

The IT Modernization Playbook

A decision framework for replacing what is holding you back — in the right order, for reasons you can defend.

Modernization is a sequencing problem, not a technology problem.

Most organisations know roughly which systems are hurting them. What they cannot answer is which one to touch first, what it costs to leave the rest alone for another year, and how to tell a genuine constraint from a loud complaint. This playbook gives you a repeatable way to answer all three.

What you will have at the end

- A complete inventory of your systems, scored on the four factors that actually drive modernization urgency
- A defensible ranked order of what to change, with the reasoning attached to each decision
- A clear Keep / Improve / Replace / Retire verdict for every system you own
- A realistic shape for the first ninety days, and an honest view of what it will cost to do nothing

Use it on your own, or bring the completed inventory to a conversation with whoever advises you. The work is in the scoring, and the scoring is yours to do.

1. Why modernization stalls

Modernization programmes rarely fail because the replacement technology was wrong. They fail earlier than that, in the choosing. Four patterns account for most of it, and you can recognise all four from the inside.

The loudest system wins

Attention follows complaint volume, not business risk. The system everyone grumbles about daily gets replaced while the quiet one with no support contract and a single person who understands it keeps running untouched. Irritation and exposure are different measurements, and only one of them shows up in a meeting.

Everything is urgent, so nothing is sequenced

When every system is labelled critical, the label stops carrying information. Work starts in several places at once, each strand moves slowly, and the programme is judged on the one that stalls most visibly. Sequencing is the single highest-leverage decision available, and it is usually the one made most casually.

The cost of waiting is never written down

Replacement has a number attached; the status quo does not. So the comparison is made between a known cost and an unstated one, and the status quo wins by default every year until something breaks badly enough to settle the argument. Putting a figure on another year of the current arrangement, even a rough one, changes which option looks expensive.

Replacing is treated as finishing

A system is counted as modernized the day the new one goes live, which is the point at which the old data, the undocumented integrations and the workarounds people built are still entirely in play. The work of actually retiring the old system is where modernization programmes quietly leave half their value behind.

The through-line: each of these is a decision-quality problem. None of them is fixed by choosing better software. All of them are fixed by scoring honestly before committing.

2. The four verdicts

Every system you own resolves to one of four verdicts. The point of the scoring in Section 3 is to reach the right one without argument.

Verdict	What it means	Reach it when
Keep	Leave it alone. Review on a set date, not continuously.	It does its job, it is supported, and nobody is working around it. Re-examine annually or when support status changes.
Improve	The system is sound; the way it is configured, integrated or operated is not.	The complaints are about speed, access, reporting or manual steps rather than the system's core function. Usually the cheapest real win available.
Replace	The system cannot be made to do what the business now needs.	The gap is structural, or support has ended, or the operating cost exceeds a credible replacement, or one person is the only reason it still runs.
Retire	Switch it off. Keep the data if you must; stop running the system.	Its function has moved elsewhere, duplicated into another system, or stopped mattering — and nobody has checked in a long time.

Retire is the verdict most often missed

Almost every inventory turns up at least one system that is still licensed, still patched, still backed up and no longer genuinely needed — kept alive because switching it off is nobody's job and the annual cost is small enough not to argue about. These are the cheapest wins in the whole exercise. Look for them deliberately, because they will not announce themselves.

Improve is the verdict most often skipped

Replacement is more satisfying to propose than reconfiguration, so systems get replaced when the actual fault was in how they were set up — and the replacement inherits the same fault, because nobody diagnosed it. Before any Replace verdict, state plainly what was tried and why Improve was ruled out. If that sentence is hard to write, you are not ready to replace.

3. The scoring model

Score every system on four factors, 1 to 5 each. The total out of 20 sets the order you work in. Score alone first, then with one other person who uses the system daily — the gaps between your two scores are where the useful conversation is.

Factor 1 — Business dependency

How much stops if this system stops? Score 5 if revenue, payroll, fulfilment or regulated activity halts within a day. Score 1 if a handful of people are inconvenienced and work continues. Be honest here: overstating dependency is the most common way inventories become useless.

Factor 2 — Fragility

How likely is it to fail or become unsupportable? Score 5 for out of support, unpatchable, running on hardware or an operating system past its end of life, or understood by exactly one person. Score 1 for current, vendor-supported and documented. Single-person knowledge counts as fragility whatever the technology underneath.

Factor 3 — Constraint on the business

Is this system preventing something the business is actually trying to do? Score 5 if a named initiative is blocked by it right now. Score 1 if it constrains nothing anyone has asked for. This factor is what separates modernization from maintenance — a fragile system that blocks nothing is a risk to manage, not a project to fund.

Factor 4 — Cost of another year

Licensing, hosting, support, the staff time spent on workarounds, and a sober estimate of incident cost. Score 5 if another year costs a material fraction of replacing it. Score 1 if it is genuinely cheap to keep. This is the factor organisations skip, and skipping it is why the status quo wins by default.

Total	What it means	Action
17–20	Acute. Fragile, depended on, blocking, and expensive to keep.	First in the sequence. Start now.
13–16	Real, but not yet acute.	Second wave. Schedule it with a date, not an intention.
9–12	Mixed signals — usually an Improve case, not a Replace one.	Diagnose before funding anything.
4–8	Not a modernization candidate.	Keep, or check whether it should be retired.

