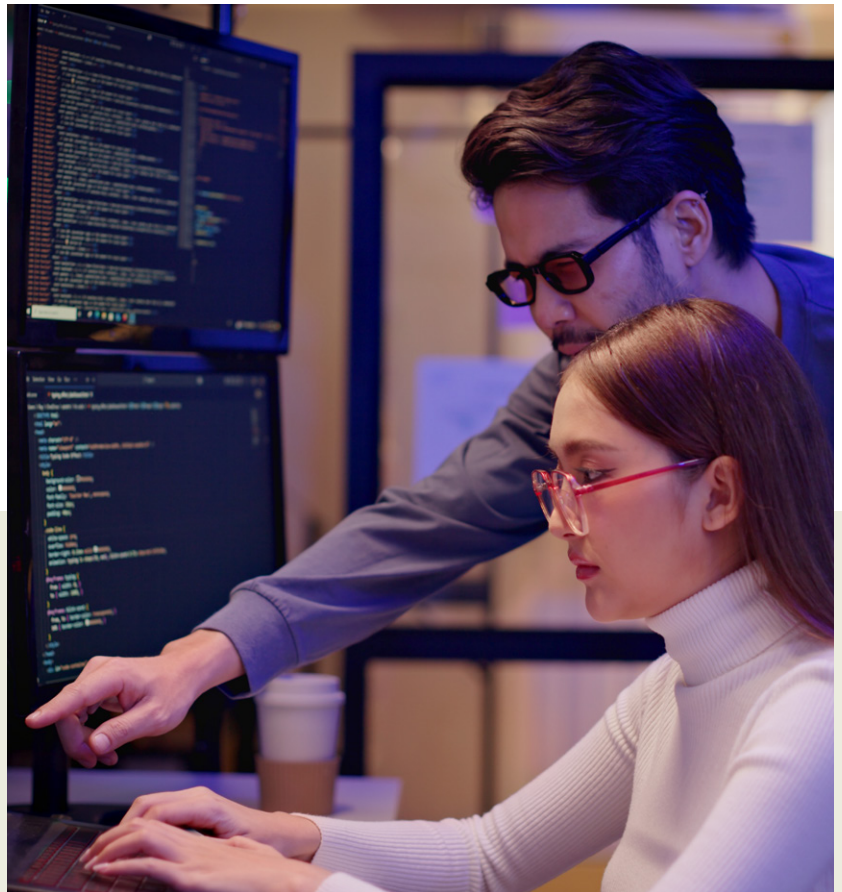


U S
T .

Analytics after generative AI



WHITEPAPER

The shift in effort
and the case for an
ontology layer

ust.com

By Heather Dawe

Chief Data Scientist and
Head of Responsible AI, UST

Contents

1. Executive summary	2
2. The change in enterprise ML and analytics	3
3. Where the risk goes	4
4. What actually reduced	5
5. Specification and verification in practice	7
6. When the business disagrees with itself	8
7. Ontology as a medium, not a catalogue	9
8. How work passes through	11
9. What controlled has to mean	13
10. Operating at scale	14
12. Data readiness	16
13. Where UST is	17
14. A first step	17

Executive summary



In UST delivery experience, generative AI has cut the elapsed build time for many analytics and ML projects by more than half. It has not, however, eliminated risk, and the use of generative AI has changed where much of this risk resides.

The time taken to produce code has collapsed. This has led to the increased importance of the detail in the specification and verification processes, as a model can produce syntactically correct code around a semantically wrong assumption without knowing it, as well as providing plausible responses to user questions that are completely wrong.

UST Chief Data Scientist Heather Dawe argues that scaling AI-assisted analytics safely requires an ontology as a live medium, not a reference catalogue, spanning data, process, and causal definitions, enforced continuously through integrated MLOps processes rather than version controlled in isolation. Her recommended first step: instrument one contested business concept end to end before committing to a full solution.

