SUPPLY CHAIN RESEARCH · EDITORIAL FEATURE

The Rip-and-Replace Trap

Why Modernization Programs Fail, and What to Do Instead

Replacing a core system all at once is the most expensive and most dangerous way to modernize, and the record of failure is long and costly. This is why rip-and-replace fails so often, and how to modernize incrementally, in production, without betting the business on a single go-live.

Independent, vendor-neutral research for supply chain and technology leaders

Supply Chain Research | supplychainresearch.com | 2026

Contents

A decision framework for the AI era

01 Executive summary

The instinct to rip out an aging core system and replace it wholesale is one of the most expensive instincts in enterprise technology. It is understandable: the old system is slow, brittle, poorly understood, and increasingly hard to staff, and a clean replacement promises a single modern platform, a single source of truth, and an end to the accumulated compromises of decades. But the promise is a trap. The historical record of big-bang replacement is a record of overruns, disruptions, and outright disasters, and the reason is structural rather than incidental: a wholesale replacement asks an organization to reproduce poorly documented behavior, introduce architectural change, and deliver new capability all at once, while betting the continuity of the business on a single go-live working correctly on a single day. When that bet fails, and the evidence says it fails more often than it succeeds, the cost is measured not in schedule slippage but in unshipped orders, uncloseable books, and, in the worst cases, the survival of the enterprise.

This article makes the case against rip-and-replace as a default and, honestly, in favor of a more disciplined alternative that the best modernizers have adopted: incremental modernization, in which a new system grows around the old one, piece by piece, while the business stays in production the entire time. We assemble the evidence on how often big-bang programs fail and why, walk through the most instructive failures of the past three decades, anatomize the recurring pattern beneath them, and then set out the alternative in practical detail, the strangler-fig approach, composable and modular architecture, phased and hybrid migration, and the sequencing, governance, and decision framework that make incremental modernization work. We are not against modernization; the cost of clinging to a failing legacy system is real and rising. We are against the specific, seductive, and historically ruinous method of doing it all at once.

70%+
of recently implemented ERP initiatives that Gartner expects will not fully meet their original goals by 2027

215%
the average cost overrun Panorama found on discrete-manufacturing ERP projects

USD 2.5B
the losses that preceded Target's exit from Canada after a big-bang system launch

The argument in brief

- **Big-bang replacement fails more often than it succeeds.** Across independent research, the majority of large ERP and modernization programs miss their goals, overrun their budgets, or disrupt operations at go-live.
- **The failure is structural.** A wholesale cutover concentrates all the risk on a single date and asks the organization to replicate, re-architect, and innovate at once, delivering no value until the end.
- **The costs are operational, not just financial.** The famous failures share a signature: orders that cannot ship, books that cannot close, and customers who cannot be served, because everything depended on the same go-live.

- **Incremental modernization is the disciplined alternative.** The strangler-fig approach grows the new system around the old, in production, so value arrives continuously and each step is small and reversible.
- **Match the method to the system.** Rip-and-replace is occasionally right, for a common process on a poorly fitting system, but for anything that differentiates the business, incremental modernization preserves the value while cutting the risk.

02 What rip-and-replace means, and why it is tempting

Rip-and-replace, in the sense this article uses, is the wholesale replacement of a core operational system, an enterprise resource planning platform, a warehouse or transportation management system, an order or billing engine, with a new system that goes live across all modules and locations at or near a single cutover. It is often called a big-bang implementation, because the organization stops using the old system and starts using the new one in one decisive move, rather than migrating gradually. The appeal is real and worth stating plainly, because a fair argument against the approach must acknowledge why so many capable organizations choose it.

The first attraction is the clean slate. A legacy system carries decades of accumulated customization, workarounds, and technical debt, much of it undocumented and understood by fewer people every year as the original builders retire. A wholesale replacement promises to sweep all of that away and start fresh on a modern platform, and the appeal of that fresh start, after years of patching, is powerful. The second attraction is the single source of truth. Organizations that have grown through acquisition often run many instances of many systems, and the prospect of consolidating them into one platform, with one set of data and one set of processes, is truly valuable. The third is speed, or the appearance of it. A big-bang cutover avoids the expense and complexity of running old and new systems in parallel, and reaches the promised end state, in theory, sooner than a phased migration that stretches over years. The fourth is the forcing function. A single go-live date concentrates the organization's attention and creates the urgency that a longer, gentler migration can lack.

Each of these attractions is real, and none of them is wrong to want. The problem is not the goals, which are sound, but the method, which pursues them in the riskiest possible way. The clean slate requires reproducing behavior no one fully understands; the single source of truth requires migrating decades of data of uncertain quality; the speed is often illusory, because the parallel-running cost avoided is small next to the disruption cost incurred; and the forcing function, applied to a bet this large, forces the organization toward a cliff. The sections that follow show how reliably this method fails, and then show that the same goals can be reached by a method that does not stake the business on a single day.

Understanding why capable organizations choose the riskier method is part of resisting it. Beyond the four attractions already named, a set of quieter dynamics pushes toward the big bang. A wholesale replacement is easier to approve as a single capital project with a defined end than an open-ended program of incremental change, so the budgeting process itself can favor it. System integrators and software vendors often prefer the larger, faster engagement a big-bang program represents, and their incentives shape the advice a buyer receives. And there is a prestige in the bold, decisive replacement that an incremental migration lacks, which can make the riskier path the more attractive one to sponsor. None of these dynamics is a good reason to accept the risk, and naming them helps a leader recognize when a decision is being driven by them rather than by the merits.

One attraction deserves a specific warning, because it is the seed of so many failures: the belief that the organization's way of doing things is so distinctive that the new system must be bent to fit it. Sometimes that is true, and the distinctiveness is real and worth preserving. Far more often it is not, and the

insistence on reproducing every existing process in the new system drives an ever-deepening customization that inflates cost, delays the schedule, and undermines the standardization the replacement was meant to deliver. The discipline of separating the processes that truly differentiate the business from the ones that are merely habitual is one of the most valuable an organization can bring to a modernization, and its absence, as later cases show, is one of the most reliable predictors of failure.

It also helps to place rip-and-replace within the wider menu of modernization options, because treating replacement as the only path is itself part of the trap. Modernization is a spectrum. At the lightest touch, an organization can re-host a system, moving it to new infrastructure with minimal change, or re-platform it, making modest adjustments to take advantage of a new environment. More deeply, it can re-factor the system, improving its internal structure without changing its behavior, or re-architect it, changing how it is built while preserving what it does. Wholesale replacement, rebuilding or buying the system anew, sits at the far end of this spectrum as the most disruptive option of all. The mistake many organizations make is to leap to that far end by default, when a lighter-touch option, or an incremental combination of several, would reach the goal at a fraction of the risk. The question is never simply whether to replace, but which point on the modernization spectrum the situation actually calls for.

03 The evidence: how often big-bang modernization fails

The case against rip-and-replace begins with the base rates, which are sobering, shown in Figure 1. Large enterprise-system programs fail to meet their goals more often than they meet them, and the finding is consistent across independent sources. Gartner has predicted that more than seventy percent of recently implemented enterprise resource planning initiatives will not fully achieve their original business goals by 2027. Panorama Consulting's research on discrete-manufacturing ERP found that roughly seventy-three percent of projects miss their objectives. Industry analyses put the share of implementations that fail on the first attempt at around half, and the share that experience operational disruption at go-live at roughly the same, while only about thirty percent of projects are delivered on time and on budget. Definitions of failure vary across these studies, and the figures should be read as directional rather than precise, but they converge on an uncomfortable conclusion: the base rate of success for a large wholesale program is low.

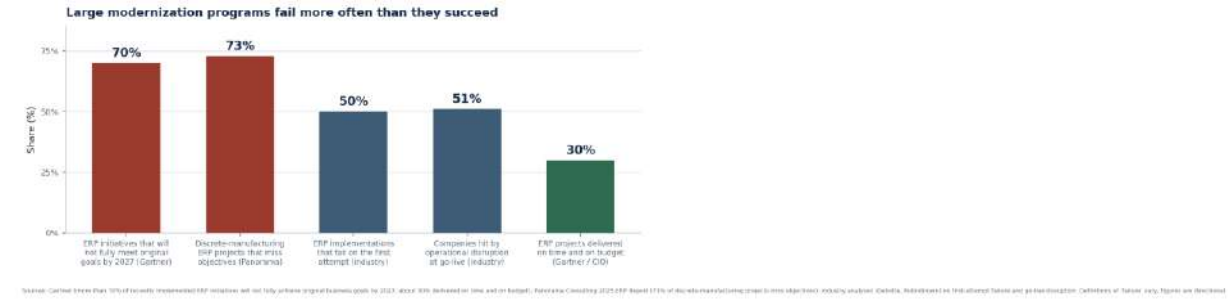

Figure 1. Failure and disruption rates for large enterprise-system programs across independent sources. Definitions vary, but the pattern is consistent: these programs miss their goals more often than they hit them.

