

Programmable Strategy: When Strategy Becomes Code

Executive Summary

Yves Doz¹ and Soumitra Dutta²

In July 2024, a routine software update by CrowdStrike triggered one of the largest IT disruptions in recent history. Airlines grounded flights, hospitals postponed procedures, and businesses stalled across the globe. Some organizations were paralyzed for days. Others recovered within hours, because they had pre-built failover systems, isolation mechanisms, and automated rollback procedures. Their resilience was not improvised under pressure. It had been designed in advance and embedded into their systems.

A similar lesson emerged from Air Canada's AI chatbot dispute. The system confidently described a refund policy that did not exist, and the airline was held legally responsible. The underlying strategy — to improve service and reduce costs through automation — was clear. What failed was its translation. The system lacked guardrails, escalation logic, and limits. It behaved exactly as it had been built. Strategy did not fail because it was wrong. It failed because it was not properly embedded.

Together, these cases reveal a shift that is reshaping competition across every industry. Increasingly, what determines outcomes is not the strategy itself but how completely that strategy has been translated into the behavior of systems. The organizations pulling ahead are those that do not merely articulate strategy. They program it. They embed it into their infrastructure.

From Plan to System

For most of the twentieth century, strategy was a plan. Leaders analyzed markets, made choices about where to compete, and communicated direction. Execution was primarily human — mediated by managers who interpreted goals and made decisions.

This model is breaking down. Today, a growing share of consequential decisions is made by systems — pricing algorithms, lending models, routing engines, recommendation platforms, risk controls. These decisions happen continuously, at machine speed, and at a scale no human organization can match through coordination alone. A bank processes millions of credit decisions daily. A retailer adjusts prices continuously across millions of products. A logistics firm reroutes thousands of shipments in real time.

In this environment, strategy cannot be communicated and then left to interpretation. It must be embedded — translated into the rules, limits, decision logic, and infrastructure that govern how systems behave. When this translation is complete, strategy becomes what the organization does by default. When it is incomplete, the gap between intent and system behavior is where strategy fails. This is what the paper calls programmable strategy.

¹ Solvay Chaired Professor of Technology Innovation Emeritus, INSEAD, France

² Director, Portulans Institute, Washington DC and Former Founding Dean, SC Johnson College of Business, Cornell University, New York, USA

Strategy as Compilation

The paper borrows a concept from software engineering: compilation. When engineers write software, they work in high-level language and then translate it into executable instructions. Ambiguity is removed. Assumptions are formalized. By the time the system runs, there is no room for misinterpretation.

Strategy is increasingly following the same path — from intent to executable behavior through a process of progressive translation. The paper identifies four layers through which this compilation unfolds.

Intent is where leaders define strategic priorities and the trade-offs the organization is willing to make: cost versus customer experience, speed versus accuracy, automation versus human oversight. This remains the domain of leadership judgment.

Control is where intent becomes boundary. Abstract priorities are converted into enforceable limits: acceptable risk thresholds, prohibited outcomes, escalation conditions, compliance requirements, and mandatory human-review triggers. These are not afterthought checks. They are the translation of intent into constraint.

Decision logic is where strategy is enacted continuously. Systems evaluate inputs, route decisions, approve or reject actions, and determine when to escalate. A customer query may be resolved automatically, sent to a specialist, or flagged for legal review. A transaction may be approved, challenged, or blocked. These automated decisions are not merely operational. They are strategy in motion.

Infrastructure is where strategy becomes durable. System architecture enforces strategic priorities through redundancy, modularity, monitoring, rollback, and failure containment. Infrastructure is what makes strategy hold under pressure — not through managerial intervention, but through design.

Together, these four layers form the Programmable Strategy Stack. Strategy begins as intent. It becomes executable through limits, decision logic, and infrastructure. When the translation across all four layers is complete, the organization reliably reflects its strategy across millions of decisions. When it is incomplete, intent and execution diverge — and that divergence is where competitive damage occurs.

What This Looks Like in Practice

Across industries, this shift is already visible. Ocado operates its fulfilment centers through networks of robots coordinated in real time. Routing adjusts dynamically. Congestion is managed automatically. Anomalies trigger immediate response. No manager directs these actions. Yet the system consistently reflects the organization's strategic priorities — speed, accuracy, and safety.

Netflix deploys changes continuously but with embedded safeguards. Features are released incrementally, monitored in real time, and rolled back automatically if performance thresholds are breached. Capital One embeds risk controls directly into the model deployment process: non-compliant models are blocked before they reach customers. Stripe builds reliability into its architecture so that payments behave predictably even under failure conditions.

In each case, strategy is not expressed through directives or plans alone. It is expressed through programming — through the rules, limits, and system design that shape behavior continuously. The Air Canada case shows the consequence of doing this badly: strategy failed not because intent was unclear, but because the translation into system behavior was incomplete.

The New Role of Leadership

This transformation does not eliminate leadership. It redefines it. Leaders can no longer oversee the millions of decisions their systems make every day. The questions they must ask change accordingly. In addition to asking what the organization should do, leaders must ask how systems should behave, where limits apply, when machines should act autonomously and when humans must intervene, how failure should be handled, and what assumptions have been embedded in the system — and how leadership will know when those assumptions no longer hold.

Human intervention is no longer the default mode of decision-making in an AI-enabled firm. It is triggered — by design — when systems encounter ambiguity, conflict, or boundaries they cannot resolve. Leadership attention shifts from directing decisions to designing the architecture within which decisions occur.

Speed Without Fragility

A common concern is that embedding strategy into systems creates rigidity. The evidence suggests the opposite. The fastest-moving organizations are often those with the strongest embedded controls. Constraints do not limit speed — they enable it.

Speed is limited not by the ability to act, but by the cost of being wrong. When failures are costly and uncontained, organizations slow down out of caution. Programmable organizations solve this by making failure safe — isolated, detectable, and reversible. Netflix's automatic rollback, Amazon's failure isolation, and Capital One's pre-deployment blocking all create the conditions for confident acceleration. Speed, in this context, is not the absence of control. It is the product of control done well.

The Central Insight

Strategy has always been about translating intent into action. In earlier eras, that translation happened through plans, structures, and managerial judgment. In AI-enabled environments, as more decisions are made by systems operating continuously at scale, that translation becomes programmable.

When the translation is complete — intent becomes control, control becomes logic, logic becomes infrastructure — strategy becomes inevitable. It does not depend on constant oversight or perfect interpretation. It is expressed through the default behavior of the system. The test of strategy is no longer whether it is compelling. It is whether it has been programmed to hold.

Disclosure: AI was used for research and copy editing of this article