

BEYOND THE BOLT-ON

Governance by Design, Today: From BCBS 239 to DORA, Cloud and AI

A Practitioner's Perspective on Embedding Data Governance in Delivery in
the Data-Product Era

Giovanni Leonardi

June 2026



Contents

1. Executive Summary	3
2. The Principle That Held	4
2.1 How the thinking has matured	4
3. What Has Changed Beneath the Principle	4
4. Automated Lineage: Governance as Code	5
5. Data Products and Distributed Ownership	6
6. The Third Party Is Now Inside the System	6
7. When the Data Trains a Model	7
8. Designing It In, Still	8
9. Where Judgement Still Beats Automation	8

Governance by Design, Today: From BCBS 239 to DORA, Cloud and AI

Transformation · No. 373
June 2026

“Automating the evidence of governance is not the same as governing; a pipeline can prove a rule was followed without anyone having decided the rule was right.”

1. Executive Summary

More than a decade ago, as BCBS 239 was reshaping expectations on risk-data aggregation, the argument that mattered was simple and, at the time, mildly contrarian: governance must be designed into delivery, not bolted on afterwards. Ownership, quality, lineage, access, retention and accountability belonged in the programme plan and the acceptance criteria from the outset, not in a policy binder written once the platform was already built. That argument has aged well in its principle and poorly in its assumptions. The principle — govern by design — is more true than ever. The world it was written for has almost entirely changed.

This essay revisits that case for the present. The regulatory perimeter has widened from risk-data aggregation to operational resilience under DORA, to data protection under GDPR, and now to the governance of artificial intelligence and the data that trains it. The architecture has changed even more profoundly: from centralised warehouses to distributed data products, from manual lineage documents to lineage generated as code, from data centres a bank controlled to cloud platforms it rents and third parties it depends on but does not operate. Each shift makes design-first governance harder to avoid and easier to automate — and each creates a fresh temptation to believe the automation is the governance.

The central claim is unchanged and, if anything, sharper: governance embedded in delivery is the only kind that survives at the speed and scale we now work at. But embedding it now means embedding it into pipelines, product definitions, contracts and model documentation — not into a document nobody reads. This is a delivery leader's account of what that looks like today, and of the places where human judgement still has to do what automation cannot.

The principle held. The implementation had to be rebuilt from the ground up. Confusing the two — keeping the old mechanics because the idea still sounds right — is the most common governance failure I see now.

2. The Principle That Held

The original argument rested on a single observation: governance added late is governance that fails. When ownership, quality and lineage are treated as things to be documented after the platform is built, they are shaped by the architecture rather than shaping it. The data model has already been decided; the pipelines already run; the definitions are already contested. Retrofitting governance onto that is expensive, partial and resented, and it produces controls that describe the system rather than constrain it.

Designing governance in reversed the order. Ownership was assigned before the data was modelled. Quality expectations were written into acceptance criteria, so a build that could not meet them was not done. Lineage was a design requirement, not an archaeological exercise conducted later by people reconstructing what the pipelines must have done. The claim was never that this made governance cheaper in the moment – it made the early phases harder – but that it made governance real, and that real governance was cheaper across the life of the asset than the theatre of governance bolted on at the end.

That much has held completely. I have not seen a single case in the intervening years where late-stage governance outperformed designed-in governance. What has changed is not whether to design governance in, but what you are designing it into – and there the ground has moved so far that keeping the old techniques while repeating the old principle has itself become a way to fail.

2.1 How the thinking has matured

It is worth being honest about what the BCBS 239 era got wrong, or at least too narrowly right. Governance then was conceived largely as control: a set of constraints applied to a largely centralised, largely internal, largely batch-processed estate. It assumed the bank owned the infrastructure, that data moved in relatively few well-understood flows, and that a capable team could hold the whole lineage in a set of documents. Each of those assumptions has quietly dissolved. The maturation of the thinking is the recognition that governance is no longer primarily an act of control over a system you own end to end. It is an act of design across a system you assemble from parts you do not fully control, running faster than any human can inspect in real time.

3. What Has Changed Beneath the Principle

Four shifts, in particular, have rewritten what design-first governance has to mean.

