

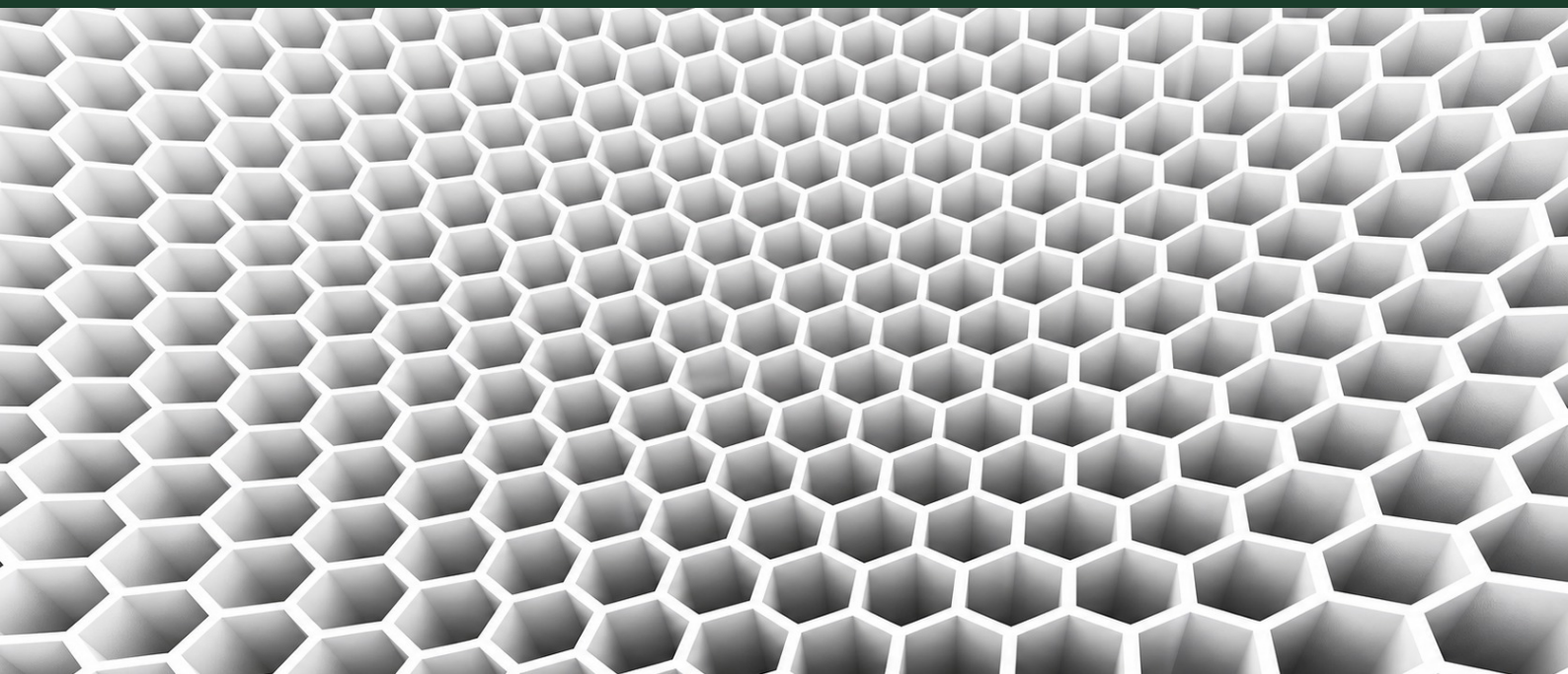
BEYOND SCALING CEREMONIES

The Programme Director in the Agile Era — Obsolete Title, Essential Function

On the Coordination Function That Enterprise Scaling Frameworks Left
Unassigned

Giovanni Leonardi

January 2018



Contents

1. Executive Summary	3
2. The Disappearing Act	3
3. What the Role Actually Does	4
4. The Gap That Opens	5
5. The Reconstruction	6
6. The Real Question	7
7. An Honest Position	8

The Programme Director in the Agile Era — Obsolete Title, Essential Function

Programme · No. 177

Originally written January 2018 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

"A framework that does not explicitly assign them is not a framework that has eliminated them — it is a framework that has left them to chance."

