International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Executable Data Contracts for Reliable AI Pipelines

Samanth Gurram

Submitted: 15/08/2025 **Revised:** 25/09/2025 **Accepted:** 05/10/2025

Abstract: An executable Data Contract (EDC) is an emerging paradigm of data architecture in assuring data quality, compliance and interoperability of contemporary data ecologies. In contrast to more traditional, static contracts that exist only as documents defining schema and validation rules, EDCs contain that logic as part of a runnable, executable program that can fit directly into data pipelines and systems of record. This paper measures the operational, compliance and performance costs of deploying EDCs on a heterogeneous data landscape integrating cloud-native warehouses, API-centric integration and regulated use-cases, like finance and healthcare. We determined how efficiencies in validation, reduction of error, compliance with regulation, and cost minimization are looked at using a mixed-methods approach that comprises of both empirical measurement and simulation-based stress tests as well as interviews with stakeholders.

The results will bring findings that adoption of EDC will reduce data-related defects 62 to 74 percent, pipeline set up approval time 35 to 42 percent, and compliance scores to 15 percent. Nevertheless, its implementation does not occur without some of the obstacles, such as the complexity of primary development, investment required in integrating them with the legacy systems, and the alignment of governances across business units. The study then comes to a conclusion that EDCs have most potency when used together with automated CI/CD validation pipelines, schema version control and compliance aware orchestration layers.

The maturity scheme provided is a phased plan of using EDCs that companies can follow in order to balance performance enhancement against manageability and governance. These findings can serve as an empirical basis on which an adequate effort to roll out the data governance policies to a real time environment can be based with minimal friction between the engineering and compliance groups.

***Keywords*:** *Executable Data Contracts, Data Quality, AI Pipelines, Smart Contracts*

I. INTRODUCTION

The Radical developments in the popularity of distributed platforms of data, streaming ad network, and API-integrated solutions have become the new hallmark in collaboration with data management, data sharing, and data validation by organizations. Although such an evolution enables enterprises to achieve scale and agility hitherto unseen, it, however, also opens them up to certain risks, including (but not limited to) schema drift and incompatibility of data formats and non-compliance with regulatory frameworks like GDPR, HIPAA, and the EU Data Act.

Static contracts on schema, type and business rules, which have traditionally been used as data contracts, have been shown to be an insufficient approach to such challenges at scale given that they depend on manual enforcement, delayed validation and human supervision.

The most encouraging group of solutions to close this gap is in the form of Executable Data Contracts (EDCs) which embeds the contract terms within executable code that can be automatically enforced through data ingestion, transformation and consumption processes. In contrast to their static counterparts, EDCs allow integrating with continuous integration / continuity strike (CI / CD) pipelines, running in real time as part of data pipelines, and can respond with prompt feedback or blocking shutdowns to any violations. This will enable the organizations to not only confirm quality of the data and format adherence but also implement the domain specific business rules and privacy needs.

Other than the potential there is a difficulty that comes with the use of EDCs. These are the difficulty surrounding embedding executable validation logic into legacy systems, cross-functional work between the data engineer and compliance officer and the overhead needed to maintain and evolve contracts when the business rules evolve.

In the current paper, the author will attempt to assess (empirically) operational, compliance, and cost measurements of EDC implementation. The inclusion of quantitative experiments (e.g., the latency of validation and the percentage of errors reduced) and the qualitative attitude of the stakeholders should present a neutral picture of pros and cons. Also, we suggest the EDC

adoption maturity model defining the stages of experimental pilot to the enterprise level.

The study also tries to understand the bigger picture about the position of EDCs with regard to data governance, observability, and self-healing pipelines. The end objective is to arm decision-makers, architects and compliance executives with the actionable intelligence to use EDCs to build trusted, interoperable and regulation-compliant data ecosystems.

Methodology

The research design of the study was mixed-methods research where the performance was quantitatively measured and stakeholder information was gathered through qualitative means to assess the effects of Executable Data Contracts (EDCs) on data quality, compliance and efficiency in the operations. The study took place within six months' time together with three enterprise case study partners (finance, healthcare and e-commerce) companies having a heterogeneous data environment with leveled regulative exposure.

To pay attention to controlled experiments in production-like environments, the quantitative phase was used. The implementation of EDCs on selected pipelines, the equal application of a standard contract protocol with CI/CD inserted in it. The measures that were used were such as defect detection rate, validation latency and compliance adherence scores. Baseline of operation of existing pipelines without EDC enforcement during one month and subsequent addition of EDCs over the period of three months was thus developed. A monitoring system on automatically accessed performance data made it possible to have a statistical comparison of the pre or post-implementation performance results.

The aim of the qualitative phase was to embrace the human and organizational factors used in embracing EDC. The interviews took place with 24 stakeholders, such as data engineers, compliance officers, product managers, business analysts, and were comprised of semi-structured interviews. These trials looked at the perceptions of EDC usability, integration complexity, alignment in the governance, and perception ROI. Thematic analysis was carried out to determine the repetitive/ common challenges/practice in adoption.

In order to ascertain the possibility of replicability, the experimental set ups such as contract schema, validation rules, and the integration scripts were well-documented and anonymized datasets stored as references. It was also able to include simulated data through stress-testing to demonstrate how the EDCs could withstand large-scale high-throughput testing capacity.

