



(RESEARCH ARTICLE)



COMMUNICATION METADATA AS A DATA PRODUCT: PLATFORM DESIGN FOR INTERACTION ANALYTICS AND AI FEATURE GENERATION

Santosh Kumar Maddali *

Staff Data Engineer.

* Corresponding Author

World Journal of Advanced Research and Reviews, 2025, 26(03), 2914–2923

Publication history: Received on 04 April 2025; revised on 12 June 2025; accepted on 15 June 2025

Article DOI: <https://doi.org/10.30574/wjarr.2025.26.3.2320>

Copyright © 2025 Author(s) retain the copyright of this article. This article is published under the terms of the Creative Commons Attribution License 4.0

ABSTRACT

Communication systems such as email, chat, calendaring, and voice generate a continuous stream of metadata: who communicated with whom, when, through which channel, and with what response latency. This metadata is a rich source of signal for interaction analytics and for machine learning features, yet in most organizations it is scattered across application logs, collected without a stable contract, and reused without clear governance. This paper treats communication metadata as a first-class data product and proposes a reference platform design for producing it. The design separates ingestion and normalization from a privacy and policy layer, publishes the product under a versioned contract with explicit quality objectives, and serves two consumer families: interaction analytics workloads that compute interaction structure and responsiveness measures, and AI feature generation workloads that require point in time correct, reproducible features. We define the product contract, the identity and event model, the minimization and pseudonymization controls, and the feature generation path, and we describe an evaluation framework built on contract conformance, freshness, completeness, and reproducibility rather than on any single deployment. The contribution is a practical, technology-neutral blueprint that platform teams can adapt to existing streaming and lakehouse infrastructure while keeping message content out of scope and keeping privacy obligations enforceable by construction.

Keywords: Communication Metadata, Data Products, Data Contracts, Interaction Analytics, Feature Engineering, Feature Store, Privacy By Design, Streaming Data Platforms

1 INTRODUCTION

Most large organizations operate several communication systems in parallel. Each of these systems records events as a side effect of normal operation: a message was sent at a timestamp, a meeting was accepted, a call was missed, a thread was replied to after some delay. Taken individually these records are operational logs. Taken together, normalized, and joined to a consistent identity model, they describe the structure and dynamics of interaction: which groups of parties communicate with each other, how quickly requests are acknowledged, how interaction volume is distributed across channels and over time, and how these patterns change. The same signals are also strong predictors in machine learning tasks such as churn risk, request routing, and demand forecasting.

Despite this value, communication metadata is rarely engineered with the discipline applied to transactional data. It is typically extracted ad hoc by individual teams, each of which writes its own parsing logic, applies its own notion of identity, and stores the output in a private table with no versioning. The result is duplicated pipelines, inconsistent metrics, and, more seriously, uncontrolled reuse of data that carries real privacy obligations. The literature on machine learning systems has documented how such undeclared data dependencies accumulate as technical debt [1], [2], and the data mesh movement has argued that the remedy is to treat data as a product with an accountable owner and a published interface [3], [4].

This paper applies the data product idea specifically to communication metadata and answers a practical question: what does a platform need to provide so that communication metadata can be consumed safely and repeatedly by both analytics and AI workloads? The scope is deliberately limited to metadata. Message bodies, attachments, transcripts, and any other content are excluded from the design. This limit is not only a simplification; it is what makes the product tractable from a governance standpoint, because the schema of metadata is small, enumerable, and amenable to minimization.

Three non-goals should be stated as clearly as the scope. The design is not a monitoring, productivity measurement, or performance evaluation system, and nothing in it supports such uses: individual level reporting is prohibited by policy in Section 7, aggregate release is subject to a minimum group size, and the feature families of Section 6 are defined for prediction tasks about requests and channels, not about persons. The paper does not describe the systems or practices of any particular organization. And it takes no position on whether any organization should collect communication metadata at all; it addresses only how such data, where it is lawfully collected for an approved purpose, can be engineered so that the purpose limitation is enforced by the platform rather than left to the discretion of each consumer.

The contributions are as follows. First, we define a canonical event and identity model for communication metadata that is independent of any single source system. Second, we specify a product contract that includes schema, semantics, quality objectives, and access policy, and we show how the contract is versioned. Third, we describe a layered architecture in which privacy controls are applied before the product is published, so that downstream consumers cannot bypass them. Fourth, we describe the feature generation path, including point in time correctness and online and offline consistency. Fifth, we propose an evaluation framework that platform teams can use to assess an implementation without depending on proprietary benchmark data.

