

AND Digital

Modernise Later. Extract Value Now.

A Strategic Case Paper by
Nathan Warner (Hub Exec)
Jamie Pullen (Solution Director)

August 2026

Modernise Later.
Extract Value Now.

Somewhere in your estate there's a set of legacy systems like an IBM AS-400 running the same core processes it's run for twenty years. Reliable, unglamorous, and quietly assumed to be a problem you'll get to eventually. For years the only responsible answer has seemed to be to replace it. But it works. It's dependable. The cost and risk of ripping it out has never quite stacked up against the business case for leaving it alone. So it sits there, generation after generation of leadership quietly agreeing it's next year's problem.

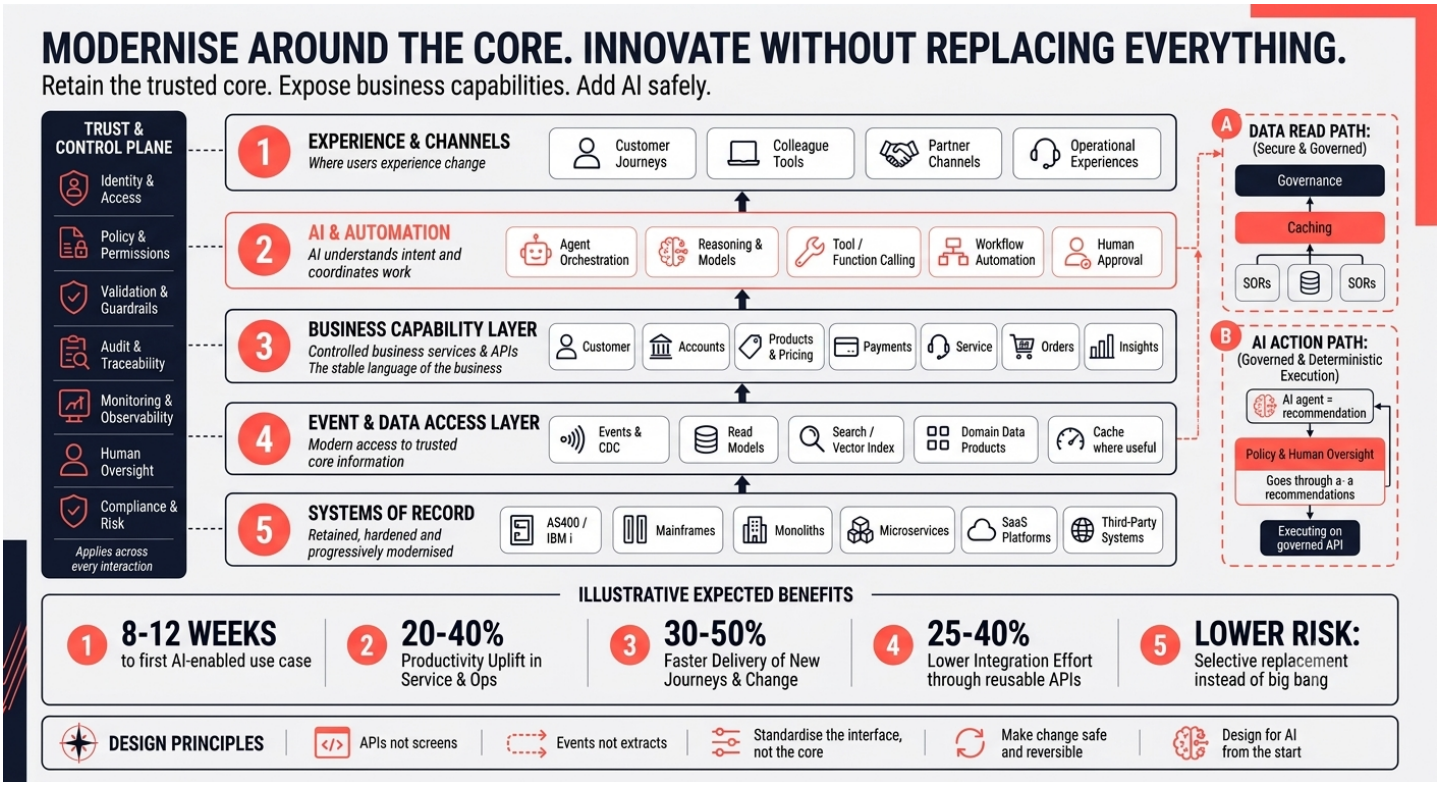
For a decade, that tension played out the same way across the industry. Break the monolith. Move to microservices. Containerise it, use Kubernetes, go all-in on serverless. Get everything into the cloud. The pitch was never really about technology. It was about value. Legacy systems held data and logic the rest of the business couldn't get to, and migration was how you cracked them open.

AI has quietly removed that reason.

Nobody migrated a 20-year-old monolith because Kubernetes is elegant. They migrated because the system was a black box, and the only way anyone could see in was to rebuild the front door. Microservices gave you APIs. APIs gave you integration. Integration gave you value: reporting, automation, new products built on old data. The architecture was the means. Exposure was the end. We've spent over ten years treating "modernise the architecture" and "expose the value" as the same project. For most of that decade, they were. There wasn't another way in.