Dimension	The BCBS 239 Assumption (c. 2013)	Today's Reality
Architecture	Centralised warehouse, few owners	Distributed data products, many owners

Dimension	The BCBS 239 Assumption (c. 2013)	Today's Reality
Lineage	Documented by people, after the fact	Generated as code, continuously
Infrastructure	Owned and operated by the bank	Rented from cloud providers and third parties
Data's purpose	Feeds reports and decisions	Also trains models that make decisions
Regulatory frame	Risk-data aggregation	Aggregation, plus resilience, privacy and AI

None of these shifts weakens the design-first principle. Every one of them raises the cost of ignoring it, because each makes late-stage, manual, control-based governance not merely inefficient but impossible. You cannot manually document the lineage of a system that rewires itself with every deployment. You cannot bolt ownership onto a hundred data products after the fact. You cannot inspect, by hand and after the event, decisions a model is making thousands of times an hour. The only governance that works in this world is the governance built into how the thing is made. The rest of this essay takes the four shifts in turn.

4. Automated Lineage: Governance as Code

The most visible change is that lineage has stopped being a document and become an output. In a modern data estate, where the data came from, what was done to it, and where it went can be captured automatically from the pipelines themselves, versioned alongside the code, and refreshed continuously. This is a genuine advance, and it vindicates the design-first principle more directly than anything else: lineage designed into the pipeline is complete and current, where lineage documented afterwards was always partial and stale.

But automation introduces its own failure mode, and it is subtle. When lineage is generated automatically, it becomes tempting to treat its existence as proof of governance. It is not. Automated lineage tells you, faithfully, what the system did. It does not tell you whether what the system did was right, whether the transformations preserve meaning, whether the owner agrees the data is fit for the use it is being put to. The map is generated; the judgement about whether the territory is acceptable still has to be made by someone.

“Automated lineage answers ‘what happened to this data’. It cannot answer ‘should it have’, and the second question is the one governance exists to ask.”

So governance-as-code is a tool of design-first governance, not a substitute for it. The discipline now is to build the automated lineage in — non-negotiable — and then to wire human decision points on top of it: an owner who signs off that a data product is fit for a stated purpose, a control that fails a deployment if lineage cannot be produced, a review that treats a break in lineage as a break in the build. The machinery is designed in. The accountability is designed in around it. Neither alone is governance.

5. Data Products and Distributed Ownership

The second shift is architectural and, for governance, the most demanding. The centralised warehouse has given way to distributed data products — discrete, owned, independently published datasets, each with its own team, its own release cadence and its own consumers. This solves the central bottleneck that made large warehouses slow and brittle. It also multiplies the governance surface by an order of magnitude, because ownership is no longer a role held in one place but a property that has to hold true across many.

Designed-in governance adapts to this well, but only if the design is explicit about a few things:

- Ownership is a product property, not an afterthought. A data product without a named, accountable owner is not a product; it is an orphan with a schema. Ownership has to be a precondition of publication, enforced by the platform, not a box ticked in a spreadsheet somewhere else.
- Interfaces carry the governance. In a distributed estate, quality and meaning are governed at the boundaries between products — the published contract, the agreed definition, the guaranteed freshness. The governance moves from the centre of a monolith to the seams between components.
- Federated does not mean absent. Distributing ownership does not distribute away the need for common standards. Someone still has to define what 'owned', 'fit', and 'published' mean across the estate, or every product will govern itself to a different standard and the whole becomes ungovernable in aggregate.

The failure I see most often here is a bank that adopts the distributed architecture for its speed and quietly drops the governance that distribution actually requires, on the assumption that autonomous teams will each do the right thing. They will each do a different thing. Autonomy without designed-in standards is not federation; it is fragmentation with better marketing.

6. The Third Party Is Now Inside the System

The third shift is that the bank no longer runs the ground its data stands on. Cloud platforms, managed services and specialist third parties are now load-bearing parts of the estate, and this is where the regulatory frame has moved most decisively. DORA reframed the question from 'is our data correct' to 'can we keep operating', and put third-party and concentration risk at the centre of it. When a handful of cloud and service providers underpin much of the industry, the resilience of any one bank is entangled with

infrastructure it depends on but does not control.

For design-first governance, this widens the scope of what 'designed in' has to cover. Governance can no longer stop at the boundary of the systems the bank operates; it has to reach into the contracts, the exit plans, the concentration analysis and the resilience testing of the third parties inside the critical path. Concretely:

1. The contract is a governance artefact. Data location, access, portability, deletion, audit rights and exit terms are governance requirements, and they have to be designed into the agreement before the dependency is created, because they are almost impossible to negotiate afterwards.
2. Concentration is a first-class risk. Where multiple critical capabilities rest on the same provider, that concentration is itself a condition to be governed — named, measured, and given an answer — not an incidental fact discovered during an incident.
3. Resilience is tested, not asserted. The ability to withstand or recover from the failure of a third party has to be demonstrated, which means designing the testability of that failure into the architecture from the start.

