Program Drift: Early Warning Signals in Large Digital Transformation Programs [1, 2]

Cornelius Greyling

Abstract

Major program failures do not usually arrive as sudden cataclysmic events. They do not hit a wall at high speed. More often, large-scale digital transformations gradually enter a degraded state where delivery remains active, governance stays in place, and reporting remains largely yellow. This paper defines that condition as "Program Drift": the gradual loss of alignment between governance, plans, decisions, and execution while work continues and formal controls remain in place. Decision latency is a major mechanism, but drift also develops through planning divergence, unvalidated work, assumption-based continuation, fragmented ownership, and apparent closure that does not change execution. The paper also points to a limit in the dominant predict-and-control strain of project management. Traditional controls assume that with enough rigor, decomposition, and method discipline, delivery remains broadly manageable. In large transformations, that assumption captures only part of the problem. Drawing on practitioner experience across enterprise transformations and large ERP deployments, this paper highlights early warning signs of drift that can appear well before delivery failure is detected. It also discusses why these signs are ignored, how the very processes meant to address them can sometimes make the problem worse, and how Program Drift can be recognized early without adding more processes or tools.

Scope and Basis

This paper draws on repeated practitioner observations across large-scale enterprise transformation programs in banking, utilities, automotive, energy, retail, and other sectors, with particular attention to SAP and S/4HANA change. The patterns described here are

[1] Editor's note: Second Editions are previously published papers that have continued relevance in today's project management world, or which were originally published in conference proceedings or in a language other than English. Original publication acknowledged; authors retain copyright. This paper was originally presented at the 18th Project Management Symposium at the University of Texas at Dallas in May 2026. It is republished here with permission of the author and conference organizers.

[2] How to cite this paper: Greyling, C. (2026). Program Drift: Early Warning Signals in Large Digital Transformation Programs; Originally presented at the 18th Project Management Symposium at the University of Texas at Dallas in May, republished in the *PM World Journal*, Vol. XV, Issue VIII, August.

presented as a qualitative diagnostic model intended to improve early visibility and intervention. Direct empirical research isolating decision latency as a standalone causal variable in large ERP or transformation programs remains limited. Most stronger evidence reaches the issue indirectly, through governance quality, role clarity, alignment, intervention timing, and decision-right allocation. That matters because those are the mechanisms through which lagging or missing decisions appear in real programs. Available evidence still points in a consistent direction: stronger governance is associated with lower delay, earlier recognition improves recovery odds, and ineffective decision structures are repeatedly linked to schedule and quality deterioration[3] [4] Research on early warning signs in complex projects already shows that weak signals are often visible before formal failure is acknowledged (Williams, Klakegg, Walker, Andersen, & Magnussen, 2012). This paper builds on that foundation but shifts the lens to active delivery in large transformation programs, where the issue is not only whether weak signals are detected, but whether governance can still convert those signals into timely decisions and coordinated action.

1. Program Drift and Decision Capacity

1.1 Motion is not Control

"He who lets the sea lull him into a sense of security is in very grave danger." – Hammond Innes

Mariners have an old problem that is easy to underestimate: a ship can appear to be making good progress and still be drifting off course. The danger is not the absence of motion. It is motion in the wrong direction, detected too late (Dekker, 2011). At sea, "set" is the direction of the current and "drift" is its speed.[5] A navigator who mistakes movement for control may discover the error only when the next fix shows the vessel is no longer where the chart assumed it should be. Experienced navigators often sense earlier that "something is not lining up" before the deviation becomes large. But that instinct is usually pattern recognition built on instruments, visual cues, timing, and sea conditions, not some vague gut feeling.

In large digital transformation programs, leaders often do not notice the moment control starts to weaken. There is no warning sign that says "failure dead ahead," and no single meeting where everyone agrees that control has been lost. Instead, the program drifts

[3] Khalid et al. found significant negative relationships between project governance and project delay (β = - 0.418), and between IT governance and project delay (β = -0.292).

[4] The UK Infrastructure and Projects Authority identifies ineffective decision-making as a recurring transformation challenge and states that ineffective decision-making bodies can severely affect schedule and quality of outcomes.

[5] In maritime navigation, "set" refers to the direction of the current and "drift" to its speed.

gradually. Status may begin green, move to amber, and remain there for months, sometimes flickering red. Delivery continues, teams remain staffed and busy, status reports are issued on time, and meetings occur as scheduled. Risks and issues are discussed, yet the program remains in a prolonged state of conditional confidence: perhaps still viable, but increasingly difficult to verify.

And yet, program leaders begin to sense a loss of control before they can fully explain it.

1.2 What is Program Drift?

In large transformation programs, drift begins when governance, plans, decisions, and execution gradually move out of alignment even though delivery continues and formal structures remain in place.

The warning signs, however, start to appear. Decisions take longer than they should. Escalations produce partial or temporary answers. Plans and local working assumptions begin to diverge. Actions may be marked complete without evidence that the underlying condition has gone away. Delivery dates slip in small increments rather than through major resets. The program exists in a constant state of conditional viability: vulnerable, but not yet in outright failure.

This paper defines that condition as Program Drift[6].

The maritime analogy is deliberate: Like a vessel under set[7] and drift, a program can preserve movement, rhythm, and the appearance of control while steadily veering off its intended course.

Drift differs from conventional project underperformance in several important ways. First, activity remains high. Plans continue to be updated, artifacts are produced, meetings are held, and milestones are still nominally tracked. There is no obvious breakdown in operational rhythm.

[6] Duncan and Schladow describe ship's drift as the difference between a vessel's dead-reckoning position and its true position, using that gap to estimate average surface current (Duncan & Schladow, 1981). Program Drift is used here as a practitioner-derived analytical construct rather than a formal category established in the project management literature. It describes a recurring condition in large transformation programs where execution activity continues, governance structures remain formally intact, and milestones are still reported, but the system's ability to convert escalation into decision and decision into coordinated action begins to erode.

[7] Navigators estimate set and drift by comparing a dead-reckoning position with a true fix and then correcting course accordingly.

Second, governance does not collapse outright. Steering committees still meet, escalation paths remain in place, and decisions continue to be made, at least formally.

Third, reported status trails behind reality. Metrics tend to reflect past completion rather than current decision readiness or delivery resilience[8]. This is the governance equivalent of relying on dead reckoning for too long without taking a reliable fix.

Drift, then, is not simply a failure of management attention or a single missed decision.

It emerges when small losses of alignment accumulate faster than the program can detect and correct them. Just as currents and wind can nudge a boat off course, programs are displaced by decision latency, unvalidated assumptions, planning divergence, fragmented ownership, misaligned incentives, and governance friction. None of these forces has to be decisive on its own. Their cumulative effect is what moves the program away from its intended course.

What creates the real danger is the false sense of control. By the time we realize something is wrong, it is usually too late to correct it quickly, and the stage for failure has already been set.

This pattern is consistent with prior research on escalation of commitment and project persistence, which shows how organizations continue investing and executing even as outcome viability declines, particularly in complex and highly visible initiatives (Staw, 1981; Flyvbjerg, 2014). It is also consistent with information systems research on software project escalation, where organizations continue committing resources to troubled initiatives even after negative information is available (Keil, Mann, & Rai, 2000).