If a system has an API, however old, however ugly, an AI agent can talk to it. It can query it, interpret the response, and translate that into something a person or another system can use. You no longer need to redesign the system to make it legible. You need to make it secure and reliably reachable. That's a much smaller job.

A lot of people are rightly cautious about connecting AI to a legacy system nobody fully understands. But that caution is the point, not a reason to stop. It means the effort goes into securely accessing it properly, not into leaving the system untouched out of fear.



For most of these black-box services, extracting value only ever needs read access.

Instead of years of phased migration, you can put a centralised, AI-driven facade over your entire estate: mainframe, monolith, half-finished microservices from 2019, all of it. The AI layer becomes the interface. It queries what it needs, stitches together an answer, and hides how messy things are underneath. The value comes out the same way it always did, through an interface. It just isn't the system's own interface any more. The legacy system stays legacy. It keeps running, and it keeps being replaced on its own timeline, or not at all, because the pressure driving that replacement has gone.

That's the shift for the AS-400 example too. Instead of a multi-year programme to rebuild the core, you harden it, secure it, and leave it doing what it does well. Around it, you build a governed layer that makes its data and capabilities reachable through modern APIs, events, and AI-enabled workflows: AS-400 core, a controlled API and event layer, governed data and business services, then AI orchestration on top. The AS-400 stays the trusted system of record. The new layer becomes the system of engagement and intelligence.

None of this makes monolith vs microservices, or Kubernetes vs Serverless, irrelevant. What becomes irrelevant is the urgency. Those architecture decisions still matter for cost, reliability, security, and how fast your own engineers can change things. If your monolith falls over every time traffic spikes, an AI layer sitting in front of it won't save you. What's gone is the idea that you have to migrate in order to extract value. That was the forcing function behind half the roadmaps I've seen, and it's no longer true.

Once extracting value stops requiring you to rebuild the thing, migration becomes an engineering decision again instead of a business-value decision. You do it because it's cheaper to run, easier to secure, or safer to change, not because it's the only door into the data. That's a healthier reason to migrate. It's also a much easier one to say no to, or to delay, which is exactly why this is going to reshape a lot of roadmaps over the next couple of years.

A good chunk of agentic AI initiatives are going to fail. Gartner's predicts over 40% by 2027, and the reasons are predictable. Poor scoping. No clear owner. Bolting an agent onto a system nobody's confident explaining. An AI layer over your estate doesn't remove that risk; it just moves where it bites.

It's also not universal. This argument is strongest where a system has some kind of programmatic surface to reach, even an ugly one, and weakest for the genuinely locked-down estate: undocumented mainframes, screen-scraped terminals, logic that only exists in one person's head that's been with the business for 40 years. That stuff still needs traditional modernisation, or it stays untouched. And there's a real difference between an agent reading your data and an agent acting on it. Exposing insight is low risk. Letting something transact on your behalf, change a record, move money, update a policy, is a different conversation, and it deserves a more careful one.

That's exactly why the AS-400 case isn't "let an AI loose on the mainframe." It's the opposite. The AI never talks to the AS-400 directly. There are deterministic, governed layers in between, with permissions, validation, audit trails, and clear rules for what the AI can read, recommend, or execute. These systems are unusually sensitive to data formatting, so keeping validation tight matters more here than most places. That distinction is what makes the approach defensible instead of reckless. None of this undoes the argument.

It just means the win isn't automatic, and it isn't everywhere.

With AS-400-style systems, the governance layer connects to a caching layer. These systems are built for heavy batch-style runs, not live queries, and hitting the core directly isn't an option due to capacity constraints. The caching layer acts as another downstream subscriber, consuming without constraining the system. The AI queries that copy, and the mainframe carries on running its jobs exactly as it always has.