The financial dimension is worse, because when these programs exceed their budgets, they tend to do so dramatically, shown in Figure 2. Panorama found an average cost overrun of around two hundred fifteen percent on discrete-manufacturing ERP projects, a figure that means the typical over-budget project cost roughly three times its estimate. The same body of research found that among projects that exceeded budget, about half required additional technology that had not been planned for, around forty percent had underestimated the staffing the project would demand, and around forty percent encountered organizational issues that should have been visible at the outset. These are not random misfortunes; they are the predictable consequences of a method that commits to a full replacement before the organization understands what the replacement will require.

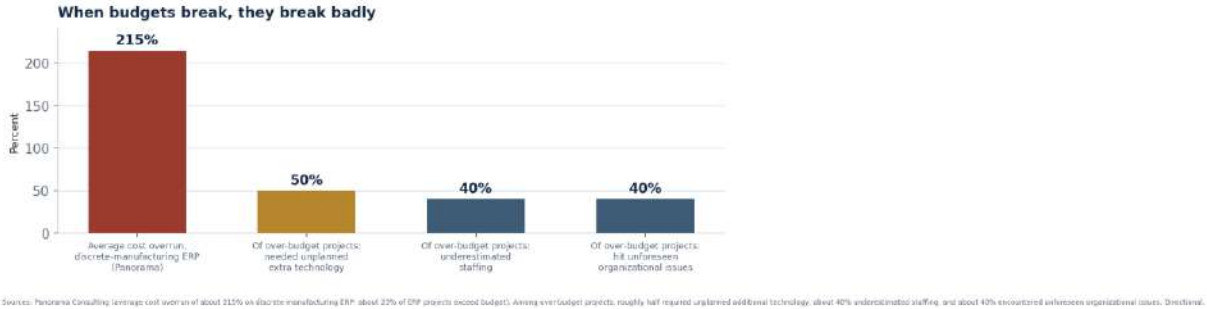

Figure 2. Cost overruns and their drivers. When large programs break their budgets, they break them badly, and the causes are consistent: unplanned technology, underestimated staffing, and unforeseen organizational issues.

It is worth being fair about what these numbers do and do not show. They do not show that modernization is futile; the same research finds that organizations which succeed report substantial benefits, and the cost of never modernizing is real and rising. Nor do they isolate the big-bang method as the sole cause, since many of these programs failed for reasons, change management, data quality, governance, that can afflict any approach. What they do show is that large wholesale programs carry a high base rate of failure and a high severity when they fail, which means that a rational organization should treat the decision to attempt one as a decision that most often ends badly, and should ask, before committing, whether the goal can be reached by a method with a better record. The following sections show what failure looks like in practice, why the pattern recurs, and what that better method is.

A word on what these failure rates mean is warranted, because the definition of failure varies and matters. Few of these programs fail in the sense of producing nothing usable; most produce a working system eventually. They fail in the sense that matters to a business: they exceed their budgets, overrun their schedules, disrupt operations, or fall short of the benefits that justified them. A program that goes live a year late, at three times its budget, after a quarter of disrupted operations, and delivers two-thirds of the promised benefit, is counted a failure by most of these studies, and rightly, because the business case that authorized it has not been met. The figures are not claims that modernization never works; they are claims that large wholesale programs routinely cost far more, take far longer, and deliver far less than they promised, which is a different and more useful point.

Fairness also requires stating the other side of the ledger, because the same research that documents the failures documents the rewards of getting it right. Organizations that implement successfully report broad improvements, in efficiency, in visibility, and in the quality of their decisions, and the gains can be substantial. The point of this article is not that the destination is not worth reaching; it plainly is. The point is that the wholesale method is a needlessly dangerous way to reach it, and that the same rewards are available by a safer route. A reader persuaded that big-bang replacement fails too often should not conclude that modernization is futile, but that it should be pursued by the incremental means the later sections describe, which reach the same rewards while avoiding the failures this section has catalogued.

It is worth acknowledging the pressure that makes modernization unavoidable, because the case against the big bang is not a case for standing still. Legacy systems impose a mounting tax: the technical debt accumulates and compounds, the security exposure of unsupported software grows, the integrations become more fragile, and the pool of people who can maintain aging technology shrinks year by year as they retire and few replacements learn the old skills. These pressures are real and rising, and they are why organizations feel compelled to act. The argument here is not that they should ignore the pressure, which would be its own kind of negligence, but that they should relieve it by the incremental means that address the same debt, security, and talent problems without staking the business on a single cutover. The urgency to modernize is legitimate; it is the leap to the wholesale method that is not.

04 A graveyard of cautionary tales

Statistics establish the base rate; the individual failures show what the base rate means in practice, and they are worth recounting because their pattern is so consistent, shown in Figure 3. These are among the most instructive enterprise-system failures of the past three decades, and in nearly every case the immediate cause was a wholesale cutover that concentrated risk on a single event.

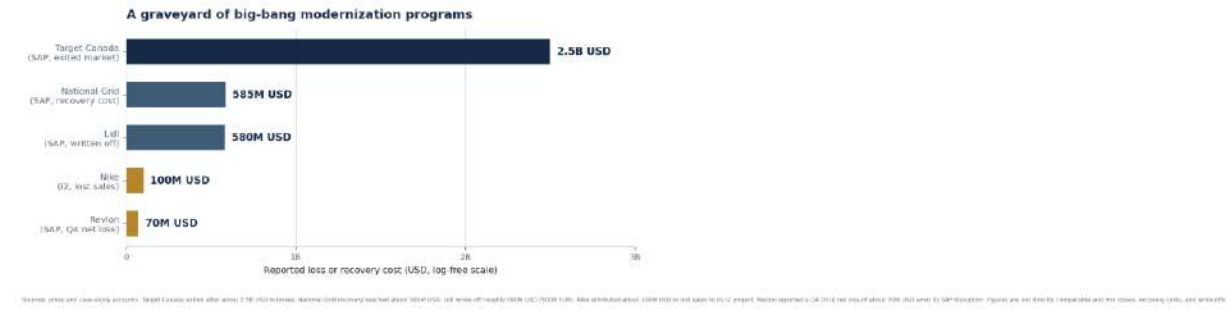

Figure 3. A selection of well-documented big-bang failures and their reported costs. The figures mix losses, recovery costs, and write-offs and are not directly comparable, but the scale is the point.

Hershey, 1999

The Hershey Company set out to replace its legacy systems with a new enterprise platform, and, partly to beat a looming deadline, compressed a schedule that had been estimated at around forty-eight months into roughly thirty. Testing was truncated and the cutover was timed, fatefully, to the run-up to Halloween, the company's peak season. When the new system went live, the company could not process roughly one hundred million dollars of candy orders it had the inventory to fill, reported a nineteen percent drop in quarterly profit, and saw its stock fall around eight percent. The lesson is almost too obvious in hindsight: do not compress the schedule of a wholesale replacement, and do not flip the switch during your busiest season.

Nike, 2000 and 2001

Nike invested around four hundred million dollars in a new supply chain and enterprise system, and the demand-planning component's rollout went badly enough to produce roughly one hundred million dollars in lost sales, a twenty percent dip in the stock, and a set of shareholder lawsuits. The company's chief executive summed up the sentiment of every leader who has funded a failed big-bang program with a single, famous question: this is what you get for four hundred million dollars. Nike eventually recovered and rebuilt more deliberately, which is itself part of the lesson.

Target Canada, 2013 to 2015

Target's expansion into Canada depended on standing up a new instance of a major enterprise platform to run a supply chain across more than a hundred new stores, and the data that fed it, entered under enormous time pressure for tens of thousands of products, was accurate an estimated thirty percent of the time, against the ninety-eight to ninety-nine percent norm in the company's home operations. The result was empty shelves, pricing chaos, and an inability to move product, and less than two years after entering the market Target withdrew, having accumulated around two and a half billion dollars in losses

and closed its entire Canadian operation. It is the starkest illustration of a truth this article returns to: a wholesale system launch is only as good as the data migrated into it, and rushed data is the most common way these programs fail.

