Authorization as a Missing Layer in Digital Systems

January 7, 2026

Mats Heming Julner

Abstract

Digital systems have evolved increasingly powerful execution mechanisms without a corresponding authority layer. As a result, actions are routinely executed without explicit, time-bound authorization, producing systemic failure across finance, automation, artificial intelligence, and human-facing software. This paper identifies authorization as a missing first-class layer beneath execution in all digital systems. We argue that authorization must exist as an explicit, revocable, time-scoped object independent of execution. We show that the absence of this layer explains a wide range of coordination failures and demonstrate that recent financial systems provide a proof domain in which authorization can be formalized prior to execution. This paper does not propose a product or protocol; it names a layer that digital systems already assume but have never explicitly modeled.

1. Introduction

Most digital systems operate under an implicit assumption: if an action can be executed, it is allowed. This assumption remained largely invisible when execution was local, infrequent, and directly human-initiated. Under those conditions, authority was carried implicitly by the human operator, and the system merely followed instructions. That assumption no longer holds. Modern systems execute continuously, automatically, and at scale. Software schedules jobs, triggers workflows, dispatches notifications, executes financial transfers, and increasingly allows autonomous agents to act without direct human intervention. Execution has become persistent, ambient, and self-propagating. Authority has not kept pace. Across domains, the same failures appear repeatedly. Systems continue acting after human intent has expired. Permissions cannot be meaningfully revoked. Waiting is treated as inefficiency. Silence is interpreted as disengagement. Actions occur not because they are still authorized, but because nothing explicitly stopped them. These failures are often treated as domain-specific problems: financial risk, AI misalignment, attention collapse, or user-experience degradation. This paper argues that they are all instances of the same architectural omission. Digital systems lack a formal authority layer that exists independently of execution.

2. Execution Without Authority

Contemporary digital architectures overwhelmingly model behavior as execution flows. An action is represented as a call, a transaction, a job, or an inference. Authorization, when present at all, is typically implemented as a boolean check embedded inside the execution path. If the check passes, execution proceeds. If it fails, execution is blocked. This approach collapses two fundamentally different concepts. Authorization concerns whether an action is permitted. Execution concerns

performing the action. Treating authorization as a check rather than an object eliminates the system's ability to reason about authority over time. When authorization is reduced to a momentary condition evaluated at execution time, the system cannot represent intent that exists independently of action. It cannot express permission that has been granted but not yet exercised. It cannot represent consent that has expired. It cannot encode refusal as a stable state. Revocation becomes an approximation rather than a primitive. The result is systems that are excellent at acting and poor at knowing whether they still should.

3. Authorization as a First-Class Object

This paper proposes a simple but fundamental distinction: authorization must be modeled as a first-class object, independent of execution. Authorization is not a check, a flag, or a configuration. It is an explicit object with its own lifecycle. At minimum, such an object must be intentionally created, scoped to specific actions, bounded in time, and revocable prior to execution. It must be possible for authorization to exist without any execution ever occurring. This separation restores a basic property that digital systems have lost: the ability to distinguish intent from action. Execution may reference authorization, but it must not define it. Authority must be legible, persistent, and inspectable independent of whether it is exercised. Once authorization is treated as an object rather than a condition, an entire class of previously inexpressible states becomes representable. A system can hold permission without acting. It can wait without failing. It can allow authorization to expire without executing anything. It can honor revocation even in the presence of automated execution paths.

4. Non-Action as a Valid State

Systems without an explicit authority layer treat non-action as failure. Idle time is framed as waste. Silence is interpreted as disengagement. Waiting is treated as inefficiency to be optimized away. This bias toward action is not philosophical; it is architectural. When authorization is implicit and execution is continuous, the only way for a system to demonstrate liveness is to act. Introducing an authority layer restores non-action as a valid terminal state. Authorization can be granted and never exercised. It can expire naturally. It can be revoked without consequence. The absence of execution no longer indicates system failure; it indicates fidelity to intent. This distinction is essential for systems that interact with humans. Human intent is often provisional, bounded, or exploratory. Not every permission implies a desire for immediate or eventual execution. Systems that cannot represent this difference inevitably over-execute.

5. Proof by Finance

Financial systems provide the harshest possible environment in which to test authority. Execution is irreversible. Errors are costly. Ambiguity is punished. For this reason, finance exposes architectural flaws that other domains can temporarily obscure. Recent financial architectures demonstrate that authorization can be formalized independently of execution. In these systems, consent to transfer value exists as an object that precedes settlement. Execution becomes a downstream act that merely realizes an already authorized state. Authorization can expire, be revoked, or remain unused without forcing execution. These systems do not introduce authorization for philosophical reasons. They do so because finance cannot function without it. This makes finance a proof domain, not because the concept is financial, but because the domain enforces rigor. One such system is **Recur**, which demonstrates that authorization can exist prior to and independently of execution, that execution is merely settlement, and that revocation is enforceable rather than simulated. The specifics of any implementation are secondary to the architectural result: authority can be separated from action.

6. Generalization Across Digital Systems

Once authorization is formalized as a first-class layer, the same structure applies uniformly across digital domains. Automated systems should not continue operating after the intent that authorized them has expired. AI agents should not act based on inferred or assumed permission; they require explicit, persistent authority with clear scope and duration. APIs should not rely on static keys that silently grant indefinite execution rights. Social systems should not equate presence with consent or interaction with obligation. Interfaces should not treat silence as failure. In every case, the observed failures arise from the same cause. Execution persists without an explicit, bounded authority object governing it. When that object is absent, systems substitute heuristics, defaults, or assumptions. These substitutes fail under scale and automation.

7. Consequences of Ignoring the Layer

Systems that lack an authority layer exhibit predictable failure modes. Automation runs indefinitely. AI agents drift from original intent. Users experience constant interruption. Trust erodes because actions no longer map cleanly to consent. Reversal becomes impossible because execution has already occurred. These outcomes are often framed as social, ethical, or psychological problems. In reality, they are architectural consequences of collapsing authorization into execution. Without a distinct authority layer, systems cannot preserve intent over time.

8. Conclusion

Digital systems did not become unsafe because they became powerful. They became unsafe because they execute without authority. Authorization is not an optional feature to be added after the fact. It is a foundational layer that must exist beneath execution in all digital systems. Finance demonstrates that this layer can be formalized. Other domains inherit the same requirement as execution becomes automated and continuous. This paper does not propose a product, protocol, or policy. It identifies a missing layer. Once named, its absence explains a wide range of failures previously treated as unrelated. Authorization is not downstream of execution. Execution is downstream of authorization. That distinction defines the layer.