I find most difficult to convey, because it runs against the instincts of anyone who cares about doing good work. The instinct is to improve things. The reality is that improvement, applied without discipline, can silently dismantle the qualities that made a system worth improving.

The system in its first year was purposefully simple. The design did not strive for perfection, intentionally glossing over nuances to get reliable answers. Users could explain its recommendations. When it produced something unexpected, the reasoning was traceable. This was not an accident. It was the result of deliberate choices early in the design, when there was still pressure to keep things simple to do less and do it clearly rather than more and do it obscurely.

Over the following two years, the system was enhanced. Each enhancement was individually justifiable. A module for tracking performance against strategic targets. More granular segmentation of the data. More precise modeling of incentive structures. Finer geographic breakdowns. Each addition improved fidelity in some narrow technical sense. None of them, in isolation, seemed like a problem.

Collectively, they transformed a system whose recommendations users could evaluate into a system whose recommendations users had to accept on faith. When the system said to do something that seemed wrong, there was no longer a clear path from the output back to the reasoning. It had become, as one internal manager later described it to me, a black box that everyone trusted until the day they stopped trusting it entirely.

The transition from trusted tool to black box is not dramatic. It happens incrementally, in the same meetings where the enhancements are approved, each of which seems reasonable. It is visible only in retrospect, when someone asks why the system is recommending

what it is recommending and nobody in the room can give a satisfactory answer.

The rule: Every parameter added to a working system is a liability. Not sometimes, every time. The question is not whether a new feature adds something, because it probably does. The question is whether what it adds is worth what it costs: the complexity of the logic, the data it requires, the maintenance it demands, the intelligibility it erodes, and the fragility it introduces. Before adding anything to a working system, require it to prove that case. If the proof is not clear, the enhancement does not go in. A system that does one thing well is more durable than a system that does many things adequately and nothing transparently.

Lesson 4: The People Using the System Must Be Able to Tell When It Is Wrong

This lesson is the practical consequence of the previous one, and it deserves to stand on its own because it is the place where system failures most often become organizational failures.

Decision support systems support decisions made by people. Those people carry knowledge, experience, and judgment that no model possesses. When the system and the practitioner disagree, one of them is right. Sometimes it is the system. Sometimes it is the practitioner. The only way to know which is for the practitioner to have enough understanding of the system's reasoning to evaluate the disagreement.

When users cannot evaluate a recommendation, they often take one of two paths, neither one good. The first is that they follow recommendations blindly, in which case the system's errors become operational errors. The second is that they override the system whenever its recommendation seems counterintuitive, which happens more frequently as the system grows more complex and its reasoning grows more opaque. At the point where overrides are frequent and unexplained, the system has become useless.

That is why transparency and clarity are critical. A user who does not understand the system knows only that something feels off. They cannot say what. If the system is wrong, or even seems so, users lose trust.

A user who understands the modeling logic can explain why a recommendation seems wrong. That user is a partner. They can help you find the flaw. But only if the modeling logic is within their reach.

The rule: Invest in transparency; ensure that users understand the system's logic well enough to engage with its recommendations critically. This is a design goal, not an afterthought. It should shape what the system does, how the outputs are presented, and what training and documentation support the system. A system that its users cannot understand is a system that its users will eventually abandon.

Lesson 5: Beware of Implicit Assumptions

Every model rests on assumptions. Some of those assumptions will eventually be violated. The critical question is how the system behaves when they are.

A system designed with this in mind produces usable recommendations when its assumptions are partially violated and produces clear warnings when the violations are severe enough to make the recommendations unreliable. It knows what it does not know. It communicates uncertainty honestly. It fails loudly enough that someone investigates before the damage becomes irreversible.

A system not designed with this in mind continues to produce confident-looking outputs after its foundation has shifted. The implicit assumption of scarce resources was violated, and the system had no way of knowing it. Nothing in the model results indicated that the inputs have become stale or the assumptions invalid. Decisions are made on the basis of those numbers. The decisions are wrong. Eventually something undeniable surfaces, and by then the postmortem reveals months of quiet malfunction that went undetected because the system had no way of flagging it and nobody was looking closely enough to see it independently.

