

The business case
for AI-led
modernisation:
A CTO's
working guide

AUTHOR

CLIFTON CUNNINGHAM
SVP ADVISORY & ALLIANCES, NEARFORM

Contents_

THE ARGUMENT

Executive summary	2
Foreword: what's actually changed	3

MAKING THE CASE

Three ways to make the case	5
Build your own	6
Choose your approach	7

THE NUMBERS

Run your own numbers: a worked example	8
--	---

THE FINANCE CONVERSATION

Is modernisation capex?	9
Questions to bring to your CFO	10

TAKING IT FURTHER

Your evidence pack	11
Where to start	12

Sources and references	14
------------------------	----

Executive summary_

Most CTOs will recognise this scenario: the system works, but everyone knows it needs modernising.

The problem is, nobody wants to touch it. The logic lives in a few people's heads, the person who built it left years ago, and no one wants to be the one who breaks it.

AI hasn't just made writing code faster. It's made modernising these systems realistic, and that changes the shape of the business case.

In this guide, Clifton Cunningham, Nearform's SVP of Advisory and Alliances and a former CTO himself, sets out three ways to make that case: lead with risk, lead with opportunity cost, or go incremental and redirect part of the maintenance budget that's keeping the old system alive.

For each, he's honest about where it tends to get approved and where it falls apart. Inside, you'll find a business case framework, a real-world example and ready-to-use worksheets. There's also a section on whether modernisation counts as capex, with questions to put to your CFO so it's treated as an asset, not an expense.

Foreword: what's actually changed_

We're into the back half of 2026. Frontier coding models, work-horse models doing the volume, and purpose-built harnesses around them have changed what it costs to modernise a legacy system.

I want to be careful with that word, harness, because it's doing most of the work in this argument. The model isn't the thing that does the migration. A migration is a repeatable, high-volume, verifiable transformation applied a few hundred thousand times, and that's a workflow problem before it's a model problem.

What makes it tractable is a harness: deterministic code that breaks the work into phases, gives each phase only the context and tools it needs, picks the model that fits that phase, and passes auditable artefacts between them. Discovery, dependency mapping, characterisation test generation, transformation, review, equivalence checking. Each phase evaluable. Each phase improvable on its own.

Point a general-purpose agent at a legacy estate and you get a general-purpose result. Build the production line and you get a measurable one.

The best independent evidence for the size of the shift comes from Google, who ran 39 internal migrations over twelve months with three engineers, landing 595 changes and 93,574 edits, roughly 74% of which were generated by the model. The engineers involved estimated the total time was cut by half against the manual equivalent. That is published, peer-reviewed work rather than a vendor claim, and it is worth reading before you believe anyone else's numbers, including ours.

I trust our own engagements more, because I can see them working.

A state-owned financial services provider we worked with had an anti-money-laundering problem of exactly this shape. Their Oracle FCCM implementation was doing its job and generating far more cases than the compliance team could clear. Each case took an analyst 15 to 25 minutes; a batch of 5,000 cases was around 125 working days of investigation, with output consistency running at about 85%. The traditional answers were to replace the platform or hire roughly thirty more people. Instead, we built an AI-native investigation agent with a four-person team and had a production-ready version in six weeks. It cleared a live batch of 847 cases in under two hours, agreed with the human analysts on 100% of both the true risks and the false positives, and removed an estimated €2m a year of additional hiring. The agent recommends. People decide.

The discipline underneath this is older than the models. When Condé Nast International needed to migrate millions of content assets from every global territory into a single CMS, with a hard three-month deadline on the first region, the answer wasn't a bigger team. It was a self-service migration platform that took accuracy from 80% to 99.9% and let territories run concurrently rather than in sequence. That's the same idea a harness formalises: treat migration as an engineered, measurable pipeline rather than a heroic project. The models have made that pipeline dramatically cheaper to build and far more capable. They didn't invent it.

The practical consequence for your business case is a change of unit. Modernisation used to be quoted in years and tens of millions. It's now reasonable to quote a first tranche in weeks, with a real artefact at the end of it: a characterised, tested, running slice of the system on a supported stack.

That's what I've seen change. Run the numbers on your own estate, and see where you land.



Clifton Cunningham

Clifton Cunningham

SVP Advisory & Alliances, Nearform

Three ways to make the case_

Even with the maths fixed, modernisation still competes for attention against wherever your business currently believes value gets created. It still needs a case. I've seen three that work, but they can also fail in different ways.

Option 1: Risk.

Lead with continuity and security. Unsupported runtimes, unpatchable dependencies, single points of human failure, regulatory exposure, the recovery time objective you're unable to meet.

