CobolCloud.

Stagnation or Demolition? No thanks!

Why Legacy Modernization Requires a Third Path

By Stéphane Croce.

Strategic Struggle

Since they first went into production, sometimes several decades ago, large in-house information systems have suffered constant questioning about their long-term viability. Historical attempts to improve or replace them have had limited success – failure rates typically reported in the 70% range - and with each passing year the pressure grows to start over from scratch. But given the high risk, is the era of big-bang transformations over? And if so, which approach truly serves the organization's interests best?

After more than three decades spent supporting, guiding, and observing legacy system modernization, CobolCloud has formed its convictions in the field, through contact with hundreds of customer projects. Our view is a first-hand perspective.

Strategy 1: Maintain the Status Quo, "Keep and Hope"

For IT modernization, the first strategy is more of a tactical default than a proactive choice. It consists of "keeping the lights on", namely maintaining existing systems while applying only the strictly necessary fixes, hoping that no critical failure occurs.

This laissez-faire approach has nevertheless kept thousands of organizations' IT systems running for years. Essential, mission-critical systems are maintained with operational capability reduced to the bare minimum.

The risks are well known: growing technical debt, rising maintenance costs, and a gradual erosion of incumbent skills and knowledge. The line is imperceptibly thin between systems seen as "too important to fail" and those that are, frankly, "too complex to still be under control." A "keep and hope" strategy lasts only as long as luck does.

Strategy 2: Start Over, "Replace and Hope"

Faced with mounting concerns about long-term viability, many leaders have chosen to make a clean cut: replace the legacy system with something more modern. And within that logic, one solution gained in popularity: translate COBOL line by line into Java, C#, or another contemporary language.

At CobolCloud, we have observed this market unfold for fifteen years. In the vast majority of cases, the result was a catastrophe.

Not because Java is a bad language. But because this approach rests on a profound misunderstanding of what a large COBOL system really is. What gets translated is lines of code. What does not get translated is intent. And that is precisely where all the value resides.

A large system is not a collection of lines. It is a digital cathedral, built by generations of developers, each with an understanding of how the elements piece together. Those intentions are not written into the syntax. They live in the structure, in the sequences, and in the architectural choices were likely never fully documented because they seemed obvious at the time.

Translating that system into another language reveals none of this. Modernization practitioners are left with business rules and architecture buried in a language that was not designed to express them. In the new incarnation, the hidden truth has not just disappeared; it has become irreversible.

And yet the argument used to justify this approach is always the same: there are no COBOL skills left, so we must migrate to a language everyone understands. But the argument is flawed - the problem was never COBOL's readability. Taken piece by piece, COBOL is very readable. The syntax is comparatively straightforward. Junior developers can become proficient quickly. Analysis tools can parse it effectively. Today's artificial intelligence systems can read it too: they identify anomalies, suggest improvements, and map dependencies. The language itself is not the challenge.

The real challenge is modernizing legacy systems is understanding the original developer's intent. What was the developer trying to do in this part of the code? What underlying logic was being encoded? Where does that logic begin, and where does it end? This problem is not specific to COBOL. It would be same if the system had been written in C, Java, or any other language. The complexity is not caused by the syntax. It is in the depths of the decisions accumulated over time.

Diving into thousands, sometimes millions, sometimes hundreds of millions of lines of code to reconstruct their original meaning is a mission that, if it succeeds at all, takes years. The promise of automatically translating such a system to recover what has been forgotten is wishful thinking. That is not true modernization. It is the destruction of an asset dressed up as transformation.

Strategy 3: Regain Control, Modernize Continuously

So, there must be another way. Continuous modernization rejects the impotency of the first approach and the unrealistic idealism of the second. It proposes a third path: a process of ongoing incremental improvement, evolution, and adaptation that offers organizations real control over their most critical IT assets.

At the heart of this approach is the objective of IT leadership recovering control of the future of its application assets. That means rebuilding capabilities, upgrading tools, and reinvesting in an expert, dedicated team that can autonomously evolve its systems according to its own needs, without relying on outside service providers or software vendors whose interests may not align.

The benefits are concrete: reduced risk, greater agility, and the ability to deliver value to the business regularly, without starting from scratch every time.

The method behind this third path, the three disciplines of Move, Master, and Manage, and the real-world case that illustrates them are documented on our web site.

Conclusion

The "keep and hope" and "replace and hope" strategies have dominated the legacy modernization market for decades, with disappointing results that rising budgets have not improved. Continuous modernization offers a credible alternative, not because it is gentler or more cautious, but because it starts from a reality the other two ignore: the value of a large system does not lie in its language. It lies in what it contains, in what it knows how to do, and in the understanding its teams have of it.

It is precisely this understanding that continuous modernization seeks to preserve, pass on, and deepen. Not as a vague ambition, but as a daily discipline built on years of experience with what works for hundreds of successful organizations.

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