How Mphasis NeoZeta is bringing banking back-end systems into the AI era

By Katy Ring April 3, 2026

Audio mode

Dyslexia mode

SUMMARY: How banking is putting down a deposit on agentic updates.

As a sector, financial services seems to forever be modernizing its IT. This is partly because banks were one of the first sectors to computerize and so have a lot of legacy systems in situ. As consumer demands have changed banks now offer digital services but that has really placed modernization efforts at the front end of processes.

In sum, there have not been many successful attempts to change the back-end banking systems and those who know the apps are no longer in the workplace. Meanwhile the market continues to change with the arrival of new fintech challengers as the traditional banking back end remains frozen. Agentic AI is positioned as a panacea for most tech woes, so can it help traditional banks?

The problem is not code migration, it's knowledge reconstruction

According to Srikumar Ramanathan, Chief Solutions Officer with Mphasis, while banks have successfully digitalized their customer-facing systems, they remain constrained by legacy back-office systems. He explains:

> *They also struggle to get budget for back-end modernization because of past failures. But now there is a faster way to do all this. This is all very well as long as we don't break things, can modernise fast and integrate new technology in a timely manner. We require a different way for the back end to work with agentic AI.*
>
> *There are multiple approaches to modernization. AI can convert one language to another, can, say, translate from Cobol to Java. However, for a back-end enterprise app translation is not modernisation because of the constraints of the original coding, which was a very monolithic way of coding with processing logic, data structures and outputs all mixed up together. To translate this, you simply get the same monolith out. The problem remains that this is a monolithic application.*

Ramanathan's experience pre-dates the AI era. His knowledge comes from modernizing legacy apps where you needed to extract the business logic first since the reason people are scared to touch the app is that they do not know what is in the app. Prior to AI people had to read all the programs involved to put that logic into language that the business understands. The challenge has always been to understand both the Cobol and the domain knowledge. In the past there were always gaps in the understanding and the confidence level remained low in the ability to succeed.

In order to modernize systems for the AI era, Mphasis teams began tackling the problem, as most do, by using tools to extract the business logic. The firm quickly realised that this is not sufficient to get an algorithm to understand the app, as it also needs to interpret its intent in business language. Ramanathan continues:

> *You must know, say, that C+M refers to a loan calculation of interest rates. I need something that understands what is going on and can map it to a business ontology. You need to create a domain ontology and use LLMs to extract logic in both a human and a model mode. Then you put the extracted logic into a knowledge graph, pulling together the intelligence from both the code and the data alongside the business logic.*

Mphasis does this using its NeoZeta platform which is designed to extract the business logic from existing domain ontology and then get the new business logic stored in a knowledge graph, using conversational English to interrogate the code. Then Ramanathan says you can safely introduce new features and decide which parts of code you do need to change.He explains:

> *We have created Mphasis Ontosphere, an engine that drives forward engineering. You can do this in Python and Java forward-engineered from the knowledge base. By using this approach, you are creating a knowledge graph basis for doing anything going forward. Today's new code is tomorrow's legacy code, but not if you are constantly updating the intelligence to power new*

read it. Mphasis is currently exploring whether to leave NeoZeta with customers after a modernisation project, and is thinking about how they would licence this. Pilots with a handful of customers have proved that the approach works. For example, NeoZeta was used with a cards platform running 50m lines of Cobol and Assembler, supporting 1.4 billion accounts and 25 billion authorizations annually. NeoZeta analysed the environment and helped re-architect it into a modular, cloud-ready platform improving operational efficiency by over 40% and compressing timelines from months to weeks.

Ramanathan concludes:

> *Productivity benefits come from the extraction of knowledge because you don't need an army of people reading the code. One of the reasons banks do not modernize is because it is not economical to do so. With NeoZeta you automate to shrink the time to extract the knowledge and so you can address more and more legacy systems.*

My take

With a greenfield app, AI can be used to describe everything and there is a school of thought that the arrival of agentic AI should be considered a rip and replace moment for enterprise technology. However, back in the real world this is unlikely to happen and agentic AI is providing ways to better understand brownfield apps and the implications of connecting them to other systems.

A lot of agentic AI tools are focused on coding but, as with many use cases for the technology, that just shifts productivity bottlenecks elsewhere. The interesting aspect of the NeoZeta approach is that Mphasis is creating agents to address forward engineering and to create epics and stories. The intention is to bring automation into many different areas of modernisation. The byproduct of using NeoZeta is that it agentifies your processes by creating machine readable knowledge, so you can then start building other agents. Not every system is available in a way that agents can consume, but NeoZeta offers a way to address this. Even for the back-end systems of banks.