The Documentation Gap: Why Divergence After Agreement Is Inevitable Without Anchoring [1]

Pradeep Billa

Abstract

Existing project failure frameworks — from PMI's governance model to McKinsey's execution model to the Standish Group's CHAOS Report — locate failure in breakdown: breakdown of process, alignment, or communication. This article introduces and defines a distinct failure category they do not capture: the Documentation Gap. The Documentation Gap describes misalignment that occurs after agreement, not because of poor communication, but because implicit expectations were never converted into written artifacts before execution began. Observed consistently across dozens of ERP implementations over 19 years — in Workday, PeopleSoft, and Oracle environments — the Documentation Gap surfaces two to six weeks after the decision point, at delivery review. The article distinguishes the Documentation Gap from adjacent failure types, presents three cross-platform case studies demonstrating its mechanism, situates it within the Silent Handshake Framework alongside the Transition Gap and Assumption Gap, and identifies anchoring — a four-primitive discipline of explicit scope, assumptions, exclusions, and definitions — as the only intervention that closes it. Agreement without anchoring is temporary alignment. Temporary alignment guarantees divergence. The only control point is the moment of agreement.

Keywords: *ERP implementation, project failure, documentation gap, Silent Handshake Framework, anchoring, scope management, Workday, PeopleSoft, implementation risk, enterprise technology*

Introduction

Project failure literature has a consensus explanation.

[1] How to cite this article: Billa, P. (2026). The Documentation Gap: Why Divergence After Agreement Is Inevitable Without Anchoring, commentary, *PM World Journal*, Vol. XV, Issue VI, June.

PMI frames failures as governance breakdowns. McKinsey frames them as execution failures. The Standish Group measures them in cost and schedule overruns. All three are correct about what they measure. None of them explain the failure mode that accounts for a disproportionate share of rework, eroded trust, and delivery misalignment in enterprise implementations.

The failure that happens after the decision is made. After both parties agreed. After the meeting ended with everyone nodding.

That failure has a name. It is the Documentation Gap. It is not a subset of communication failure. It is not scope creep. It is a distinct failure category, operating through a mechanism that existing frameworks do not isolate because they are not looking in the right place.

And it is not a risk. It is an inevitability, unless one specific act is performed at the moment of agreement.

What Existing Models Miss

PMI's governance model assumes that if proper process exists, failure is a deviation from it. McKinsey's execution model assumes that if the right people are aligned, delivery follows. Both models locate failure in breakdown: breakdown of process, breakdown of alignment, breakdown of communication.

The Documentation Gap operates differently. There is no breakdown. The communication happened. The alignment was real. The decision was mutual.

The failure is that implicit expectations were never converted into explicit, transferable artifacts before execution began. Both parties left the exchange carrying different versions of the same agreement, not because they disagreed, but because the agreement lived in memory rather than in writing. Memory drifts. Memory fills gaps with assumptions. Memory, over six weeks of a live implementation, becomes unreliable in ways that neither party notices until delivery.

This is not what PMI measures. This is not what McKinsey fixes. This is a micro-level failure anatomy that sits between agreement and execution, invisible until the delivery lands wrong.

Governance and execution models fail to prevent post-agreement divergence because they assume alignment is stable once achieved. The Documentation Gap demonstrates

that assumption is false. Alignment achieved in conversation is not alignment. It is the beginning of alignment. It becomes stable only when it is anchored in writing.

Recognizing the Documentation Gap as its own failure category shifts teams from correcting misalignment after delivery to preventing it through pre-execution anchoring, materially reducing rework and increasing delivery predictability across enterprise implementations.

What the Documentation Gap Is, and What It Is Not

Precise classification matters here. The Documentation Gap is not interchangeable with related failures. It has hard boundaries.

A Documentation Gap exists when:

1. A decision is explicitly made and acknowledged by both parties
2. No breakdown in communication occurs at the time of agreement
3. Misalignment emerges at delivery due to expectations that were never externalized in writing

It is not a Documentation Gap when:

- Requirements were unclear at the time of discussion
- Parties disagreed during the decision
- Information was lost during a handoff between teams or phases

That last point is critical. The Documentation Gap is not a handoff failure. The handoff succeeded. Both parties left the exchange believing the same thing. The failure is that the belief was never converted into a written artifact that both parties could check against six weeks later.