Recent SAP migration research points to the same pattern: many programs effectively fail on paper long before they fail in production, as early weaknesses in governance, sequencing, ownership, and scope quietly erode delivery viability before any visible technical breakdown occurs (ISG, 2026a).[9]

[8] The comparison is to dead reckoning in navigation: an estimated position advanced from a known fix using assumed course and speed. It is useful for continuity, but it becomes less reliable the longer it runs without correction from an actual fix. The project equivalent is status reporting that extends prior assumptions forward without re-establishing whether the program's real decision position is still the one being reported.

[9] ISG states that many SAP migrations "fail on paper long before they fail in production," emphasizing that early choices around governance, sequencing, ownership, and scope shape later outcomes before technical failure becomes visible.

1.3 Decision Capacity as a Major Drift Mechanism

A common pathway into drift opens when the demand for decisions outpaces the program's ability to make them, close them, and translate them into execution.

As programs gain speed and complexity, they need more direction, more trade-off decisions, and faster calls from leadership. The issues behind those decisions are rarely simple. They cross workstreams, carry competing constraints, and usually arrive under time pressure. As governance comes under strain, these issues take longer to resolve than execution can tolerate. That is when the gap starts to open between what the program needs decided and what leadership can actually decide in time.

This gap rarely appears all at once. It builds gradually as unresolved questions accumulate faster than they can be cleared. Design choices remain open longer than downstream work can tolerate. Escalations return without closure. A decision is not closed merely because it has been recorded or an action has been assigned. It is closed when the direction is clear, ownership and plans have been updated, downstream dependencies have been adjusted, and execution reflects the outcome. Without that translation, teams continue forward on partial assumptions while governance lags behind the needs of execution.

But drift is not only a decision-capacity problem, and it is not adequately explained as a delivery problem.

That point matters beyond this paper's immediate argument. Much of conventional project management still assumes a more mechanical model of control: define the work, sequence it, track variance, and escalate exceptions. Those disciplines remain necessary, but in large transformations they are not sufficient. Control also depends on whether the program can sustain decision quality, shared understanding, and coordinated execution as complexity rises.

The teams on these programs are often experienced, committed, and working at full stretch. Yet control can still begin to weaken. One of the earliest breakdowns is often the path from issue identification to decision closure. If that path becomes too slow, teams move forward on assumptions long before governance catches up. In other programs, the first visible break may be plan divergence, fragmented ownership, or work accepted without sufficient validation. These mechanisms then reinforce one another.

The maritime analogy remains useful here. A vessel does not drift off course only when steering stops. It also drifts when corrective action is too slow, too small, or too late to compensate for the forces acting on it. Programs behave in much the same way. The issue is not the absence of control, but the inability of the control system to detect, interpret, and

correct deviation before displacement becomes material.[10]

Over time, the gap can widen between the decisions a program needs and the decisions governance can actually make in time. It may not create visible problems immediately. The program keeps moving, but each unresolved point is carried forward into plans, dependencies, and execution, making later correction more difficult.

At that point, continuing without correction can do more damage than stopping work long enough to regain control.

1.4 Why the Usual Fixes Fail

Common recovery responses can make this worse. Under pressure, teams may add analysis, generate more options, extend decision windows, or create further mitigations to preserve progress. These responses are often professionally motivated: they protect prior work, avoid waste, and keep customer commitments alive. But unless they remove ambiguity or change execution, they can convert closure into another cycle of activity.

It also does not announce itself. It grows out of the same practical adjustments that keep a program moving: small changes made under pressure, each one reasonable enough at the time (Dekker, 2011; Dekker & Pruchnicki, 2013). It takes shape in the space between governance structures, incentive systems, reporting practices, and the limits of human judgment under sustained pressure.

Drift is not solely a delivery-side phenomenon. It is often reinforced by demand-side behavior, including delayed business decisions, shifting priorities, and internal misalignment within the client organization. These conditions increase decision latency and introduce assumptions into execution. Drift sits in the space between delivery and demand. It does not belong to either side on its own.

That interaction becomes even harder to manage in multi-party SAP programs, where unclear ownership across clients, system integrators, software vendors, and niche partners can fragment decision rights and delay closure without any one actor fully owning the end-to-end outcome (ISG, 2026b).[11] Research on distributed and offshore collaboration shows how overlapping boundaries, status differences, and organizational separation can inhibit effective coordination across parties, which helps explain why decision rights and

[10] Current SAP migration findings are consistent with this view. Delays and overruns are often linked less to technical platform difficulty than to weak governance, underestimated complexity, scope expansion, and internal capacity constraints that reduce the program's ability to make and sustain timely decisions (ISG, 2026b).

[11] ISG identifies fragmented accountability, unclear decision rights, and weak acceptance ownership in multi-vendor SAP environments as recurring contributors to delay and underperformance.

accountability become especially fragile in large multi-vendor delivery models (Levina & Vaast, 2008).

By the time drift is widely acknowledged, recovery options are already constrained. In mature drift, decisions are often not merely delayed; they are delayed until no real decision remains to be taken, because elapsed time, downstream commitments, assumption-based execution, and accumulated rework have already collapsed the option space.

Mitigation is necessary, but closing a mitigation action is not evidence that the risk condition has disappeared. If ownership, plans, dependencies, and execution still reflect the unresolved condition, the program remains exposed. The same is true of analysis: more information does not necessarily improve decisions and, beyond a certain point, can make them worse (Simon, 1957; Kahneman, 2011).

Effective intervention works in the opposite direction. It narrows the decision space, forces explicit trade-offs, and concentrates governance attention on the small number of decisions that actually change program trajectory.

2. Detecting Program Drift

Drift leaves traces well before delivery outcomes degrade. These signals are frequently noticeable months beforehand, yet they are faint, unsettling, and easily dismissed. This is consistent with earlier research on IT project failure, which found that warning signs are often visible before failure becomes unavoidable, but are not always acted on early enough (Kappelman, McKeeman, & Zhang, 2006).

These traces typically become visible in four practical areas: decision alignment, decision closure, planning integrity, and execution anchoring.

2.1 Decision Alignment

These signals reflect how quickly and reliably the program reaches a shared understanding of its current state.

- **Time to reach status alignment**

Alignment should be fast and largely uncontroversial, so discussion can focus on implications and next steps. As drift develops, alignment itself becomes the work. Time is spent reconciling interpretations rather than acting on them. Meetings end without clear agreement, or agreement is reached only after side conversations. This indicates that the program no longer shares a stable operating reality. Without a common fix, each group starts navigating from its own assumed position.

This is not just a communication problem. It reflects growing uncertainty about whether reported progress still corresponds to real delivery readiness.

- **Divergence between official plans and working assumptions**

What is formally reported and what teams are actually operating to should remain closely aligned. As drift develops, teams begin to operate on assumptions that are not formally acknowledged.

These assumptions allow work to continue in the absence of decisions, but they also introduce silent divergence. When this gap widens, governance loses its ability to steer delivery.

Over time, the official decision record diverges from actual decision behavior. The chart still shows one course, but the vessel is being steered through a series of informal corrections nobody has fully plotted.