The change in enterprise ML and analytics

Implementation is no longer the constraint

Analytics and ML development has changed shape over the past two years. The part of the work that used to dominate the schedule now takes very little time. On UST delivery work, builds that would previously have occupied a conventional team for around three months are landing in four to seven weeks.

The reduction in elapsed time is the obvious change. The redistribution of effort underneath it is the more interesting one.

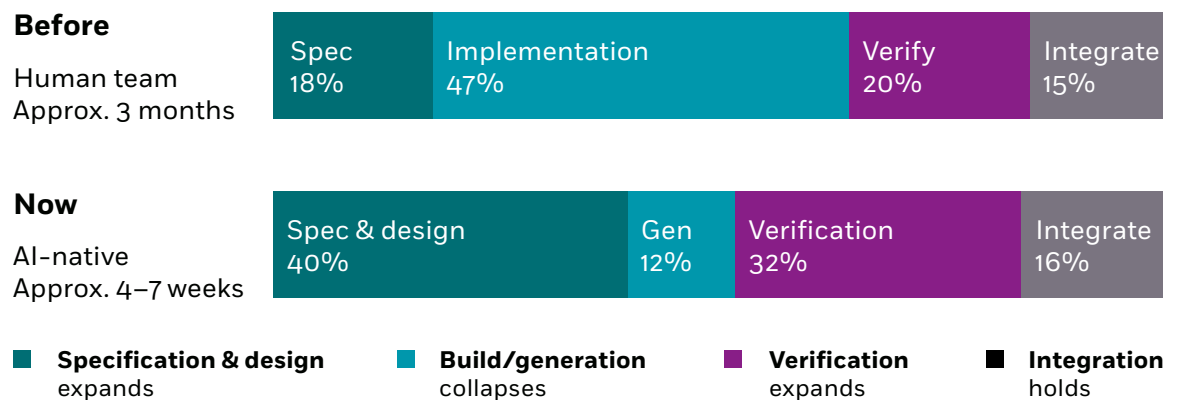


Figure 1. Where the effort goes. These proportions are indicative, drawn from UST delivery experience rather than a formal benchmark, and will vary by project.

Specification and design roughly double. Build falls to around a tenth of total effort. Verification rises by half again, and integration and hardening hold steady. Generation and verification are a cyclical, iterative process, and verification can be carried out by agents or humans.

This isn't a temporary effect of immature tooling. The cost of producing a plausible answer has fallen much faster than the cost of establishing that the answer is correct.

Where the risk goes

Total risk doesn't fall. It moves.

Risk that falls ▼	Risk that rises ▲
- Schedule risk “Can we build it” risk - The cost of exploring an option - Rework from late requirement change	- Plausible-wrong output - Over-production - Verification debt - Silent divergence between markets

Figure 2. The risk profile inverts alongside the effort profile

It is important to note that the risk profile has changed, and how it has changed.

The risks that fall are familiar ones. Delivery organisations have spent decades learning to manage schedule, feasibility and the cost of late change. Exploration is now cheap enough that options previously ruled out on effort grounds can simply be built and compared.

The risks that rise are less visible. Generated work is fluent by its nature of construction: incorrect work doesn't look incorrect. Output volume rises faster than the capacity to review it. And verification debt accumulates quietly, as work is generated, merged and put in front of a business user before anyone has established it's right.

Team shape follows. The teams working this way effectively are generally smaller, with data scientists and AI engineers orchestrating and reviewing rather than authoring code. Cadence moves towards continuous flow with frequent review gates rather than fixed sprints. While the same inversion holds in regulated and safety-critical domains, the net saving there is smaller, as verification still takes a larger share.

The mix shifts as well as the size. Build effort has fallen, but operational effort hasn't, so the balance moves towards platform and operations engineering rather than shrinking uniformly across every role.