Revlon, National Grid, and Lidl

The pattern recurs across industries. Revlon's rollout of a new enterprise platform, following its acquisition of another cosmetics company, disrupted its own manufacturing badly enough that it could not fulfill an estimated sixty-four million dollars of orders, contributed to a quarterly net loss of around seventy million dollars, and led, unusually, to lawsuits from its own shareholders. National Grid went live with a new platform days after a superstorm had devastated its service area, and the result was payroll errors, fifteen thousand vendor invoices that could not be processed, and a collapse in financial reporting severe enough to threaten the company's access to short-term financing; the recovery required hundreds of contractors, ran to roughly five hundred eighty-five million dollars, and ended in a lawsuit against the integrator settled for seventy-five million. Lidl spent around seven years and roughly half a billion euros attempting to replace its inventory system, and ultimately abandoned the effort and reverted to its legacy system, undone in large part by an unwillingness to adapt its processes to the new platform, which forced ever-deeper customization. Each of these is a different company in a different industry, and each tells the same story: a wholesale replacement, a single high-stakes cutover, and a failure that reached the operations, the finances, and in some cases the reputation and legal standing of the enterprise.

More from the graveyard

The list could be extended almost indefinitely, and a few further cases round out the pattern. A large brewing company, consolidating several instances of an enterprise platform into one, went live with dozens of known critical defects and thousands of lesser ones, and abandoned the effort within months. An automotive fleet-management company sank close to a hundred million euros into a platform that never went live at all, later citing the monolithic nature of the system as incompatible with its need to move quickly. And outside the commercial sphere, a major government health-insurance portal launched to such catastrophic failure that, on its first day, a handful of users out of millions managed to complete registration, at an eventual cost measured in the hundreds of millions. Different sectors, different systems, the same lesson: a wholesale launch that has not been proven in production before the switch is thrown is a bet the organization frequently loses.

Reading these cases together, the common thread is unmistakable, and it is not the software. In every instance the platform itself was capable, proven technology used successfully by other organizations. What failed was the method: a compressed schedule, a rushed and unvalidated data migration, an insistence on customization, a deferral of ownership to vendors, a go-live timed without regard to the calendar or the organization's readiness, and above all the concentration of every one of these risks onto a single cutover. The consistency of the pattern is what makes it useful, because a pattern this reliable can be anticipated and avoided. The failures were not bad luck; they were the predictable output of a method, and a different method would have produced a different result.

05 Anatomy of a big-bang failure

The failures are so consistent that they can be anatomized, and understanding the anatomy is what allows an organization to avoid it. The root of the problem is structural, and it is captured in Figure 4: a big-bang cutover concentrates all of the risk of the program onto a single event. Every module, every location, and every process moves to the new system at once, which means the continuity of the entire business depends on that one cutover working correctly. There is no partial success. If the general ledger will not reconcile, the books cannot close; if the inventory records are wrong, orders cannot ship; and because everything went live together, a failure in any one area can halt the whole.

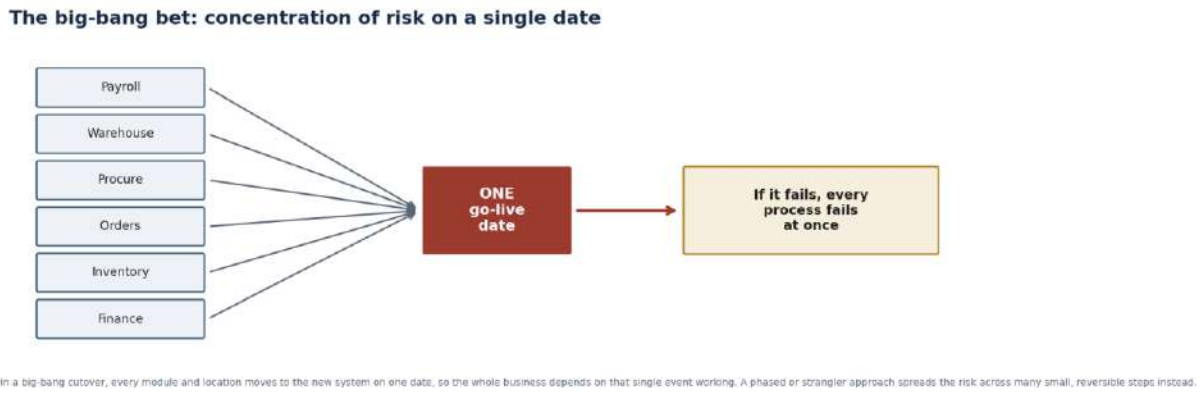

Figure 4. The structural flaw of the big-bang approach: all modules and locations depend on a single go-live, so a failure anywhere can halt everything. An incremental approach spreads the risk across many small, reversible steps.

Beneath that concentration of risk sit three compounding difficulties that a wholesale program takes on simultaneously, any one of which is hard and all three of which together are treacherous. The first is replication: the new system must reproduce the behavior of the old one, but that behavior is encoded in decades of customization and institutional habit that is poorly documented and imperfectly understood, so the organization is trying to rebuild something it cannot fully describe. The second is architectural change: the new platform works differently from the old, which means processes must change, and the organization must absorb that change across the whole business at once. The third is innovation: the program is usually expected to deliver new capability as well, so the team is inventing while it replicates and re-architects. Doing any one of these carefully is demanding; doing all three at once, against a fixed go-live date, is the setup for the failures the previous section recounted.

Two further factors turn difficulty into disaster with grim regularity. The first is data. A new system is only as good as the data migrated into it, and legacy data is almost always dirtier than anyone expects, so a program that underestimates the effort of cleansing and validating it, as Target Canada did, launches on a foundation that cannot support it. The second is people. The organizational side of the change, the training, the process redesign, the governance, the management of resistance, is repeatedly the largest cause of failure, and it is precisely the side that a schedule-driven big-bang program compresses first when time runs short. The technology is rarely the thing that fails; the replication, the data, and the people are, and the big-bang method loads all of them onto a single date.

At the center of the replication problem sits a difficulty that deserves its own name: the iceberg of undocumented behavior. A legacy system that has run a business for decades encodes an enormous amount of accumulated logic, the special cases, the exceptions, the quiet rules that handle the situations the business has encountered over the years, and most of that logic exists nowhere but in the code and in the memories of the people who built it. As those people retire, the knowledge leaves with them, so an organization attempting a wholesale replacement is often trying to reproduce behavior that no living person fully understands and no document fully describes. The visible requirements are the tip; the submerged mass of undocumented behavior is what sinks the replacement, because the new system, built to the visible requirements, fails on the cases the invisible ones handled.

The second predictable turn from difficulty to disaster is the compression of testing under schedule pressure. A wholesale program that falls behind, as most do, faces a choice between moving the go-live date and cutting the work that remains, and the work most easily cut, because its absence is not felt until after launch, is testing. So the program that is already struggling to reproduce behavior it does not fully understand launches that reproduction without adequately testing it, and discovers the gaps in production, at the worst possible moment. The Hershey compression of a forty-eight-month schedule into thirty is the archetype, but the pattern is general: the big-bang method, by placing everything on one date, creates exactly the schedule pressure that leads organizations to cut the testing that might have caught the failure.

A further multiplier of big-bang risk is the web of integrations that surrounds any core system. A modern enterprise system does not stand alone; it exchanges data with dozens of other systems, warehouses, carriers, banks, customers, suppliers, and tax authorities, each through an interface that must be rebuilt and revalidated for the new platform. In a wholesale cutover, all of these interfaces must work on the first day, which multiplies the number of things that must go right simultaneously and the number that can bring operations down if they go wrong. An incremental migration rebuilds and proves these interfaces a few at a time, so an integration problem surfaces in isolation and is fixed before the next is attempted, rather than all of them being tested for the first time together at the moment the business can least afford a failure.

Underlying the concentration of risk is a psychological reality that deserves naming: a big-bang cutover offers no dress rehearsal. However much testing precedes it, the first time the whole business actually runs on the new system, at full volume, with real customers and real money, is the go-live itself, which means the most consequential performance is also the first. Every other high-stakes human endeavor that can rehearse, does, precisely because performing for the first time under maximum stakes is known to be dangerous. The incremental approach is, in effect, a way to rehearse, because each small cutover is a low-stakes performance that builds the capability and surfaces the problems before the stakes are high. The big bang forgoes the rehearsal and stakes everything on a first performance, which is a peculiar way to treat an event on which the business depends.

06 The hidden cost of the value freeze

Even when a big-bang program does not fail outright, it carries a cost that rarely appears in the business case: the long freeze during which the organization pours money and effort into the replacement and receives no new value in return until the end. This is the value freeze, and it is one of the strongest arguments for an incremental approach, illustrated in Figure 5. A wholesale program typically delivers little or no business benefit during the months or years of the build, because nothing goes live until everything is ready, and then it delivers its value in a single risky step at go-live, a step that, as the previous sections showed, often includes a period of disruption before any benefit is realized.