Why it works: it's concrete, it's auditable, and it has a natural sponsor – the risk committee with a standing agenda item and a budget line. If you have a red-rated system with no remediation funding, you have a paper that writes itself.

Where it fails: fear-based cases get funded to the level of the fear, and no further. You'll get the money to remove the specific risk you named, not to modernise the system, so you end up doing the least interesting version of the work, and you'll be back next year with the next scary thing.

Option 2: Opportunity cost.

Lead with what the system prevents. Your AI strategy needs data and events out of these systems; they have no API, no test coverage, and no one who can safely change them. Every integration prices in a legacy surcharge.

Why it works: it connects to value creation, which is how to get attention (and cash). It puts modernisation on the critical path of something the board already wants, rather than in competition with it.

Where it fails: if the initiative slips or gets deprioritised, your case evaporates with it. You've also publicly declared yourself the blocker, so the pressure lands on you to deliver the enabling work at a pace set by someone else's roadmap.

Option 3: Incremental.

Don't raise a programme. Redirect. Your maintenance budget already exists and is already approved, and now that the cost of transformation work has fallen by an order of magnitude, that budget covers modernisation work it never used to. Spend the same money differently: modernise the slice you were going to patch anyway.

Why it works: you already have approval, so you skip the entire competition. It suits the harness model perfectly, because a harness pays back over repetition, you build it for the first slice and amortise it across everything after.

Where it fails: it's invisible. Nobody senior knows that it's happening, so it gets no credit, no air cover, and no protection when budgets tighten.

Build your own case_

Tick the cell in each row that matches your situation, then count the ticks in each column. The column with the most is the case to lead with.

	Risk	Opportunity cost	Incremental
 Sponsor	☐ Risk/audit committee, CISO	☐ CEO, CPO, whoever owns the AI agenda	☐ You
 Evidence needed	☐ Support dates, CVEs, RTO gaps, key-person analysis	☐ The specific initiative and the specific blockage	☐ Current maintenance run-rate and its trend
 What you get funded for	☐ Removing the named risk	☐ Unblocking the named initiative	☐ The next slice
 Time to money	☐ One risk cycle	☐ One planning cycle	☐ Now
 Failure mode	☐ Minimum viable remediation, repeated annually	☐ Case dies when the initiative slips	☐ No visibility, no air cover, no scale
 Best used	☐ When you have a red-rated system and a date	☐ When legacy is genuinely on the critical path	☐ Always, and first

RISK

OPPORTUNITY COST

INCREMENTAL

Choose your approach_

Ask these questions of the one system you're most worried about right now.

- Does it carry a named, dated risk: an end-of-support deadline, an unremediated audit finding, a compliance exposure you can point to? That points to risk.
- Is there a specific initiative already on your roadmap, an AI programme, a new product, an integration, that this system is visibly blocking? That points to opportunity cost.
- Do you already have an approved maintenance budget for this system, with no fresh sign-off needed to redirect it? That points to incremental.
- Could you produce real evidence, a working slice, a real number, within one quarter, without asking anyone for anything? If yes, incremental is available to you right now, regardless of your other answers.
- Who would actually sponsor this work: a risk or audit committee, the owner of the blocked initiative, or just you? That's who you need to get on side.

My honest advice: this isn't a choice of one.

Start with the incremental case to produce evidence, because it needs no permission.

Use that evidence to make the opportunity-cost case credible: 'we can unblock this in eleven weeks, and here's the one we already did' is a different conversation to 'we estimate two years'.

Keep the risk case in reserve and use it precisely on the one system where the risk is real and dated, rather than as your general argument. Risk is a good scalpel and a terrible hammer.

Run your own numbers: a worked example_

Here's how this looked on a real system.

The system:

A mid-sized order management system: about 1.1 million lines of Java, framework two major versions past end of support, no meaningful test coverage, running on VMs in a data centre being exited. Current annual run cost: 3.5 FTE of maintenance and on-call at fully loaded cost, roughly £420k. Licences and support £180k. Hosting £240k. Total, call it £840k a year, indefinitely, for zero change.

The 2023 attempt:

A replacement programme was quoted at £9m over 30 months, and never funded, correctly.

The 2026 difference:

A harness built around this specific estate: six weeks of discovery and characterisation, then transformation in slices. Call it seven months and £1.3m all-in, including the team, the tokens, and the infrastructure to run it.

Now run yours.

Fill in your own figures on your system.

[\[TRY THE CALCULATOR\]](#)

The two most important numbers: How many years of your current run cost does the thin-slice quote represent? And how much do you get back every year once it's done? In the example above, the thin-slice quote was under two years of the current run cost, with a lower one afterwards, and the single point of human failure removed.

Do this sum before you write anything else. If two to three years of your current run cost covers the modernisation, you have a case that needs no rhetoric at all.