What actually reduced

Analytics and ML development isn't software development

The change is in code production rather than in analytics as a whole. Pipeline code, feature engineering, model scaffolding and evaluation harnesses are now generated in minutes. The activities around them have moved much less, and they haven't moved evenly.

Activity	Effect of generative tooling
Framing the question	Minimal. Deciding what to model, and whether it's worth modelling at all, still depends on knowing the business
Feature engineering	Code production collapses. Knowing which features are legitimate, and when each becomes available relative to the point of prediction, does not
Model selection and tuning	Limited or reduced time – dependent on the nature of ML development (i.e. autoML or more manual methods)
Evaluation design	Harness code is generated in minutes. Judging whether the evaluation design is valid is unchanged
Productionisation	Scaffolding is generated quickly. Operating the result across its life isn't addressed at all
Interpretation into a decision	Minimal. Turning an output into a defensible decision still rests on causal reasoning

There's a simple reason for this. An AI system cannot reliably infer business knowledge from the data and code available to it. It can see the shape of the data but not what any of it means, and nobody has told it what decision the work is for. So it can tell you the code runs. It can't tell you the answer is nonsense.

Take an example. The task is to predict customer churn. The customer table has a field recording why someone cancelled. For customers still with the business, it's blank, and for customers who have left, it's filled in.

The AI assistant will use that field because it predicts leaving almost perfectly. And it's useless, because the field is only filled in after someone has already gone. Run the model on current customers, which is the entire point, and it's blank for every one of them.

The code is correct. The accuracy score is higher than a sound model would produce. That's what makes it dangerous rather than merely wrong.

The same gap in understanding causes other problems. A field that's only filled in for online orders, used as though it covered every sales channel. A model trained on the wrong group of customers. A forecast covering a period that doesn't match the decision anyone is trying to make.

Someone who knows the business spots all of these quickly. The AI assistant cannot, because it does not know the business in the same way. And when this assistant is producing forty reports a week, it is unlikely that the same someone is checking each of them.

When it comes to insight generation, the challenge to ensure correctness is harder still. Where an agent interrogates data and returns narrative to the user rather than a model, there's no artefact to inspect. There is prose that reads authoritatively. It's both the fastest growing use of this tooling and the most difficult to govern. The controls described later — grounding each request in defined concepts, and capturing the reasoning trail behind every answer — are what bring even this case within reach, since the narrative still has to resolve through the ontology to be produced.

The requisite activities that survived the collapse have something in common. Each depends on knowing what the data means and how the business actually works, rather than on the ability to write code.



Specification and verification in practice

What has to be written down, and what has to be checked

SPECIFICATION

Where generation is cheap, the specification carries most of the value. This isn't necessarily about longer documents but ensuring the things an AI generator will otherwise guess at are clearly and determinedly written down. These include:

- Which population, and who is excluded from it
- What time window, and where the point of prediction sits
- What counts as the outcome
- What the model may and may not see at that point of prediction
- What a good enough answer looks like, and what should happen where data is missing

None of this is new to anyone who has built analytics carefully. What's new is that it can no longer be left until somebody decides to check the outcome. A human in the loop will close down these risks, and such checks will either lead to agreed verification or reiteration. This is not realistic when analytics is expected to be automated.

VERIFICATION

In addition, verification means something different here too compared to software build. Testing that a pipeline executes is cheap and tells you very little. Checking an analytics artefact means asking whether the population is right, whether the outcome has been defined the way the business defines it, whether every feature was legitimately available at the point of prediction, and whether the evaluation reflects how the result will actually be used. Some of these the ontology can enforce structurally — ensuring a concept resolves to a single definition for the given context, or a method the process layer already fixes. The ontology can also surface and make auditable other details, such as whether a feature was legitimately available at the point of prediction, but settling them still takes judgement.

These are different activities from code review, and they require someone who knows the business rather than someone who knows the coding language. Whether an artefact stays valid after deployment is a separate question again, and one this paper returns to later.