2 RELATED WORK

2.1 Data products and data mesh

The data mesh emerged as a socio-technical response to the bottlenecks of centralized data lakes, with domain-oriented ownership, data as a product, a self-serve platform, and federated governance as its four principles [3], [4]. A data product in this sense is discoverable, addressable, trustworthy, self-describing, interoperable, and secure. The design in this paper adopts those properties as requirements and makes them concrete for one specific domain. Gebru et al. proposed datasheets that document the motivation, composition, collection process, and recommended uses of a dataset [5]; we treat such documentation as a mandatory part of the product contract rather than as an optional attachment.

2.2 Streaming and lakehouse infrastructure

Communication metadata arrives as an unbounded stream. Log-based messaging systems such as Kafka provide the durable, replayable transport that makes late-arriving and reprocessed events manageable [6]. The Dataflow model formalizes the distinction between event time and processing time and introduces windowing and triggering semantics that are essential when computing responsiveness measures over out-of-order events [7]. Kleppmann gives a broad treatment of the reliability and consistency trade-offs involved [8]. On the storage side, the lakehouse pattern combines open table formats with transactional guarantees, which allows a single copy of the product to serve both batch analytics and feature generation [9], [10], with Spark as a common execution engine [11]. Kimball and Ross remain the standard reference for dimensional modeling of the resulting analytic tables [12].

2.3 Machine learning data management

Sculley et al. identified data dependencies, undeclared consumers, and pipeline jungles as sources of hidden technical debt in machine learning systems [1]. Polyzotis et al. surveyed data lifecycle challenges including validation, cleaning, and provenance [2]. Breck et al. proposed a production readiness rubric that includes tests for feature and data quality [13]. Schelter et al. described a declarative approach to large-scale data quality verification that can be integrated into pipelines [14]. Production machine learning platforms such as TFX established shared, validated data and feature

pipelines with explicit checks for training and serving skew as standard components [15]. This paper builds on these ideas by placing the feature generation step downstream of a governed product rather than directly on raw logs.

2.4 Analysis of communication metadata

Social network analysis provides the formal tools for describing interaction structure, including centrality, density, and community measures [16]. Eagle and Pentland demonstrated that metadata alone, in their case device proximity and call logs, is sufficient to infer meaningful interaction patterns [17]. That finding cuts both ways: it establishes the analytic value of metadata and it establishes that metadata is personal data requiring protection. Sweeney's k-anonymity [18] and Dwork's differential privacy [19] provide the two main families of formal protection for aggregate release, and the General Data Protection Regulation sets out the purpose limitation and data minimization principles that a lawful design must satisfy in the European context [20].

2.5 Recent developments

More recent work has consolidated several of these threads. Work on decentralized governance for data mesh platforms has argued that data product ownership, policy definition, and policy enforcement must be built into the platform itself rather than layered on afterwards [21]. Data contracts have since been treated as a first-class engineering artifact with their own tooling and lifecycle [22]; the contract in Section 5 follows that treatment. On infrastructure, comparative analysis of lakehouse table formats has clarified the trade-offs among transactional log designs and their implications for time travel and concurrent writers [23], and feature stores have matured into managed systems with explicit point in time join semantics and online and offline consistency guarantees [24]. On privacy, practical guidance for applying differential privacy to machine learning has narrowed the gap between formal guarantees and deployable systems [25]. Finally, empirical analysis of communication metadata collected at scale has shown that sensitive facts about parties can be inferred from metadata alone, without any access to content [26], which reinforces both the analytic value and the governance obligations that motivate this paper.

3 PROBLEM DEFINITION AND DESIGN GOALS

We consider an organization with N communication source systems, each producing events with its own schema, identifier space, timestamp precision, and delivery guarantees. Several consumer teams need a consistent view of interaction patterns across those systems. In the absence of a shared product, each consumer builds its own extraction, leading to the following common failure modes: metric disagreement because identity resolution differs; silent breakage when a source changes its export format; inability to reproduce a feature value as of a past date; and audit findings because data collected for one purpose was reused for another without review.

The platform must therefore satisfy the following design goals.

