

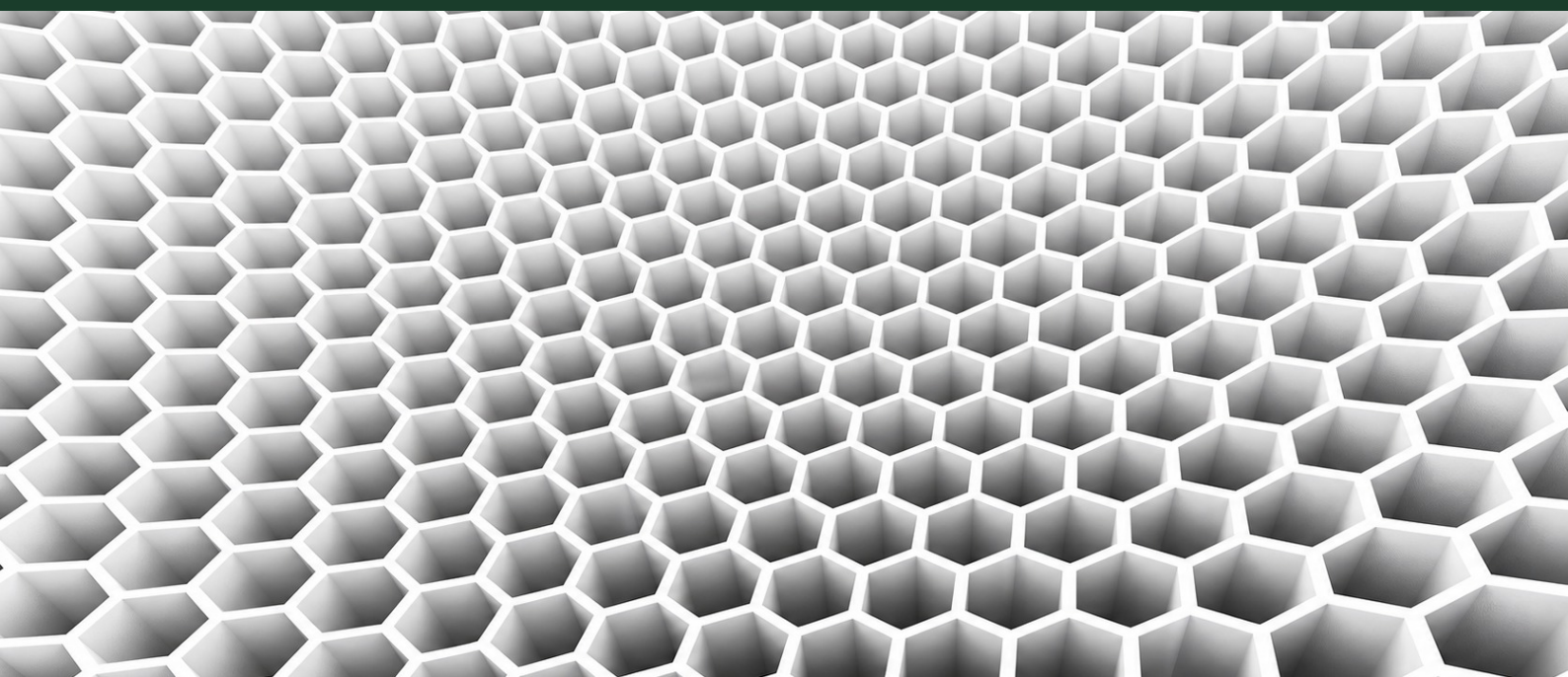
BEYOND THE REGISTER

The Dependency Is the Delivery — Why Cross-Programme Dependencies Deserve First-Class Management

The Case for Treating Cross-Programme Dependencies as Bilateral Promises

Giovanni Leonardi

September 2024



The Dependency Is the Delivery — Why Cross-Programme Dependencies Deserve First-Class Management

Programme · No. 266

Originally written September 2024 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

“We punish the messenger, so the message arrives late.”

1. The Green Status Report That Should Have Been Red

The quarterly portfolio review runs through its slides. Programme Alpha is green — milestones on track, scope controlled, team velocity stable. Programme Beta, three floors up, reports the same. And yet neither programme will deliver what the business case promised, because something that lives between them — a data migration that Alpha owns and Beta needs by Q3 — has been slipping quietly for eleven weeks. It appears in both programmes' dependency registers, described in slightly different language, owned by neither programme director with any real conviction. When the slip finally surfaces at the executive steering group, it will arrive not as a managed risk but as a surprise, and the conversation will be about blame rather than recovery.

This pattern recurs so consistently across complex delivery environments that it ought to be treated as a systemic condition, not an occasional oversight. Programmes rarely fail inside their own boundaries. They fail at the seams.

2. Delivery's Most Junior Discipline

We have sophisticated approaches to scope, risk, resourcing, benefits — the things that live inside a programme's perimeter. Dependency management, by contrast, remains the discipline we handle with the least rigour. In most organisations I have observed, it consists of a register: a spreadsheet or tool entry capturing a brief description, a target date, and a nominal owner. The register is reviewed at the programme board, sometimes weekly, more often fortnightly, and the review consists of confirming that the dates have not moved — or, when they have, noting the new date with a resigned sigh.

The problem is not that dependencies are unrecognised. It is that they are recognised and then under-managed. We record them as lines in a register rather than treating them as what they actually are: commitments between two parties, each with something at stake, each needing to trust that the other will deliver to a standard and a timeline.