The design-first principle holds exactly as before; it simply now extends across an organisational boundary it did not need to cross in 2013. Governance that stops at the bank's own perimeter governs a shrinking fraction of the system that actually matters.

7. When the Data Trains a Model

The fourth shift is the newest and the least settled. Data is no longer only something that feeds reports and human decisions; increasingly it trains models that make or shape decisions directly. This changes the governance question in kind, not just degree. It is no longer sufficient to govern the data at rest and in flight. The data now has a downstream life inside a model, and the model's behaviour inherits the properties — and the flaws — of the data it learned from.

Designed-in governance extends into this territory along lines that will feel familiar, which is the point: the principle transfers even where the object is new.

- Training data needs lineage too. What a model was trained on, whether that data was permitted for the purpose, and whether its provenance can be reconstructed are governance questions identical in spirit to the ones BCBS 239 asked of risk data — now asked of the corpus behind a model.
- Purpose and permission travel with the data. Data gathered under one basis, or one privacy consent, cannot be silently repurposed to train a model for something else. Under GDPR this is not a nicety; it is the difference between lawful and unlawful processing, and it has to be designed into the pipeline that assembles training data, not checked afterwards.
- Model behaviour is a data-quality outcome. A model that produces biased, unstable or unexplainable outputs is very often expressing a property of its training data. Governing model behaviour therefore begins upstream, in the governance of the data — which is precisely where a delivery-led, design-first discipline is strongest.

The emerging regulatory expectations around AI push in the same direction as everything above: document the data, establish accountability, make the system explainable and auditable by design. A bank that has genuinely embedded governance in its data delivery is far better placed to meet them than one now scrambling to reconstruct where its training data came from. The design-first bank has the lineage already; the bolt-on bank is doing archaeology under a deadline.

8. Designing It In, Still

Draw the four shifts together and the practical shape of governance-by-design today becomes clear. It lives in the same places it always should have — the plan, the acceptance criteria, the definition of done — but those places now describe pipelines, products, contracts and models rather than a warehouse and a set of reports.

- Acceptance criteria that fail a build if lineage cannot be produced or an owner is not named.
- Platform controls that make ownership and published contracts a precondition of a data product going live.
- Third-party agreements written as governance artefacts before the dependency is committed.
- Training pipelines that record provenance and enforce permitted purpose as they assemble data.
- And, above all, human decision points designed in on top of all this automation — the owner who signs, the reviewer who challenges, the accountable name against each product and model.

Governance you can see in the pipeline, the contract and the model documentation is real. Governance you can only see in a policy document is a description of governance, not the thing itself.

9. Where Judgement Still Beats Automation

It would be easy to read all of this as a story of governance becoming automated, and to conclude that the delivery leader's job is to install the right machinery and step back. That is the one conclusion I want to resist. Everything above automates the evidence and the enforcement of governance — the lineage, the controls, the gates. None of it automates the judgement at the centre.

Automating the evidence of governance is not the same as governing; a pipeline can prove a rule was followed without anyone having decided the rule was right. Someone still has to decide whether a data product is genuinely fit for a purpose, whether a concentration risk is acceptable or intolerable, whether training data assembled lawfully is nonetheless being used in a way the bank should stand behind. These are not questions a control can answer, because they are questions about what the rules should be, not whether they were obeyed. The machinery can tell you the system is behaving as designed. It cannot

tell you the design was wise.

This is why I have argued throughout as a delivery leader rather than a governance specialist. The specialist's instinct is to perfect the control framework. The delivery leader's instinct is to ask whether the thing we are building will actually be trustworthy in use — and that question is answered in the design, by people who own the outcome, not in a framework maintained beside it. The two need each other. But when governance is designed into delivery, the delivery leader is the one holding the pen at the moment the decisions are actually made, which is the only moment governance can truly be designed in at all.

Thirteen years on, the sentence I would still defend without changing a word is the one the original essay was built around: govern by design, or do not really govern at all. Everything else — the architecture, the regulation, the automation, the presence of models where there were once only reports — has changed almost beyond recognition. The principle has not needed to.