1. Executive Summary

The scaling frameworks that have reshaped enterprise delivery — SAFe, LeSS, the Spotify-inspired squad models — addressed a real problem: programme management had grown too heavy, too ceremonial, and too distant from the work. But in removing the programme director, many organisations inadvertently removed four load-bearing functions: cross-team integration at the level of outcomes, political absorption, cross-cutting decision authority, and outcome ownership across time horizons. The pattern that follows is remarkably consistent — a period of apparent success, a cross-cutting crisis that reveals the gap, and a reconstruction phase in which the function is rebuilt under different names with less authority. This essay argues that the debate should shift from whether the role is obsolete to a functional question: what coordination and integration functions does complex delivery require, and how should they be deliberately distributed? The title may not survive; the function will outlast every framework that fails to account for it.

2. The Disappearing Act

The reorganisation email arrives on a Tuesday afternoon, as these things tend to. The language is careful — "streamlining," "empowering teams," "removing unnecessary layers" — and the org chart attached to it is clean, geometric, full of the symmetry that consulting firms favour. Where the programme director once sat, there is now a dotted line and the words "Release Train Engineer." The teams that reported through the programme are redrawn as squads, each with a product owner. A coaching layer appears where the governance layer used to be. The email closes with the phrase "this is an exciting step in our agile journey."

What it does not close with — what no reorganisation email ever closes with — is an answer to a practical question: who, precisely, now owns the outcome that cuts across all six of those squads?

I have watched this scene play out, in different organisations and with different scaling frameworks providing the choreography, enough times to see it as a pattern rather than an incident. The programme director disappears from the structure. The function the programme director performed does not disappear. And the gap between those two facts is where a surprising amount of enterprise delivery quietly comes undone.

This is not a lament for a role under pressure. The agile critique of heavyweight programme management is substantially correct: too many layers, too much ceremony, too little trust in the people doing the work. But the scaling frameworks that followed — SAFe with its Agile Release Trains, LeSS with its radically simplified structure, the various Spotify-inspired squad models that now populate enterprise org charts from Sydney to Stockholm — have in many cases conflated a valid objection to the manner of coordination with an invalid assumption that the need for coordination has been resolved. It has not. It has been redistributed, often to people who do not know they are carrying it.

3. What the Role Actually Does

To understand what happens when the programme director is removed, we need first to be honest about what the role does — which is not what most role descriptions say it does.

The formal account emphasises planning, tracking, governance, and reporting. These are real activities, but they are not the load-bearing ones. Strip them away and a programme can survive, at least for a while. Strip away the deeper functions and it cannot. In my experience, the programme director's actual work — the work that matters, as distinct from the work that is visible — falls into four categories that scaling frameworks handle unevenly at best.

Cross-team integration at the level of outcomes. Individual teams optimise locally. This is not a defect; it is the correct behaviour of a well-functioning team with clear ownership of its domain. But programmes exist precisely because there are outcomes that no single team owns — outcomes that emerge from the interaction between multiple teams' work, and that require someone to hold the whole picture. The Release Train Engineer in SAFe is positioned to do some of this, but the role as typically practised is facilitative rather than directive. The RTE runs the PI Planning ceremony and surfaces dependencies; the programme director decides what to do when the plans of three teams are mutually incompatible and none of them can see it because each plan looks perfectly sound from the inside. That distinction — between surfacing a conflict and resolving it — is precisely the one that matters under pressure, and it is precisely the one that framework documentation tends to gloss.

Political absorption. This is the function nobody writes about in framework documentation, and it may be the most important one. Large organisations are political environments. Priorities conflict not because people are unreasonable but because the organisation genuinely wants more things than it can have simultaneously, and every prioritisation is a loss for someone with the authority to object. Senior stakeholders have competing interests. Budgets are contested territory. Strategic direction shifts in ways that are announced to the programme weeks after they were decided in corridors.

Someone needs to absorb, translate, negotiate, and — on occasion — simply shield the delivery teams from forces that would disrupt them. The programme director sits at the boundary between the organisation's political reality and the teams' need for stable, focused working conditions. I have watched a programme director spend an entire week negotiating the sequence of three regulatory deliverables, not because the sequencing was technically complex but because each deliverable was sponsored by a different divisional head who wanted theirs first. The teams never knew that conversation happened. That was the point.

Remove that boundary and one of two things follows: the politics floods downward into the teams, or the teams become invisible upward to the organisation. The first produces chaos; the second produces irrelevance. Neither is viable for long.

Cross-cutting decision rights. Scaling frameworks are remarkably clear about who owns decisions within a team and admirably vague about who owns decisions that span teams. When a shared platform team's priorities conflict with two consuming teams' roadmaps, who arbitrates? When a regulatory deadline requires three squads to change direction simultaneously, who makes the call and ensures it sticks? When an architectural decision taken by one team imposes costs on four others, who authorises the trade-off?