Is modernisation capex?_

This is the least talked-about part of this case, and possibly the most useful. How the spend is classified often decides whether it happens.

I'm not an accountant and this isn't advice, but you should have the conversation with your CFO early. The answer is more favourable than most technology leaders assume, and it's currently in motion.

The short version: under both IFRS (IAS 38) and US GAAP (ASC 350-40), routine maintenance is an expense, but development work that creates or materially enhances an asset (new functionality, an extended useful life, a genuine increase in capacity or performance) can be capitalised. A lift-and-shift that changes nothing is a hard sell. A modernisation that produces a better asset is a much easier one.

Two things worth knowing right now:

- FASB's ASU 2025-06 has reworked the internal-use software rules, dropping the old sequential 'project stages' model in favour of a probable-to-complete threshold: management has authorised and committed funding, and there's no significant remaining development uncertainty. It's effective for annual periods beginning after 15 December 2027, with early adoption permitted.
- The new test is explicitly about resolved uncertainty. Thin-slice modernisation, where you characterise the system and prove the transformation on a representative slice before committing to the rest, is close to a purpose-built way of clearing that bar early.

There's also a cost-mix effect. As the work shifts from person-months to compute, some of what used to be salaried engineering time, capitalisable if attributable to the project, becomes token and infrastructure spend, and the treatment of that isn't uniform yet.

Questions to bring to your CFO_

- Does our current run cost on this system qualify as routine maintenance, or has it already drifted into something closer to redevelopment?
- If we prove a thin slice first, does that count as resolving development uncertainty early, under ASU 2025-06's probable-to-complete test?
- How do we want to treat compute and token spend that replaces what used to be salaried engineering time?
- What would it take to capitalise this as an asset, rather than expense it as maintenance?

“Get finance into the room while you're designing the programme, not after. This is the sort of detail that decides whether the number in front of your board is one they can say yes to.”

Your evidence pack

Need further evidence to take to your board? Here's what the industry's saying.

28%

of central government systems sit in the legacy category, with red-rated systems often carrying no remediation funding at all.

State of Digital Government Review, UK Department for Science, Innovation and Technology

£330m

TSB's 2018 core banking migration cost the bank this much in the year of the incident, followed by a £48.65m fine from the FCA and PRA.

FCA/PRA final notice

60%

of CIOs say technical debt has grown noticeably in three years, already eating 20–40% of estate value and up to a fifth of new-product budget.

McKinsey, Tech debt: Reclaiming tech equity

74%

of code changes across 39 internal Google migrations were AI-generated, cutting total migration time by around half.

Google Research, arXiv:2504.09691

43%

of IT budget goes to running and maintaining legacy systems, against just 29% directed to transformative technology.

Capgemini Research Institute, via CIO Dive

42%

of a developer's week, roughly 17.3 hours, is lost to technical debt and maintenance rather than building.

Stripe, The Developer Coefficient

Where to start_

If any of this is familiar, the useful next step is small and doesn't need a business case.

Pick one system you're maintaining. Not the scariest, the second scariest: the one where a failure would be embarrassing rather than existential. Spend a few weeks characterising it: what it does, what depends on it, what tests exist, who's left who understands it. Do that with a harness rather than a discovery workshop, because the artefacts are reusable and the cost is a rounding error. Then modernise one slice of it out of money you already have.

You'll end the quarter holding evidence, which is what this conversation has always lacked. A real timeline, a real cost, a real before and after, from your own estate rather than a vendor deck. Every one of the three cases above gets easier to make once you have it, and the argument stops being about whether it's possible.

“Pick one system you're maintaining. Not the scariest, the second scariest.”

Nearform has been modernising systems people can't afford to break for over a decade, and we now build the harnesses that do it: the discovery, the characterisation tests, the transformation pipeline, and the evals that keep it honest. If you want to work out what a first thin slice would look like on one of your own systems, **get in touch.**

*"Modernisation
used to be quoted
in years and
tens of millions.
It's now
reasonable
to quote a first
tranche in weeks."*