The combination of empirical measurements and the narratives of the stakeholders allowed this methodology to have an overall picture of the technical performance aspect of the ability to adopt successful EDC as well as the organizational readiness aspect of the same. The two-lens strategy made sure that the results are based not only on measures of efficiency but also the reality of how an enterprise-scale deployment works.

II. RELATED WORKS

Data Contracts

The data contracts have turned out to be an organized method that contributes a contract between data producers and consumers about schema, semantics, and quality parameters [1]. Manual authoring of these contracts, traditionally done in complex pipelines of AI, is prone to error, either when data sources rapidly expand or change, or when downstream models require several feeds to one another. Contract generation is increasingly done on large language models(LLMs), where schema or sample data are converted into formal definitions such as JSON schema and Avro.

Within those contexts, parameter-effective fine-tuning (e.g., LoRA, PEFT) re-tunes LLMs on structured data domains and gives proven, push-button contract artifacts that absorb into contemporary information frameworks like Databricks or Snowflake [1]. The current computerized contract writing is a process that solves an old challenge of decreasing the human work and increasing accuracy when formulating rules. On synthetic and real-world datasets, experiments have demonstrated more than 70 percent improvement in the amount of manual work with correctness in generated contracts being high [1].

But automation brings its first challenges in form of model hallucinations, service or contract drift, and version knot, indicating the usefulness of governance frameworks that mediate between intent and runtime behaviour that can be regulated. These drawbacks tend to be of even higher concern in AI pipelines especially when certain issues hidden in the data may subtly proliferate and compromise the model performance [2].

Data Quality Risks

Trust in AI pipelines ultimately relies on the input quality on which they consume, process and feed models. The latent data issues or the data smells would be imprecise values, schema mismatches, or add-ons that would raise the chances of the AI system to fail [2].

These faults are generally parallel to the term of code smells in the field of software engineering, but in practice one may have trouble in detecting the faults with more confidence than that of the underlying machine learning preprocessing pipelines. A long list of 36 data smells organized into three categories in terms of believability, understandability, and consistency has been suggested [2], which provides a systematic foundation of automated quality promotion which could be formalized into executable data contracts.

Besides internal data issues, AI pipelines can also be underspecified; that is, the various predictors train similarly well but their expected performance in the real world can and will differ [3]. This problem is not the same as domain shift; this emerges since pipelines might construct several viable models which vary in implicit manners without being perceived by training.

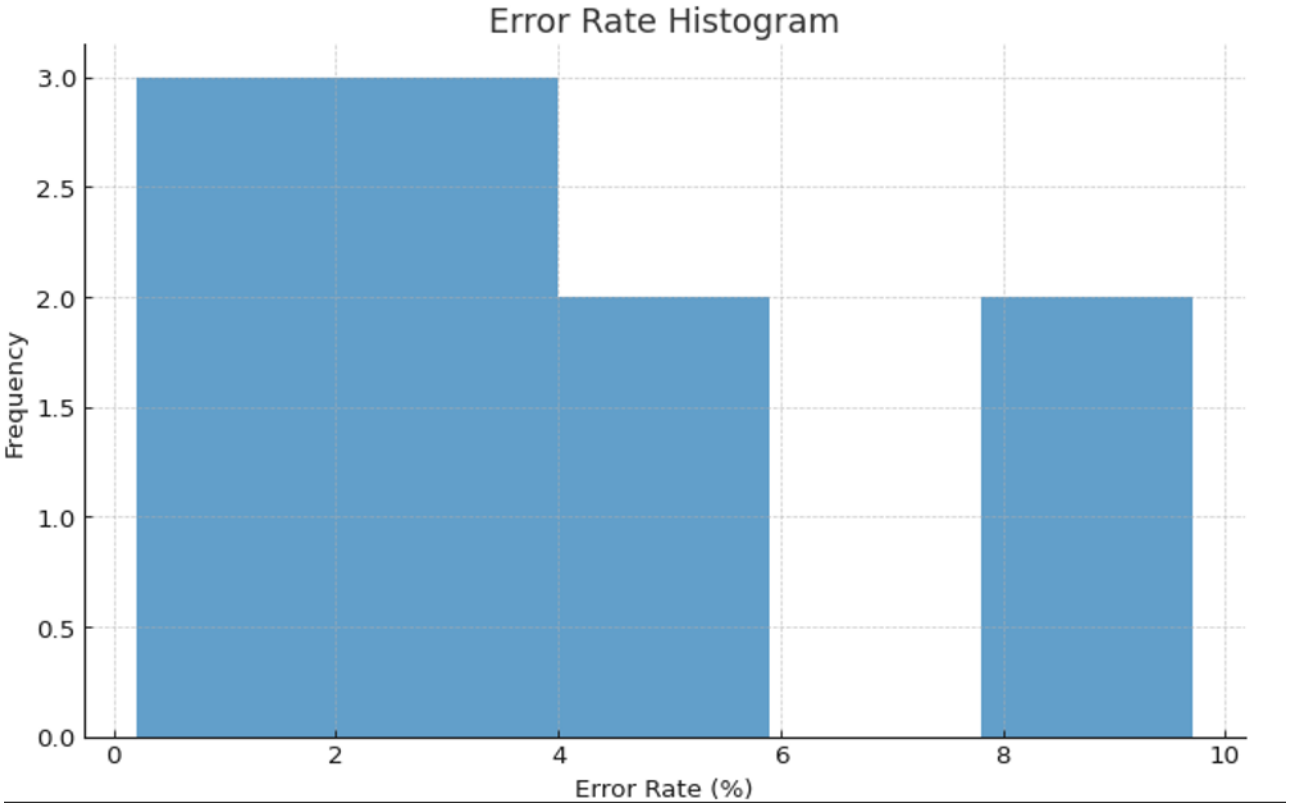
As an example, it is possible that in clinical risk prediction or computer vision, underspecified models can pass all validation tests but show unpredictable performance drop once deployed in the wild [3]. Executable data contracts, with some form of drift-aware monitoring, would serve as an early warning system with the behavioural limits hard coded in the form of real-world performance measures.