2.2 Decision Closure

These signals reflect whether escalated topics produce outcomes that are explicit, owned, embedded in plans, and visible in execution.

- **Percentage of escalations resolved within one governance cycle**

Escalation should lead to a decision within a predictable governance cadence. As drift develops, escalation produces activity but not closure. Topics are deferred, reframed, or partially addressed. A decision recorded in minutes, or an action assigned for follow-up, is not yet evidence that the underlying issue has been resolved.

A declining resolution rate indicates that governance is no longer converting escalation into decision.

• Frequency of repeat agenda items

Issues reach governance forums but return without clear direction. Actions are assigned for further analysis or mitigation and may later be marked complete even when the underlying condition remains. The same topics reappear week after week with minor reframing.

From the outside, escalation appears to be working. In reality, it has stalled.

Similar dynamics have been observed in studies of decision latency and governance overload, where formal escalation mechanisms exist but resolution is delayed due to risk aversion, accountability diffusion, and consensus-seeking behavior (Eisenhardt, 1989; Kahneman, Lovallo, & Sibony, 2011).

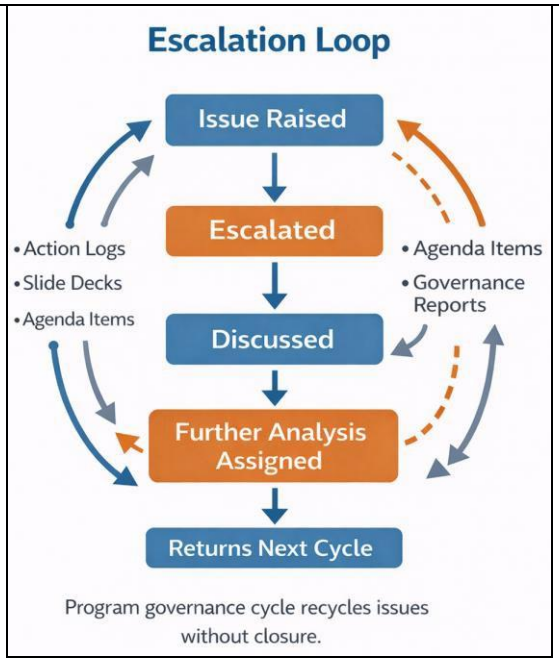

Figure 1 – The Escalation Loop
Issues can continue moving through formal governance cycles without reaching decision closure.

2.3 Planning Integrity

These signals reflect whether the program is still operating against a coherent and shared plan.

• Consistency between integrated plan and local workstream plans

Local plans should align with the integrated plan, even as updates occur. As drift develops, inconsistencies persist. Workstreams adjust locally while the integrated plan catches up late, if at all. Coordination shifts from planned to reactive adjustment.

• Emergence of parallel or unofficial plans

A program should operate against one shared plan, even if imperfect. As drift develops, multiple versions and formats of the plan emerge. These are often informal and exist to compensate for missing decisions. Once parallel plans take hold, alignment becomes increasingly difficult to restore.

Different parts of the program begin using different tools, timelines, and assumptions. Integration points multiply, and reconciliation becomes manual and retrospective. No single view reflects the actual delivery status. In project governance terms, this is close to the normalization of deviance problem: local adaptations that seem reasonable in isolation can become accepted operating practice even as they weaken control across the wider program (Dekker & Pruchnicki, 2013; Vaughan, 1996; Pinto, 2014).

This becomes visible when identical process requirements, such as a global 'Order-to-Cash' flow, begin to show distinct, uncoordinated configuration paths across workstreams or regions. When the "Global Template" becomes nominal rather than functional, realigning the program becomes harder, slower, and more politically loaded.

2.4 Execution Anchoring

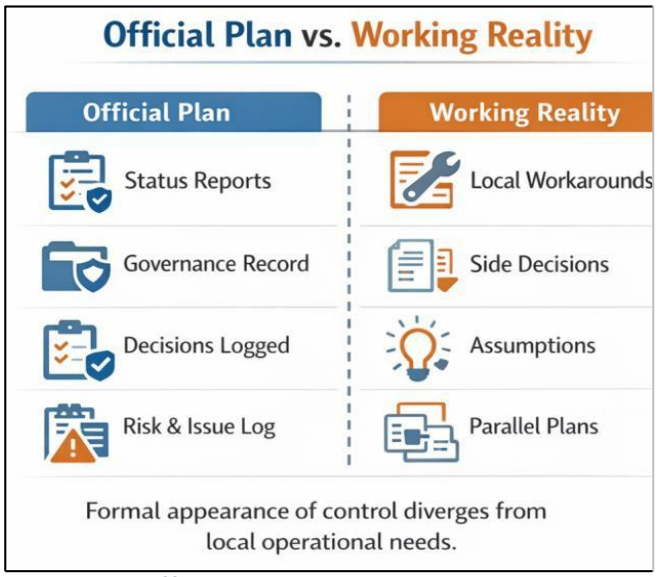

Figure 2 – Official Plan vs. Working Reality
As drift develops, formal reporting and local execution assumptions progressively diverge.

These signals reflect whether execution remains tied to formally agreed decisions.

- **Volume of work proceeding under assumptions**

Assumptions should be limited and explicitly visible. As drift develops, assumptions become the default mechanism for maintaining progress. Work continues, but not on confirmed decisions.

Once enough work proceeds on assumptions, governance may still appear to retain authority. In practice, it is no longer choosing between viable options. It is ratifying the least disruptive path left standing.

That is how dead reckoning becomes dangerous. It is not useless, but the longer it runs without correction, the less trustworthy it becomes (Duncan & Schladow, 1981).

- **Rework driven by late or reversed decisions**

Rework should be limited and traceable. As drift develops, rework increases as decisions arrive late or change. This is not primarily a delivery issue. It is a symptom of delayed or unstable decision flow.

These dimensions are intentionally limited. Drift does not need to be visible everywhere at once. A sustained deterioration in any one dimension is often enough to show that control is weakening.

3. Why Drift Stays Invisible

When drift is eventually acknowledged, it is often attributed to execution weaknesses: insufficient rigor, lack of accountability, poor planning, or resource shortfalls. These explanations are appealing because they fit familiar narratives and justify adding more controls.

They are often incomplete.

3.1 Technical and Metric Blindness

In practice, drift emerges even in programs with mature PMOs, strong vendor governance, and comprehensive reporting. The problem is not the absence of information, but the dilution of meaning. As programs grow in complexity, the volume of status data increases while the clarity of decision-relevant signals declines (Roetzel, 2019). This is why weak-signal detection matters: high- reliability research emphasizes the need to remain sensitive to small anomalies and avoid simplifying uncomfortable signals too quickly (Weick & Sutcliffe, 2007).

This creates a paradox: the more reporting a program produces, the harder it becomes to see drift forming (Roetzel, 2019). A navigator can produce a neat dead-reckoning plot and still be off track. The quality of the chart does not guarantee the accuracy of the position.

Drift is not explained by execution weakness alone. It becomes visible when the mechanisms that connect signals, decisions, plans, ownership, and coordinated execution begin to degrade.

