

# Escaping the legacy black hole with **AI/works**<sup>TM</sup>

Bharani Subramaniam

**/thoughtworks**

Design. Engineering. AI.

<b>The legacy black hole</b>	<b>3</b>
<b>The modernization imperative</b>	<b>5</b>
<b>Legacy modernization through reconstruction</b>	<b>7</b>
<b>Blackbox legacy modernization in action</b>	<b>9</b>
<b>Start plotting your roadmap today</b>	<b>11</b>



## The legacy black hole

Every organization relies on systems that were once well understood, well maintained, and supported by the people who built them. Over time those conditions eroded, as documentation fell behind and key architects left. Operational patches accumulated until no one could map cause to effect with confidence. The system still runs, but the organization can no longer explain why. This is the defining characteristic of a legacy black-box system: predictable at the surface, opaque at its core.

This opacity is not benign. It accumulates quietly, like structural fatigue in a bridge. Each year the system becomes harder to maintain, harder to integrate and harder to modify safely. When enough opacity accumulates, the black box becomes something more dangerous: a legacy black hole. This is the point where complexity collapses in on itself, until escape begins to feel impossible. You aren't just dealing with limited visibility anymore; you're dealing with a system that actively resists understanding, absorbing effort without yielding clarity.

The telltale signs are there for those that look: the majority of IT projects fail to meet time and budget expectations. Where legacy systems are missing source code, either partially or fully, the projects take between two to four times to reach fruition, turning modest modernization into multi-year programs. These overruns aren't caused by incompetence, they result from information asymmetry: you cannot plan for how to deal with something that defies understanding.

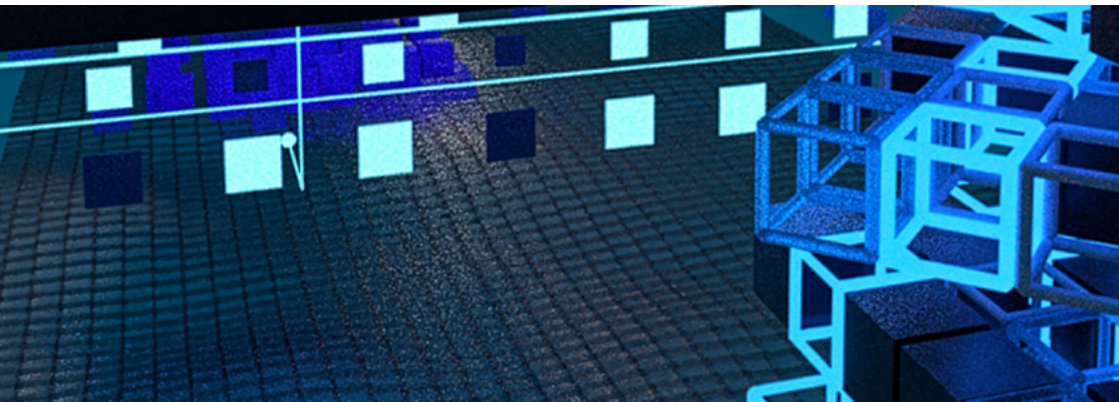
This legacy burden creates a strategic paradox: you have a system that is both too risky to touch and too important to ignore. Many organizations delay modernization to avoid disruption but delay compounds the very risk they fear.

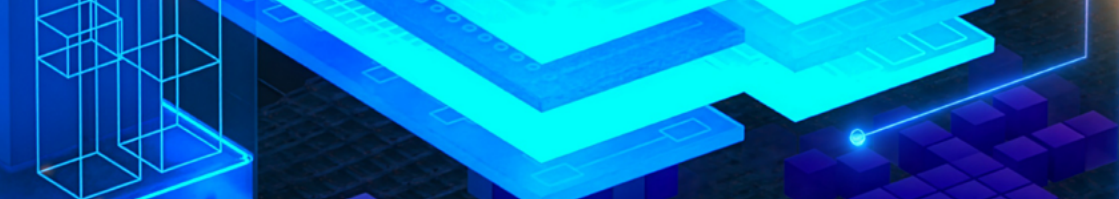


**In financial services alone, failure to modernize is projected to cost \$57 billion in lost revenue by 2028.**

This is the reality many CIOs inhabit: dependent on systems they cannot fully understand, lacking the institutional memory to safely evolve them and absorbing rising operational drag year after year. The impetus to act arrives not as a spark of innovation but as an unavoidable recognition: modernization is no longer optional. It has become a matter of operational resilience, regulatory compliance and, perhaps, competitive survival.

The first step is acknowledging the truth: your organization doesn't have a technology problem. It has a visibility problem. And visibility cannot be restored by wishful thinking, partial documentation or incremental patching. It requires a deliberate shift from assumptions to evidence.