This distinction matters because it demands a different diagnosis and a different intervention. Fixing handoff processes does not close a Documentation Gap. Improving communication training does not close a Documentation Gap. The only thing that closes it is anchoring.

Why Divergence Is Inevitable Without Anchoring

The mechanism is structural, not behavioral.

When a client makes a request, they are not describing a deliverable. They are describing a compressed version of an outcome they have already visualized, shaped by organizational history, unstated requirements, and context that feels self-evident from inside their business. The artifact they produce — an email, a Slack message, a verbal request — is a representation of that vision. It is not the vision itself.

The vendor receives the compression. They have no access to the original. Nothing in the exchange signals that the compression is incomplete, because the client does not experience it as incomplete. They experienced alignment. They left nodding.

The deterministic reality that follows from this structure: as long as expectations remain internal and artifacts remain compressed representations of those expectations, divergence between intent and delivery is not a risk. It is an inevitability. The gap is not created by error or negligence. It is created by the structure of how expectations are held and transferred. Two parties cannot execute against the same understanding when one party's understanding was never fully externalized.

Alignment achieved in conversation degrades over time when it is not anchored in writing. That degradation is predictable, measurable, and consistent across implementations.

Three Implementations: The Same Mechanism

Observed across dozens of ERP implementations over 19 years, the Documentation Gap manifests consistently. It typically surfaces two to six weeks after the decision point, at the moment of delivery review. The pattern holds across systems, project types, and client contexts.

Workday Implementation: Analytics Report Request

A business team submitted a casual email requesting a performance analytics report. The vendor built and delivered against the email. Weeks later, the head of the business team responded: the delivery did not honor the request as intended. The original ask had never been scoped. The business carried a comprehensive analytics vision. The vendor built against a few lines of text.

The verdict from the client: "the vendor should have known."

That phrase is the signature of the Documentation Gap. "Should have known" means: the expectation existed in full inside my head, and I assumed the compressed artifact I sent you was sufficient to transfer it. It was not. Because the communication felt complete to the sender, the gap was invisible until delivery.

PeopleSoft to Workday Migration: Job Data Conversion

The ask: convert all Job Data history from PeopleSoft so managers can see complete employment timelines. Both sides understood the goal. Neither documented the structural assumption underneath it — that the two systems' data models were compatible at a one-to-one level.

They are not. PeopleSoft allows overlapping effective dates, multiple concurrent jobs, and custom action codes. Workday's Position Management model rejected thousands of rows. HR expected full history. The migration team had to collapse, cleanse, and drop significant portions of historical data. The decision had been made. The assumption had been shared. Neither had been written down, examined, or challenged before the work began.

PeopleSoft HCM and Payroll: Retro Pay Configuration

The ask: enable retro pay so that compensation changes trigger automatic back pay calculations. The assumption — undocumented and unexamined — was that the delivered PeopleSoft Retro Pay process would handle any compensation change without additional configuration.

It does not. Retro Pay triggers only for specific action and reason combinations and requires Job Data entries in a precise sequence. HR entered changes out of order. Retro did not trigger, or calculated incorrect amounts. Payroll audited and adjusted manually across dozens of checks.

Across all three cases, the failure did not originate in execution. It originated at the moment of agreement, where expectations were assumed complete but never externalized. The execution team was not wrong. The delivery team was not negligent. The gap was structural, created at the moment the decision was made without an anchor.

The Silent Handshake Framework: Three Gaps, One Pattern

The Silent Handshake Framework defines three distinct failure categories that account for the majority of misalignment in enterprise ERP implementations. These are not communication failures. They are not execution failures. They are structural gaps that open in the space between teams, phases, and assumptions — in the places where existing project failure models do not look.

The three layers are distinct in their timing, root cause, and visibility:

Gap Type	Failure Moment	Root Cause	Visibility
Transition Gap	During handoff	Loss of information between phases or teams	Visible at transfer point
Documentation Gap	After agreement	Unanchored expectations diverge over time	Visible at delivery
Assumption Gap	Before decision	Implied, unspoken logic treated as shared	Invisible until failure

Table 1: The Silent Handshake Framework — Three Failure Categories Compared

The Transition Gap is a handoff failure. Something present in one phase does not survive the transfer to the next. It is detectable at the boundary.