The rule: Before the system goes into production, walk through its major assumptions and ask the following: What happens to the outputs if this assumption is wrong by 10%? By 50%? By the most extreme plausible amount? Build sanity checks. Bound the outputs and trigger alerts when bounds are exceeded. Give users a visible mechanism for flagging recommendations that seem inconsistent with their experience. A system that expresses its own uncertainty appropriately is more trustworthy, over time, than one that is always confident. Confidence without epistemic honesty is not a virtue. It is a liability waiting to mature.

Lesson 6: The Project Is Never Finished

This is the lesson that practitioners most reliably resist, because it is the most inconvenient.

There is enormous pressure, in any project, to reach the point at which the work is done. Budgets have end dates. Contracts have deliverables. Developers have other clients. Organizations want to move on. Everyone involved has a strong incentive to declare victory, close the books, and stop spending money.

A decision support system declared finished is not a finished decision support system. It is a system that has entered the phase of its existence most likely to determine whether it survives. The environment it operates in will continue to change. The data it depends on will continue to drift. The business processes it supports will continue to evolve. The system, if it is not maintained and updated, will gradually

become a system designed for an organization that no longer exists, producing recommendations calibrated to conditions that have already passed.

The right answer, when a client asks when the project will be done, is as follows: It will not be done, but the first version will be operational by a specific date, at which point we transition from building to maintaining. Maintenance is not a lesser activity than building. It is a different activity that requires its own skills, its own budget, and its own sustained commitment. It is also the activity that determines whether the investment in building was worthwhile.

The rule: Budget for ongoing support before the project begins. The standard expectation in analytical software is 15%–20% of development cost per year for maintenance and improvement. There is no reason to believe that OR models are exempt from this. An organization not prepared to make this commitment should think carefully about whether it is actually prepared to implement a decision support system, or whether it is prepared to implement one for a year or two and then discard it. Both choices can be rational; what is not rational is to make the second choice while pretending it is the first.

A Final Word

The painful truth about the failure I have described is that it was preventable. The warning signs were visible in advance to anyone who knew where to look and was looking. The forecasts were drifting. Assumptions were violated. The system had grown more complex than anyone fully understood. The institutional ownership had quietly evaporated. Any one of these conditions, addressed early, could have been corrected.

Instead, each was left unaddressed through the entirely human tendency to trust what has been working and direct attention toward what has not. The system had been working. It deserved the trust it had earned. What it also deserved, and did not receive, was the ongoing attention that would have preserved the conditions that made it trustworthy.

The glamour in this field is in the building. The papers get written about the systems that went into production and returned 20 times their investment in the first year. Those stories are true and worth telling. But the story that comes after is also worth telling. The story of what it takes to keep a working system healthy, of what happens when that effort is not made, and of what a practitioner learns when a system he is proud of is turned off.

System builders would do well to take this seriously at the institutional level, not merely the technical one. The systems we build are often invisible precisely because they are working. The system builder's responsibility does not end with delivery. It extends to ensuring that the organization understands, at every level of leadership turnover, what it has and what would be lost without it.

The wisdom is in the tending. I wish I had understood that sooner.

Acknowledgments

This column owes its existence, its format, and its philosophy to Gene Woolsey, whose Fifth Column pieces demonstrated there is more than just the modeling when it comes to implementing a system.

Michael F. Gorman is Niehaus chair in business analytics and operations management at University of Dayton School of Business. He focuses on practical applications of analytics, optimization, and OR to transportation, logistics, and operations problems. He is the former editor-in-chief of *INFORMS Journal on Applied Analytics* and founding president of the INFORMS Analytics Society. He was a finalist for the Edelman and Wagner Prizes and received INFORMS' Prize for the Teaching of OR/MS Practice.