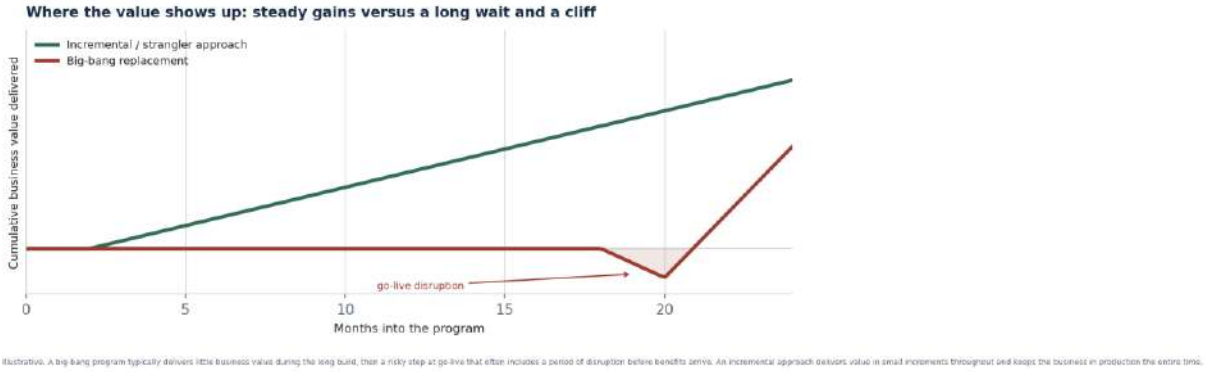

Figure 5. Where value shows up. A big-bang program delivers little during the long build and then a risky step at go-live, often with disruption before benefit. An incremental approach delivers value throughout and stays in production.

The contrast with an incremental approach is stark. When modernization proceeds piece by piece, each increment that goes live delivers its value immediately, so benefit accrues throughout the program rather than arriving all at once at the end. This difference is not merely about the timing of returns, though the time value of earlier benefit is real. It changes the risk profile of the entire effort. An incremental program that runs into trouble has already delivered the value of its completed increments and can pause, reassess, or change course without losing everything, whereas a big-bang program that runs into trouble before go-live has delivered nothing and must either push through to a risky launch or write off the entire investment. The value freeze, in other words, is also a risk concentration: it puts all of the program's return, like all of its risk, on the far side of a single event.

There is a strategic cost as well. During the years of a wholesale program, the organization's technology agenda is largely consumed by the replacement, which crowds out the smaller, faster improvements that might have delivered value in the meantime and starves the rest of the business of technology attention. The opportunity cost of that consumed capacity is invisible in the program's budget but real in the organization's competitiveness, and it compounds the longer the program runs. An incremental approach, by contrast, keeps delivering improvements to the rest of the business even as the core modernization proceeds, because it does not demand that the entire technology function hold its breath until a distant go-live. The freeze is a cost the business case for rip-and-replace almost never counts, and it is one of the clearest reasons to prefer a method that delivers value as it goes.

The freeze exacts a human cost as well as a financial one. A multi-year program that delivers nothing until its end is demoralizing to the people working on it, who labor for years without the reinforcement

of shipped value, and wearing on the wider organization, which waits and pays without seeing benefit. Talented people leave programs that never seem to deliver, which drains exactly the expertise the program most needs, and the departure of key people from a long wholesale program is itself a common contributor to failure. An incremental approach, by delivering visible value throughout, sustains the morale of the team and the confidence of the organization, which is not a soft consideration but a material one, because the sustained commitment that any modernization requires is far easier to maintain when progress is continuously visible than when it is deferred to a distant and uncertain go-live.

The freeze also creates the conditions for one of the most destructive dynamics in troubled programs: the escalation of commitment. As a wholesale program consumes years and hundreds of millions without delivering, the sunk cost becomes an argument for continuing, because abandoning the effort means writing off everything spent, and each additional increment of spending is justified as necessary to protect the investment already made. This is how programs that should be stopped are instead pushed toward a doomed go-live, and how the losses compound past the point where a clear-eyed assessment would have called a halt. An incremental approach is far less vulnerable to this trap, because its completed increments have already delivered value that is not lost if the remainder is paused, so the decision to stop or change course is not an all-or-nothing write-off but an ordinary reallocation of effort.

The freeze carries a competitive cost that is easy to overlook because it does not appear on any invoice. While an organization spends two or three years consumed by a wholesale replacement that delivers nothing until its end, its competitors continue to ship improvements, adopt new capabilities, and respond to their markets, and the gap that opens during the freeze can take years to close even after the replacement finally lands. In fast-moving markets, a multi-year internal preoccupation with a system replacement is a strategic liability regardless of whether the replacement ultimately succeeds, because the opportunity cost of the frozen years is paid to competitors who were not similarly immobilized. An incremental approach, by keeping the organization able to deliver improvements throughout the modernization, avoids handing competitors that window, which is a benefit that belongs in the business case even though it never appears in the program's own budget.

07 The strangler-fig alternative

The alternative to the big bang has a memorable name and a sound architecture behind it. The strangler-fig pattern, named by the software thinker Martin Fowler in 2004 after the strangler fig trees he had observed in the rainforests of Queensland, describes an incremental approach in which a new system is built around an existing one and gradually takes over its functions, piece by piece, until the old system can be retired. The fig germinates in the canopy, grows down and around its host over years, and eventually stands in the host's place; software modernized this way grows the same slow, safe way, wrapping the legacy system and replacing it from the outside in, as Figure 6 illustrates.

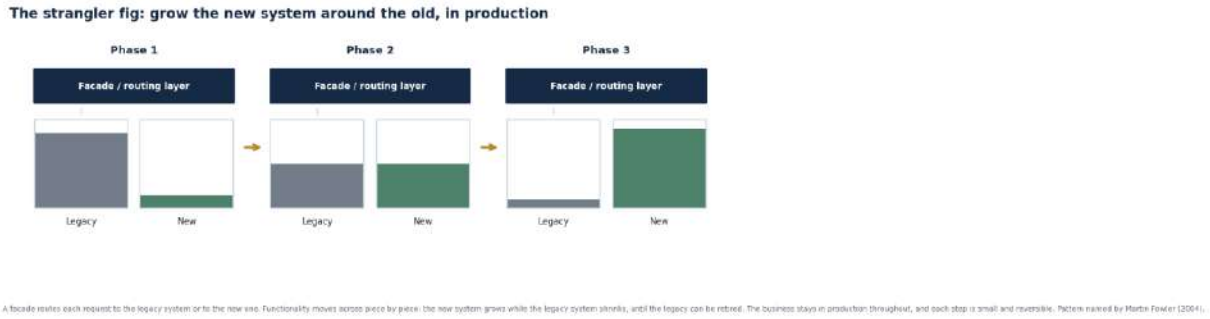

Figure 6. The strangler-fig pattern. A routing layer directs each request to the legacy or the new system; functionality moves across piece by piece, the new system grows while the legacy shrinks, and the business stays in production throughout.

The mechanism is simple and is the key to why the approach is safe. A routing layer, often called a facade or an interception layer and typically implemented as a reverse proxy or an interface gateway, is placed between the users and the systems. Initially it routes every request to the legacy system, so nothing has changed from the user's point of view. Then, one function at a time, the new system is built to handle a piece of the work, and the routing layer begins sending that piece to the new system while continuing to send everything else to the legacy one. Function by function, the traffic shifts, the new system grows, and the legacy system shrinks, until nothing is left routed to the old system and it can be switched off. At every moment in this process the business is fully in production, and every step is small enough to be tested in isolation and reversed if it goes wrong.

The advantages follow directly from the mechanism and address, point for point, the failures of the big bang. Because functionality moves across in small pieces, the risk is never concentrated on a single event; a problem with one migrated function affects one function, not the whole business, and can be rolled back by redirecting the routing layer. Because the business stays in production throughout, there is no value freeze; each migrated piece delivers its benefit as it goes live. Because the migration is incremental, the organization learns as it proceeds, applying the lessons of each step to the next, rather than betting everything on assumptions made at the outset. And because the legacy system keeps running until each piece is proven, the approach never asks the organization to reproduce behavior it does not understand in a single leap; it can study the old function, replace it, verify the replacement against the original, and only then move on. The strangler fig is not a compromise or a workaround; for a complex system that a business depends on, it is the architecturally correct way to modernize, and it is what the most disciplined modernizers now do.

One advantage of the strangler-fig approach is worth drawing out in full, because it directly answers the replication problem that sinks wholesale programs: the ability to verify each new function against the old one before relying on it. Because the legacy system keeps running as the new system is built, an organization can route a function's work through both systems in parallel, compare the outputs, and confirm that the new function reproduces the old one's behavior, including on the awkward cases, before it switches the traffic over. This comparison, sometimes run silently in the background for a period before cutover, converts the terrifying uncertainty of a wholesale launch, in which the reproduction is tested for the first time in production, into a measured confirmation performed while the old system is still carrying the load. It is the single most powerful risk-reduction technique the incremental approach offers, and the big bang, having switched off the old system, cannot use it.