The register captures the what. It rarely captures the how, the to what quality, or the verified by whom. It is a record of existence, not a mechanism of governance.

3. The Reframe: Promises Between Programmes

The shift I am arguing for is conceptual before it is procedural. It is to stop thinking of dependencies as items on a list and start thinking of them as promises between programmes — explicit, bilateral commitments that carry the same weight as any other contractual obligation in the delivery environment.

This reframe changes the questions you ask. A register entry asks: does this dependency still exist? A promise asks: is this commitment on track, at the right quality, and are both parties confident it will be honoured?

The distinction matters because it changes behaviour. When a dependency is a line on a register, it belongs to the person who typed it in. When it is a promise, it belongs to two programme directors — the one who made it and the one who is depending on it. Both have skin in the game. Both have a responsibility to manage it actively.

The strongest objection to this reframe is that it adds bureaucracy to what should be a simple coordination task. Dependencies, the argument runs, are just things programmes need from each other; they do not need the weight of formal contracting. In smaller, well-aligned delivery environments, that argument has some force. But the moment you are running three or more interdependent programmes with different sponsors, different delivery partners, and different planning horizons — which is to say, in any complex portfolio — the informal approach consistently produces the pattern described above: silent slippage, late escalation, surprise failure.

4. What a Dependency Contract Looks Like

If we accept the premise that dependencies are promises, then they deserve the basic apparatus of any promise: clarity about what is being committed, by when, to what standard, and how both parties will know the commitment has been met.

A dependency contract does not need to be a weighty document. It needs five things:

1. What is being delivered — described in terms the receiving programme can verify, not in the language of the providing programme's internal plan.
2. When it will be delivered — with a firm date and an early-warning trigger date, the point at which the providing programme must escalate if the commitment is at risk.
3. To what quality — the acceptance criteria the receiving programme will apply, agreed in advance rather than discovered on delivery.
4. Verified how — who will confirm delivery, using what evidence, and how disputes about whether the commitment has been met will be resolved.
5. Owned by whom on both sides — a named individual in each programme who is accountable for the health of this specific commitment, not a generic dependency owner drawn from the PMO.

The contract is reviewed bilaterally — the two owners, meeting regularly, reporting on the health of the commitment and escalating when it is threatened. This is distinct from the programme board's review of the register, which tends to be a one-sided, status-reporting exercise.

The cultural shift this demands should not be underestimated. It requires programme directors to accept accountability for commitments they make to other programmes with the same seriousness they bring to commitments they make to their sponsors. In many delivery cultures, a missed internal milestone is treated as a planning issue; a missed dependency is treated as someone else's problem. The contract makes it explicitly shared.

5. The Portfolio-Level Heatmap

Individual dependency contracts manage the commitment between two programmes. But the portfolio needs to see the system — to understand where the density of dependencies is highest, where the fragile commitments sit, and where a single failure could cascade.

A dependency heatmap serves this purpose. It is not a Gantt chart of activities — we have enough of those. It is a visual map of commitments across the portfolio, coloured by health:

Health	Meaning	Signal
Green	Both parties confident; no early-warning triggers activated	Bilateral review confirms on track
Amber	Early-warning trigger activated; providing programme has flagged risk	Remediation plan in place, escalation not yet required
Red	Commitment will not be met on the current trajectory	Escalated to portfolio level for decision
Black	Commitment has failed; receiving programme activating contingency	Impact assessment underway

The heatmap does two things the register cannot. First, it shows density — if seven of your ten cross-programme dependencies sit on the same critical path, you have a systemic vulnerability that no individual programme board will surface. Second, it shows direction — it reveals which programmes are net providers, making more promises than they receive, and which are net consumers. The net providers are your portfolio's load-bearing walls. When they slip, everything above them shifts.

Reviewing the heatmap at the portfolio board should be a standing agenda item, given the same weight as the portfolio risk register. In practice, most portfolio boards spend twenty minutes on risk and thirty seconds on dependencies. The heatmap makes that imbalance visible.

6. The Cultural Step

The hardest part of this is none of the above. The hardest part is changing the incentive around surfacing a threatened dependency.

In most delivery cultures, reporting that a dependency you own is at risk is heard as reporting that you have failed. The rational response, therefore, is to protect the status as long as possible — to keep it green until it is unmistakably red, at which point recovery options have narrowed. We punish the messenger, so the message arrives late.

Treating dependencies as first-class means treating the early surfacing of a threatened commitment as good delivery behaviour — as a sign of mature programme management rather than a confession of weakness. This is a leadership behaviour, not a process change. It requires portfolio directors to model it, programme directors to practise it, and governance forums to reward it.

The programme that flags an amber dependency eight weeks before the due date, with a remediation plan and a clear ask, is doing exactly what we need. The programme that reports green until three weeks out and then escalates a crisis is not — regardless of whether the crisis was ultimately resolved.

7. The Integrated Plan, Reconsidered

We talk a great deal in this profession about the integrated plan. Too often, what we mean by that is a consolidated timeline — a master Gantt chart that stitches together the individual programme plans into a single view of activities and milestones.

That is not integration. Integration is understanding the commitments that flow between programmes — what each programme has promised to the others, and what each programme needs from the others to deliver its own outcomes.

The integrated plan is not a chart of activities. It is a map of promises.

When we manage those promises with the rigour they deserve — contracted explicitly, owned bilaterally, tracked at portfolio level, and culturally supported — we address the place where complex programmes actually fail. Not inside their boundaries, where we have decades of practice and mature disciplines. At the seams, where we have a register and a hope.