



BEYOND THE GATEWAY

The API Economy Didn't Remove Your Integration Debt — It Hid It

A Practitioner's Perspective on Integration Complexity in Large-Scale Change Programmes

Giovanni Leonardi

September 2016

The API Economy Didn't Remove Your Integration Debt — It Hid It

Programme · No. 190

Originally written September 2016 | Re-edited and re-rendered for the WhereBeyond Edition, 2025

"We have swapped a technical problem we knew how to staff for an organisational problem most programmes are not structured to hold."

The Promise That Sold Itself

Every few years our profession falls in love with an idea that appears to dissolve a stubborn problem. This year that idea is the API. Expose everything as a service, the argument runs, and integration stops being a programme's private tax on delivery and becomes something closer to a marketplace. Systems connect through clean contracts. Teams consume one another's capabilities without raising a project. The tangle of point-to-point interfaces that defined the last decade quietly disappears.

I have sat through enough steering committees to recognise the appeal of that story. It is not wrong, exactly. But the pattern I have observed across integration-heavy programmes is less comfortable than the slide deck: the API economy does not remove integration complexity. It moves it — out of the middleware layer where we could at least see it, and into the spaces between teams, where we cannot.

Integration Debt Does Not Disappear — It Distributes

A decade ago, integration complexity had an address. It lived in the enterprise service bus, in the mapping documents, in the small team of specialists who understood how the billing system spoke to the customer database. It was ugly and it was centralised, and centralisation had one great virtue: you knew where to look when it broke.

The API-led model breaks that concentration deliberately, and for good reasons. But it also breaks the visibility. When every capability is a service and every service has an interface, the number of integration points does not fall — it multiplies. What changes is that no single team owns the whole picture any more. Each team sees its own producers and consumers and assumes someone, somewhere, holds the map. Usually no one does.

I have come to call this distributed integration debt. It is the same debt we always carried, fragmented into pieces small enough that no individual piece looks like a problem. A brittle point-to-point coupling used to be a visible architectural sin.

Dressed as a published API with a portal in front of it, the same coupling now reads as progress.

The point-to-point interface never went away. We gave it a gateway, a developer portal and a versioning policy, and stopped counting it.

Why Programmes Keep Missing It

The reason this pattern persists is not technical naivety. The engineers usually understand the risk perfectly well. The failure is one of framing at the programme level.

Most API initiatives I have observed are chartered as delivery efforts. The objective is expressed in outputs: expose forty services by year end, stand up the gateway, publish the developer portal. These are measurable, and being measurable, they are what gets reported. What does not get reported is the thing that actually determines whether the API economy pays off — the ongoing cost of keeping all those contracts honest as the estate changes underneath them.

- A programme counts the APIs it has published, not the ones that have quietly drifted out of use.
- It tracks gateway uptime, not whether two teams have agreed what a "customer" record actually means.
- It celebrates consumption metrics, not the growing number of consumers now coupled to a version that the producing team is desperate to retire.

The result is a programme that reports green while the integration estate silently accrues exactly the kind of debt the whole exercise was meant to eliminate.

The Governance the Textbooks Skip

The standard literature on this subject treats API management as a product-selection problem. Choose a gateway, define a design standard, publish a style guide, govern by review board. All of that is necessary. None of it is sufficient, because it governs the creation of APIs and says almost nothing about their life.

The questions that decide whether an API estate stays healthy are not design questions. They are lifecycle and ownership questions:

1. Who owns this interface as a product — not who built it, but who is accountable for it a year from now, when the person who wrote it has moved on?
2. What is the contract, and what is the deprecation commitment — how much notice does a consumer get before a breaking change, and who enforces it?
3. How do we know what depends on what — is there a living record of producers and consumers, or does that knowledge live only in people's heads?

4. What does a version cost — every version kept alive for a reluctant consumer is a standing liability; who is watching that ledger?

An organisation that cannot answer these has not built an API economy. It has built a more fashionable integration layer, and handed itself a harder maintenance problem than the one it started with — because the old service bus at least had a team who knew it was there.

The Contrast Worth Holding Onto

Dimension	The service-bus era we are leaving	The API era we are entering
Where complexity lives	Centralised in middleware	Distributed across teams
Who can see it	A specialist team	Nobody, by default
Failure mode	Visible, blocking, addressed	Invisible, cumulative, deferred
The real discipline	Technical integration	Ownership and lifecycle

The column on the right is not worse than the column on the left. In most respects it is a genuine improvement, and I would not counsel any organisation to turn back. But the discipline it demands is different in kind, and that is the part the enthusiasm tends to skip. We have swapped a technical problem we knew how to staff for an organisational problem most programmes are not structured to hold.

What I Would Do Differently

If I were standing up such a programme now, I would change three things before writing a line of interface specification.

First, I would refuse to treat APIs as deliverables and insist on treating them as products with owners, budgets and end-of-life plans. A service with no named owner and no deprecation policy would not be allowed to publish.

Second, I would fund the map. A living, trusted record of who produces and who consumes each interface is not documentation overhead — it is the single artefact that turns distributed debt back into something visible. It costs real money to keep current, and it is the first thing programmes cut. That instinct is exactly backwards.

Third, I would change what the programme reports. Not the count of APIs shipped, but the health of the estate: how many interfaces carry no owner, how many versions are being kept alive past their time, how many consumers are coupled to something scheduled to die. Report the debt, and the debt gets managed. Report only the deliveries, and you will get a great many deliveries and a slow, quiet accumulation of everything they leave behind.

The API economy is real, and the organisations that master it will move faster than those that do not. But it rewards a discipline that has very little to do with technology and a great deal to do with ownership. The interface is the easy part. It always was.