This is not necessarily because the instruments are absent or poorly implemented. It is because they were not designed to measure enough of what drift requires leaders to see (Koskela & Howell, 2002). They reflect a classical project management logic in which

progress can be understood primarily through decomposition, plan adherence, and variance against baseline. That logic works reasonably well for stable and decomposable work, but it is less effective in environments defined by uncertainty, ambiguity, and complexity (Koskela & Howell, 2002; Pich, Loch, & De Meyer, 2002; Bosch-Rekveldt et al., 2011).

Drift emerges where these instruments stop being enough: when they confirm activity but cannot show whether signals, decisions, plans, ownership, and execution still align under pressure.

The gap becomes clearer when the usual control instruments are tested against what drift actually requires leaders to see.

Traditional instrument	What it is designed to show	Why it fails under drift	What it does not capture
Milestone tracking	Progress against planned deliverables	Milestones can be achieved under assumptions or partial alignment	Whether the decisions underpinning those milestones are stable and aligned
RAID logs (Risks, Actions, Issues, Decisions)	Visibility of known risks and issues	Escalations can remain open or recycle without resolution	Whether escalation leads to timely and binding decisions
Status reporting (RAG)	Overall program health and confidence	Status reflects negotiated consensus, not underlying execution readiness	Divergence between reported status and actual delivery conditions
Defect and test metrics	Quality and completeness of delivered functionality	Defects surface late and reflect consequences, not causes	Whether upstream decisions are driving instability and rework
Integrated plan and schedule tracking	Alignment to baseline timelines	Plans can remain formally intact while local execution diverges	The existence of parallel or unofficial plans and assumption-driven work
Governance forums and escalation structures	Formal decision-making pathways	Forums can generate discussion and analysis without closure	Decision latency and the conversion of escalation into resolution

Current S/4HANA migration evidence supports that conclusion: programs can show extensive delivery activity while still slipping for reasons rooted in weak governance and fragmented accountability rather than in the technical migration itself (ISG, 2026b).[12]

What they do not fully measure is whether the program can still convert signals into

[12] ISG's public reporting on S/4HANA programs identifies weak governance and fragmented accountability as primary drivers of slippage, even where delivery activity remains high and technical migration work remains underway.

decisions, decisions into updated plans and ownership, and those decisions into coordinated execution at the pace required to sustain delivery (Cooke-Davies et al., 2007; Pich, Loch, & De Meyer, 2002). Drift emerges in the gaps between those steps.

Part of the difficulty is that drift rarely appears as a clean, isolated signal. It overlaps with the familiar problems every large program already has: scope growth, schedule variance, resource pressure, defect trends, unresolved dependencies, and execution noise. Those problems matter, but they can also hide the deeper pattern. What looks like ordinary delivery turbulence may actually be evidence that the decision system is no longer keeping scope, plans, dependencies, and execution aligned.

Drift is therefore not a failure of governance presence. It is a failure of governance effectiveness under load.

3.2 Governance Theater

A second reason drift stays invisible is that governance can remain highly active even as its practical control weakens. Meetings continue. Status decks are updated. Risks are reviewed. Actions are assigned. From the outside, the program appears engaged and responsive.

But visible governance activity is not the same as effective control.

Under drift, reporting often becomes a staging mechanism rather than a detection mechanism. Its purpose shifts, usually without anyone saying so explicitly. Instead of helping the program confront uncertainty, it helps the program preserve the appearance of order. Status is negotiated into something acceptable. Escalations are absorbed into further analysis. Open issues are reframed as work in progress. The form of control remains intact while the substance of control weakens.

This is why drift can persist inside mature reporting environments. Governance does not disappear. It becomes performative. It continues to generate evidence of oversight, but not necessarily evidence of resolution.

The result is a dangerous mismatch between what governance appears to do and what it actually achieves. The program looks controlled because the machinery of control is still running. But the machinery is no longer producing timely closure, shared reality, or reliable execution. Reporting begins to mask drift rather than expose it.

3.3 Behavioral and Structural Resistance

These warning signs are not just hard to see. They are often filtered out by the way programs are governed, incentivized, and commercially structured.

Incentives reward stability over accuracy

Most large programs reward leaders for maintaining confidence, not for surfacing uncertainty. Yellow status is tolerated. Red status triggers scrutiny, escalation, and reputational risk. The rational response is to delay escalation until certainty is unavoidable. By then, the range of viable options is usually narrower.

Empirical work on organizational incentives shows that reporting systems often evolve to protect legitimacy rather than accuracy under uncertainty, reinforcing optimistic bias and suppressing early negative signals (Meyer & Rowan, 1977; Flyvbjerg & Gardner, 2023).

Defensive governance behavior

Resistance to calling out drift is rarely overt. More often, it shows up in the operating behavior of the program itself. Identifying drift or moving to red status triggers immediate audits, additional oversight, and status-heavy meetings. For delivery leads already under pressure, the effort required to manage this response is often experienced as a distraction from actual recovery. The predictable result is that early warning signals are softened, delayed, or suppressed.

A second mechanism is the protection of institutional legitimacy. Because governance is often treated as the control room of the program, admitting drift can be read as admitting that the control mechanism itself is failing. Stakeholders therefore have an incentive to normalize divergence rather than escalate it. The program absorbs instability instead of naming it.

Structural commercial misalignment

A significant driver of drift is the built-in misalignment between client goals and traditional SI commercial models.

This risk becomes sharper in multi-vendor environments, where unclear responsibility boundaries, weak decision rights, and fragmented acceptance ownership allow delay to accumulate while each party can still plausibly claim that the core problem sits elsewhere (ISG, 2026b).

In many time-and-materials transformations, the cost of drift falls faster on the client than on the implementation partner. Delay, rework, and prolonged ambiguity damage the business case, but do not always create an immediate financial penalty for the provider. Under those conditions, organizational friction and unresolved divergence can be normalized as ordinary project complexity rather than surfaced as a structural threat to delivery.

4. Measuring Drift

Drift is not hard to measure because tools are missing or because reporting has failed. It is hard to measure because most instruments were not designed to detect drift in the first place.

Traditional program controls are built to track activity, output, compliance, and schedule movement. That reflects a classical project management logic: structure the work, measure variance, and correct execution against plan. Those disciplines still matter, but they are better suited to relatively stable and decomposable problems than to environments defined by uncertainty, ambiguity, and complexity (Koskela & Howell, 2002; Pich, Loch, & De Meyer, 2002; Bosch-Rekveldt et al., 2011). Drift becomes visible where that logic reaches its limit. What starts to fail is not activity tracking alone, but the relationships that keep signals, decisions, plans, ownership, and execution aligned under pressure (Cooke-Davies et al., 2007).

4.1 The Dimensions of Drift

If traditional instruments do not fully reveal drift, the next question is what else needs to be observed.

For measurement, the four practical areas can be translated into five dimensions:

1. Decision Flow: Can the program make decisions at the pace required?
2. Decision Closure: Do escalations result in clear, binding outcomes?
3. Planning Coherence: Is there still one plan, or multiple competing realities?
4. Execution Anchoring: Is work based on confirmed decisions, or assumptions?
5. Cross-View Consistency: Do different parts of the program operate on the same understanding of reality?

Decision Latency as a Major Drift Mechanism

Decision latency is the widening gap between when a decision is needed and when it is

actually made.