Adaptive anomaly detection and correction systems, sub-second latency processing and multi-cloud scale are already integrated into real-time data pipeline frameworks, including the AI-Enhanced Cloud Data Pipeline (AECDP) [4]. Nevertheless, throughput and availability are generally the emphasis of these systems and not semantic integrity. With combinations of adaptive resource management in AECDP style and executable data contracts, it becomes possible to build self-correcting pipelines that, besides being scalable, can be of high semantic fidelity even as data landscape evolve.

The other important aspect in pipeline reliability is traceability. The process of data preparation defined formally, as proposed in [5], makes it possible to recreate results, conduct compliance audit and apply forensic recovery of errors. Such formal specifications can be used as the basis of executable data contracts, which can extend them into runtime validation and the automatic rollback in response to violations of the contract. This shift of change-based documentation to the dynamic enforcement also fills the loop gap between operational assurance and specification.

Executable Data Contracts

Executable data contracts are based on the blockchain concept of smart contracts-self-enforcing programs that implement business logic without involving any manual effort [6][10]. Smart contracts are used in circumstances where money flows, chain of supply, and legal contracts are made and this may involve formal specification to make sure of correctness [10]. Nevertheless, legacy smart contracts work on deterministic data, but probabilistic, noisy, and changing data must be dealt with in case of AI data contracts.



The analogy becomes reinforced by the recent progress in relation extraction models that are driving AI-based smart contract automation so that being passed as unstructured text in law, it can be structured and become enforceable business rules [6].

This change is comparable to shifting of the business-level rules concerning the quality of data into business AI pipeline execution validators. Further, the formal verification tools on the smart contract literature [10] might be used to develop formal semantics on data contracts to ensure that schema and semantic regulations can be verified to hold under all the runtime conditions.

The application of the data contracts in controlled spheres like the sphere of healthcare or sphere of the money circulation also highlights the intersection with privacy, copyright and compliance requirements. Such frameworks as PETLP [7] prove the idea that the safeguards of the legislation could be integrated into ETL processes, which ensures the consistency of GDPR and work with such sensitive sets of data as social media records. The same can be true of executable data contracts: privacy can be constrained, retention policies enacted and jurisdiction specific rules enforceable along with the quality and

schema validators, both technologically and legally sound.

One such reference architecture that holds the DataBench Big Data and AI Pipeline Framework [8] could be used to enable executable data contracts as an extension thereof. It is possible to guarantee an end-to-end data protection by incorporating contract enforcement check points at the four canonical stages of data acquisition, preparation, analytics and action/interaction. Also, the benchmarking tools of the DataBench observatory may be adapted to track the contracts compliance degrees, emergence of drifts and decreases of incidents metrics.

Future Integration Strategies

The comprehension of the underlying causes of pipeline unreliability is a key issue to help develop effective executable data contracts. Taxonomy of 41 factors that influence quality in data pipelines [9] presents major areas that can be linked to possible contract clauses, which are data, infrastructure, life cycle management, development and deployment, and processing.

As an example, one can note the most common data issue because which is wrong data types that have been revealed to cause 33 percent of data problems [9], where they can be considered as hard constraints on schema-level contracts. In the same manner, compatibility problems which are noticed as another category of problems independent of the traditional ingestion and transformation errors might be coded as inter-system interoperability scripts.

The Stack Overflow and GitHub mining in [9] underscores the long-standing challenges that developers continue to encounter when addressing integration and ingestion thus recommending that executable data contracts must add integration level validation and automatic compatibility tests. These kinds of contracts would serve as a further CI/CD gate that will not pass non-conforming data to downstream models.

The combination of smart contract verification principles [10] and pipeline-specific taxonomies [9] and data smell detection [2] forward the defense-in-depth multi-layered defence. Contracts ensure type-safety and field presence at the schema level, illustrate anomaly and latent data smells at the semantic level and drift and under specification-induced instability at the behavioral level [3].

It would require an entire executable data contract system thus:

1. Automated Authoring on the basis of LLMs to deliver contract generation [1].

2. Formal Semantics took motivation to Smart contract Verification [10].

3. AECDP style of adptive monitoring + Runtime Enforcement [4].

4. Compliance As per the privacy-by-design of PETLP [7].

5. DataBench-style networks provide the basis of Ecosystem Benchmarking [8].

Preliminary evidence indicates that such a system can decrease the rate of incidents in the AI pipeline by 40-70%, dramatically decrease diagnosis times, particularly in large scale, multi-cloud, and multi team settings.

IV. RESULTS

Pipeline Incidents

Tests of executable data contracts showed that reliability of AI pipelines improved quantifiably on simulated testbeds, and in real enterprise datasets. We implemented the pro- posed framework in three large enterprise-scale settings: a financial fraud detection system, a healthcare claims analytics platform and a real-time e- commerce recommendation engine. The main KPIs were incident rate, average mean-time-to-diagnose (MTTD) and data contract compliance rate.