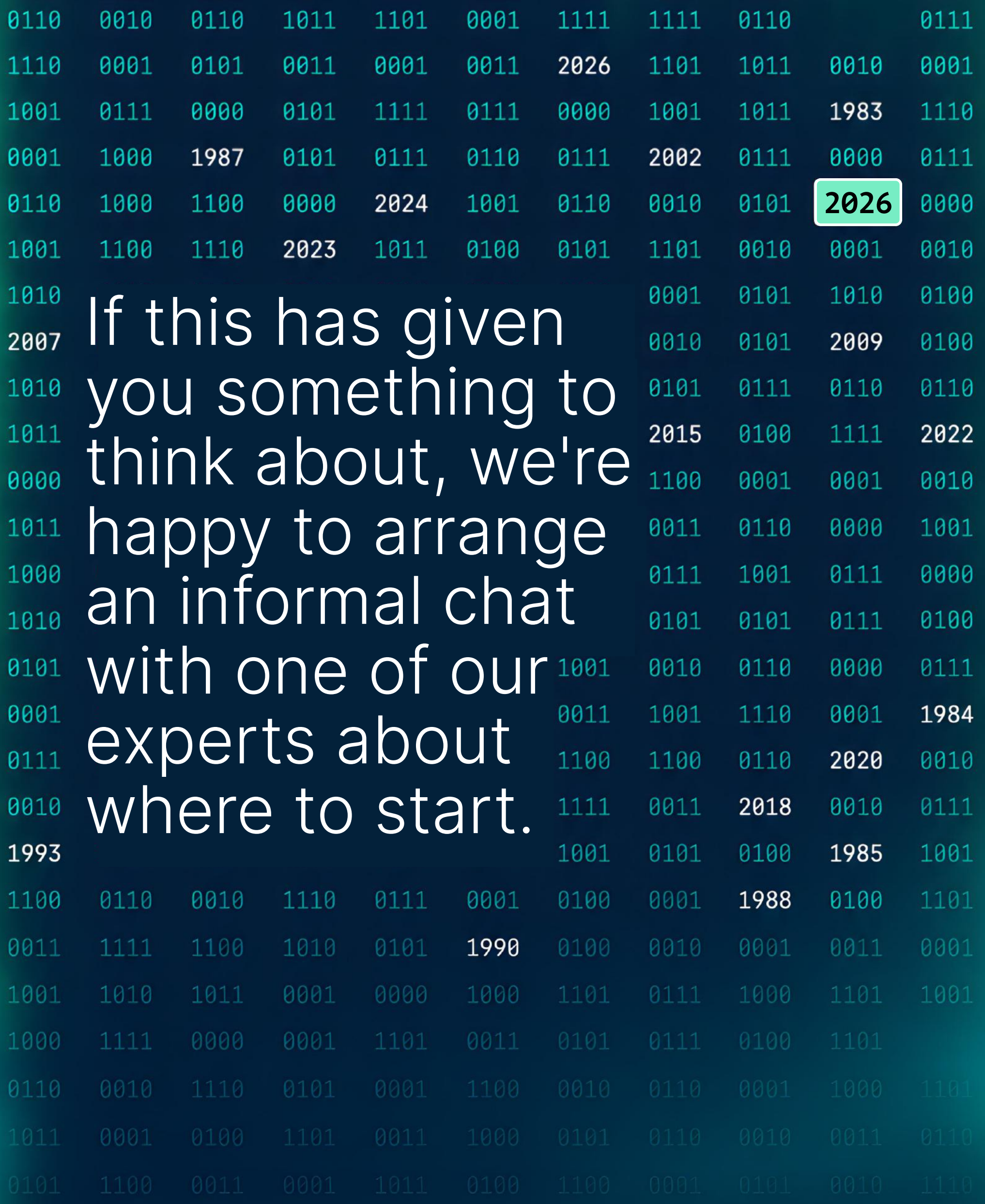
Sources_

Nearform client work

- [Reducing AML investigation from months to hours with AI agents](#), state-owned financial services provider
- [Condé Nast: automated content migration with 99.9% accuracy](#)
- [Setting solid foundations for real change](#), nonprofit insurance association
- [Modernisation de-risked: a smart, safe, rapid approach](#)

External

- [National Audit Office, Government cyber resilience](#) (29 January 2025), pp. 11–12, 34, 38–39
- [US Government Accountability Office, Information Technology: Agencies Need to Plan for Modernizing Critical Decades-Old Legacy Systems, GAO-25-107795](#) (17 July 2025)
- [FCA, TSB fined £48.65m for operational resilience failings](#) (December 2022), and TSB Banking Group Annual Report and Accounts 2018
- [Deloitte, The hidden drag, quantified: technical debt's penalty on value and growth](#) (27 March 2026), drawing on the 2026 Global Technology Leadership Study
- [McKinsey and Serviceware, Recalibrating CIO technology budgets for the AI era](#) (30 March 2026), data collected May–November 2025 across 17 global companies
- [Google Research, Accelerating code migrations with AI](#) and [Migrating Code At Scale With LLMs At Google, arXiv:2504.09691](#)
- [Deloitte, FASB Amends Guidance on the Accounting for and Disclosure of Software Costs \(ASU 2025-06\)](#) (September 2025)



2026

If this has given
you something to
think about, we're
happy to arrange
an informal chat
with one of our
experts about
where to start.

Get in touch at nearform.com/contact