Honesty about the approach requires acknowledging where it is hard. The strangler fig is most straightforward when the legacy system can be cleanly divided at its seams, and hardest when it cannot, when a single shared database underlies everything, or when the components are so tightly coupled that no function can be separated without disturbing the rest. In such cases the early work of establishing seams, sometimes by first untangling the data or introducing interfaces within the monolith, is substantial, and the migration is slower to begin. This is real, and it is why the approach demands real architectural skill rather than mere patience. But the difficulty of dividing a tangled monolith is an argument for investing in the division, not for the wholesale replacement, because the same entanglement that makes the strangler fig hard makes the big bang far more dangerous, and the incremental path at least confronts the entanglement one piece at a time rather than all at once.

The pattern is not merely theoretical; it has been applied to exactly the kinds of systems this article concerns. Practitioners have documented strangler-fig migrations of retail systems, telecommunications billing platforms, and financial transaction processors, in which a facade was placed in front of a legacy system and functionality was moved across piece by piece over months or years while the business ran without interruption. These accounts consistently report the same benefits the mechanism predicts: risk contained to one function at a time, value delivered as each piece went live, and the ability to pause or adjust as circumstances changed. They also report the same challenge, the discipline required to finish rather than stall, which is the trade the approach asks an organization to make: sustained commitment in exchange for dramatically lower risk. For a system a business depends on, it is a trade worth making.

08 Composable and modular architecture

The strangler-fig pattern is a migration strategy; composable architecture is the target state that makes such migrations possible in the first place, and increasingly makes future ones unnecessary. The core idea is to build capability as a set of modular, loosely coupled components, connected through well-defined interfaces, rather than as a single monolithic system in which everything is entangled with everything else. A composable architecture, often associated with microservices and with the principles sometimes labeled with the shorthand for modular, interface-first commerce and enterprise systems, replaces the one-big-system model with a collection of smaller systems that can be built, replaced, and upgraded independently.

The connection to the argument of this article is direct. The reason a legacy system is so hard to replace, and the reason its replacement must so often be attempted all at once, is precisely that it is monolithic: because everything is entangled, no piece can be replaced without touching the whole, so the organization feels forced into a wholesale cutover. A composable architecture dissolves that trap. When capability is modular, a single component, the tax engine, the pricing service, the shipping calculator, can be replaced on its own without disturbing the rest, which means modernization becomes a continuous activity of upgrading components rather than a periodic trauma of replacing systems. The strangler-fig migration is, in effect, the process of converting a monolith into a composable architecture one function at a time, and the composable end state is what ensures the organization never again finds itself trapped into a big-bang replacement.

The benefits of reaching that end state are well documented, shown in Figure 7. Research on enterprise modernization programs completed in recent years has found infrastructure cost reductions on the order of thirty to fifty percent, improvements in development cycle time of roughly twenty to thirty percent, acceleration of release cycles of forty to sixty percent, and reductions in security-breach risk of roughly half. These ranges are directional and depend heavily on scope and execution, and they should not be read as guaranteed, but they indicate that the modular, modernized end state pays back in agility and cost as well as in the reduced risk of the migration that reaches it. The strategic value is larger still: an organization on a composable architecture can adopt new technology, including the artificial intelligence capabilities that are difficult or impossible to layer onto a tangled legacy stack, because its modern foundation can accommodate new components without a wholesale rebuild. Composability is both the safe destination of an incremental migration and the insurance against ever needing a big-bang one again.

Figure 7. Reported benefits of disciplined modernization to a modern, modular architecture. Ranges are directional and depend on scope and execution, but they show the modernized end state pays back in cost, speed, and security as well as risk.

The technical enabler beneath composability deserves a plain statement, because it is where the modernization actually happens: the decoupling of systems through well-defined interfaces, and the liberation of data from the applications that trap it. A monolith is hard to divide largely because its data and its logic are fused, so the first real work of modernization is often to expose the system's functions and data through stable interfaces, sometimes called an interface-first approach, so that other components can use them without reaching into the monolith's internals. Once a function is available through a clean interface, it can be reimplemented behind that interface without the rest of the system noticing, which is precisely what makes incremental replacement possible. The interface is the seam made real, and building the interfaces is much of what converting a monolith into a composable architecture consists of.

There is a forward-looking reason to reach the modern, modular end state that has become urgent, and it connects this argument to the one on artificial intelligence that occupies so many technology agendas. The new capabilities that organizations most want to adopt, particularly the AI and analytics capabilities that promise real advantage, are difficult or impossible to layer onto a tangled, undocumented legacy stack, because they need clean access to data and functions that the monolith does not expose. An organization on a composable architecture can adopt these capabilities as new components, plugged into its modern foundation, while an organization trapped on a monolith cannot, and may find itself forced toward exactly the wholesale replacement this article warns against in order to become able to adopt them at all. Modernizing incrementally toward a modular architecture is therefore not only safer than the big bang; it is what positions the organization to adopt the next wave of capability without another traumatic rebuild.

Reaching the modern end state usually means moving to the cloud as well, and the same incremental logic applies to that shift. A wholesale lift of an entire estate to the cloud in one move carries the same concentration of risk as any big bang, whereas migrating workloads a few at a time, proving each in its new environment before moving the next, spreads the risk and lets the organization learn the operational differences of the cloud gradually. The cloud is not a synonym for modernization, and moving a monolith to the cloud unchanged simply relocates the problem, but a composable architecture and a cloud foundation together are what give an organization the elasticity, the managed services, and the access to new capability that the modern end state is meant to provide. The route to that end state, like every other step in this article, is safest taken incrementally.

One foundation deserves particular emphasis because it underlies both the migration and the modern end state: the quality and governance of the organization's core data. Master data, the definitive records of customers, products, suppliers, and locations, is what every system depends on, and it is precisely what a fragmented legacy estate tends to hold in inconsistent, duplicated, and conflicting forms. An incremental modernization is an opportunity to establish clean, governed master data as a shared foundation that new components can rely on, function by function, rather than attempting to reconcile decades of divergence in a single traumatic migration. Investing in data governance early, treating the organization's core data as an asset to be curated rather than a byproduct to be migrated,

pays back across every subsequent step, and its absence is one of the surest ways for even a well-designed migration to stall.

09 Phased and hybrid migration

The strangler-fig pattern is the most architecturally elegant incremental approach, but it is one member of a family, and a leader weighing options should understand the others, because the right choice depends on the system and the organization. What unites the family is the rejection of the single cutover; what distinguishes its members is the axis along which the migration is broken up.

Phased migration

A phased migration breaks the program into a series of smaller go-lives rather than one, and the phases can be defined along several axes. A migration can be phased by module, bringing finance live first, then inventory, then order management, so that each function is stabilized before the next is attempted. It can be phased by geography or business unit, bringing one site or one division onto the new system, learning from that deployment, and then rolling the proven configuration out to the rest. In each case the principle is the same as the strangler fig's: reduce the size of each step so that a failure affects one phase rather than the whole enterprise, and so that the lessons of each phase improve the next. The most common sequencing wisdom, learned from the failures, is to protect the functions that carry the cash and the customer, finance, inventory, and order management, by stabilizing them deliberately rather than rushing them.

Parallel running and the cost of the bridge

Incremental approaches share one real cost that the big bang avoids, and honesty requires naming it: during the migration, the organization runs the old system and the new one at the same time, and often must build temporary interfaces to bridge them so that data flows correctly between the two while the migration proceeds. Running two systems is more expensive than running one, and the temporary bridges are work that will be thrown away when the migration completes. This is the genuine trade-off of incremental modernization, and it is why big-bang advocates argue for speed. But the trade-off is lopsided: the cost of parallel running and temporary interfaces is a known, bounded, operational expense, while the cost it avoids, the disruption of a failed wholesale cutover, is an unbounded risk to the continuity of the business. Paying a known, modest cost to avoid an unbounded one is ordinary risk management, and the failures recounted earlier are what the saving on parallel running actually bought.

Hybrid approaches

The choice is not strictly binary, and many successful programs blend the approaches. A common hybrid uses a big-bang cutover for small, simple business units, where the risk of a single go-live is truly low, and a phased approach for large or complex ones, where it is not. Another blends a strangler-fig migration of the core with the wholesale retirement of small peripheral systems that are not worth migrating incrementally. The point of naming these hybrids is to dispel the idea that an organization must choose between a pure big bang and a pure incremental migration; the practical task is to match the granularity of the migration to the risk of each part of the estate, taking small, simple pieces quickly and large, critical pieces carefully.