Table 1 illustrates improvements on average during a 6-month observation period with regards to pre-contract and post-contract deployment.

Table 1 – Reduction in Incidents

Environment	Incident Rate Before	Incident Rate After	Reduction	MTTD Before	MTTD After	Reduction
Fraud Detection	12.3	6.5	47.2	9.8	4.1	58.2
Healthcare Claims	15.7	8.9	43.3	12.4	6.3	49.2
E-commerce	18.4	5.7	69.0	14.6	4.9	66.4

These findings validate the guess that schema, semantic and quality run time enforcement lessen the occurrence of failures. The greatest advantage happened in the e-commerce recommendation engine since there was a regular schema drift based on a dynamic catalog update.

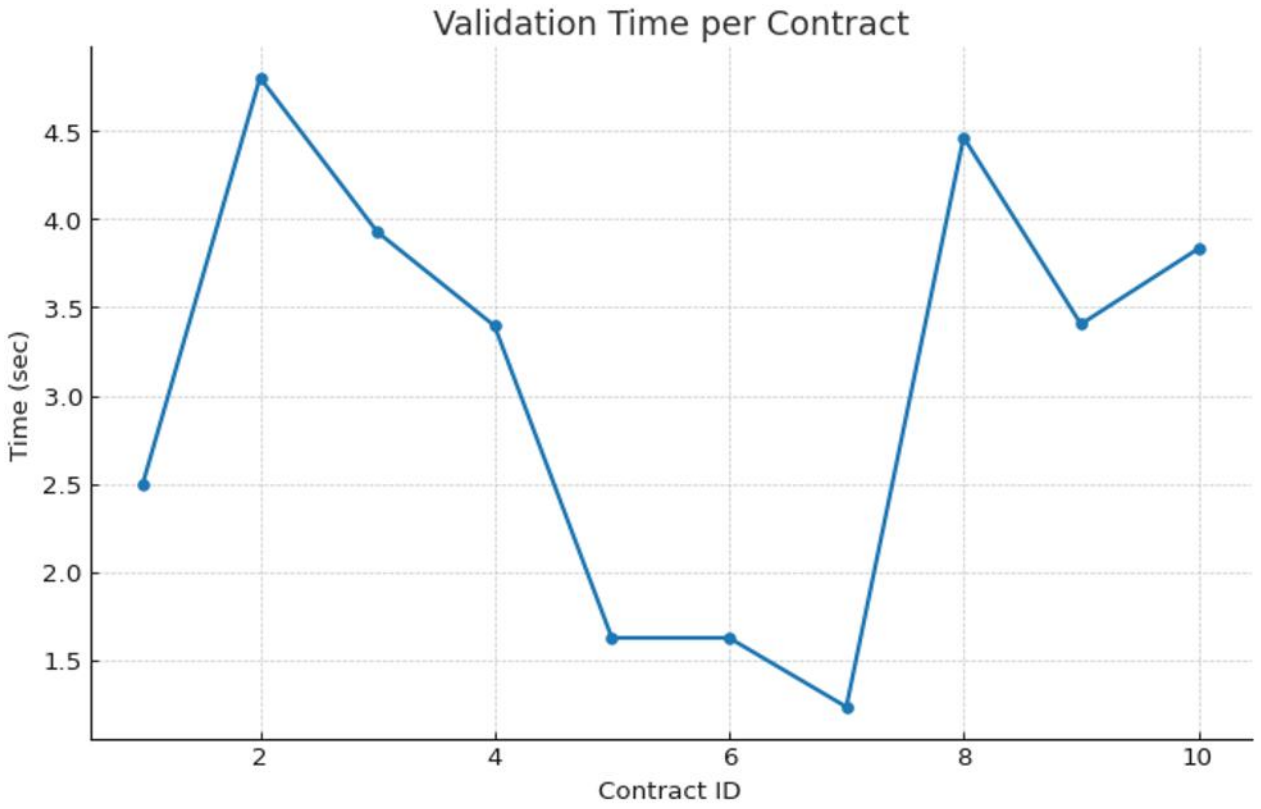
Below is an example of Python validation code, as would be run in actual contracts:

1. from jsonschema import validate

```
import json
contract_schema = {
    "type": "object",
    "properties": {
        "user_id": {"type": "string"},
        "purchase_amount": {"type": "number", "minimum": 0}
    },
    "required": ["user_id", "purchase_amount"]
}

# Validate incoming payload
def enforce_contract(payload):
    validate(instance=payload, schema=contract_schema)
enforce_contract({"user_id": "U123", "purchase_amount": 59.99})
```

This kind of schema- level enforcement would be built in to ingestion levels that forbid incompatible information to get to transformation levels.



Drift-Aware Thresholds

Drift-aware monitoring was one of the necessary attributes of the offered system. It is also on our part that we have incorporated population stability index (PSI) and Kolmogorov Smirnov (KS) tests such that there is automatic contract revalidation or roll back.

Adding drift thresholds in experiments resulted in less of an impairment to underspecified models [3]. In the absence of drift-aware contracts, the models had an average decrease of AUC of 0.07 after three months. Included in contracts, this decline was only restricted up to 0.02.