Both changes point the same way. The intuitive response to more output is more review, but inspection is a per-artefact cost, and the number of artefacts is now effectively unbounded, so more review simply moves the bottleneck onto the most senior people available. To be effective, correctness has to become a property of the layer the work passes through, rather than a check applied afterwards.

When the business disagrees with itself

The same term means different things to different teams and markets

There is a further problem which has more to do with human ambiguity than machine interpretation. Even where someone does know the business, the business may not agree with itself about what its own terms mean.

Within an enterprise, customer analytics frequently shows this ambiguity very clearly, as it carries one of the widest and most contested semantic surfaces in the business. A large global CPG will typically hold several defensible definitions of active customer, household, loyalty and incremental volume. They differ by team, market, channel (DTC, retail, and distributor), and consent regime. Each is legitimate in context.

This inconsistency was previously limited by how much analysis people could physically produce. Generative tooling removes that limit. The result is a large number of internally coherent and mutually incompatible answers, all defensible, arriving faster than they can be reconciled.

Because each answer is defensible on its own terms, the disagreement surfaces late. Usually in a meeting, where two numbers that should match do not.



Ontology as a medium, not a catalogue

Why the distinction from a data catalogue and deterministic nature of the medium matters

In this paper an enterprise's ontology refers to its governed knowledge layer spanning semantic definitions, process knowledge and causal assumptions.

Using an ontology as a medium is different from using a data catalogue as a reference. Work can proceed without consulting the data catalogue and generally does. In contrast, a medium is the substance the work moves through. When the ontology is the medium, every request, every execution and every publication cross a boundary at which meaning is resolved, and policy applied.

The ontology itself is built in three layers, each depending on the one beneath it.

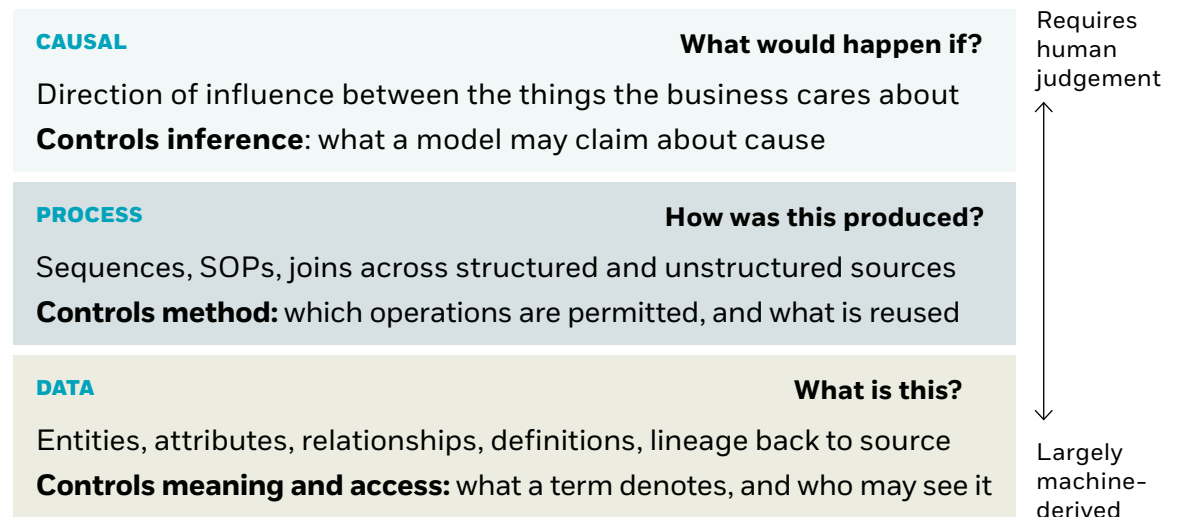


Figure 3. The three layers, and how derivable each one is.

