Successful mainframe modernization: Taking a behavior-first, AI-powered approach

BY FEDERAL NEWS NETWORK STAFF

For federal agencies grappling with aging mainframe systems, the path to modernization has long been paved with risk, uncertainty and failed attempts.

Many government systems still run on code written decades ago, often by developers who have long since retired. Documentation is sparse, institutional knowledge is rapidly disappearing, and the cost of maintaining these systems — both financially and operationally — is growing.

But recent attempts to address this situation have yielded disappointing results, said Edward Hieatt, chief operating officer at Mechanical Orchard.

"The traditional way to go about this mainframe modernization would be, to consider it a translation problem first and foremost, which makes sense. You've got COBOL, you want Java: Translation's the name of the game," Hieatt said. "The typical approach now is using artificial intelligence to translate it and then use people to try and prove it's the same and put it in production somehow. That last part is what takes all the time and creates all the risk."

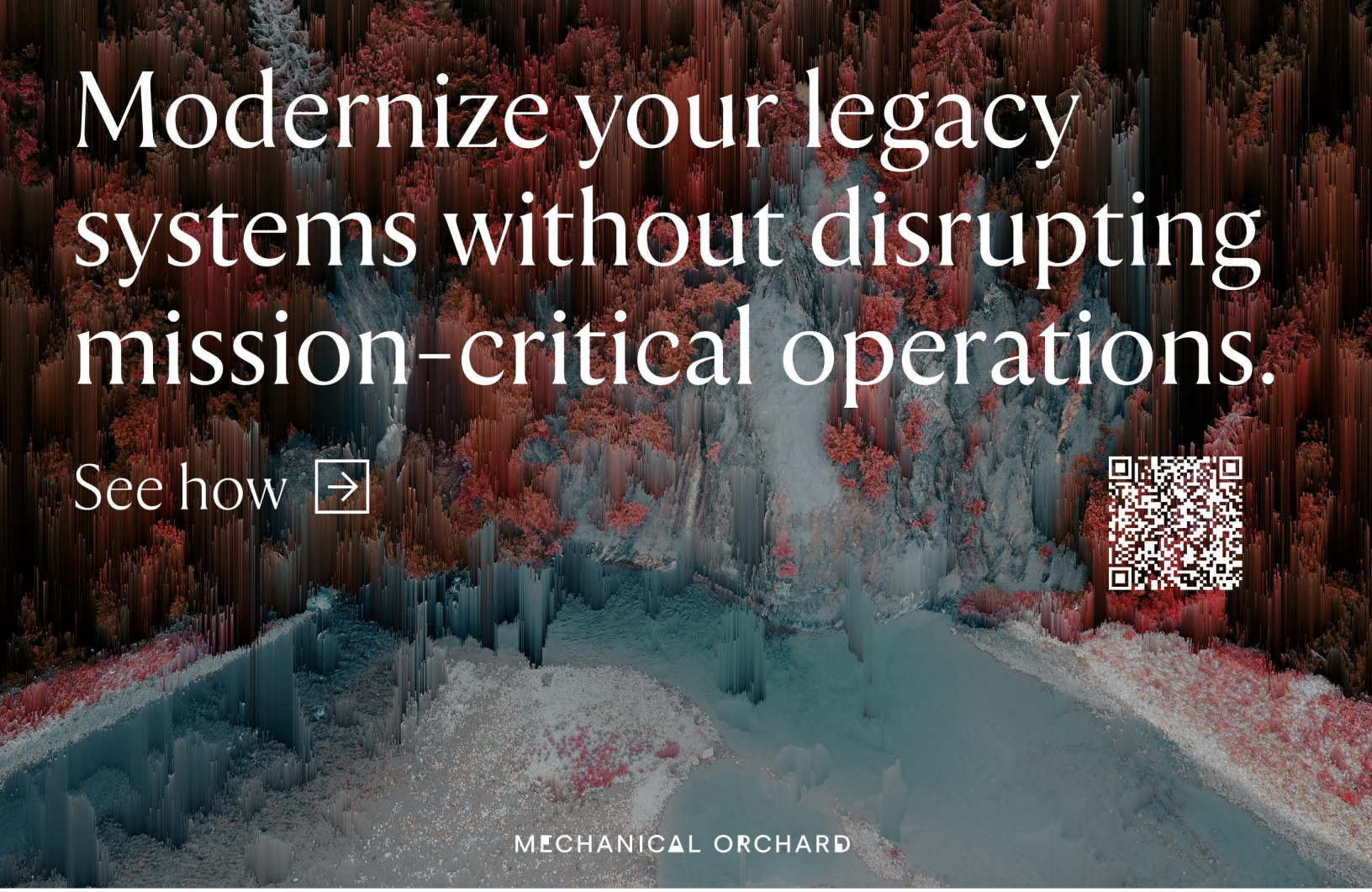
But Hieatt said there's an alternative.