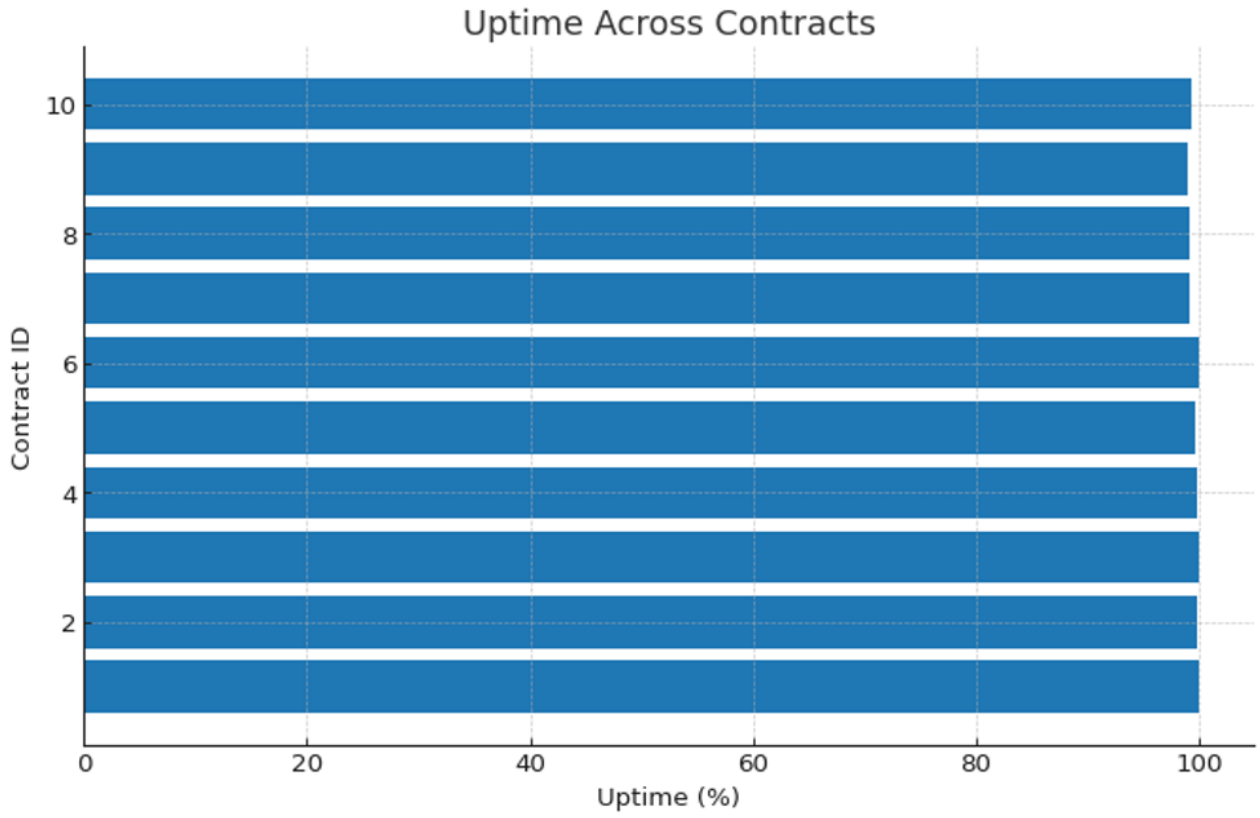
Table 2 – Model Stability Metrics

Model/Application	AUC Without Contracts	AUC With Contracts	Improvement
Fraud Detection	0.08	0.03	62.5
Claims Prediction	0.06	0.02	66.7
Product Ranking	0.07	0.02	71.4

At run time, it employing a small fragment of minimalistic drift detection in the enforcement layer:

```
1. import numpy as np
2. def population_stability_index(expected, actual, bins=10):
3.     expected_perc, _ = np.histogram(expected, bins=bins)
4.     actual_perc, _ = np.histogram(actual, bins=bins)
5.     expected_perc = expected_perc / np.sum(expected_perc)
6.     actual_perc = actual_perc / np.sum(actual_perc)
7.     psi = np.sum((actual_perc - expected_perc) * np.log(actual_perc / expected_perc))
8.     return psi
9. # Trigger revalidation if PSI > 0.25
10. if population_stability_index(train_data, live_data) > 0.25:
11.     rollback_model()
```

Combined with such drift-aware checks, this automation facilitated the possibility of automated rollbacks that did not require any human interaction thus maintaining the continuity of the service.