THE DATA LAYER

The data layer holds entities, attributes, relationships, definitions, and lineage back to source. It answers what a thing is and where it came from. The control it contributes is over meaning and access, as a concept that resolves to a single definition can be governed and one that doesn't can't be. Most of this layer can be derived from schemas, profiling and existing lineage, with human input needed mainly to settle the conflicts the machine has surfaced rather than to author definitions from scratch. In customer analytics, this is the layer that reconciles household across a loyalty scheme, panel data and DTC records.

THE PROCESS LAYER

The process layer describes how the work is actually done, covering the sequences, standard operating procedures, the joins between structured and unstructured sources, and the transformations that produce a reported figure. It answers how a number was produced. The control it contributes is over method, in that it determines which operations are permitted and what should be reused rather than reinvented. It's partly derivable from code, logs and orchestration metadata, but a meaningful part of it sits with the people who run the process. In customer analytics, this is the layer that captures the real sequence behind a market's trade promotion effectiveness figure, including the manual reconciliation step nobody wrote down.

THE CAUSAL LAYER

The causal layer represents the organisation's validated assumptions about what influences what, together with the evidence and conditions under which those assumptions are considered valid. This isn't correlation, and it isn't sequence, it's the direction of influence between the things the business cares about. It can be used to answer the 'what if?'. The control this layer contributes is over inference, as it determines what a model is permitted to claim about cause. This layer is the least derivable of the three. In customer analytics, this is what determines whether price elasticity observed in one market can be assumed to transfer to another, and whether an observed lift came from the promotion, the display, or the seasonal baseline.

Two things get harder moving up the stack. The questions become more valuable and more difficult, from what a thing is, through how it was produced, to what would happen if. And the work becomes less automatable. AI can assist in proposing, extracting and maintaining tactical and operational causal structure, but those relationships still require subject-matter validation. That need becomes still more important for strategic causal claims.

Industry templates can give each of the three layers a starting point rather than a blank sheet, which shortens the data and process work considerably. They don't remove the need for validation at the causal layer.

Control at each layer is a property of that layer rather than something wrapped around it. This matters because governance implemented elsewhere sits outside a system that doesn't understand its own contents.

