CobolCloud.

Move, Master, Manage: the method of continuous modernization

Turning IT legacy into lasting advantage

Turning Ideas to Action

In a previous article, we set out a core belief: legacy systems are not obstacles to overcome, but assets to understand. That belief is supported by proven methods, practices, and results.

This article explores the continuous modernization model that CobolCloud has developed over three decades in the field, structured around three phases that continuously reinforce one another.

These phases of Move, Master, and Manage coexist and strengthen each other over the course of an evolutionary path of transformation.

Move: moving forward without breaking everything

The *Move* phase addresses a fundamental question: how do you evolve a critical system toward modern environments, whether Linux, the cloud, containers, or whatever architecture emerges tomorrow, without rewriting what already works, and without creating new dependencies to replace the old ones?

The goal is not to migrate once and for all to a predefined destination. It is to make business logic inherently portable, capable of running in any relevant environment without altering either its form or substance. It is this freedom of movement that makes long-term continuity possible.

In practice, this requires rigorous, incremental progress. You start with the most critical modules, remove unnecessary dependencies on proprietary formats and outdated licensing models, and automate delivery processes. Research confirms it: nearly 70% of modernization projects pursued through a disruptive, break-from-the-past approach fail to meet their objectives. The poor results reflect a flawed approach, as much as any technical complexity.

Master: protecting against any future disruption

The Master phase addresses an equally important question: how can you ensure that the tools and solutions chosen today remain under the organization's control in ten, twenty, or thirty years?

Legacy systems operate over time spans that few technology products are built to cover. Vendors get acquired, licenses change, roadmaps evolve without consultation, and support is discontinued. An organization that depends on technology it does not fully control is exposed to disruptions it can neither anticipate nor control, whether they are financial, sovereignty, or technical.

Master is precisely the answer to this risk. Managed Open Source is its cornerstone: neither proprietary software subject to unilateral decisions, nor community open source whose governance burden you bear alone. It is a third path: shared source code maintained by a professional company, with the guarantees of a software vendor and the freedom of an open ecosystem. But beyond the code itself, it also means real and complete transfer of know-how, including recipes, practices, and methods, so that the client truly owns its tools, builds capability, and develops lasting autonomy.

Technical Hub 2.0 is the CobolCloud operational framework for that control. Deployable directly within the client organization or operated by a trusted third party, it ensures that the team is never held hostage by an external decision when running its most critical system.

Manage: regaining the clarity of day one

The *Manage* phase exists because of an issue all experienced IT leaders will recognize: a system that works is not necessarily a system that is understood. Business rules encoded decades previously, dependencies buried in modules that no one dares touch, calculation logic whose original intent has been lost along with the founding teams. This phenomenon of organizational amnesia takes hold quietly, worsens over time, and its cost ultimately becomes considerable.

The ambition of Manage is to recreate what CobolCloud calls the clarity of day one: understanding the system as if it had just gone live, knowing why it is built this way, how it behaves, and where to intervene with confidence. It is a GPS for code: not a tool that replaces the judgment of teams, but an instrument that allows them to navigate decades of accumulated decisions without losing their way.

AI-assisted analysis tools offering dependency mapping, identification of business rules, and detection of high-risk areas, act here as accelerators. They transform technical debt into

documented capital, and opaque code into an intelligible asset. Reaching further than static analysis could alone, Manage provides true augmented understanding, serving developers, architects, and managers who can then act faster, more safely, and without accumulating new debt.

A real case: when discipline pays off over fifty years

These phases are the result of years of experience built through real client cases, of which the following is a concrete illustration.

The system in question was designed in the late 1970s. It is a large-scale COBOL calculation engine, approaching one million lines of code, combining online supervision with intensive batch processing. Its results affect several million individuals across Europe. Since its inception, it has never stopped running.

What makes this case remarkable is not its longevity. It is the way it has gone through five decades of technical transformation without ever losing control over what it produces.

Move in action: from the mainframe to the cloud.

Originally deployed on IBM mainframe infrastructure, the system gradually migrated to distributed Unix environments, then Linux, before reaching the AWS cloud. At no stage was the core business logic rewritten. The COBOL programs themselves remained intact. What changed were the execution and orchestration layers, not the functional core. A sign of the maturity achieved: validation of the full CobolCloud stack on AWS Graviton ARM64 processors was completed literally in just a few hours. The move to public cloud was delivered six months ahead of the initial projections.

Master in action: autonomy as a founding decision.

Throughout this journey, a dedicated European team retained full control of the system: daily operations, maintenance, and functional enhancements—without ever allowing a critical dependency on a third-party vendor to take hold. From the earliest changes onward, the teams chose to build the simulation environment on the same COBOL source code base as production. Not an approximation, not a stripped-down copy: an identical working base. That decision, made in the 2000s, laid the foundations for autonomy that has never since been called into question. The fundamentals endured. Only the techniques evolved to multiply the possibilities of the calculation application.

Manage in action: rethinking the application without distorting it.

After nearly fifty years of continuous service, most of the gains no longer came from the business logic itself, but from the way calculations were executed. By understanding the system well enough to rethink its execution model without touching the COBOL modules, the teams transformed sequential processing into parallel in-memory pipelines, with up to 64 pipelines running simultaneously. The results are measurable: simulation times dropped from 30 seconds to less than one second. Full processing for the most intensive clients fell from 8 minutes to 8 seconds.

The success of this project was built around continuously understanding, and adapting, the application.
(Learn more on our website)

What this journey teaches us

An IT system that endures is something real, something solid: maintained, understood, and prepared for evolution. It functions like a living organism that remains healthy because it is continuously cared for.

Move, Master, and Manage represent three disciplines to be practiced continuously, each making the other two possible. Move ensures freedom of movement. Master builds long-term independence. Manage ensures that knowledge is never lost. Each empowers the others.

IT leadership today must decide whether they should endure another modernization, or whether they can truly lead it.

Organizations that succeed will not be shy away from short-term fixes or snake-oil replacements. They will be the ones with the discipline to understand them, master them, and continuously evolve them—without disruption and without losing control.

A version of this article first appeared in the French language IT publication, Programmez!

Learn More : CobolCloud Products Overview