In large transformation programs, this gap rarely appears as open indecision. More often, it appears as analysis requests, conditional approvals, deferred commitments, and repeated escalation cycles that generate activity without resolution. Decisions are technically "in progress," but delivery continues without the clarity those decisions were meant to provide (Pich, Loch, & De Meyer, 2002).

As decision latency increases, teams compensate locally. They make assumptions, proceed conditionally, or resolve issues in side conversations to avoid blocking progress. This may preserve momentum in the short term, but it fragments authority and accelerates drift by disconnecting execution from formal governance.

Program drift is therefore not just slow delivery. Decision latency becomes a drift mechanism when unresolved choices are carried forward as assumptions, fragment plans, and detach execution from agreed direction.

4.2 Drift Indicators

Drift does not require complex measurement. It is observable through a small number of signals.

Dimension	Observable signal	Practical proxy	What it indicates
Decision Flow	Time to decision	Days / governance cycles to close key decisions	Slowing governance throughput
Decision Closure	Escalation resolution rate	% resolved within one cycle	Escalation without outcome
Decision Closure	Repeat agenda items	Same topic reappearing without progress	Decisions not landing
Planning Coherence	Plan divergence	Mismatch between integrated and local plans	Loss of single source of truth
Planning Coherence	Parallel plans	Existence of unofficial plans	Governance losing control
Execution Anchoring	Assumption-driven work	Volume of work without confirmed decisions	Execution detached from governance
Execution Anchoring	Rework from late decisions	Reversal or reconfiguration effort	Cost of delayed decisions
Cross-View Consistency	Signal variance	Different statuses across forums/workstreams	Fragmented reality

Not all drift is equal. Programs move through stages, and a simple temperature model is more

useful than a complex index.

- **Stable**: Decisions close within expected cycles. Alignment is fast. One coherent plan exists. Assumptions are limited and explicitly tracked.
- **Strained**: Decisions take longer, but still close. Some agenda items repeat. Minor divergence begins to appear between integrated and local plans. Assumptions are increasing but remain visible.
- **Drifting**: Decisions are frequently delayed or conditional. The same topics return without resolution. Parallel plans begin to emerge. Work proceeds under assumptions.
- **Critical**: Escalations no longer produce timely decisions. Alignment increasingly depends on side conversations. Plan divergence becomes material. Rework is rising.
- **Terminal**: Governance no longer drives execution. Decisions are made outside formal structures. No single version of reality holds. Delivery instability becomes visible externally.

4.3 The Lifecycle of Drift

Drift tends to emerge in a predictable pattern across a large enterprise implementation lifecycle. In early phases, visible drift is often low. This does not necessarily mean the program is healthy. It usually means the decision system has not yet been tested under real strain.

The first real inflection point typically appears during design, for example in the Explore phase of SAP Activate. This is where solution fit is tested, requirements become concrete, and cross-functional dependencies begin to surface. At that point, governance is no longer processing abstractions. It is processing real decisions under time pressure.

Decision latency often rises early, but it is not the only entry point. Drift can also begin through unvalidated work, local plan changes, fragmented ownership, or incomplete closure. These mechanisms reinforce one another as teams work around uncertainty through assumptions, conditional progress, or local interpretation. Visible instability usually lags behind them. It tends to become materially apparent during build and testing, for example in Realize, and intensifies toward deployment.

Drift becomes most visible during integration, testing, and cutover. These phases compress timelines, concentrate dependencies, and expose decisions that were deferred earlier

(Bosch-Rekveldt et al., 2011).

As programs progress through integration, testing, and cutover preparation, the volume of risks, dependencies, and decision permutations increases sharply. Governance structures, however, often remain largely static. Senior forums face more topics, tighter time pressure, and higher stakes, without a corresponding increase in decision throughput or clarity. This is when the current finally becomes visible in the track made good.

Ironically, this is also when programs are least willing to acknowledge instability. Schedules are public. Go-live dates are politically sensitive. The cost of admitting uncertainty rises sharply.

As a result, governance often becomes more formal and less effective at exactly the wrong moment.

ERP implementation research consistently identifies coordination, communication, project management, and integration complexity as major risk drivers (Aloini, Dulmin, & Mininno, 2007; Somers & Nelson, 2001).

The point is not that drift is created late. It is that drift becomes visible late. By the time delivery instability is obvious, the underlying decision system has usually been under strain for some time.

That pattern is consistent with current SAP migration research arguing that many failures are shaped early, during planning and operating-model decisions, long before visible production distress makes the problem undeniable (ISG, 2026a).[13] Recent EY and Oxford research on SAP S/4HANA transformations points in a similar direction: transformation outcomes depend not only on technical migration activity, but also on leadership behavior, role clarity, decision discipline, psychological safety, and the ability to treat S/4HANA as a business transformation rather than a narrow IT project.

4.4 The Economics of Drift

The cost of drift is not limited to schedule slippage. It appears as rising burn, avoidable rework, degraded quality, and the progressive loss of recovery options.

Current SAP transformation evidence suggests that this degradation is not confined to

[13] ISG states that many SAP migration failures originate in the planning phase, with early operating-model, governance, and sequencing decisions shaping later outcomes

timeline alone. In a 2025 Horváth study, the strongest deviations from plan were reported in quality and budget, with scope expansion cited as a leading cause. That reinforces the point that drift erodes delivery performance more broadly than schedule alone (Horváth, 2025).

A major amplifier is demand-side latency. When business stakeholders delay alignment or decisions, delivery teams proceed on placeholders, assumptions, or provisional interpretations. That creates hidden debt, which is usually repaid later during integration, testing, and cutover, when correction is far more expensive.

The cost of correction is therefore not linear. It rises as decision latency combines with workstream divergence. A six-week delay in a core design decision can trigger a sharp increase in downstream rework, because work built on incorrect assumptions must later be reversed, reconciled, or rebuilt.

This is the Drift Premium: the cost of maintaining the appearance of progress while structural control is weakening. [14]

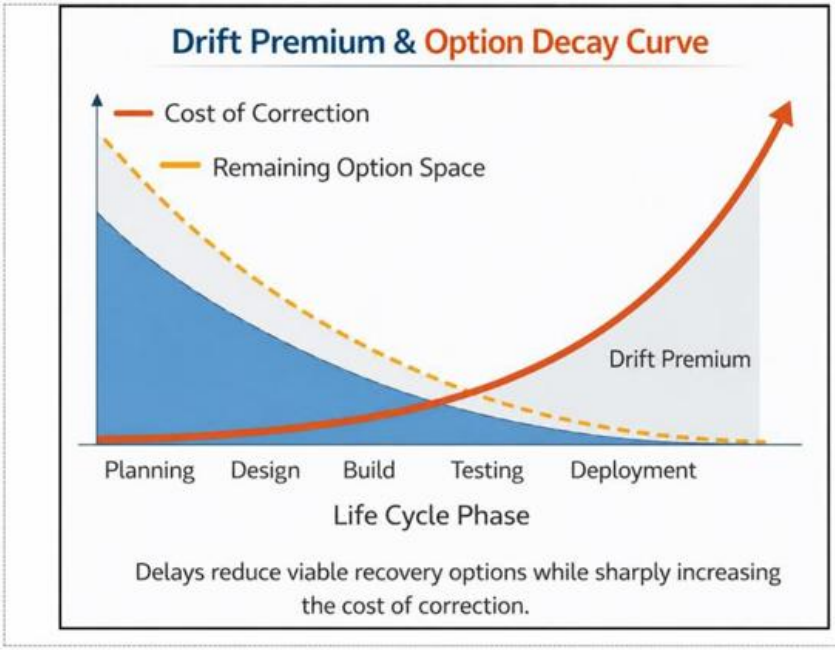

Figure 3 – Drift Premium and Option Decay Across the Program Lifecycle

[14] The Drift Premium is used here as a practitioner-derived term for the additional cost created when work continues under unresolved conditions and must later be reconciled, reversed, or rebuilt.