How work passes through

Every request, execution and publication crosses a control point

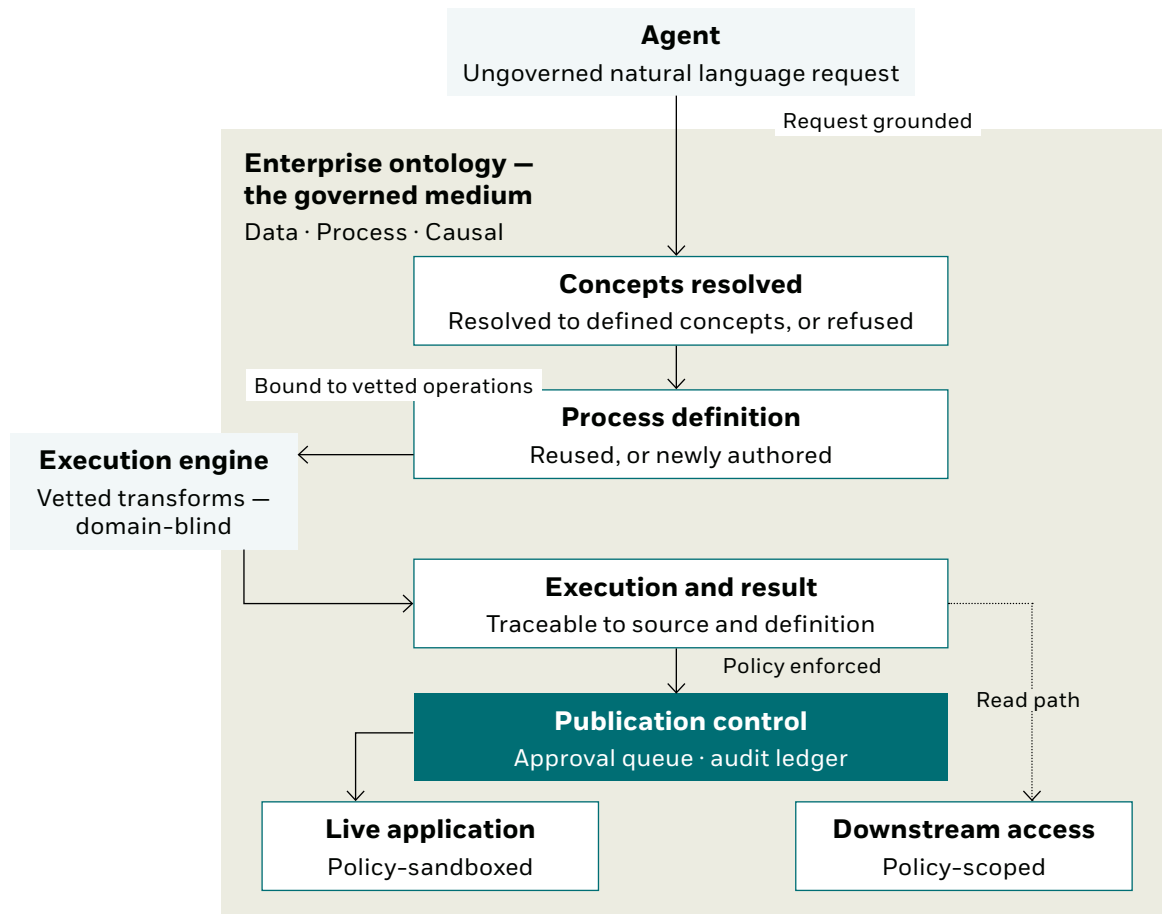


Figure 4. A generalised view. The specific bindings will vary by client.

The flow runs top to bottom. An agent submits the business request as ungoverned natural language, and that request is either resolved into defined concepts or refused. The execution engine remains domain-blind, bound to vetted operations rather than told what the business means, which is what allows it to be shared, hardened and reused. Results carry their lineage. Nothing reaches production without passing the publication gate, and read access is policy-scoped.

GROUNDING THE AGENT'S REQUEST

The agent sits outside the ontology, which is deliberate. It receives the request in whatever form the business user phrases it, and it has no independent authority to decide what any business term means. Interpretation is the ontology's job rather than the agent's, and this is where a good deal of the control sits.

That interpretation is constrained in three ways. Every business term in the request has to resolve to a concept the data layer defines, so a request about active customers in Germany resolves to the German market's definition rather than an average or an inferred one. Where a term is ambiguous or contested across markets, the agent isn't permitted to choose, and the request is either refused or returned as a decision for a named owner. And where the process layer already defines how a figure is produced, the agent is bound to that method rather than free to compose its own calculation.

The effect is that the agent's latitude is bounded by what the ontology defines. This is what makes an agent reasonable to put in front of business users at all. An agent with the same generative capability and no grounding layer will answer the same question fluently, differently, and without indicating that it has made any interpretive choices.

The refusal path is worth noting. A system that answers every question can't be governed, as it has no means of signalling that a question was malformed or that it's got the answer wrong. The ability to decline is a control rather than a limitation, and it's the cheapest control available as it operates before any work has been done. The countervailing risk is that a system which refuses is one users can route around, instead reaching for any available ungoverned tool. The governed path therefore has to be the easier one to use, not only the safer one.



What controlled has to mean

Properties worth requiring

- Data remains in place. The semantic model is the portable asset rather than the data.
- Models receive the minimum necessary context, so no single model holds the complete picture of a sensitive question. This is important as it reduces the exposure of Enterprise IP.
- Every answer can be reconstructed afterwards — the request, semantic bindings, evidence retrieved, tool calls, transformations, model and ontology versions, policy decisions and resulting recommendation.
- Gaps and conflicts are raised as decisions for named owners rather than resolved silently.
- Accumulated reasoning and memory are held independently of any single model vendor.
- The ontology evolves with use, but under review. Self-learning without a governance surface will produce drift.
- The ontology is versioned and deployed like any other production artefact, so that an answer can be reproduced against the ontology as it stood when it was given.