## The modernization imperative

Organizations often talk about modernization as a strategic priority but treat it as a discretionary project. The data contradicts this. Legacy systems degrade performance, slow change velocity and create existential business risk. Modernization is not an upgrade; it is risk mitigation at an enterprise scale.

The operational burden is already well evidenced. Legacy environments consume disproportionate maintenance budgets, driven by scarce skills and brittle architectures. For organizations operating in competitive or regulated markets, this is not simply a technical limitation — it is a strategic handicap.

Security risk compounds the challenge. Older systems carry unpatched vulnerabilities that cannot be easily remediated because teams don't fully understand the internal logic. Regulations increasingly require provable controls, yet undocumented behaviors make compliance a guessing game. When a system's behavior is partly unknown, leaders cannot attest to its security posture with confidence.

Then there is the talent paradox: the people who understand the system are retiring or have already left. Modern engineers are not trained on COBOL, Fortran or proprietary vendor frameworks. What's more, they often lack sufficient understanding of the existing system: with no architecture decision records and little-to-no test coverage, there is no safety net for incremental feature development. The result? Studies show outdated systems are costing businesses hundreds of millions each year in lost productivity.

Legacy systems directly erode morale and retention, creating a feedback loop in which the system's obscurity expands as experienced staff exit.

And yet, modernization offers significant upside. With a coherent understanding of existing business logic, data flows and system dependencies, organizations can replatform, refactor or selectively replace systems with confidence. Transformation shifts from a high-stakes gamble to a structured evidence-driven program. Modern platforms improve agility, reduce operating cost and enable strategic initiatives that have been blocked for years.

This opportunity is only accessible when leaders confront a simple truth: modernization without visibility is reckless, but modernization with a blueprint is transformative.





## Legacy modernization through reconstruction

When source code is missing, incomplete or unusable, traditional modernization tactics break down. You cannot rewrite what you cannot inspect. The only viable strategy is reconstruction: building an architectural and behavioral map of the system based entirely on observable evidence.

This approach draws from digital forensics, reverse engineering and AI-augmented analysis. It uses multiple lenses to triangulate truth:



### **User interface and workflow analysis.**

The UI reveals operational behavior: the paths users follow, the data they manipulate and the implicit assumptions encoded in each interaction. While incomplete, this perspective anchors the system in real-world usage.



**Data lineage and database analysis.** Data does not lie. Whether systems are thick clients, with two-tier architectures or more modern three-tier designs, tracing mutations across tables, transactions and the interfaces that mediate them allows teams to infer hidden business rules, undocumented transformations and dependencies that SMEs routinely overlook. Even when source code or service contracts are missing, the database and its interaction patterns still describe the system's logic, reducing reliance on incomplete API documentation.



**Integration and network tracing.** By observing how the system communicates with external services, one can reconstruct protocol behaviors, timing patterns, and

implicit contracts. This is essential when replacing parts of the system or implementing the strangler fig pattern.



**Binary analysis and AI lifting.** When code cannot be read directly, AI-driven tools can analyze executables, stored procedures, configuration files and logs to produce human-readable summaries. Modern GenAI systems accelerate dependency mapping, business rule extraction and documentation generation, mitigating the knowledge gap created by missing code and brain drain.



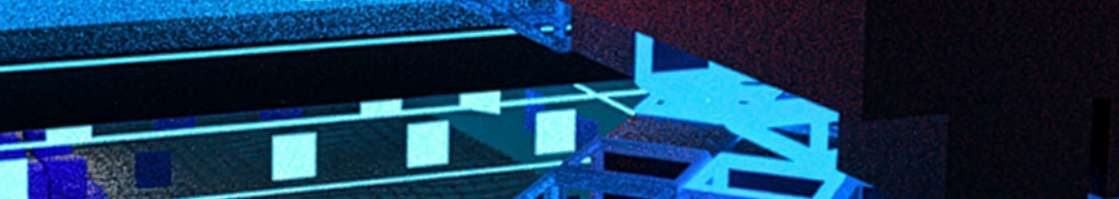
**Documentation mining and institutional recovery.**

Forgotten diagrams, emails and support tickets contain fragments of institutional memory. When aggregated and analyzed, they reveal constraints, historical decisions and behavioral expectations that would otherwise remain invisible.

The result of triangulation is a functional blueprint: a comprehensive view of workflows, rules, data flows and dependencies. This blueprint becomes the lodestar for modernization — accurate enough to guide investment decisions, detailed enough to reduce risk, and structured enough to support compliance and test planning.

Reconstructing the map is the moment an organization crosses the threshold from speculation to evidence.





## **Blackbox legacy modernization in action**

One global retailer faced a modernization nightmare: a mission-critical platform whose source code was partially missing after decades of vendor transitions and internal attrition. Leadership believed the system was stable and well understood. A modernization initiative uncovered a different reality.