In current SAP programs, that often means protecting near-term stability, timeline survival, and budget predictability while postponing the harder process, data, and operating-model changes needed to create longer-term transformation value (ISG, 2026b). [15]

If left unresolved, this accumulation leads to what can be described as the Cliff Effect.

The Cliff Effect occurs when the cost of correction exceeds the organization's practical ability to recover. At that point, drift stops being an internal governance problem and becomes an external operational failure.

Programs approaching this point usually show visible signals in advance, especially persistent decision delays, repeated escalation cycles, and rising rework. The problem is not that the signals are absent. It is that they are often normalized until failure becomes visible at go-live or in production.

4.5 Modeling and Limits

The framework presented in this paper is intentionally diagnostic. It is designed to show where drift becomes visible and how it can be countered in practice. It does not attempt to define drift as a single precise variable.

The constructs below are directional and exploratory. They outline a path toward more explicit modeling of drift, but they are not presented here as validated measures.

One possible construct is a composite **Drift Factor** built from signals such as decision latency, escalation closure, repeat agenda frequency, divergence between official plans and working assumptions, assumption-driven work, and rework attributable to delayed decisions.

A second construct, **Drift Variance**, reflects the degree to which these signals differ across workstreams or governance layers. High variance indicates fragmentation, with different parts of the program operating on different assumptions and different versions of reality.

Correction effort is not linear. When drift is detected early, intervention can focus on restoring decision flow and alignment. Once drift has propagated into planning and execution, recovery requires reconciliation of plans, re-alignment of workstreams, and reversal of work already performed. Correction cost therefore rises nonlinearly as drift spreads.

[15] ISG reports that many enterprises are prioritizing short-term stability over deeper transformation, with limited process re-engineering despite migration to S/4HANA.

At the same time, drift should not be reduced to a false precision. It is multi-dimensional, context-dependent, and influenced by organizational behavior. Attempts to over-quantify it risk shifting attention back to reporting rather than reality.

These indicators are diagnostic, not predictive. They are directional, not absolute.

Programs do not need perfect measurement. They need enough clarity to recognize when decision flow is degrading and to intervene before drift becomes embedded in planning and execution.

5. Restoring Decision Flow

As established earlier, increasing information volume under pressure can degrade decision quality. Effective intervention should narrow the decision space, make trade-offs explicit, and reconnect decisions to plans and execution.

Effective intervention focuses on restoring decision flow and reconnecting it to planning and execution.

Not all decisions serve the same purpose. Some give direction by choosing between competing paths, priorities, or scope boundaries. Others are corrective, taken after drift has begun, to restore alignment, reverse assumption-driven work, and steer execution back onto a coherent track. In both cases, closure requires more than a stated choice: the outcome must be reflected in ownership, plans, dependencies, and observable execution. Corrective decisions become more costly and constrained the longer drift is allowed to spread.

The goal is not to make the chart prettier. It is to regain a reliable fix, understand the set, and steer back onto the intended track.

- **Reduce decision surface**
  Limit the number of topics entering governance forums. Prioritize decisions that materially affect program trajectory. Remove or defer non-critical topics.

- **Force explicit trade-offs**
  Replace conditional approvals and partial agreements with clear choices. Where continued optionality prevents direction, require an explicit choice and record the consequences.

- **Re-anchor execution to decisions**
  Pause work proceeding under unconfirmed assumptions where feasible. Make dependencies visible and require explicit confirmation before continuation.

- **Collapse parallel plans**
  Re-establish a single integrated view of the plan. Where multiple versions exist, force reconciliation rather than allowing divergence to persist.

- **Shorten decision cycles**
  Increase the frequency or focus of decision forums only where it improves closure. The goal is faster resolution, not more discussion.

These interventions are deliberately minimal. They do not require new tools or structural redesign. Their purpose is to restore the program's ability to resolve decisions, reflect them in the plan, and keep execution aligned at the pace required to sustain delivery.

In practice, drift rarely becomes visible across all dimensions at once. It may first appear in decision alignment or closure, but it can also surface through plan divergence, unvalidated work, or repeated rework. Once present, the mechanisms tend to reinforce one another.

For a rapid diagnostic, decision alignment and decision closure are often the best entry points. They provide early, observable signals that the program is no longer converting escalation into an executable outcome.

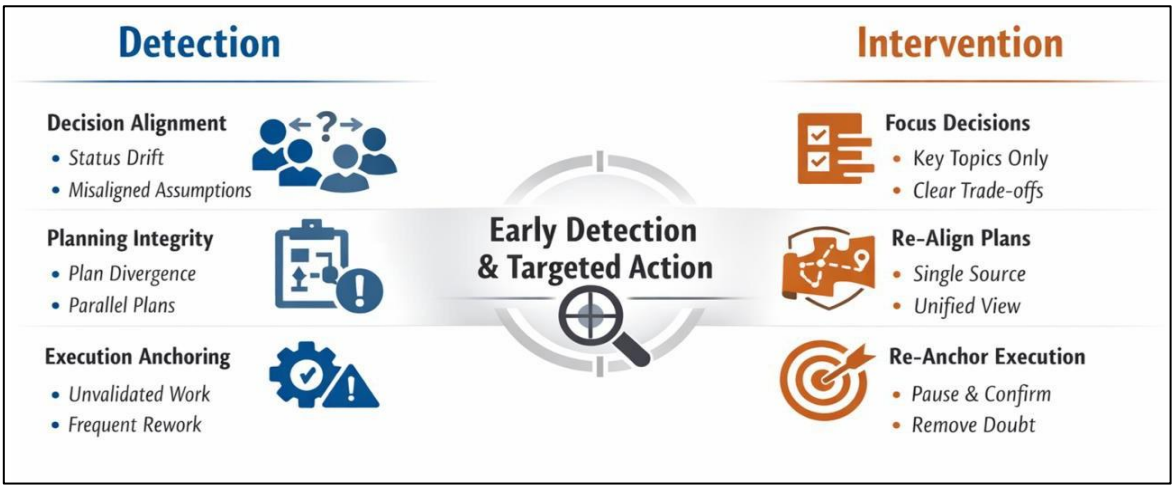

Figure 4 – Early Detection and Targeted Action
Drift becomes manageable when weak signals are identified early and matched with focused intervention in decision alignment, decision closure, planning integrity, and execution anchoring.

5.1 Decision Surface Optimization

To stabilize a drifting program without triggering defensive resistance, leadership must reduce the amount of decision load placed on senior governance. The issue is not authority. It is overload.