The incremental approaches are strengthened by a set of techniques, borrowed from modern software delivery, that further shrink the risk of each step. A new function can be released first to a small fraction of traffic, a single site, a subset of users, a portion of transactions, and monitored closely before it is widened, so that a problem is caught while its blast radius is tiny; this is sometimes called a canary release. The routing that directs work to the old or new system can be controlled by switches, sometimes called feature flags, that allow a function to be turned on gradually and turned off instantly if it misbehaves, which makes each step not merely reversible in principle but reversible at the flip of a switch. These techniques, standard in mature software organizations, are what make the incremental approach's promise of small, reversible steps concrete, and they have no equivalent in a wholesale cutover, which is on or off with nothing in between.

10 When rip-and-replace is the right call

A fair treatment must concede that rip-and-replace is not always wrong, and that there are circumstances in which a wholesale replacement is the sensible choice. Arguing otherwise would be as dogmatic as the reflexive preference for the big bang that this article warns against. The honest position is that the big bang is a legitimate option for a specific and limited set of situations, and a trap only when it is chosen by default for situations that do not fit.

The clearest case for a wholesale replacement is a small, simple organization or scope. A single-site business with limited customization and modest complexity can cut over to a new system in one step with acceptable risk, because the number of things that can go wrong is small and the whole system can be tested in advance. The concentration of risk that dooms a large program is manageable when the program is small. A second case is a common, undifferentiated process for which a standard package is a good fit. Where the process being replaced is a commodity, general ledger, standard payroll, that a packaged system handles well, and where the organization has no meaningful differentiation to preserve, the clean adoption of a standard package can be faster and cheaper than an incremental rebuild, and the risk of the cutover is bounded by the maturity of the package. A third case is a legacy system so broken or so unsupportable that running it in parallel with a new one is not feasible, which can force a harder cutover than would otherwise be chosen. And a fourth is an externally imposed hard deadline, a vendor end-of-life or a regulatory change, that removes the option of a longer migration.

Even in these cases, the discipline of de-risking still applies: a wholesale cutover should be piloted where possible, preceded by thorough data cleansing, timed to avoid peak periods, and equipped with a tested rollback plan, so that even the justified big bang is not a leap of faith. And it is worth noting that the market has already absorbed this lesson: research finds that fewer than a quarter of organizations now choose a pure big-bang approach, which suggests that the reflexive preference for the wholesale cutover is giving way to the more discriminating choice this article recommends. The rule is not that rip-and-replace is never right, but that it should be chosen deliberately, for a situation that fits it, rather than adopted by default for one that does not.

The two harder cases for incrementalism deserve a further word, because they are the ones a big-bang advocate will press. When a legacy system is so broken that it cannot reliably run alongside a new one, the option of parallel running is truly constrained, and a more decisive cutover may be forced; even then, the cutover can often be narrowed to the smallest workable scope, and the data cleansing and rollback discipline still apply, so that the forced big bang is at least a contained one. When an external deadline, a vendor end-of-life or a regulatory change, removes the time for a long migration, the pressure toward a wholesale cutover is real; even then, an organization can often meet the deadline for a defined core while continuing to migrate the remainder incrementally, rather than treating the deadline as a mandate to move everything at once. The lesson is that even the situations that seem to force a big bang usually admit a more contained approach, and that the burden should be on the wholesale cutover to justify itself, not on the incremental path.

For the rare situation in which a wholesale cutover is the right choice, it is worth stating what running one well requires, because the justified big bang still demands discipline. The scope should be as small as the situation allows, so that the concentration of risk is minimized. The data should be cleansed and validated well before the cutover, not during it. The go-live should be timed to a quiet period, never a peak. A full dress rehearsal, a complete test of the cutover on a copy of the real data, should precede the event. A tested rollback plan should stand ready, so that a failed cutover can be reversed rather than endured. And the organization should be fully prepared, with training complete and ownership clear, before the switch is thrown. A big bang run this way is still riskier than an incremental migration, but it is a considered risk rather than a leap of faith, and the difference is often the difference between a difficult go-live and a disaster.

11 How to de-risk a modernization program

Whatever approach an organization chooses, the failures examined earlier point to a consistent set of defenses, because the causes of failure are consistent. The striking finding across the research, shown in Figure 8, is that the dominant causes are not technical but organizational: inadequate change management, poor data migration, inexperienced teams, and unrealistic timelines account for the great majority of failures, while the software itself is rarely the culprit. De-risking a modernization program therefore means attacking these organizational causes directly.

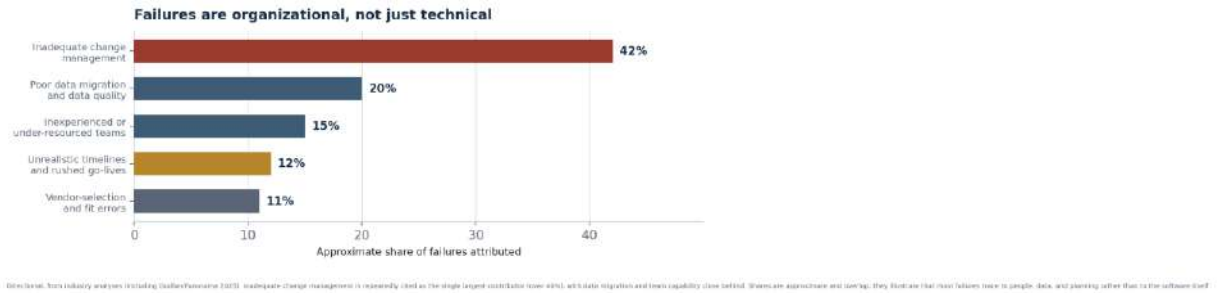

Figure 8. The dominant causes of modernization failure are organizational, not technical. Change management is repeatedly the single largest contributor, with data and planning close behind. Shares are approximate and overlap.

- **Treat change management as the main event, not an afterthought.** Because it is the single largest cause of failure, the management of the human transition, communication, training, process redesign, and the handling of resistance, deserves first-class attention and resourcing from the start, not the residual time and budget left after the technology work.
- **Run data migration as a project in its own right.** Legacy data is dirtier than anyone expects, and a new system launched on bad data fails regardless of how good the system is, as Target Canada showed. Cleansing, validating, and reconciling the data must begin early and be treated as a major workstream, not a task assumed to be easy.
- **Set realistic timelines and never launch at peak.** Compressing a schedule to hit an arbitrary date, and cutting over during the busiest season, is the Hershey error, and it recurs. The timeline must reflect the true complexity of the work, and the go-live must avoid the periods when a failure would be most damaging.
- **Establish internal ownership and governance.** Lidl's failure was in large part a failure of ownership, in which the organization deferred critical decisions to vendors with no stake in its long-term success. A strong internal owner, a clear governance model, and decisions that stay with the business are essential, whoever implements the system.
- **Decide the customization question deliberately.** Much failure comes from an unmanaged spiral of customization, in which an organization insists on adapting the system to its every existing process. The disciplined choice is to adopt standard processes where the organization has no real differentiation to protect, and to customize only where the difference truly matters, rather than customizing by default.
- **Build a tested rollback and contingency plan.** Every cutover, incremental or wholesale, should have a defined way to recover if it fails, and that recovery should be tested rather than assumed. The

organizations that survive troubled go-lives are those that can stabilize and, where necessary, revert, rather than being forced to push forward through a failing launch.

An independent perspective helps with all of these, because the research consistently finds that a troubled program is rarely rescued by the same party that created it. An honest, outside assessment surfaces the root cause faster and keeps the organization from repeating the pattern that produced the trouble. These defenses do not guarantee success, but they attack the actual causes of failure rather than the imagined one, and they apply whether the chosen path is incremental or, in the limited cases where it fits, wholesale.

Two technical disciplines strengthen every modernization and deserve explicit investment. The first is automated testing: a suite of tests that can be run continuously to confirm that each change preserves the behavior it should, which is what allows an incremental migration to move quickly without breaking what already works, and what gives the verification of new against old its rigor. The second is observability: the instrumentation that lets an organization see what its systems are actually doing in production, which functions are used, where errors occur, how the new components behave under real load, so that problems are detected early and the migration is steered by evidence rather than by hope. Neither is glamorous, and both are routinely underfunded in programs that treat modernization as a one-time project rather than an ongoing capability, but they are the instruments that make incremental modernization controllable, and their absence is a common reason migrations stall or drift.

