



BEYOND THE RAID LOG

The Risk That Lives Between Projects

A Practitioner's Perspective on Cross-Portfolio Risk in Complex Organisations

Giovanni Leonardi

April 2013

The Risk That Lives Between Projects

Programme · No. 172

Originally written April 2013 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

“The dependency that destroys a programme is almost never the one that was logged – it is the one that sat in the space between two programme managers who assumed the other was managing it.”

1. The Dependency Problem Nobody Owns

Ask any programme manager about dependencies and they will point you to their RAID log. It will contain entries – sometimes dozens, sometimes hundreds – each carefully categorised, assigned an owner, given a RAG status, and reviewed at regular intervals. The programme manager will tell you, with justifiable confidence, that their dependencies are being managed.

And yet programmes continue to fail because of dependencies. Not the dependencies in the RAID log – those are visible, tracked, and generally handled competently. The dependencies that cause real damage are the ones that exist between programmes, across the portfolio, in the organisational spaces where no single programme manager has line of sight and no governance forum has been designed to look.

This is the dependency problem that nobody owns, and it has persisted across every sector and every era of portfolio management I have observed. It persists not because organisations are unaware of it, but because the structural forces that sustain it are more powerful than the ad hoc mechanisms typically deployed to address it.

2. What Cross-Portfolio Dependencies Actually Look Like

It is worth being precise about what we mean. A within-programme dependency – module A cannot begin testing until module B delivers its API – is a scheduling problem. It is complex, certainly, but it is bounded: one programme manager, one governance structure, one set of trade-offs. The tools for managing it are well understood, and competent programme managers handle these dependencies routinely.

A cross-portfolio dependency is structurally different. It exists when the success of one programme depends on an output, a decision, a capability, or a resource controlled by another programme, and when neither programme has authority over the other. The

dependency may be technical: two programmes building systems that must integrate, but designed by different teams with different architects operating to different timescales. It may be organisational: a transformation programme that depends on the operating model changes being delivered by a separate organisational design initiative. It may be resource-based: two programmes drawing on the same pool of specialist skills, where one programme's acceleration inevitably delays the other.

What makes these dependencies dangerous is not their complexity — individually, each is manageable — but their invisibility. They exist in the spaces between programme boundaries, and the portfolio governance structures in most organisations are not designed to see into those spaces.

3. Why the Standard Approach Fails

The standard organisational response to cross-portfolio dependencies follows a predictable pattern. Someone — usually a portfolio office or a senior programme manager — recognises the problem and establishes a dependency management process. A cross-programme dependency register is created. Programme managers are asked to identify their external dependencies and submit them. A regular dependency review meeting is established, typically monthly, where the register is walked through and RAG statuses are updated.

This approach fails, repeatedly, for reasons that are structural rather than procedural.

3.1 The identification problem

Programme managers can only identify dependencies they can see, and their field of vision is bounded by their programme. A programme manager delivering a new customer platform knows they depend on the data migration programme for clean customer data. They may not know that the data migration programme has, in turn, a dependency on a regulatory programme that is redesigning data classification standards — standards that will invalidate the migration approach currently being built. The transitive dependency — customer platform depends on data migration depends on regulatory compliance — is invisible from any single programme's vantage point.

The standard dependency register captures direct, known dependencies. It systematically misses indirect dependencies, emergent dependencies that arise as programmes evolve, and — most critically — dependencies that neither programme manager has recognised because they would require understanding the internal architecture of someone else's programme.

3.2 The ownership problem

A within-programme dependency has a clear owner: the programme manager. A cross-portfolio dependency has, by definition, no natural owner. It sits between two programmes, each of which has its own governance, its own timeline, and its own priorities. When the dependency creates a conflict — when Programme A needs something from Programme B that Programme B cannot deliver on Programme A's timeline — neither programme manager has the authority to resolve it.

The dependency review meeting surfaces the conflict but cannot resolve it, because the people in the room are peers with competing priorities. Resolution requires escalation to someone with authority over both programmes, and in many organisations, that person sits at a level where their attention is scarce and portfolio-level dependency management is not their primary concern.

The structural truth is that cross-portfolio dependencies require cross-portfolio authority to resolve, and most organisations vest that authority at a level where it is too remote to exercise effectively on operational timescales.

3.3 The timing problem

Dependencies are most manageable when identified early and most dangerous when discovered late. The standard dependency management process — a monthly register review — operates on a cadence that is structurally misaligned with the speed at which dependencies become critical. A dependency that shifts from green to red between two monthly reviews has, by the time it is discussed, already begun to cause damage. The programme waiting for the dependency has either absorbed the delay silently, built a workaround that will create technical debt, or escalated through informal channels that bypass the dependency management process entirely.

