

BEYOND THE DEPENDENCY LOG

Dependencies Across the Portfolio — The Risk Nobody Owns

A Practitioner's Perspective on Cross-Programme Dependency Risk in Large Change Portfolios



Dependencies Across the Portfolio – The Risk Nobody Owns

Portfolio · No. 9

Originally written May 2006 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

“Green plus green does not equal green when both are standing on the same bridge.”

The Dependency That Nobody Owns

Ask a programme manager to show you their dependencies and they will produce a log. It will list the things their programme is waiting on and the things other people are waiting on from them, each with a date, an owner, and a status. It is a competent artefact. It is also, in my experience, almost never the place where the real danger lives.

The dependency that brings a portfolio down is rarely the one inside a single programme. Within a programme, dependencies are visible because someone is paid to see them. The plan holds them, the programme board reviews them, the manager loses sleep over them. The dependencies that go unmanaged are the ones that run between programmes – the shared platform two initiatives both assume will be ready, the specialist team quietly committed to four places at once, the sequencing assumption that only holds if everything else lands on time. These live in the white space between programmes, and the defining feature of white space is that nobody owns it.

A dependency inside a programme has an owner by construction. A dependency across the portfolio has an owner only if someone deliberately assigns one – and most organisations never do.

Why Local Discipline Produces Global Blindness

Here is the paradox I have watched play out across sector after sector: the better each programme is run, the more exposed the portfolio can become. A well-run programme optimises for its own delivery. Its manager negotiates hard for the resources it needs, protects its critical path, and books the shared architects and the integration environment early. Multiply that by a dozen capable managers, all doing exactly what we ask of them, and you arrive at a portfolio in which every programme has locally sensible commitments that are collectively impossible.

The mechanism is not incompetence. It is the absence of a vantage point. Each programme sees its own dependency log clearly and everyone else's not at all. The programme board reviews delivery, not the seams between deliveries. And the one function that supposedly sits above it all – the portfolio office, or whatever the organisation has christened it – is usually consumed with something else entirely.

The Reporting Reflex

Most portfolio offices spend their energy on aggregation. They collect status from every programme, roll it up into a single report, apply a traffic-light, and present it upward. This is useful for assurance and nearly useless for dependency management, because aggregation destroys exactly the information you need. Two programmes can both report green while sitting on a collision that neither has been asked to see. Green plus green does not equal green when both are standing on the same bridge.

Dependency risk is not additive; it is relational. You cannot find it by summing the parts. You find it only by looking at the connections between the parts — and connection analysis is a different discipline from status collection.

- Status reporting asks: is each programme on track against its own plan?
- Dependency analysis asks: is each programme's plan compatible with everyone else's?
- The first is a vertical question. The second is horizontal — and horizontal questions have no natural owner.

Three Shapes the Danger Takes

Across the portfolios I have seen come under strain, the unmanaged cross-programme dependency tends to take one of three shapes.

Form	What it looks like	Why it hides
Shared resource	One scarce team — architects, testers, a data group — committed across several programmes beyond its capacity	Each programme books it in good faith; no single plan shows the total load
Sequence assumption	One programme's business case quietly depends on another delivering first	The slippage is one programme's problem on its own report; the other's exposure is invisible until late
Shared foundation	Two or more initiatives assume the same new platform or capability will exist	Nobody owns the platform as a dependency; it is everyone's assumption and no one's commitment

The common thread is that each of these is perfectly visible from above and nearly invisible from within. The programme manager cannot see the shared resource's total load, because they see only their own booking. They cannot see the sequence risk, because the programme they depend on reports separately. The only place these become legible is the portfolio level — which is precisely the level that is usually busy summing traffic-lights.

What Actually Changes It

I want to be careful not to prescribe a new bureaucracy. The instinct, when a risk is discovered to be unowned, is to build machinery — a dependency board, a tool, a weekly

return. More process is rarely the answer, and a cross-portfolio dependency log that nobody has the authority to act on is just another artefact to keep warm.

Three things, in my experience, make the difference, and only the first is non-negotiable.

1. Give the cross-programme dependency an owner. A dependency without a named individual accountable for it is a wish. This does not mean one person delivers both sides; it means one named person is accountable for seeing the seam and raising it when it strains. Ownership is the whole game — everything else is technique.
2. Make dependencies a first-class object, not a column. Track them as things in their own right, with two programmes attached, a single owner, and a status that reflects the relationship rather than either programme's health. The moment a dependency's status is derived from the connection rather than from one side of it, collisions become visible early.
3. Change the cadence to look sideways. The portfolio review that only proceeds programme by programme will only ever surface vertical news. Once in the cycle, ask a different question: where are two programmes standing on the same bridge, and who is watching the bridge?

None of this requires a new methodology. The guidance we already have tells us to manage dependencies; it simply assumes someone is looking at them across the boundary, and in most organisations no one is. The gap is not in the guidance. It is in the ownership.

The Uncomfortable Conclusion

Cross-portfolio dependencies are the risk nobody manages not because they are hard to understand, but because managing them requires someone to hold a view that cuts across the programmes we have so carefully separated — and our structures, our boards, our reports, our incentives, are built almost entirely to look up and down rather than across.

The organisations that handle this well are not the ones with the most elaborate portfolio tooling. They are the ones where a small number of senior people habitually ask the horizontal question, hold the seams personally, and treat a dependency between two programmes as a real thing with a real owner rather than a line item that belongs to neither. That is a matter of attention and accountability far more than method. And attention, unlike method, cannot be bought in from a framework.