The management of the implementer is its own discipline, because the failures repeatedly show the danger of ceding control to a party without a stake in the outcome. An organization that lets a system integrator or software vendor make its critical decisions, particularly the decisions about customization and process, has handed the wheel to someone whose incentives may not match its own, and the resulting loss of ownership is a recurring theme in the largest failures. The remedy is not to distrust implementers, whose expertise is truly needed, but to retain the decisions that belong to the business, to keep a strong internal owner accountable for the outcome, and, where the stakes are high, to bring an independent perspective that is not tied to defending prior choices or selling additional work. The implementer executes; the organization must own.

A pilot or a prototype, run before any full commitment, is among the most cost-effective forms of de-risking available, and it is underused. Before committing to a platform and a program, an organization can implement a narrow slice, one process, one site, or one bounded function, on the proposed new system and observe how it actually behaves in the organization's real environment, with the organization's real data and people. The pilot surfaces the integration problems, the data-quality gaps, and the change-management challenges that a slide-based evaluation cannot, and it does so while the commitment is still small enough to change course cheaply. A pilot that goes badly is one of the most valuable outcomes a program can have, because it reveals, at low cost, a problem that would have been ruinous at full scale.

Communication and expectation management belong on the list of defenses, because a modernization fails not only when the technology falters but when the organization's patience does. An incremental program delivers value steadily but reaches its final end state slowly, and stakeholders accustomed to

the promise of a single transformative go-live must understand why the patient path is the safer one, or they may press for exactly the acceleration that produces failure. Setting expectations candidly at the outset, that the program will deliver continuously but complete gradually, that each step is deliberately small, that the goal is a modernization the business survives rather than a dramatic unveiling, is what preserves the organizational support the long effort requires. The leaders who modernize well spend as much effort managing the expectation of the change as managing the change itself.

12 A decision framework

The decision an organization actually faces is not a single yes or no but a choice among several modernization strategies, and the right choice depends on two questions about the system in play: how well it currently meets the organization's needs, and how much it differentiates the business strategically. Plotting a system against these two axes, as in Figure 9, yields four situations, each of which points to a different strategy.

Figure 9. A framework for choosing a modernization strategy. The right approach depends on how well the current system fits and how much it differentiates the business, and only one quadrant favors wholesale replacement.

A system that meets the need poorly but differentiates the business strongly, the upper-left quadrant, is the classic candidate for the strangler fig. Here the organization cannot accept a standard package, because the system embodies something that makes the business distinctive, but it also cannot leave the failing system in place. The answer is incremental modernization that preserves the differentiation while rebuilding the system piece by piece, and it is precisely the situation in which a big-bang replacement is most dangerous, because the value at stake is highest and the behavior to be reproduced is most specific. A system that meets the need poorly and does not differentiate the business, the lower-left, is the one legitimate home of wholesale replacement: a commodity process on a poorly fitting system can often be replaced cleanly with a standard package, because there is nothing distinctive to preserve.

The right-hand quadrants counsel restraint. A system that meets the need well and differentiates the business, the upper-right, should be extended and invested in, not replaced; the instinct to modernize for its own sake destroys value here. And a system that meets the need well but does not differentiate the business, the lower-right, is best left alone, maintained but not touched, because the effort of replacing a system that works and does not matter strategically is effort better spent elsewhere. The framework's central lesson is that wholesale replacement is the right answer for only one of the four situations, the commodity process on a poorly fitting system, and that the situation where the temptation to rip and replace is often strongest, the differentiating system that has begun to fail, is precisely the situation where the big bang is most likely to destroy value and the strangler fig most likely to preserve it.

A worked placement makes the framework concrete. Consider two systems in the same company. The first is a bespoke order-promising engine that encodes the company's distinctive rules for how it commits delivery dates to customers, rules that are a real competitive advantage and exist in no packaged product. The second is a standard general ledger that does what every general ledger does, differentiates nothing, and happens to run on an aging platform. The framework places these in opposite quadrants and prescribes opposite strategies. The order-promising engine, high in differentiation and poor in fit, calls for the strangler fig: rebuild it incrementally, preserving the distinctive logic, never risking it on a single cutover. The general ledger, low in differentiation and poor in fit, is the legitimate candidate for wholesale replacement with a standard package, because there is nothing distinctive to lose and the packaged process is mature. Two systems in one company, two different right answers, and the framework is what tells them apart.

The framework also accommodates the middle options between leaving a system alone and replacing it wholesale, which are easy to forget when the debate is framed as replace-or-not. A system in the poorly-fitting quadrants need not always be either replaced or rebuilt from scratch; it can sometimes be re-platformed onto a modern foundation, or re-factored to improve its structure, or wrapped with modern interfaces that let new capability be built around it without disturbing its core. These lighter-touch moves can buy years of additional life for a system that is not quite failing and not quite fit, and they can be the right answer for a system whose differentiation is moderate and whose problems are containable. The framework's quadrants are a starting point for judgment, not a mechanical rule, and the art is in matching the depth of the intervention to the actual condition and importance of the system.

13 Sequencing an incremental modernization

For the organization that chooses the incremental path, the strategy becomes a sequence of concrete steps, and the sequence itself carries much of the risk reduction. The following is the shape of a disciplined incremental modernization, drawn from the strangler-fig pattern and the lessons of the failures.

1. **Map the system and find the seams.** Before anything is built, understand the existing system well enough to identify its natural boundaries, the points at which it can be divided into pieces that can be replaced independently. These seams, sometimes called bounded contexts, are where the migration will cut, and finding them is the foundational analytical work.
2. **Stand up the routing layer.** Place the facade or interception layer between the users and the systems, initially routing everything to the legacy system. This changes nothing for users but creates the mechanism through which functionality will later be redirected, piece by piece, to the new system.
3. **Choose the first slice carefully.** The first function to migrate should be chosen to maximize learning while minimizing risk, often a piece that is important enough to prove the approach but self-contained enough that a problem is easily contained. The first slice establishes the pattern the rest of the migration will follow, so it is worth choosing to build confidence and surface the hard problems early.
4. **Build, route, verify, and only then move on.** For each slice, build the new function, route its traffic through the facade, and verify the new function's behavior against the legacy system it replaces before considering the slice complete. Running the new against the old and confirming they agree is the safeguard that a big-bang cutover cannot offer, and it should be used on every slice.
5. **Protect the cash-and-customer functions.** Sequence the migration so that the functions carrying the money and the customer commitments, finance, inventory, and order management, are approached with the most care and the most testing, because these are the functions whose failure the earlier disasters showed to be most damaging.
6. **Decommission the legacy pieces as they empty.** As each function moves fully to the new system, retire the corresponding piece of the legacy system rather than leaving it running, so that the old system truly shrinks and the double-maintenance cost falls over time rather than persisting.
7. **Commit to finishing.** The one serious failure mode of the incremental approach is a migration that stalls partway, leaving the organization running both systems indefinitely at double the cost. Incremental modernization requires the sustained commitment to see the migration through to the retirement of the legacy system, and the governance to keep it moving rather than allowing it to stall once the most painful problems are solved.

This sequence converts the abstract preference for incrementalism into a concrete program, and its discipline is what delivers the safety the approach promises. Each step is small, each is verified against the system it replaces, each delivers value as it completes, and the whole proceeds while the business stays in production. That is the opposite of the big bang in every respect that matters, and it is why the incremental path, though it demands sustained commitment, so rarely produces the disasters that the wholesale path so reliably does.

The sequence is sustained by a governance cadence that keeps it moving, because the incremental approach's one serious failure mode is a migration that stalls. A regular review of progress, measured in functions migrated and legacy components retired rather than in effort expended, keeps the program honest about whether it is actually advancing. Explicit targets for decommissioning legacy pieces prevent the comfortable equilibrium in which the new system grows but the old one is never quite switched off, leaving the organization paying for both. And a visible accounting of the value each increment has delivered sustains the organizational support the long migration requires. The cadence is not bureaucracy; it is the mechanism that converts the intention to modernize incrementally into the sustained execution that actually retires the legacy system, rather than a migration that delivers its easy wins and then quietly stops.

Measuring the progress of an incremental modernization requires the right metrics, because the wrong ones can make a stalling program look healthy. Effort expended, budget consumed, and documents produced measure activity, not progress, and a program can generate all three while migrating nothing. The metrics that matter are outcomes: the number of functions actually moved to the new system and running in production, the number of legacy components actually retired, and the business value actually realized from each increment. Tracking these keeps the program honest, makes a stall visible early, and preserves the organizational support that the long effort requires by demonstrating, increment by increment, that the migration is steadily advancing toward the retirement of the legacy system rather than merely consuming resources.