The monthly cadence is not arbitrary; it reflects the practical limits of how much governance overhead programme managers can absorb. Increasing the frequency to weekly or daily would capture dependencies faster but would consume so much management attention that it would become counterproductive. The timing problem has no easy solution within the standard approach.

4. The Deeper Structural Forces

If the standard approach fails repeatedly, we must ask why organisations continue to use it. The answer lies in structural forces that are rarely made explicit.

4.1 Programme boundaries as organisational boundaries

Programmes are not neutral containers for work. They are organisational units with budgets, teams, governance structures, and — critically — accountable leaders. The boundaries of a programme typically mirror the boundaries of organisational authority: a programme sits within a division, a function, or a business unit, and its programme manager reports into that structure.

Cross-portfolio dependencies, by their nature, cross these organisational boundaries. Managing them effectively requires a level of organisational integration that most structures are not designed to support. The programme manager who identifies a dependency on another division's programme is, in effect, identifying a coordination problem between two parts of the organisation that have been deliberately separated for management purposes. The same organisational design principles that create clear

accountability within programmes create coordination failures between them.

4.2 Incentive misalignment

Programme managers are measured on the delivery of their programme. They are not measured on the health of the portfolio. When a cross-portfolio dependency creates a conflict, the rational response for each programme manager is to protect their own timeline and escalate the problem. There is no incentive to absorb delay on behalf of another programme, and the programme manager who voluntarily slows their delivery to accommodate a cross-portfolio dependency is, from their governance perspective, underperforming.

This incentive structure is remarkably resistant to intervention. Portfolio-level metrics and shared objectives help at the margin, but they cannot overcome the fundamental reality that programme managers are accountable to programme boards, not portfolio boards, and programme boards care about programme delivery.

4.3 The illusion of portfolio governance

Most organisations have a portfolio board or investment committee that nominally oversees cross-portfolio matters. In practice, these boards spend the majority of their time on investment decisions — approving new programmes, reviewing business cases, allocating capital — and very little time on cross-portfolio operational risks like dependencies. The portfolio board that devotes thirty minutes per quarter to dependency management, sandwiched between two hours of investment approvals, is not governing dependencies; it is acknowledging their existence.

The governance gap is not a failure of design but a reflection of attention scarcity at senior levels. Portfolio boards are composed of the organisation's most senior leaders, whose time is the organisation's scarcest resource. Asking them to engage with operational dependency management at the level of detail required to be effective is asking them to do a different job.

5. What Would Actually Work

The pattern that I have observed in the minority of organisations that manage cross-portfolio dependencies effectively is not more process but different architecture. These organisations share several characteristics.

- They establish a portfolio integration function — not a PMO that tracks status, but a small team with the technical and programme management expertise to understand the internal architecture of each programme well enough to identify dependencies that programme managers cannot see from their own vantage point.
- They invest in architectural governance that spans the portfolio, particularly for technology programmes. A chief architect or architecture review board with the authority to mandate integration standards and the capacity to review programme designs for cross-portfolio compatibility catches technical dependencies at the design stage, when they are cheapest to resolve.

- They create escalation paths that are fast and lightweight. When a cross-portfolio dependency becomes critical, the resolution mechanism is a direct conversation between a portfolio-level decision-maker and the affected programme managers, not a governance paper that works its way through committee cycles.
- They accept that some dependencies cannot be managed and must be eliminated. The most effective response to a particularly complex cross-portfolio dependency is sometimes to restructure the portfolio itself — merging two programmes, creating a shared workstream, or sequencing the portfolio so that dependent programmes do not run concurrently.

Managing cross-portfolio dependencies is ultimately not a process problem but a design problem: the question is whether the portfolio is structured so that its most critical interdependencies fall within programme boundaries rather than between them.

6. The Persistence of the Pattern

Perhaps the most striking aspect of the dependency problem is how little progress has been made against it. The challenges I have described are not new — they were visible in the first generation of large-scale transformation programmes and they remain visible today. The tools have improved, the processes have been refined, and the vocabulary has become more sophisticated, but the fundamental pattern persists: organisations design portfolios with structural interdependencies, manage those portfolios through structures that cannot see or resolve those interdependencies, and then express surprise when dependencies cause failures.

The persistence of this pattern suggests that it is not a problem waiting for a better solution but a structural feature of how organisations manage complex change. The separation of work into programmes is a necessary response to cognitive and managerial limits — no individual or governance forum can manage a portfolio of any complexity as a single integrated endeavour. But the act of separation creates boundaries, and boundaries create dependencies that the separated structures are not equipped to manage.

Accepting this tension — rather than pretending it can be eliminated through better process — is perhaps the first step toward managing it honestly. The dependency that nobody manages is not a failure of attention; it is the predictable consequence of how we organise complex work. The organisations that manage it best are those that acknowledge this structural reality and design their portfolios, their governance, and their leadership attention accordingly.