- G1. Canonical model. All sources map to one event schema and one identity model so that consumers reason about interaction patterns, not about source systems.
- G2. Contract stability. The product exposes a versioned contract; breaking changes require a new major version and a migration window.
- G3. Privacy by construction. Minimization, pseudonymization, purpose tagging, and retention are applied before publication and cannot be bypassed by consumers.
- G4. Dual consumption. The same product serves aggregate interaction analytics and record level feature generation without divergence.
- G5. Bounded reproducibility. Any feature value or analytic aggregate can be recomputed as of a stated event time from the product alone, for any event time within a published reproducibility horizon. Beyond the horizon, only approved aggregates and the audit artifacts describing their derivation are retained.
- G6. Measurable quality. Freshness, completeness, and conformance are stated as objectives and monitored continuously.

4 REFERENCE ARCHITECTURE

Figure 1 shows the proposed architecture. It is organized as four producer-side layers that end in the published product, and three consumer-side families that read from it. The direction of dependency is strictly left to right and top to bottom; consumers never read from ingestion or source layers directly.

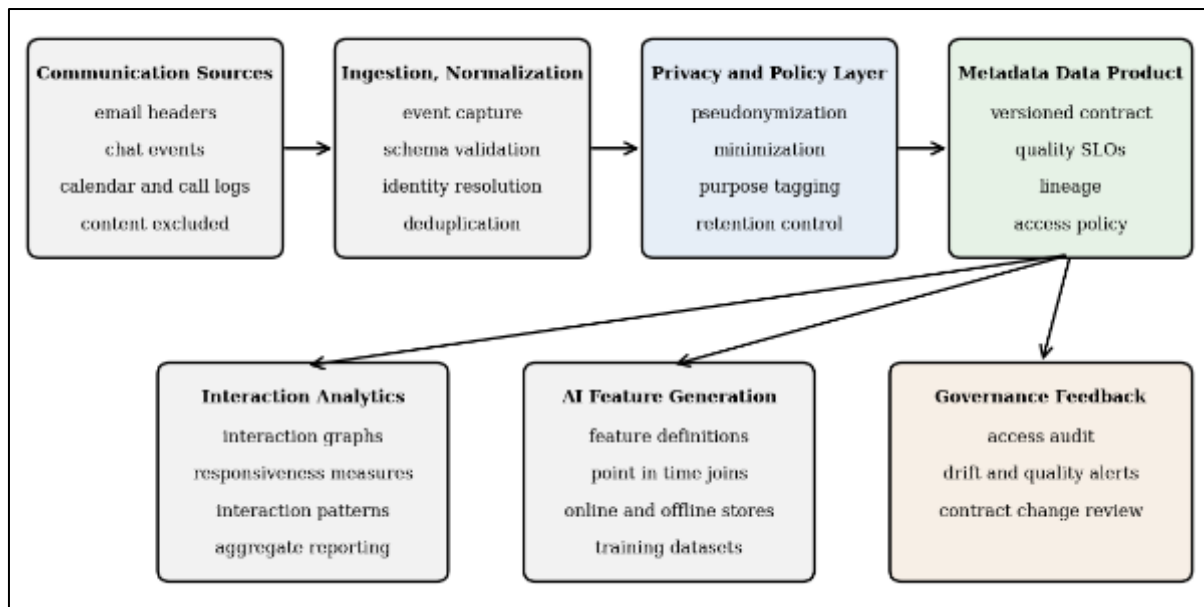


Figure 1: Reference architecture for communication metadata as a data product

4.1 Communication sources

Sources include email header logs, chat platform event exports, calendar systems, and telephony call detail records. Only header level fields are collected: sender, recipients, timestamps, thread or conversation identifiers, channel type, and delivery state. Subject lines, message bodies, attachment names, and transcripts are excluded at the connector. Excluding content at the earliest point simplifies every later control and removes the most sensitive category of data from the platform entirely.

4.2 Ingestion and normalization

Each connector publishes raw events to a durable log topic partitioned by source. A normalization stage validates each event against the source schema, rejects or quarantines malformed records, and maps the event to the canonical schema described in Section 5. Identity resolution maps source-specific addresses and handles to a stable internal party identifier; the mapping table is itself a governed asset with its own change history. Deduplication uses a deterministic event key composed of source system class, source instance identifier, source event identifier, and event type so that replays from the log are idempotent. The source instance identifier distinguishes separate tenants or deployments of the same platform, whose local event identifiers are not guaranteed to be globally unique; without it, otherwise valid events from two instances could collide on the same key and one would be silently dropped as a duplicate. Event time is preserved from the source and processing time is recorded separately, following the Dataflow model [7].