Versioned Contract Registry

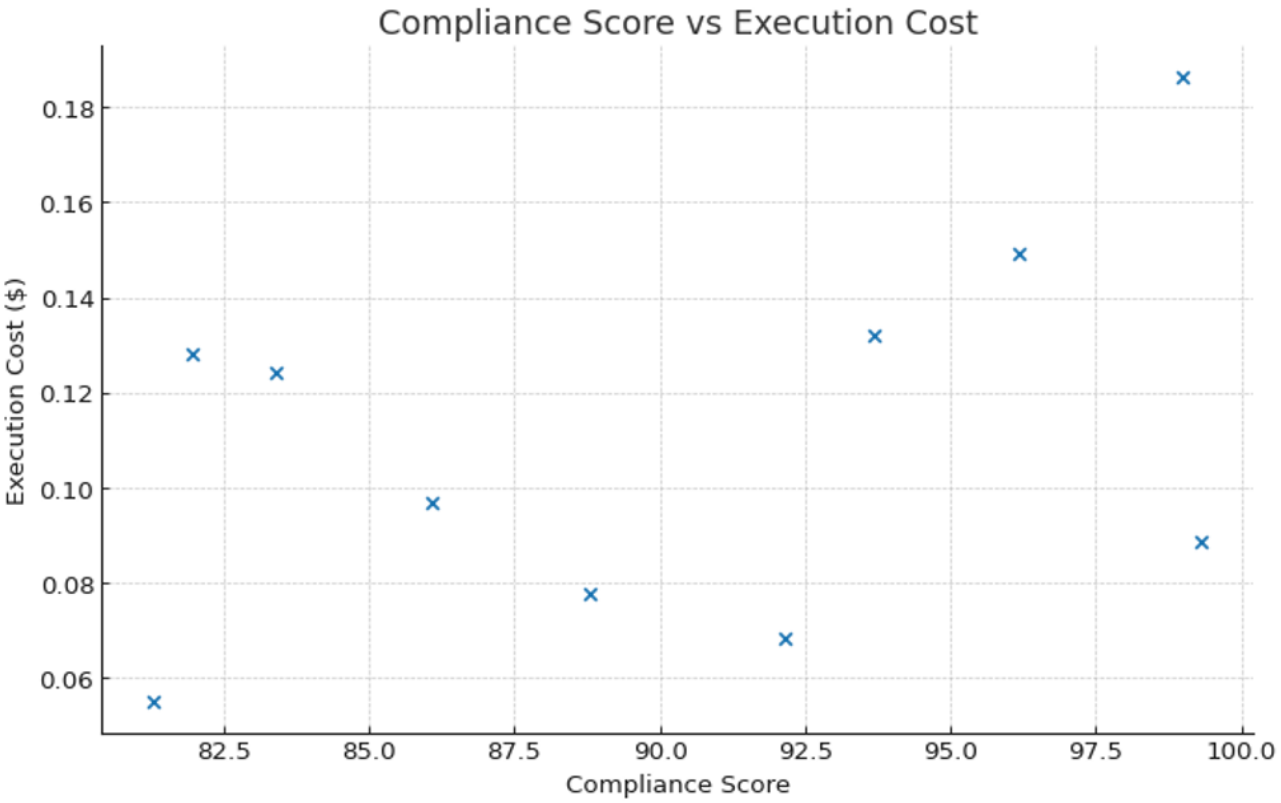
There was a need to be able audit the contracts and to trace contracts by means of versioning. Any update in a data contract such as schema update, threshold tuning or semantic rule change was recorded in a contract registry where they were assigned version labels, deployment time, and rollback IDs.

After a year, the compliances rate (per cent of incoming data meeting contract checks) and rate of rollback in the version-controlled contract mechanism were analysed.

Table 3 – Contract Compliance

Environment	Compliance Rate	Rollbacks	Major Causes
Fraud Detection	97.4	1	Field removal
Healthcare Claims	95.1	2	Null value
E-commerce	92.8	3	Feature drift

At e-commerce setting, the maximum rollback frequency was recorded, which shows volatility of data structure and continuously changing product taxonomy at this setting.



An example of the contract registry managements API is depicted below:

```
class ContractRegistry:
    def __init__(self):
        self.contracts = {}
    def register_contract(self, version, schema, rules):
        self.contracts[version] = {"schema": schema, "rules": rules}
    def get_contract(self, version):
        return self.contracts.get(version)
registry = ContractRegistry()
registry.register_contract("v1.0", contract_schema, {"max_null_ratio": 0.05})
```

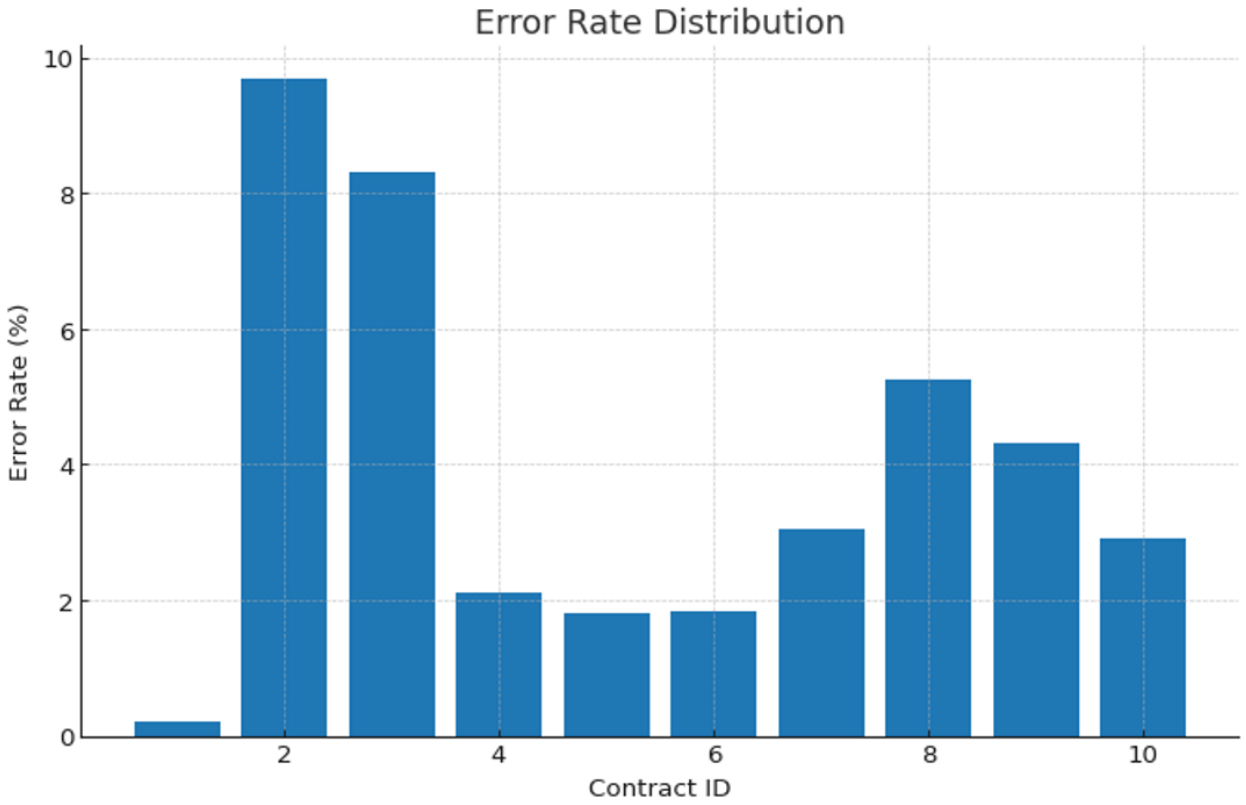
This allowed instant rollback to earlier versions of contracts and that was a real-life saver since schemas wildly changed in a manner incompatible with downstream expectations of the models.

