

Data & BI platform migration

Move years of database and BI logic into your new platform, already compiled.

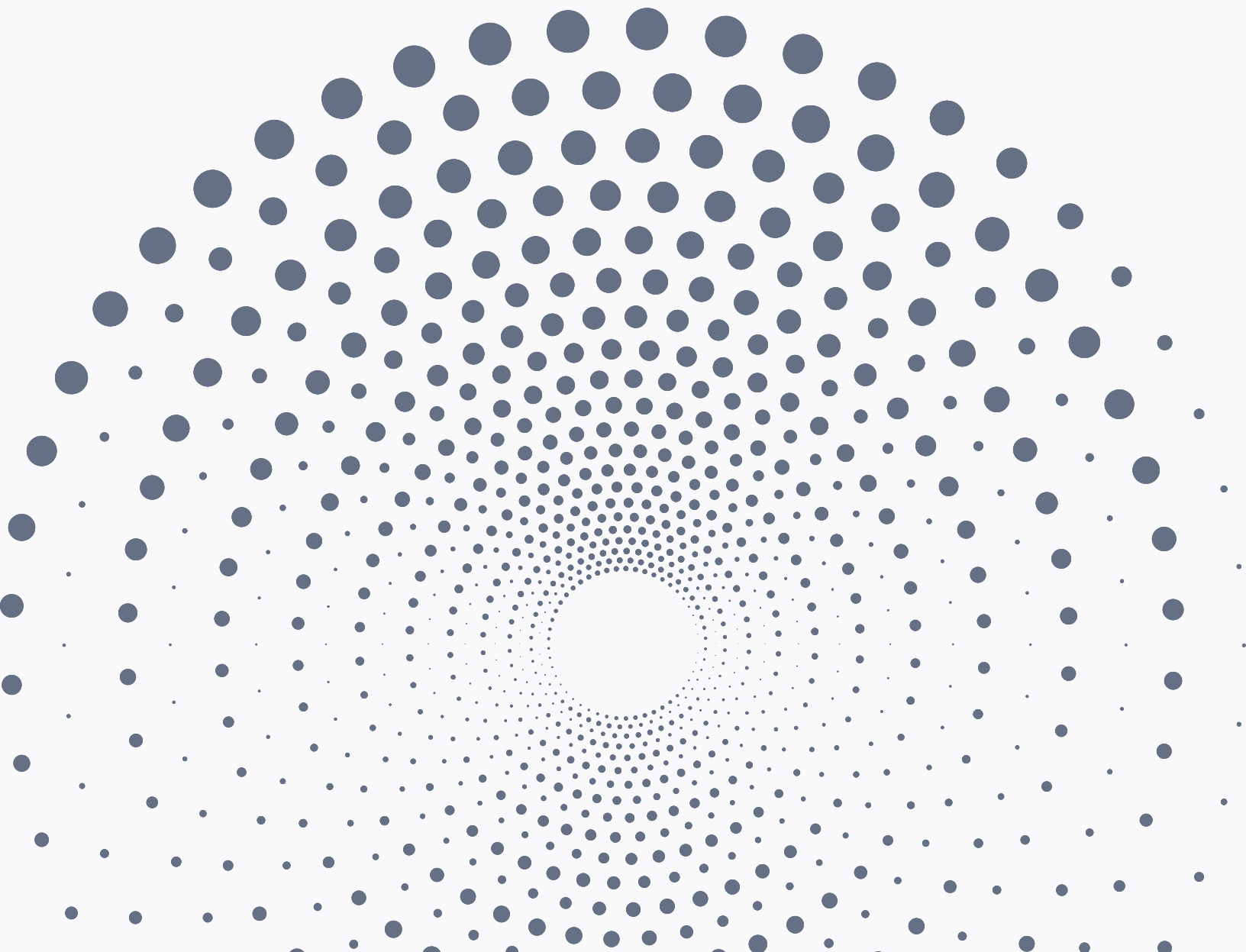


Table of Contents

The dependency problem	03
The semantic debt problem	03
How MetaKarta handles both	04
The proof to ask for	04
The metric that moves	05
Where this leaves the estate	05

Migrations fail in two ways, and neither one surfaces until it's expensive to fix.

The first is a production incident months after go-live. A column was renamed. A table restructured. The dependency never appeared in the impact analysis, because that analysis was built on log-based lineage and the downstream consumer hadn't run during the observation window. Now a dashboard is wrong, an executive wants answers, and the root cause traces back to a migration decision made before anyone mapped what depended on what.