Mechanical Orchard flips the script. Instead of starting with code translation, it begins by observing how legacy systems manipulate data in production, he said.

This approach creates a behavioral specification — a record of inputs and outputs that defines

> If you end up with crappy code, you haven't really solved it. For us, the code quality, meaning human or AI readability, is key. The metric around complexity should be less complex when we're done.

Edward Hieatt, Chief Operating Officer, Mechanical Orchard



what the system actually does, Hieatt explained. Capturing the behavior of the data maps out the data flows, which allows modernization to begin with an intended result instead of a direct translation. The code doesn't have to translate directly, as long as it accomplishes the same goal.

Hieatt compared it to foreign language translations. Word-by-word translations can lack grammatical syntax or context. The same thing happens when developers or AI attempt a line-by-line translation from COBOL to Java. They often get what Hieatt called "JOBOL," an overly complicated version of Java that mirrors the structure of COBOL.

"It's gobbledygook. It might work, but it's not maintainable," he said. "Often the motivating factor to get off of the mainframe is for the code to be maintainable. So if you end up with crappy code, you haven't really solved it. For us, the code quality, meaning human or AI readability, is key. The metric around complexity should be less complex when we're done."

Modernizing incrementally, building confidence

Another key difference in Mechanical Orchard's approach is its incremental nature. Rather than attempting to modernize an entire system at once, the team works component by component, validating each step before moving on.

There's no big bang modernization; AI captures the data at a low level, a few thousand lines of code at a time, implements the new version and then checks against the behavioral patterns to verify the new implementation works as

expected, Hieatt said. Once the system owner is confident in the equivalence of the new code, the new code moves to the cloud environment, and the old workload is retired.

This incremental process provides continuous oversight and validation, reducing the risk of failures while building trust in both the process and the new code, he said. By identifying risks early and minimizing disruptions, teams avoid costly rework and accelerate with growing confidence. The result is a predictable, steady path to successful modernization, Hieatt said.

"Certainty, predictability and incremental delivery make it much safer, giving customers more agency," he said. "Done means done as you go, ratcheting forward progress, as opposed to translating with AI and then crossing your fingers, which is what most people end up doing."

Ready to make an impact

This approach also delivers measurable benefits because projects are completed faster, with higher code quality and reduced complexity, Hieatt said. This not only improves maintainability but also enhances security and resilience, which are critical concerns for federal systems.

Less complex code means fewer vulnerabilities and easier compliance too, he pointed out. By reducing code volume and improving readability, systems become both more secure and future-proof.

"All code is debt. All code is legacy code. The more code you have, the more possibilities there are for exploits," Hieatt said. "Any code that you have represents danger. You want as little code as possible — always — because the more code you have, the more possibilities there are for exploits that you don't know about yet. The more possibilities for human error, the more difficult it is to cover with good test cases, the more likely something is to go out of support from some third-party vendor."

> Any code that you have represents danger. You want as little code as possible — always — because the more code you have, the more possibilities there are for exploits that you don't know about yet.
>
> ***— Mechanical Orchard's Edward Hieatt***

Additionally, this method lets agencies reduce spend on legacy vendors sooner. Instead of paying for legacy infrastructure throughout a multiyear project, agencies can retire components incrementally and begin realizing cost savings early, Hieatt said.

"The appetite and the hunger for modernization is actually higher than a few years ago. There's less tolerance for the status quo. There are captive contracts sitting around in many of the default contractors just waiting to be used for modernization. So it doesn't take acts of Congress to get new awards. We're primed for making an impact in federal government."