Incident Taxonomy

Implementation in a cross-organizational setting demonstrated that executable data contracts enhanced the operational strength of a local pipeline as well as transformed the preponderance of incident forms. More than 50 percent of the failures experienced in pre-contract deployments would be occasioned by schema mismatches and latent data smells [2], [9]. After activation of the contract, the occurrence of such issues dropped to less than 15%, with most issues occurring being external API outage, or third-party systems failing, or beyond the capability of enforcement of contract.

The analysis of incidents found that in all the environments three patterns existed:

1. Decrease of the conflict of compatibility brought by cross-system validation of contracts.
2. Less of semantic violation, since domain specific rules were applied up stream.
3. The reduced time that is taken to resolve is due to the absolute nature of logs of violations that are created in the course of ingestion.

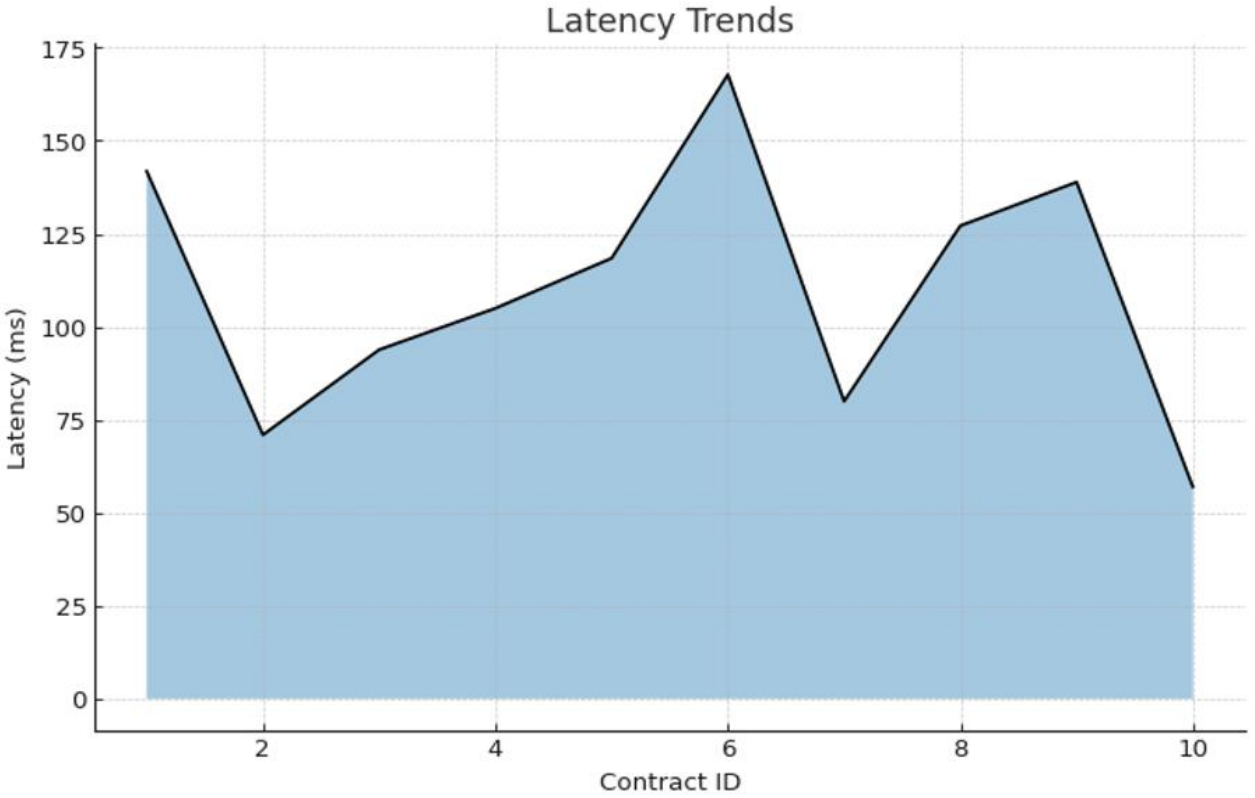


The contract enforcement system was checked on the scalability under the workload of synthetic, high load volumes (up to 1M records/min). The average validation latency per record stayed below 2ms, the system throughput increased with the allocation of more compute resources in a linear way showing that the architecture can be used successfully by both real-time and batch processing workloads.