The benefits of using an ontology as a medium should not be understated. The deterministic nature of the ontology removes the reliance on inherently probabilistic agents to be correct. It enforces agreed definitions and policy constraints, so standardising responses to the enterprise's own concepts and rules, enabling reproducibility and making governance enforceable. It also enables materially stronger control of Enterprise IP and helps maintain data sovereignty. These are key challenges for Enterprise AI right now.



Operating at scale

MLOps is where these controls are either enforced continuously or not at all

Everything discussed so far has been about building. But building isn't running.

For simplicity, this paper uses the term 'MLOps' to refer to the production operations layer across analytics, ML and AI. Running analytics and ML in production is MLOps: the pipelines, deployment, monitoring and retraining that keep a model or a metric working once it has been handed over.

Generative tooling has helped here too, and considerably. Pipeline code, infrastructure definitions, deployment configuration and monitoring setup are all code, and all of it is generated far faster than it was. Much of the technical scaffolding for an MLOps capability has never been cheaper and faster to produce.

What hasn't changed is the running. Judging whether what's in production is still correct, and still permitted, takes what it always did. And there is now far more in production to judge. The build got cheaper, so the operating burden went up rather than down.

This matters most for governance. Approving something once isn't approving it forever, and MLOps is where that continuous checking either happens or doesn't. An ontology that resolved meaning correctly in March governs nothing in September unless something is checking that it still does. This is where an AI governance platform such as UST Prism can connect to the MLOps estate, drawing on its monitoring to check continuously that the ML in production is still valid for its application rather than relying on the approval it was given at deployment.

STATISTICAL DRIFT AND SEMANTIC DRIFT ARE DIFFERENT PROBLEMS

Classical MLOps monitoring watches feature distributions, prediction distributions, and performance decay. This remains necessary, but it won't detect the failure mode this paper is concerned with. Where a market changes what active customer means, the feature distributions can look entirely stable while every downstream figure quietly changes meaning. No statistical monitor is designed to catch that, as nothing in the data has moved. What has moved is the definition. Detecting semantic drift requires a governed representation of the definition and its changes, kept live and monitored rather than held only in a design document.

THREE VERSIONS, NOT TWO

Reproducing an answer requires the model version, the data version and the ontology version as it stood at the time. Mature MLOps practices typically version models and increasingly track the corresponding data state. Ontology or semantic-model versioning is currently much less commonly treated as an aspect of reproducibility. This is the point at which the ontology stops being a design asset and becomes part of the deployed estate, with the release discipline that implies. We envisage that ontology versioning as a core aspect of reproducibility will become increasingly standard practice.

EVIDENCE HAS TO BE A BY-PRODUCT

Audit evidence that has to be assembled by hand won't be assembled at the volume generative tooling now produces. The reasoning trail, the definitions in force, the ontology version and the approvals behind them all need to fall out of the pipeline automatically. A control that only works where someone remembers to operate it isn't really a control.

WHERE UST PRISM SITS

This operational layer is what UST Prism – UST's AI Governance platform – is intended to govern. Rather than sitting alongside the MLOps estate, it's designed to read from it, covering model registries, deployment records, monitoring telemetry and ontology version state, and to hold policy across the whole of it.

That distinction matters in practice. Any large Enterprise will run a mixed estate of vendor platforms, internal builds and embedded model capability, and governance reaching only the models built on one vendor's platform reaches relatively little of the estate. It also means the governance layer has to be able to interface with MLOps tooling a client has already standardised on, rather than requiring that tooling to be replaced.