Early analysis used a multi-lens approach. What began as a straightforward discovery phase quickly revealed that core behaviors contradicted SME descriptions. Data lineage work surfaced transformations that had never been captured anywhere. Binary inspection uncovered business rules buried inside compiled components written more than a decade earlier. In this instance, the retailer owned the software IP, which gave it the rights to do this reverse engineering — it's worth noting that this isn't always the case. Integration tracing exposed dependencies on external systems long assumed retired. Each pass revealed more hidden structure, expanding — not reducing — the organization's understanding of risk.

This is the moment every organization fears: discovering that the system they rely on is not the system they believe they have.

**The breakthrough came from combining structured analysis with AI-assisted reverse engineering — a form of the automated understanding that's a key component of Thoughtworks' AI/works™ platform. This allowed the team to reconstruct a full blueprint of the platform's logic and flows.**

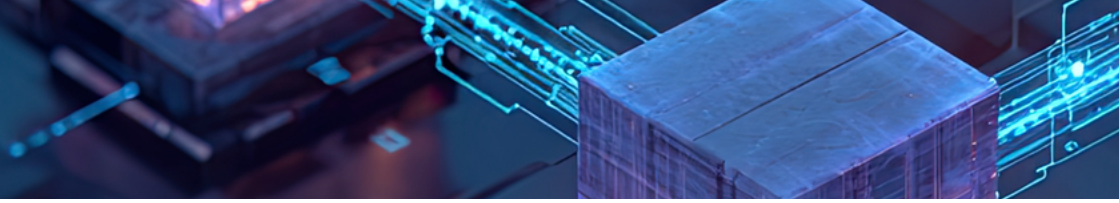
Instead of an open-ended exploration, the work became a progressively tightening evidence base. Automated documentation pipelines accelerated comprehension, restoring years of lost institutional knowledge. With no source code available, dependency-mapping tools treated deployed software as analyzable artifacts, reduced volatility during discovery by surfacing dormant execution paths, hidden modules and structural hotspots long before any rebuild work began.

With a trustworthy blueprint in hand, leaders could finally manage risk instead of guessing at it. They sequenced migration using the strangler-fig pattern — where a legacy system is updated piece by piece — prioritized increments based on measurable impact, and made informed calls about which components needed full replacement versus encapsulation. The retailer avoided the spiraling overruns that plague black-box modernization efforts, delivering a staged transformation aligned to its business outcomes.

The ultimate benefit wasn't just technical — it was strategic. The company moved from inherited complexity to intentional architecture, from uncertainty to control and from operational fragility to a roadmap grounded in observable truth.

This kind of work is now embedded in how AI/works™ approaches legacy systems: structured understanding first, accelerated regeneration second and confident evolution throughout.





## Start plotting your roadmap today

The path forward begins with a simple premise: you cannot modernize what you do not understand. But once understanding is achieved, modernization can proceed with confidence rather than fear.

### A practical roadmap includes:

**Identify the black box.** Choose systems where opacity creates risk — security, compliance, operational bottlenecks or vendor lock-in.

**Start with a thin, representative slice.** Focus on a critical workflow or module to expose key behaviors without overwhelming scope.

**Apply multi-lens analysis.** Combine workflow evaluation, data lineage tracing, integration mapping, binary analysis and AI-assisted documentation.

**Validate with SMEs.** Not to confirm accuracy but to reveal gaps in institutional memory — which are often larger than anyone expects.

**Build the functional blueprint.** This becomes the foundation for modernization strategy, test planning, sequencing, and compliance evidence.

**Choose the right modernization strategy.** Encapsulation for immediate interoperability, refactoring where logic is understood, or replacement where complexity is irreducible. You should choose your modernization pathway based on facts, not gut feelings.

**Commit to continuous visibility.** A modernized system can still become tomorrow's black box unless the organization changes how it captures, documents, and governs system behavior.

This is where agentic development platforms enter — not as a sales pitch but as an elegant structural solution to a pernicious structural problem. Modern systems evolve faster than humans can document them. Teams with deep expertise in forward engineering can use platforms such as AI/works™ to ensure behavior remains observable, explainable and continuously mapped. This doesn't just prevent your next generation of systems from decaying into opacity, it keeps them constantly fine-tuned to your business needs.

Modernization resolves the past. Continuous understanding protects the future. Your systems don't grow old, they grow up.

We are a global technology consultancy that delivers extraordinary impact by blending design, engineering and AI expertise.

For over 30 years, our culture of innovation and technological excellence has helped clients strengthen their enterprise systems, scale with agility and create seamless digital experiences.

We're dedicated to solving our clients' most critical challenges, combining AI and human ingenuity to turn their ambitious ideas into reality.

[thoughtworks.com](https://www.thoughtworks.com)