The standard framework answer is that the relevant stakeholders align in a ceremony — PI Planning, a Scrum of Scrums, a community of practice, a system demo. The operational reality is that alignment ceremonies surface the conflict but rarely resolve it. Resolution requires someone with the authority, the context, and the accountability to make the call and absorb the consequences. Ceremonies are excellent at generating shared awareness. They are poor at generating binding decisions in the face of genuine disagreement.

Outcome ownership across time horizons. Agile teams work in iterations, typically of two weeks. The product owner manages the backlog with a horizon of one or two programme increments. But a programme-level outcome — a regulatory remediation, a platform migration, a market entry — has a time horizon measured in quarters or years, and the path to it is not a backlog but a sequence of interdependent moves that must be orchestrated. Someone needs to hold the thread across those time horizons, adjusting the sequence as the landscape shifts but never losing the destination. This is not project management in the traditional sense; it is something closer to campaign management, and it requires a kind of sustained, cross-cutting attention that no framework role adequately describes.

4. The Gap That Opens

When I describe these functions to people who have worked inside scaling frameworks but never in a programme structure, the most common response is recognition — not recognition of the role, but recognition of the gap. "That is what we have been missing" is a phrase I have heard enough times to stop being surprised by it.

The pattern is consistent enough to sketch with some confidence. An organisation adopts a scaling framework. The programme director role is removed or renamed into something with less authority — an "agile programme manager" with no direct reports and no decision rights, or a "delivery lead" whose scope has been narrowed to a single train. For

the first two or three programme increments, things appear to work. The teams are energised by the new structure. The ceremonies are novel and feel productive. The velocity numbers look good because the teams are measuring what they individually control.

Then the first significant cross-cutting problem arrives. I watched one organisation — a mid-tier financial services firm running SAFe across four release trains — hit this wall eight months into their transformation. A regulatory change required coordinated modifications across their payments platform, their customer data layer, and their reporting infrastructure. Three trains, twelve teams, one deadline. Each product owner understood their own scope. None of them could see the whole. The dependency map that the programme director would have maintained did not exist, because dependency mapping had been classified as "waterfall thinking" during the transformation.

The product owners escalated to their respective business stakeholders, who escalated to a steering committee that met monthly and lacked the operational detail to decide anything useful. The Scrum Masters escalated to the RTE, who could facilitate a conversation but could not compel a trade-off between trains. The agile coaches observed that the teams were being disrupted and recommended more retrospectives. The regulatory deadline did not move.

What happens next, in every case I have observed, is that someone steps into the vacuum. Sometimes it is a senior product owner who quietly begins coordinating across teams, adding a responsibility that was never in the role description and nobody authorised. Sometimes it is the RTE who exceeds the facilitative boundaries of the role and starts making directional calls, discovering that servant-leadership has limits when the building is on fire. Sometimes it is a delivery manager or a senior business analyst who has been around long enough to understand the dependencies. And occasionally — particularly in financial services, where the regulatory stakes make the gap most dangerous — it is a programme director who has been rehired under a different title, six to eighteen months after the original role was removed.

The function is load-bearing. Organisations that remove it do not remove the load — they redistribute it to people who may not recognise they are carrying it, and who certainly have not been given the authority to bear it well.

5. The Reconstruction

The most interesting phase of this cycle is the reconstruction. Having removed the programme director, organisations begin to rebuild the function — but they do so in ways that reveal how deeply the original role was misunderstood.

The rebuilding rarely looks like a reinstatement. The agile transformation has generated its own political momentum, and acknowledging that a removed role was needed after all would be an uncomfortable admission. Instead, the reconstruction takes one of several forms, each of which recaptures part of the original function while quietly losing other parts.

The first and most common is the elevated product owner. A senior product owner is given cross-team scope, a seat at the leadership table, and an implicit mandate to "make it all work together." This recovers the integration and outcome-ownership functions but typically lacks the political-absorption capacity that comes from sitting outside the product hierarchy. The elevated product owner is still, fundamentally, an advocate for a product vision; the programme director was an advocate for a programme outcome, which is a different and sometimes competing allegiance. When the product vision and the programme outcome diverge — and in a complex programme, they always eventually do — the elevated product owner has no framework for navigating the conflict, because the framework insists that product ownership is the atomic unit of direction.