The objective is to protect senior forums for the few trade-offs that materially affect program trajectory. Technical detail, local misalignment, and non-critical issues should be resolved in faster tactical channels with clear decision rights. This is not gatekeeping. It is governance triage.

Rather than presenting escalation as an open-ended discussion, leaders should present a small number of explicit trade-offs. Conditional approvals and ambiguous direction preserve motion without resolving the underlying issue. Where a decision is required, the forum should conclude with a named outcome, owner, and effective date, followed by confirmation that plans and execution have changed accordingly.

5.2 The Political Triage of Subtraction

Subtraction is mechanically necessary, but politically sensitive. In many organizations, inclusion is treated as influence, so removing topics from senior governance can be misread as devaluing them.

That is why subtraction must be framed carefully. Topics are not being dismissed as unimportant. They are being routed according to their likely impact on decision flow and program trajectory.

Items that do not materially alter the direction of the program should move to asynchronous review or delegated tactical forums. This keeps input flowing while protecting key decision forums from overload.

5.3 Implications for Project Managers and PMOs

For project managers and PMOs, the implication is uncomfortable but clear: drift cannot be prevented through better reporting alone. In some cases, reporting becomes part of the problem.

The role of the PMO is therefore not just to produce information, but to protect signal quality. That means ensuring that early warnings reach decision-makers without being diluted, delayed, or buried under non-critical noise.

This is a different kind of PMO contribution. It is less about volume, coverage, and reporting

completeness, and more about preserving the program's ability to recognize emerging instability before it becomes operationally expensive.

6. Designing Against Drift

6.1 Psychological Safety as a Diagnostic Prerequisite

Early drift can only be detected if people are willing to surface what is not lining up: delayed alignment, unvalidated assumptions, repeated escalation, and rework. If the culture punishes red status or treats disagreement as failure, those signals will be softened, delayed, or kept out of sight. This is consistent with Edmondson's work on psychological safety, which shows that learning behavior depends on whether team members can speak up, ask for help, discuss errors, and challenge assumptions without fear of interpersonal penalty (Edmondson, 1999).

The most sophisticated detection model is useless if the organization punishes the messenger. Early detection depends on a culture in which red status is treated as a request for attention and corrective action, not an admission of execution failure.

This matters especially for workstream leads, architects, and PMO analysts. If surfacing a drift signal results in a scrutiny tax, the signal will not disappear, but its visibility will. By the time it becomes undeniable, the cost of correction is usually much higher.

6.2 Structural Prevention

Steering committees are usually set up to keep senior stakeholders aligned. Under pressure, that often leads to compromise decisions, conditional approvals, and deferred commitments.

The result may be apparent alignment without an outcome that teams can execute.

Drift is often treated as an execution problem to be corrected once it becomes visible. In practice, many of the conditions that allow drift to form are introduced much earlier through the design of governance, decision rights, escalation paths, and reporting structures.

These design choices do not eliminate drift. They reduce the conditions under which it forms, delay its onset, and improve the program's ability to detect and respond before it propagates into planning and execution.

Designing against drift requires deliberate choices about how governance reaches closure; adding process alone will not achieve that.

Governance should be designed for decision throughput, not representation. Many programs optimize for coverage, ensuring that all functions, stakeholders, and topics are present in governance

forums. Under pressure, this leads to overloaded agendas, diluted accountability, and slower decisions.

Decision ownership should be explicit and singular. Shared accountability often delays closure because participants defer commitment in the name of alignment. Clear ownership does not remove the need for input, but it does prevent responsibility for closure from dissolving across the group.

Escalation paths should function as resolution mechanisms, not visibility mechanisms. In many programs, escalation surfaces issues to higher forums without ensuring that those forums are structured to resolve them. The appearance of control increases, but decision latency does too.

Reporting and decision-making should be separated. When governance forums are dominated by status updates, decision time shrinks and attention fragments. Reporting should provide the evidence required for governance to decide; allowing the two activities to consume the same forum weakens both.

The number of active decision forums should be constrained. As programs grow, additional forums are often introduced to manage complexity. While this may improve local coordination, it also increases the risk of fragmented decision-making and inconsistent outcomes. A smaller number of clearly defined forums with explicit scope and authority is usually more effective.

Finally, incentives should favor early signal visibility. If leaders are rewarded for maintaining confidence rather than surfacing uncertainty, drift will be delayed rather than prevented.

These design choices do not remove drift as a risk. They preserve the reliability of decision flow under pressure and improve the odds that instability is detected while it is still manageable.

6.3 Why Drift Persists

The following dynamics are rarely the result of individual bad faith or negligence. They are more often the predictable outcome of structural misalignment inside complex programs.

When a transformation measures success through milestone completion rather than whether decisions have actually changed plans and execution, stakeholders are naturally

pushed toward preserving the appearance of stability rather than surfacing emerging risk. In that environment, drift is usually not deliberate. It is the product of competing pressures inside the system.

Three patterns are especially common.

Risk-averse decision behavior

For senior leaders, a definitive high-stakes decision creates a visible point of accountability. Delay, conditionality, or non-decision can feel safer. Drift therefore preserves a form of conditional viability in which the program remains active, but hard choices are deferred.

Incentives tied to scale and recovery

In some environments, drift creates space for a mild form of empire building. Programs under strain often justify additional resources, oversight structures, recovery teams, and management attention. That can expand budgets, headcount, and internal influence, even when simplification or earlier decision closure would be healthier for the program.

This does not mean stakeholders want failure. It means the system does not always reward early reduction of complexity and can sometimes reward the visible management of instability instead.

Commercial and technical opacity

When implementation partners face product limitations, delivery gaps, or unresolved design weaknesses, drift can create a convenient fog of war. Complexity, dependency, and alignment language absorb attention while underlying technical or delivery weaknesses remain partially obscured.

These dynamics do not make drift inevitable. But they help explain why drift can persist even when many participants recognize that the program is no longer operating cleanly. The issue is not simply poor discipline. It is that some of the surrounding structures make ambiguity easier to sustain than resolution.

7. Conclusion

Program drift is not an anomaly. It is a predictable outcome of how large organizations govern complex change under pressure (Kreiner, 1995). Delivery activity continues, milestones are reported, and governance structures remain in place, but the program's ability to keep signals, decisions, plans, ownership, and execution aligned weakens over time

(Dekker, 2011; Dekker & Pruchnicki, 2013).

The practical risk is that drift becomes visible late. Traditional indicators usually measure output, completion, and variance after the fact. They are less effective at showing whether the program still has a shared operating reality, whether decisions are closing fast enough, and whether execution remains anchored to those decisions.[16] The difficulty is that drift signals rarely appear in isolation. They mix with familiar project symptoms such as scope growth, schedule variance, defect pressure, resource constraints, and unresolved dependencies. Those problems may be real, but they can also obscure the deeper pattern: the program's decision system is no longer keeping scope, plans, dependencies, and execution aligned.

That is why early detection matters. Drift usually appears first in decision alignment and decision closure, then propagates into planning and execution. Recognizing drift early does not guarantee success, but missing it makes loss of control much more likely.[17] The presence of drift signals does not guarantee collapse. Many large transformations operate for long periods under elevated decision latency and still reach cutover. The difference is whether leadership recognizes the pattern early enough to reduce ambiguity, close decisions, and protect the remaining recovery options.