Read the factors, not just the total. A system scoring 4 on fragility and 1 on constraint is a different problem from one scoring 1 and 4, even though both total the same. The first needs risk management; the second needs a project. Treating them alike is how budget gets spent on the wrong thing.

4. The inventory worksheet

One row per system. Include everything that runs, including the spreadsheet that quietly became a system and the tool one department pays for on a card. Print this page or rebuild it in a sheet.

System	Owner	Dep.	Frag.	Constr.	Cost	Total	Verdict

Dep. business dependency · **Frag.** fragility · **Constr.** constraint on the business · **Cost** cost of another year. Each 1–5.
Verdict: Keep / Improve / Replace / Retire.

Before you score, make sure the list is complete

- ☐ Finance, payroll and billing systems — including anything run by an outside bookkeeper
- ☐ Customer records, wherever they actually live rather than where policy says they live
- ☐ File storage, in every location people genuinely use
- ☐ Email, calendars and the identity system that controls access to all of it
- ☐ Line-of-business applications, including any a single department bought independently
- ☐ Websites, and whatever hosts, backs up and secures them
- ☐ Backup and recovery — and the last date a restore was actually tested

-
- ☐ Networking and connectivity, including remote access
 - ☐ Anything running on a machine physically in your building
 - ☐ Spreadsheets that have become systems of record
 - ☐ Tools still being paid for that you believe nobody uses

5. Sequencing: what goes first

The ranked list from Section 3 is a starting point, not the final order. Four rules adjust it, and they override raw score.

Rule 1 — Identity and access come before almost everything

How people sign in, and what they can reach once they have, sits underneath every other system. Modernizing applications on top of a weak access layer means doing the access work later anyway, once per application, at higher cost. If identity scores anywhere near the top, it goes first regardless.

Rule 2 — Fix what you cannot recover from before what merely annoys

An untested backup is a higher priority than a slow application, however much louder the complaints about the application are. Recoverability is the floor the rest of the programme stands on. Test a restore before you schedule anything else.

Rule 3 — Take the retirements early

Switching off systems nobody needs costs little, frees budget and attention, and produces a visible result early in a programme that will otherwise show nothing for months. Do these in parallel with the first real project rather than queueing them behind it.

Rule 4 — One significant change at a time, per team

Capacity to absorb change is a per-team constraint, not an organisational one. Two teams can each take a change simultaneously; one team taking two will do neither properly, and the failure will be blamed on the technology rather than the scheduling.

Write the sequence down with dates, and write down what you are deliberately not doing this quarter. The second list is what stops the first one being reopened every time someone complains.

The shape of a sound first ninety days

Weeks	Focus	What exists at the end
1–2	Complete the inventory and score it with the people who use each system.	A scored list, and a short note on every disagreement about a score.
3–4	Test recovery on the systems scoring highest on dependency.	A tested restore, and an honest record of what did not work.
5–8	Start the first project. Begin retirements in parallel.	One change genuinely underway; at least one system switched off.
9–12	Finish the retirements. Decide the second wave on evidence.	A dated plan for the next quarter, and a decision you can defend.

6. Five ways this goes wrong

Scoring in a room with only one perspective

A system scored only by the people who maintain it, or only by the people who use it, produces a confident and misleading number. Score it twice and treat the gap as the finding rather than something to average away.

Replacing before diagnosing

If you cannot write one clear sentence explaining why Improve was ruled out, the Replace verdict is not ready. The replacement will inherit the undiagnosed fault and the same complaints will return against a new system and a spent budget.

Counting go-live as done

Budget the decommissioning work in the same plan as the implementation, or it does not happen. Running both systems indefinitely costs more than either and leaves the risk you were trying to remove exactly where it was.

Letting the loudest system reset the sequence

The sequence will be challenged, usually by whoever is most inconvenienced this week. This is what the written reasoning is for: the answer is the scoring, not the seniority of whoever is asking.

Modernizing a process that should have been removed

Before replacing the system that supports a process, confirm the process is still required. A meaningful share of what gets carefully rebuilt existed only because the previous system made it necessary.

What to ask before you commit to any replacement

- ☐ What exactly does this do that the current system cannot? One sentence.
- ☐ What have we tried on the current system, and why was it not enough?
- ☐ Who owns this after go-live, and is that in their actual workload?
- ☐ What happens to the data in the old system, and who decides what is kept?
- ☐ What is the date the old system is switched off, and who is accountable for it?
- ☐ What does leaving as-is cost over the next twelve months?
- ☐ If this is not finished, what state are we left in — and is that recoverable?
- ☐ How do we get our data out of the new system if we leave in three years?

Where to take this next. The inventory you have just filled in is the hard part, and it is now yours regardless of who does the work. If you want a second read on the scoring or the sequence, bring the completed worksheet to the conversation — it is a far better starting point than a list of complaints.

General business guidance only. This playbook is a planning aid, not professional, legal, financial, security or regulatory advice, and it does not assess compliance with any standard or regulation. Modernization decisions involving personal data, regulated records, payment systems, safety-critical operations or contractual obligations carry requirements this framework does not evaluate. Take qualified advice on your own circumstances before acting. © Atomix Technology. Shared freely — please pass it on unaltered.