The Documentation Gap is a post-agreement failure. The handoff succeeded. The decision was mutual. The failure is that no written artifact was created to hold the agreement stable through execution. It is detectable at delivery.

The Assumption Gap is a pre-decision failure. No decision was actually made. Both parties proceeded on implied logic that was never surfaced or verified. It is frequently invisible until a project fails entirely.

Together, these three layers map a failure anatomy that existing frameworks do not capture because existing frameworks look for breakdowns. The Silent Handshake fails without breaking. Communication succeeds. Agreement is reached. Teams execute. And the misalignment accumulates silently in the space between what was said, what was meant, and what was written down.

The Four Primitives of Anchoring

The Documentation Gap does not require a documentation policy to close. Policies do not survive schedule pressure on live implementations.

What it requires is anchoring, treated as a discipline with the same standing as any other implementation gate. Anchoring requires four elements, not three. These four primitives are necessary and sufficient to stabilize an agreement through execution. Remove any one of them, and the remaining three degrade. They are not a checklist. They are a system.

1. **Explicit scope.** What is included in this deliverable.
2. **Explicit assumptions.** What must be true for the work to succeed.

3. **Explicit exclusions.** What is not included and will not be delivered.
4. **Explicit definitions.** Shared meaning of key terms, data objects, statuses, and outcomes.

The fourth primitive is the one that collapses without attention, and when it collapses, it takes the other three with it.

In ERP implementations, terms that appear universal behave differently across systems, modules, and teams. When definitions are not anchored, scope becomes unstable because the scope was written against an undefined term. Consider what happens when the following terms are left undefined between a PeopleSoft team and a Workday team:

- "Active employee"
- "Position"
- "Termination date"
- "Retro pay event"
- "Primary job"
- "Effective date"
- "Headcount"
- "Eligibility"

Each of these terms sounds unambiguous. Each carries system-specific logic, module-specific behavior, and team-specific interpretation. Explicit definitions are the load-bearing element of anchoring. Without them, the other three primitives are written in a language that both parties believe they share and do not.

All four elements must be confirmed in writing by both parties before execution begins. Not in a meeting summary sent afterward. Not in a follow-up email that assumes the other party will flag gaps. In a scoped, confirmed artifact that both parties have reviewed and accepted as the basis for the work.

The common substitutes for anchoring — meeting summaries, ticket descriptions, and email confirmations — fail for the same reason. They capture what was said. They do not capture what was meant. They preserve the conversation. They do not stabilize the agreement.

Agreement without anchoring is temporary alignment.

Diagnosing the Silent Handshake in the Field

When misalignment surfaces on an implementation, the failure moment determines the gap type and the intervention.

Before a decision is made, both parties proceed on implied logic that was never surfaced or verified. That is the Assumption Gap.

After a decision is made but before anchoring occurs, both parties execute against the same agreement and arrive at different deliveries. That is the Documentation Gap.

During a handoff between phases or teams, something present in one phase does not survive the transfer. That is the Transition Gap.

Locating the failure moment is not a forensic exercise. It is a diagnostic discipline. The right intervention for a Transition Gap will not close a Documentation Gap. The right intervention for a Documentation Gap will not surface an Assumption Gap. Teams that apply this model can diagnose misalignment in minutes rather than weeks and intervene at the correct failure point.