Data readiness

Start with the data as it is. But scaling needs a scaled supply that can be trusted.

Many large enterprises need a data improvement programme, and often a substantial one.

The difficulty for analytics and AI progression is one of sequencing. Treating the data programme as a gate, on the assumption that nothing of value from an analytics and AI perspective can be built until it completes, imposes a delay that will last months if not over a year. It also deprives the programme of the information that would tell it what to prioritise.

Structured and unstructured sources can both be mapped into the ontology, so in reality data can be used in whatever state it's currently in, provided the constraints on it are explicit and travel with the answer.

WHAT THAT GETS YOU, AND WHAT IT DOESN'T

But it's worth being realistic about how far this goes. Working with data as it stands is enough to get started, and enough to prove whether an approach holds. It isn't enough to scale.

A pilot tolerates manual intervention. A handful of governed concepts can be maintained by the people who care about them. An Enterprise running hundreds of analytics products across dozens of markets can't work that way. At that point the binding constraint stops being the definition and becomes the supply: whether data arrives reliably, on schedule, at a known quality, without someone having to intervene.

Scaling therefore depends on the parts of data engineering that are unglamorous and unavoidable. Data pipelines that run without intervention. Quality checks that fail loudly at ingest rather than quietly somewhere downstream. Sufficient observability to know that today's data arrived and looked as it should. Where this holds, the ontology has something dependable to bind to. Where it doesn't, the ontology is describing data that may or may not be there.

So, data and analytics development programmes can run together rather than in sequence. Start with the data as it is, and ensure you invest in automating and quality-assuring the supply, because that is what turns a governed pilot into governed analytics and AI assets at scale.

UST has significant data architecture and data engineering programmes running with our Fortune-500 Clients, and we have transformed our approaches to Client data modernisation to be AI-native. Reliable, well-described data supply is what allows governed analytics to scale past a pilot.

Where UST is

Integrated platforms for analytics and governance

UST Pulse is a platform built against the argument set out above. It constructs the three layers described here across structured and unstructured Enterprise sources and uses industry templates to give the data and process layers an initiation point.

Its ontology studio provides the human governance surface, where data and business teams review what the system has inferred, resolve the conflicts it has surfaced, and extend the model deliberately. Answers carry their reasoning trail.

UST Prism, described above, is the governance control centre across the Enterprise's operating estate, which can include UST Pulse.

A first step

Testing these ideas using a contested concept, instrumented end to end

Treating the subjects of this paper as a platform decision is premature. To begin with, we recommend taking one contested concept in customer analytics, ideally where two markets each defend a different definition, and instrument it end to end – definition, lineage, causal treatment, access control and evidence trail.

This is usually enough to establish whether the approach holds in the given estate, and small enough to commit to nothing further. It also contributes to the data improvement programme, as running one concept properly reveals the true state of the data where it matters.

Three questions are worth being able to answer before scaling AI-assisted analytics further:

- Who owns each business-critical definition the models depend on?
- If an AI-generated answer were challenged nine months from now, could the route to it be reconstructed?
- Does AI governance span the wider estate, or sit inside a single vendor's product?

More details and demos of UST Pulse and UST Prism are available on request.

Together, we build for boundless impact

Since 1999, UST has worked side by side with the world's best companies to make a powerful impact through transformation. Powered by technology, inspired by people, and led by our purpose, we partner with our clients from design to operation. Our digital solutions, proprietary platforms, engineering, R&D, products, and innovation ecosystem turn core challenges into impactful, disruptive solutions. With deep industry knowledge and a future-ready mindset, we infuse expertise, innovation, and agility into our clients' organizations—delivering measurable value and positive lasting change for them, their customers, and communities around the world. Together, with 30,000+ employees in 35+ countries, we build for boundless impact—touching billions of lives in the process.

ust.com

© 2026 UST Global Inc.

Version 0101-20260902

U ■
S T