The second is slower and costs more. Years of semantic model investment sit in the old platform: Power BI reports built on custom DAX, Tableau workbooks with layered calculated fields, LookML definitions encoding business logic that exists nowhere else in writing. The new platform reads none of it. The team rebuilds from incomplete documentation, and the people who wrote the original definitions have moved on.

Both failure modes are avoidable, though most migration toolchains handle neither.

The dependency problem

Log-based lineage records what ran. Consumers that haven't executed recently stay invisible: jobs on irregular schedules, dormant tables, reports that run quarterly.

Column-level impact analysis built on logs inherits the same limit. It captures direct dependencies where execution records exist, and leaves out the structural relationships encoded in the code itself. A transformation that hasn't run since last quarter still depends on the column being renamed, and the gaps that leaves in the plan show up after go-live.

The semantic debt problem

Every BI platform stores metric logic in its own dialect. Power BI keeps it in DAX measures inside the semantic model, Tableau keeps it in calculated fields in workbooks and published data sources, and Looker defines it in LookML. None of the three reads the others.

So when an organization moves from Tableau to Power BI, or from Looker to Fabric, the semantic models get rebuilt. Business logic that took years to accumulate, encoding decisions made by people who have since left, gets reconstructed from institutional memory and spreadsheets. The reconstruction is always slightly off, and the errors surface in production when someone finally runs the right comparison.

The expensive part of semantic debt is the divergence between what the new platform computes and what the old one produced. The organization finishes the migration with less confidence in its numbers than it had going in.

How MetaKarta handles both

MetaKarta's parser-based lineage engine reads the code artifacts that define the estate: SQL, ETL definitions, stored procedures, dbt models, and legacy sources across 400+ connectors. Lineage is computed from what's defined, so every downstream consumer of any column appears in the impact analysis regardless of when that consumer last executed. Lineage by design, not by observation.

Before anything moves, the dependency map is complete. Every object that references the column, table, or schema being changed appears in it, traced to the specific line of code that creates the dependency. The analysis lands in the planning phase, when the plan can still change.

For semantic debt, Semantic Hub's reverse-engineering engine reads existing Power BI and Tableau semantic models, along with legacy models from MicroStrategy, SAP BusinessObjects, IBM Cognos Framework Manager, and Oracle OBIEE, and imports them as governed starting points in the shared metadata repository. Calculation logic, dimension relationships, and business rules all come in, and all of it is governable in the repository. Years of accumulated investment become the foundation the migration is built on.

Semantic Hub then compiles those governed definitions into the native format of the target platform. The organization arrives at the new platform with the semantic fidelity it had in the old one, plus a governed repository that holds consistency in place going forward.

The proof to ask for

The demo runs both halves of the problem in a single session:

- ✓ Point the parser at a schema change and produce the column-level impact analysis, including consumers that haven't run recently.
- ✓ Reverse-engineer an existing Power BI or Tableau semantic model into a governed definition in the shared metadata repository.
- ✓ Compile that definition into the target platform's native format.
- ✓ Open the compiled artifact in the target tool and compare it against the original definition.

The compiled artifact is the proof.

The metric that moves

Two numbers, both available before the migration starts.



Dependency coverage.

The count of downstream consumers in a parser-based impact analysis, against the count the current log-based tool returns for the same change. The gap between them is the production-incident risk being carried into go-live.



Rebuild scope.

How many existing semantic models import cleanly as governed definitions. Each one is rework that comes off the migration plan, and its business logic arrives intact instead of reconstructed.

Where this leaves the estate

Both failure modes get handled in the planning phase, while they're still schedule items. Dependencies are mapped from the code before the first schema change ships. Semantic models arrive as governed definitions and compile into the new platform's native format.

The destination platform launches with governed definitions, column-level lineage back to source, and a shared metadata repository that keeps the estate coherent as the organization builds on top of it. When the next platform decision arrives, that migration starts from the repository and the definitions carry forward again.

Request a demo to see the impact analysis surface dependencies a log-based tool misses, and an existing semantic model imported, governed, and compiled into a new platform's native format.

[Request Demo](#)

About Meta Integration Technology

Meta Integration Technology, Inc. (MITI) is a pioneer in metadata integration, semantic modeling, and data lineage with decades of experience serving global enterprises and technology providers. MITI's technology is embedded in products from leading data and analytics vendors and powers MetaKarta, its enterprise platform for metadata-driven data management. MetaKarta delivers data lineage, data cataloging, data governance, and semantic enforcement through a single integrated system, connecting the full enterprise data estate across 400+ native connectors, from legacy to modern and on-premises to the cloud.

650 Castro Street, Suite 120-431, Mountain View, California 94041-2055 USA
Phone: 650.429.8853

info@metakarta.com
www.metakarta.com