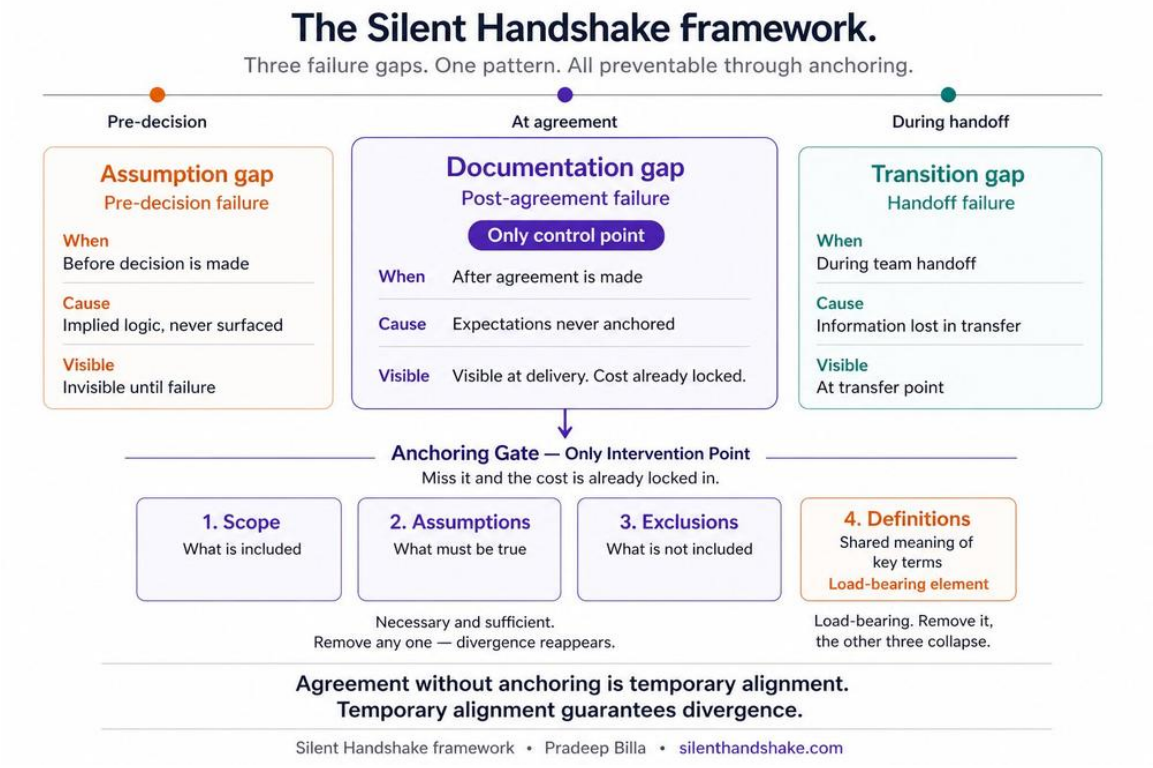

Figure 2: The Silent Handshake Framework — Failure Timing, Cause and Intervention Points

Conclusion

The Documentation Gap converts low-cost clarification into high-cost rework by shifting misalignment detection from agreement to delivery. The work to close a gap at the moment of agreement is measured in minutes. The work to close it at delivery is measured in weeks, budget, and trust.

The cost is distributed across the project in rework cycles, tense emails, and relationships that grow slightly more guarded after deliveries that land wrong. It never traces cleanly to its source. The source is always the same. An agreement was reached. It felt solid. The implicit expectations beneath it were never converted into explicit artifacts before execution began.

The divergence that follows is not bad luck. It is not poor communication. It is the structural outcome of leaving expectations internal when they needed to be external.

Agreement without anchoring is temporary alignment. Temporary alignment guarantees divergence. The only control point is the moment of agreement. Miss it, and the cost is already locked in.

The Silent Handshake Framework maps the structural gaps that open between teams, phases, and assumptions in enterprise technology implementations — in the places where existing project failure models do not look. silenthandshake.com

---

Declaration of AI Use

The author used AI-assisted tools for structural editing and language refinement. All observations, frameworks, named concepts, implementation case studies, and conclusions are the author's original work, drawn from 19 years of enterprise ERP implementation experience across Workday, PeopleSoft, and Oracle environments.

References

Bughin, J., Doogan, J., and Vetvik, O. J. (2012). Delivering large-scale IT projects on time, on budget, and on value. McKinsey Digital, October 2012.

Project Management Institute (2023). Pulse of the Profession 2023: Power Skills. PMI.

Standish Group (2020). CHAOS Report 2020. The Standish Group International.

About the Author

Pradeep Billa

San Antonio, Texas
USA

Pradeep Billa is a senior enterprise ERP practitioner with 19 years of implementation experience across Workday, PeopleSoft, and Oracle HCM environments. He holds Workday Pro HCM, PMP, and PMI-ACP certifications. The Silent Handshake Framework is his original methodology for diagnosing structural failure in enterprise technology implementations. Pradeep is based in San Antonio, Texas, USA. He can be contacted at www.silenthandshake.com