Large digital transformation programs do not usually fail for lack of rigor, funding, or talent alone. They lose control when governance activity no longer keeps decisions, plans, ownership, and execution aligned. Decision latency is a major contributor, but not the only one. False closure, assumption-based continuation, plan divergence, unvalidated work, fragmented ownership, and the well-intended preservation of continuation paths can produce the same condition. This paper has focused on how those mechanisms erode control under rising complexity; the quality of the decisions themselves deserves separate treatment.

The central responsibility of program leadership is therefore to preserve the program's

[16] The point is not that conventional project management disciplines such as planning, sequencing, baseline control, and variance management are wrong or obsolete. They remain necessary. The narrower claim is that they explain only part of what determines outcomes in large transformations. Under sustained complexity, control depends not only on plan quality, but also on whether the program can sustain decision quality, preserve shared reality, and keep execution aligned over time.

[17] Oxford Saïd Business School and EY report that 96% of transformation programs encounter issues serious enough to create a turning point. In their analysis, early recognition of those issues increased the likelihood that the turning point would significantly improve performance by 107%. They also found that delegated decision authority and clarified roles were far more common in successful turning points than in unsuccessful ones.

ability to recognize deviation, reach timely decisions, translate those decisions into plans and execution, and verify that the underlying condition has changed while meaningful recovery options still exist.

References

Aloini, D., Dulmin, R., & Mininno, V. (2007). Risk management in ERP project introduction: Review of the literature. Information & Management, 44(6), 547-567.

Bosch-Rekveldt, M., Jongkind, Y., Mooi, H., Bakker, H., & Verbraeck, A. (2011). Grasping project complexity in large engineering projects: The TOE (Technical, Organizational and Environmental) framework. International Journal of Project Management, 29(6), 728-739.

Cooke-Davies, T., Cicmil, S., Crawford, L., & Richardson, K. (2007). We're not in Kansas anymore, Toto: Mapping the strange landscape of complexity theory, and its relationship to project management. Project Management Journal, 38(2), 50-61.

Dekker, S. W. A. (2011). Drift into failure: From hunting broken components to understanding complex systems. Ashgate Publishing.

Dekker, S., & Pruchnicki, S. (2013). Drifting into failure. Theoretical Issues in Ergonomics Science, 15(6), 534-547.

Duncan, C. P., & Schladow, S. G. (1981). World surface currents from ship's drift data. The International Hydrographic Review, 58(2), 91-112.

Edmondson, A. (1999). Psychological safety and learning behavior in work teams. Administrative Science Quarterly, 44(2), 350-383.

Eisenhardt, K. M. (1989). Making fast strategic decisions in high-velocity environments. Academy of Management Journal, 32(3), 543-576.

EY & Saïd Business School, University of Oxford. (2025). Maximize value in SAP S/4HANA transformations. EY.

Flyvbjerg, B. (2014). What you should know about megaprojects and why: An overview. PM World Journal, 3(2), 1-10.

Flyvbjerg, B., & Gardner, D. (2023). How big things get done: The surprising factors behind every successful project, from home renovations to space exploration. Currency.

Horváth. (2025). Business transformation unlocked: Maximizing the benefits of SAP S/4HANA.

Information Services Group. (2026a). 2026 state of SAP migrations report. ISG.

Information Services Group. (2026b, February 2). Firms moving to SAP S/4HANA prioritize short-term stability over long-term transformation: ISG study. ISG.

Infrastructure and Projects Authority (UK Government). (2022). Bad omens: Signs your transformation programme will struggle and how to avoid it. UK Government.

Kahneman, D. (2011). Thinking, fast and slow. Farrar, Straus and Giroux.

Kahneman, D., Lovallo, D., & Sibony, O. (2011). Before you make that big decision. Harvard Business Review, 89(6), 50-60.

Kappelman, L. A., McKeeman, R., & Zhang, L. (2006). Early warning signs of IT project failure: The dominant dozen. Information Systems Management, 23(4), 31-36.

Keil, M., Mann, J., & Rai, A. (2000). Why software projects escalate: An empirical analysis and test of four theoretical models. MIS Quarterly, 24(4), 631-664.

Khalid, M., Khan, R. A., Khushnood, M., Aslam, S., Khattak, Z. Z., & Abbas, S. (2022). An empirical analysis of the influence of project governance and information technology governance on project delay. Global Business Review. Advance online publication. https://doi.org/10.1177/09721509221121179

Koskela, L., & Howell, G. A. (2002). The underlying theory of project management is obsolete. Paper presented at PMI Research Conference 2002: Frontiers of Project Management Research and Applications, Seattle, Washington. Newtown Square, PA: Project Management Institute.

Kreiner, K. (1995). In search of relevance: Project management in drifting environments. Scandinavian Journal of Management, 11(4), 335-346.

Levina, N., & Vaast, E. (2008). Innovating or doing as told? Status differences and overlapping boundaries in offshore collaboration. MIS Quarterly, 32(2), 307-332.

Meyer, J. W., & Rowan, B. (1977). Institutionalized organizations: Formal structure as myth and ceremony. American Journal of Sociology, 83(2), 340-363.

Pich, M. T., Loch, C. H., & De Meyer, A. (2002). On uncertainty, ambiguity, and complexity in project management. Management Science, 48(8), 1008-1023.

Pinto, J. K. (2014). Project management, governance, and the normalization of deviance. International Journal of Project Management, 32(3), 376-387.

Roetzel, P. G. (2019). Information overload in the information age: A review of the literature from business administration, business psychology, and related disciplines with a bibliometric approach and framework development. Business Research, 12, 479-522.

Saïd Business School, University of Oxford, & EY. (2024). Transformation leadership: Navigating turning points. Saïd Business School, University of Oxford and EY.

Simon, H. A. (1957). Administrative behavior. Macmillan.

Somers, T. M., & Nelson, K. (2001). The impact of critical success factors across the stages of

enterprise resource planning implementations. Proceedings of the 34th Hawaii International Conference on System Sciences.

Staw, B. M. (1981). The escalation of commitment to a course of action. Academy of Management Review, 6(4), 577-587.

Weick, K. E., & Sutcliffe, K. M. (2007). Managing the unexpected: Resilient performance in an age of uncertainty. Jossey-Bass.

Williams, T., Klakegg, O. J., Walker, D. H. T., Andersen, B., & Magnussen, O. M. (2012). Identifying and acting on early warning signs in complex projects. Project Management Journal, 43(2), 37-53.

About the Author

Cornelius Greyling

Florida, United States

Cornelius Greyling is a senior project and program leader with nearly 30 years of experience working on large, complex digital transformation programs. His career began in Swiss financial services and has since taken him across Europe, Latin America, and the United States, leading enterprise transformations across a wide range of industries. He has held senior delivery and leadership roles at SAP, Natuvion, SNP, and international consulting organizations. His work focuses on how large transformation programs lose control while continuing to appear active, structured, and governed. He is PMP-certified and the author of *Designed to Drift*. He can be contacted through linkedin.com/in/greyling