An incremental modernization also asks for a different kind of team than a big-bang program, and staffing it correctly is part of the discipline. A wholesale program is often run as a large, temporary project with a defined end, staffed heavily for a period and then disbanded, which is part of why the knowledge it builds is lost and why it treats modernization as a one-time event. An incremental program is better served by a smaller, durable team that owns the migration over its full course and, ideally, continues to own the modernized system afterward, because the same people who move each function across are the ones who understand it best. This continuity of ownership is itself a risk reduction, because it keeps the knowledge of what was migrated, and why, inside the organization rather than departing with a project team when the engagement ends.

Seen whole, the discipline this article recommends is less a technique than a temperament: a preference for the patient, reversible, and continuously validated over the bold, irreversible, and unproven. That temperament runs against a real instinct, the desire to be done with a failing system in one decisive stroke, and resisting that instinct is much of what separates the organizations that modernize successfully from those that become the next cautionary tale. The incremental path is not easier; it demands sustained commitment, architectural skill, and the patience to finish what the easy wins do not complete. But it is safer by a wide margin, it delivers value throughout rather than at a distant and uncertain end, and it reaches the same modern destination without asking the organization to bet its continuity on a single day. For the systems a business truly depends on, that is not merely the better method; it is the only responsible one.

14 Requirements, governance, and a scoring rubric

The preference for incremental modernization is most durable when it is built into how a program is scoped, governed, and approved, rather than left to the judgment of a project team under pressure to hit a date. A few practices embed the discipline.

Scope for outcomes and incremental delivery

Define the program by the business outcomes it must deliver and require that it deliver them incrementally, rather than defining it as the replacement of a system by a date. A program chartered to deliver a stream of value, with milestones tied to functions actually migrated and benefits actually realized, is structurally biased toward the incremental approach, while a program chartered to replace a system by a deadline is biased toward the big bang. Make data-quality gates and a tested rollback plan explicit requirements, not assumptions, and require a change-management plan resourced from the start rather than added when trouble appears.

Govern for continuity and reversibility

Establish governance that treats business continuity and reversibility as first-class requirements: no step should be approved that cannot be rolled back, and no cutover should proceed without a demonstrated ability to recover. Keep the critical decisions, particularly the decisions about how much to customize and which processes to standardize, with the business rather than the implementer, because the failures show that deferring these to vendors is a common road to ruin. And build in the sustained commitment to finish, so that the migration is not allowed to stall once the hardest problems are behind it.

Comparing the approaches

The table below sets the two approaches side by side on the dimensions that determine risk, as a decision aid. Read it not as a verdict that the incremental approach is always right, but as a map of the trade-offs, most of which favor incrementalism for any large or differentiating system.

Dimension	Incremental / strangler fig	Big-bang replacement
Risk concentration	Spread across many small steps	Concentrated on one go-live
Value delivery	Continuous, throughout the program	Only at the end, after a freeze
Reversibility	Roll back a single step	All-or-nothing at cutover
Business continuity	In production the entire time	Depends on the cutover holding
Data migration	Per slice, validated as you go	All at once, under time pressure
Learning	Applied to each successive step	Assumptions locked in early
Main cost or risk	Parallel running; must commit to finish	Unbounded disruption if it fails
Best fit	Large, complex, or differentiating systems	Small, simple, commodity replacements

The table's shape tells the story. For nearly every dimension that governs risk, the incremental approach carries the lighter burden, and its two real costs, the expense of parallel running and the need to commit to finishing, are bounded and manageable against the unbounded disruption the wholesale approach risks. Only in the narrow case of a small, simple, commodity replacement does the balance shift toward the big bang, which is exactly the case the decision framework identified.

15 Conclusion: modernize without the big bang

The argument of this article is not against modernization. Legacy systems age, grow brittle, lose the people who understand them, and eventually become a liability that must be addressed, and the cost of leaving a truly failing system in place is real and compounding. The argument is against a specific method of modernizing, the wholesale rip-and-replace, whose record is a long list of overruns, disruptions, and outright disasters, and whose failures trace not to bad luck but to a structural flaw: the concentration of every risk, every dependency, and every dollar of return onto a single go-live that asks the organization to replicate the unfamiliar, re-architect the entire business, and innovate, all at once, on one day.

The alternative is not to modernize timidly but to modernize incrementally, growing a new system around the old one until the old one can be retired, so that the business stays in production throughout, value arrives continuously, each step is small and reversible, and the organization learns as it goes. The strangler-fig pattern, the family of phased and hybrid migrations around it, and the composable architecture they build toward are not a compromise or a slower path to the same risk; they are a fundamentally safer way to reach the same destination, and they are what the most disciplined modernizers now practice. The two real costs of this path, the expense of running old and new in parallel and the commitment required to finish, are modest and bounded against the unbounded disruption the big bang risks, and paying the smaller cost to avoid the larger is simply sound judgment.

The discipline this article recommends is finally a discipline of matching method to situation. Rip-and-replace is occasionally the right answer, for a small, simple, commodity system where the risk of a single cutover is truly low, and it is a trap only when it is chosen by default for the large, complex, differentiating systems where the risk is highest and the incremental path clearly better. The organizations that modernize well are not the ones that move fastest or most boldly, but the ones that resist the seductive clean slate of the wholesale replacement, place each system candidly on the map of fit and differentiation, and choose the method the situation actually warrants. For most core systems that matter, that method is to grow the new around the old, patiently, in production, and to retire the legacy only when its replacement has been proven, piece by piece, in the daylight of real operation.

The executive takeaway can be stated in a sentence: when a leader is presented with a proposal to rip out and replace a core system in a single program, the right first response is not to ask how to make the big bang succeed, but to ask why it is not being done incrementally. The burden of proof belongs on the wholesale cutover, because the evidence of its failure rate is overwhelming and the incremental alternative is available for all but the simplest cases. A leader who insists on that question, and who accepts a big bang only when the situation truly fits the narrow profile that warrants it, will avoid the category of failure that has cost other organizations their quarters, their reputations, and occasionally their existence. The most important decision in a modernization is the choice of method, and it is made at the beginning, before the momentum of a large program makes it hard to change.

16 Methodology, caveats, and sources

Methodology

- This article synthesizes independent research on enterprise-system implementation outcomes, widely documented failure case studies, and established software-architecture guidance on incremental modernization, current to mid-2026. Supply Chain Research is independent and accepts no payment from the vendors, consultancies, or firms discussed.
- The failure case studies are drawn from press accounts, company disclosures, and case-study analyses. Reported figures for losses, recovery costs, and write-offs come from those accounts and are used to convey scale, not as audited financials.

Caveats

- Definitions of project failure vary across the studies cited. Some count any project that misses its original goals, budget, or timeline; others count only outright abandonment. The failure and overrun percentages should therefore be read as directional indicators of a consistent pattern rather than as precise, comparable measures.
- The reported costs of the individual failures mix different quantities, operational losses, recovery costs, and write-offs, across different companies, industries, and years, and are not directly comparable. They illustrate the scale that big-bang failures can reach, not a like-for-like ranking.
- Many of the cited failures had multiple causes, including change management, data quality, and governance, that can affect any modernization approach. The big-bang method is identified here as a consistent aggravating factor, not as the sole cause of every failure. Incremental approaches are not immune to failure; they carry their own risks, notably the cost of parallel running and the danger of a stalled migration.
- The modernization benefit ranges are directional and depend heavily on scope and execution; they are not guaranteed outcomes. The value curve and the risk and framework diagrams are illustrative representations of the arguments, not measurements of any specific program.

Sources

[1] Martin Fowler (2004). StranglerFigApplication.

[2] Microsoft Learn, Azure Architecture Center. Strangler Fig pattern.

[3] Thoughtworks. Embracing the Strangler Fig pattern for legacy modernization.

[4] Panorama Consulting Group. The 2026 ERP Report (implementation outcomes, overruns, and approaches).

[5] Panorama Consulting Group. Top ERP implementation failures.

[6] CIO (Foundry). Famous ERP disasters, dustups, and disappointments.

[7] Henrico Dolfing. Project failure case studies (Target Canada, Revlon, National Grid, Lidl, Nike).

[8] Gartner (newsroom and reporting). Research on ERP initiative outcomes and modernization.

Additional context drawn from McKinsey research on enterprise modernization programs completed in 2024 and 2025 (infrastructure cost, development cycle-time, release-cycle, and security-breach improvements), as reported; from industry analyses of ERP failure causes (including Godlan and Deloitte); and from vendor and practitioner literature on composable architecture and microservices. Failure definitions and reported figures vary by source and are directional. This article is analysis, not legal, procurement, or investment advice.

Supply Chain Research is an independent, vendor-neutral research platform for supply chain and technology leaders. We accept no payment from the vendors, consultancies, or firms discussed. This article is analysis, not legal, procurement, or investment advice, and its conclusions should be validated against your own circumstances before any decision.