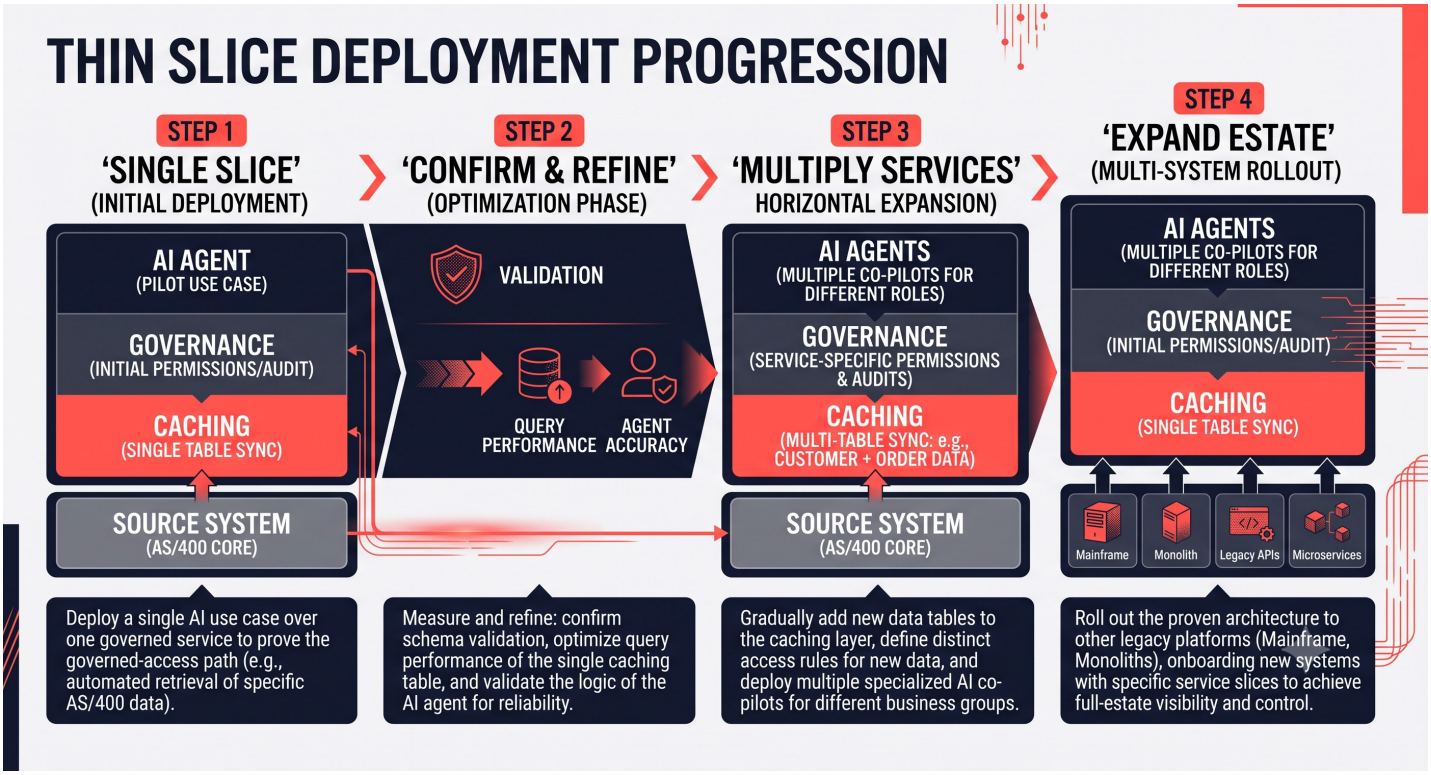
A few things follow from all this.

The AI layer is the new priority, not the migration behind it. If you can put a well-governed, AI-accessible layer over your estate, you've unlocked most of the value you were chasing with the multi-year replatform. Build that first. For the AS-400, that means better customer journeys, automated manual processes, and colleagues with faster access to information, all without touching the core.

"It has an API" is now a strategic asset, even a bad one. A clunky SOAP interface from 2008 is still a door. You don't need to knock it down and rebuild it before an AI agent can walk through it. Migration pressure moves from value to risk. The conversation with the business changes from "we need to modernise to unlock the data" to "we need to modernise because this thing is a security and reliability liability." That's a smaller, more honest conversation, and it competes for budget differently. The investment case stops being "spend tens of millions replacing the system" and becomes "unlock new value now, contain the risk, and progressively retire the parts that genuinely constrain us." You still modernise, just selectively: replace what genuinely limits resilience, data access, product speed, or regulatory control, and leave what's dependable alone.

The £50 million that would have gone on a programme to replace the legacy system doesn't need spending. A fraction of it builds the API facade instead, and puts the rest towards removing the technical debt that's actually causing pain, not refactoring an entire estate that mostly still works fine.

Thin Slice Deployment Progression



And governance stops being a footnote and becomes the actual project. Centralising access through an AI layer means centralising risk too. Who's allowed to ask the layer what. What it can touch versus what it can only read. Who's accountable when it gets something wrong, and how you'd know. That's the design work that now deserves the energy that used to go into arguing about service boundaries, and it's not a one-off decision; it's ongoing operational discipline.

Every big cloud and microservices push of the last decade was, underneath the technology, an argument about access. AI has settled that argument in a different way. You don't need to tear a system apart to get value from it any more. You need to make it answerable, and you need to control who's allowed to ask.

The AS-400 can keep ageing, keep running, keep being difficult, and slowly move on its own schedule. The pressure's off. What's left is architecture for architecture's sake, cost, reliability, security. Worth doing. Just no longer the thing forcing your hand. That's not using AI to hide technical debt. It's retiring the idea that wholesale replacement was ever the only route to innovation.

The multi-year, multi-million-pound replatform is no longer the mandatory entry fee for innovation. You don't need a pristine, modern architecture to start leveraging agentic AI; you just need a secure, governed door to the data you already have.

If you want to skip the migration gridlock and map out what a pragmatic, thin-slice AI layer looks like for your own estate, reach out for a chat. https://www.linkedin.com/in/jamie-pullen/