The second is the empowered RTE. The Release Train Engineer is given more authority — decision rights over cross-team priorities, a role in stakeholder management, a voice in budget discussions. This is closer to the original function, but it creates a tension within SAFe's own model: the RTE is supposed to be a servant-leader who facilitates alignment, and the programme director function requires someone who can, when necessary, direct. Organisations that try to have it both ways usually end up with an RTE who directs while describing everything as facilitation, which satisfies nobody and confuses the teams about where authority actually resides.

The third is the shadow programme office. A small group — sometimes called an "agile PMO," sometimes a "delivery leadership team," sometimes simply "the people who meet on Thursdays" — coalesces around the coordination vacuum. This group performs the integration and decision-rights functions, but because it has no formal authority and no place in the framework's vocabulary, it operates through influence and institutional memory. This works until it does not. The first time the shadow programme office needs to override a product owner's priorities in service of a cross-cutting deadline, its lack of formal authority becomes visible and painful.

Each of these reconstructions is a partial answer. None recovers the full function because none begins from a functional analysis of what was needed — they start from the framework's vocabulary and try to stretch an existing role to cover the gap. The result, in most cases, is a role that does the programme director's job with less authority, less clarity, and less organisational permission than the original.

6. The Real Question

The debate, as it is usually framed, asks whether the programme director is obsolete or essential. This is the wrong question, and it generates the wrong answers — either a defensive insistence that nothing has changed and the role should be preserved exactly as it was, or a dismissive assertion that scaling frameworks have solved the coordination problem and any remaining gap is simply an adoption failure.

The right question is functional, not titular: what are the coordination, integration, and decision functions that a complex, multi-team delivery requires, and how should they be distributed across roles in an agile operating model?

Once the question is posed this way, several things become clear.

The functions are real and do not disappear because a framework does not name them. Cross-team integration, political navigation, cross-cutting decision authority, and long-horizon outcome ownership are properties of the organisational problem, not of the management paradigm. They needed to be performed in waterfall structures, they need to be performed in agile structures, and they will need to be performed in whatever comes after agile. A framework that does not explicitly assign them is not a framework that has eliminated them — it is a framework that has left them to chance.

The traditional programme director role bundled these functions in ways that were partly accidental and partly a product of its era. The role accumulated responsibilities over decades, and not all of them are load-bearing. The reporting ceremonies, the status decks, the gate reviews, the RAG-rated dashboards — these are the visible, ceremonial layer that agile rightly challenged. They can be reduced or eliminated without loss. What cannot be eliminated is the core integrative function beneath them, and any honest redesign must start by identifying it clearly before distributing it.

The distribution itself is a design problem, not a framework selection. SAFe, LeSS, and the Spotify model each offer a distribution, but none of them was designed with a functional analysis of the programme director's actual work as the starting point. They were designed with a different starting point — the team — and the cross-cutting functions were either assigned to roles that cannot fully carry them or left unassigned entirely. This is not a criticism of the frameworks' intentions; it is an observation about their common blind spot.

We are fluent in method; we are far less fluent in examining what the method has quietly discarded and whether the discard was intentional or accidental.

7. An Honest Position

I write from inside this role, and I should name the bias that creates. I have a professional interest in the programme director function surviving, and that interest inevitably colours the analysis. The reader should weigh it accordingly.

But I also write with the accumulated evidence of watching the removal-and-reconstruction cycle play out enough times to trust the pattern rather than the ideology. The organisations that have handled this transition most effectively are, without exception, the ones that began with the functional question rather than the framework question. They asked: what do we actually need someone to do across these teams? They mapped the functions honestly — including the political ones that nobody puts on a slide. They distributed them deliberately, with commensurate authority attached to each. Some of those functions went to the RTE. Some went to a senior product owner. Some required a new role, designed from the ground up rather than inherited from either tradition.

What none of them did was assume that the function would take care of itself. That assumption — that coordination at scale is an emergent property of well-run teams rather than a designed capability — is the one that costs organisations the most, and it is

the one that the scaling frameworks, for all their considerable virtues, most consistently encourage.

The programme director title may not survive the agile era. Titles rarely survive intact across paradigm shifts; the function they name simply migrates and acquires new language. But the function itself — the sustained, cross-cutting, politically aware integration of work that no single team can see whole — will outlast every framework that fails to account for it. The only question is whether organisations will design for it deliberately or continue to rediscover it painfully, one failed scaling attempt at a time.