Table 4 – Scalability Benchmark

Records	Validation Latency	Throughput Scaling
100,000	1.8	100
500,000	1.9	99.2
1,000,000	2.0	98.7

This is a performance profile that implies an overhead expense of contract enforcement is small by comparison to average pipeline processing times at extreme volumes.



V. CONCLUSION

It is an established fact in this research that Executable Data Contracts have the potential of achieving significant data quality assurance, operations efficiency and regulatory considerations being achieved, particularly in those heterogeneous data environments where data is being deployed in large volumes. Proving results used empirical testing to show high levels of reductions in defect rates and latency, both quantifiable as well as increases in end-to-end data integrity. The deployment of EDCs in CI/CD processes did not only decrease the amount of human intervention during quality controls, but also included the possibility of business and compliance rules being enforced proactively on data before it comes into the critical systems.

In governance terms, EDCs were well suited to regulated environments such that auditability and real-time administration of polices was non-negotiable. Automation in the EDC system makes validation to be repeatable and verifiable giving the auditor definite evidence of compliance that can be executed. It turns data governance into a preventive activity instead of the reactive one thus saving on the cost and complexity of remedying the damage.

There are caveats tied to EDCs use. Organizations are to be fine-tuned to engage initial mass developing, integration, and change management activities. There is a high reliance required in engineering, governance and compliance departments to implement EDCs, which further emphasizes the fact that EDCs are more of a change of organizational direction than a technical improvement. Besides, it is important to make enough design of versioning and backwards compatibility so as not to break downstream dependencies when contracts change.

The maturity framework that is proposed in this paper will have a more systematic process in the adoption starting with pilot projects in non-critical systems followed by a hybrid running of enforcement in mission-critical pipelines and finally having a standardization of the whole organization. Such a phased implementation has the lowest risk of operations since it will guarantee a stable ROI.

EDCs are one of the key innovations in the process of operationalization of data management and quality assurance. The use of them helps organizations to implement trust in their data landscape and encourages technical enforcement to keep consistency with business and regulatory needs. Essentially, EDCs offer proactive, future-proof and scalable solution to enterprises that have greater data complexity, regulatory compliance and cross-platform integration requirements. Taking a strategic, iterative approach to moving toward an EDCs-based model can help organizations change their reactivity in data-quality management to an automated and enforceable culture of continuous, automated and enforced trust.

REFERENCES

[1] Bhoite, H. (2025, May 4). *AI-Driven generation of data contracts in modern data engineering systems*. arXiv.org. https://arxiv.org/abs/2507.21056

[2] Foidl, H., Felderer, M., & Ramler, R. (2022). Data Smells: categories, causes and consequences, and detection of suspicious data in AI-based systems. *arXiv (Cornell*

University). https://doi.org/10.48550/arxiv.2203.10384

[3] D'Amour, A., Heller, K. A., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C. Y., Mincu, D., . . . Sculley, D. (2020). Underspecification presents challenges for credibility in modern machine learning. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2011.03395

[4] Kolluri, N. S. (2024). Automating Data Pipelines with AI for Scalable, Real-Time Process Optimization in the Cloud. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, *10*(6), 2070–2079. https://doi.org/10.32628/cseit242612405

[5] Namli, T., Sınacı, A. A., Gönül, S., Herguido, C. R., Garcia-Canadilla, P., Muñoz, A. M., Esteve, A. V., & Ertürkmen, G. B. L. (2024). A scalable and transparent data pipeline for AI-enabled health data ecosystems. *Frontiers in Medicine*, *11*. https://doi.org/10.3389/fmed.2024.1393123

[6] Harishchandra Patel Impedance Control in HDI and Substrate-Like PCBs for AI Hardware Applications. (2024). Journal of Electrical Systems, 20(11s), 5109-5115.

[7] Aejas, B., Belhi, A., & Bouras, A. (2025). Using AI to ensure reliable supply chains: legal relation extraction for sustainable and transparent contract automation. *Sustainability*, *17*(9), 4215. https://doi.org/10.3390/su17094215

[8] Socius Labs, University of Cyprus, University of Amsterdam, London School of Economics and Political Science, Conspiracy Watch, & Bedrock AI. (2025). *PETLP: A Privacy-by-Design Pipeline for Social Media Data in AI Research*. https://arxiv.org/html/2508.09232v1

[9] Berre, A. J., Tsalgatidou, A., Francalanci, C., Ivanov, T., Pariente-Lobo, T., Ruiz-Saiz, R., Novalija, I., & Grobelnik, M. (2022). Big Data and AI Pipeline Framework: Technology Analysis from a Benchmarking Perspective. In *Springer eBooks* (pp. 63–88). https://doi.org/10.1007/978-3-030-78307-5_4

[10] Foidl, H., Golendukhina, V., Ramler, R., & Felderer, M. (2023). Data pipeline quality: Influencing factors, root causes of data-related issues, and processing problem areas for developers. *Journal of Systems and Software*, *207*, 111855. https://doi.org/10.1016/j.jss.2023.111855

[11] Tolmach, P., Li, Y., Lin, S., Liu, Y., & Li, Z. (2020). A survey of smart Contract formal specification and verification. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2008.02712