CobolCloud.

What If "Legacy" Wasn't the Problem?

Reconsidering the value of existing systems

By Stéphane Croce.

Restoring Reputations

In the world of information technology, so-called "legacy" systems suffer from a poor reputation. Associated with outdated environments, they are often portrayed as obstacles to progress. Yet that view ignores what they really represent. Far more than mere lines of code, they are the living reflection of accumulated business knowledge and organizational history. The word "legacy" refers to what is passed on. It is precisely this attribute that this article seeks to restore: that of an inheritance from the past that continues to shape the present and chart the future.

Defining Legacy and Modernization

What is a legacy system?

In information technology, systems generally described as "legacy" are those perceived as old or nearing the end of their life cycle. That perception is explained in part by the historical origins of certain technologies, many of which date back to the 1950s and 1960s. But equating age with obsolescence is, at best, an oversimplification.

Many so-called legacy technologies have, in reality, never stopped evolving. IBM continues to improve its mainframe systems; COBOL, too, is regularly updated to adapt to changes in platforms and software environments. Like the automobile or telecommunications industries, some innovations span decades by reinventing themselves without ever losing their relevance. By contrast, technological history is also full of recent innovations—sometimes highly promising ones—that failed to stand the test of time. The idea that older technologies are destined to disappear therefore deserves to be seriously challeng

A Very Real Dependence

The global market for legacy-system modernization is now estimated at roughly $15 billion; it is expected to exceed $30 billion by the end of the decade. Firms such as IDC and Gartner regularly devote studies and recommendations to the topic, given how critical the stakes are.

In France, sectors as strategic as public administration, banking, insurance, industry, and retail still rely heavily on systems built decades ago. Many of them are based on mainframe architectures and developed in COBOL, and they perform critical functions: financial transactions, contract management, payroll, administrative processes, and logistics. Meanwhile, in the United Kingdom, the Financial Conduct Authority noted that more than 90 percent of financial institutions still rely on legacy technologies. Evidence is telling us that these systems are not merely still around, they sit at the heart of how organizations operate.

Modernization: A Multifaceted Concept

To modernize a system is to evolve it so that it better meets today's requirements, in service of a company's vision and objectives. But that evolution can take many different forms: incremental improvement, the integration of new tools, adaptation to new environments, or deep transformation. In practice, it can affect many layers: application code, infrastructure, databases, interfaces, and mechanisms for exchanging information between systems.

An entire ecosystem has taken shape around these challenges, offering a broad spectrum of approaches, from gradual transformation to radical reengineering. The diversity of viewpoints expressed by specialist analysts reflects that reality. But that diversity also reveals a more fundamental truth: beyond technical choices, the real question is often how well these systems are truly understood.

A Legacy to Understand and Preserve

Organizations evolve under constant pressure: shifting markets, regulatory constraints, user expectations, and accelerating technological change. Transformation initiatives are multiplying, often out of necessity. But maintaining the continuity of essential business functions is just as imperative. Striking the right balance between change and stability remains one of the most delicate challenges to address.

Over the years, systems accumulate business rules, processes, and structural decisions. A large share of that knowledge exists

nowhere other than in the code itself. In that sense, systems become the most faithful (and sometimes the only) representation of how an organization actually works. What defines a legacy system, then, is not only its age: it is also the extent to which its internal logic is no longer fully mastered by those who depend on it.

But a system cannot be reduced to its code alone, either. It exists within a broader environment: operational practices, data management, continuity arrangements, and security frameworks. Together, these form an inseparable whole, reflecting an organization's accumulated experience, the decisions it has made, the constraints it has overcome, and the adaptations it has carried out over time. It is that combination, including the code, that constitutes the true legacy.

What Leaders Underestimate

The issue is not merely technical. It is also human, organizational, and often urgent.

The first risk is the silent erosion of knowledge. Over the years, the people who designed these systems retire, move to other companies, or shift into different roles. Their successors inherit systems they keep running without always understanding their deeper workings.

Those of us who helped build these systems beginning in the 1980s have lived through every technological wave that accompanied them: from the mainframe to distributed architectures, from client-server to the cloud, from the earliest migration attempts to full-scale rewrite projects. That field experience, accumulated over nearly four decades, shapes a distinctive view of what has worked and what has failed. It is not uncommon to encounter critical applications that have run without interruption for 30 or 40 years even though the source code had disappeared, or the link between source and executable had been broken somewhere along the chain of successive changes. What this reveals is fundamental: a system that works is not necessarily a system that is understood, or even one that remains truly under control.

The second risk is confusing transformation with destination. At one time or another, every organization has launched modernization programs. Some succeeded, others did not, and many were interrupted along the way. But they all share one thing in common: the work is never truly finished. Meanwhile, the environment keeps changing: new regulatory requirements, renewed user expectations, and the rise of artificial intelligence and hybrid architectures. To ignore this truth is to condemn tomorrow's systems to the same fragilities as yesterday. The cycle begins again, and with it, the same costs, the same disruptions, the same regrets.

Redefining Legacy to Act Differently

Given these realities, a change in perspective is necessary. Treating legacy systems as assets rather than liabilities first requires truly understanding them before deciding how they should evolve. It also means accepting that modernization draws on technical, organizational, and human dimensions alike, and that its success depends as much on the methods and skills of those leading it as on the tools they use. Business leaders must recognize the urgency: without appropriate action, the fragilities of these systems will only grow. Legacy is a gift from the past. It deserves to be treated as such.

Accepting that legacy is an asset is a necessary condition. But it is not sufficient. One must also know how to move it forward. COBOL lies at the heart of this challenge: as the emblematic language of this category, it carries some of the most critical, and least well understood, business logic in our economy. From that reality emerges an approach to continuous modernization, proven in the field and built around three complementary disciplines of Move, Master, and Manage, for those who have decided to act.

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