4.3 Privacy and policy layer

This layer is the boundary between operational data and the product. Four controls are applied. Pseudonymization replaces the internal party identifier with a keyed pseudonym; the key is held by the platform and rotated on a schedule, and re-identification is possible only through an audited service. Minimization drops fields that are not required by any approved purpose and coarsens fields where precision is not needed, for example truncating timestamps to the minute for analytics grade outputs. Purpose tagging attaches the set of approved purposes to each product version so that access policy can be evaluated against declared use. Retention control enforces a maximum age per field class and deletes or aggregates records that exceed it. The retention period of event level records is the product's reproducibility horizon (Section 5.2): within it, event level replay is possible; beyond it, only approved aggregates and the audit artifacts that describe how they were produced remain. Because these controls run before publication, consumers cannot obtain raw identifiers or fields beyond the precision authorized for their access class.

4.4 Metadata data product

The product is published as a set of versioned tables in an open table format on the lakehouse [9], [10], together with a machine readable contract, lineage metadata linking each table to its upstream sources and transformations, and a policy document that defines who may read which columns for which purposes. The product owner is a named team accountable for the contract and for the quality objectives. Discovery is through the organization's catalog, where the contract and datasheet are the primary documentation.

4.5 Consumer families

Interaction analytics consumers compute aggregate measures such as interaction graphs, response time distributions, and interaction patterns across groups. They typically read analytics grade tables with coarsened fields and produce reports at group level. AI feature generation consumers compute record level features for training and inference. They read feature grade tables at full retained precision, subject to purpose approval, and write results into offline and online feature stores. Governance consumers read access logs, quality metrics, and contract change requests to operate the oversight process. Each family is discussed further in Sections 6 and 7.

5 THE PRODUCT CONTRACT

The contract has four parts: the canonical schema, the semantic definitions, the quality objectives, and the access policy. Table 1 lists the core fields of the canonical communication event.

Table 1: Core fields of the canonical communication event

Field	Type	Description
event_id	string	Deterministic key: source system, source instance, source event id, event type
event_type	enum	sent, received, replied, forwarded, invited, accepted, declined, call_started, call_ended, missed
channel	enum	email, chat, calendar, voice
actor_pid	string	Pseudonymous identifier of the initiating party
target_pids	array	Pseudonymous identifiers of receiving parties
thread_id	string	Pseudonymous conversation or thread key
event_time	timestamp	Source event time, UTC
ingest_time	timestamp	Platform processing time, UTC
latency_ms	integer	Time since the most recent event in the same thread by a party other than the actor; null if no such event exists
party_count	integer	Number of distinct parties on the event
source_system	string	Originating system class, for example email or chat platform
source_instance_id	string	Opaque token for the originating tenant or deployment; used in key generation and not exposed at analytics grade
schema_version	string	Contract version that produced the record

5.1 Semantic definitions

Several fields have non-obvious semantics that must be fixed in the contract to prevent metric drift. A reply is defined as an event whose thread identifier matches a prior event and whose actor was a target of that prior event; a message in the same thread from the original actor is a follow-up, not a reply. Response latency, stored in `latency_ms`, is measured from the most recent event in the same thread by a party other than the actor, not from the immediately preceding event regardless of actor and not from thread creation; a follow-up by the same actor therefore carries a null latency. A meeting acceptance is recorded at the time of acceptance, not at the meeting start. These definitions are written in the contract in plain language and are implemented once in the normalization stage, so that all consumers inherit the same meaning.

5.2 Quality objectives

The contract states a small set of objectives that the owner commits to and that the platform measures: freshness, defined as the lag between event time and availability in the product; completeness, defined as the fraction of source events that appear in the product after the freshness window; conformance, defined as the fraction of records that pass schema and semantic checks; uniqueness, defined as the absence of duplicate event keys; and the reproducibility horizon, defined as the minimum age up to which event level records are guaranteed to be retained at feature grade precision. The horizon is the explicit boundary between goal G5 and the retention control of Section 4.3, and it is published in the contract so that consumers can plan training and backfill windows against it. The specific thresholds are chosen by the owner in consultation with consumers and are recorded in the contract. This paper does not prescribe

numbers, because appropriate values depend on the source systems and use cases of each organization; the requirement is that thresholds exist, are published, and are monitored.

5.3 Versioning

The contract uses semantic versioning. Adding an optional field is a minor change and is backward compatible. Adding a new enumeration value is also treated as a minor change, but it is backward compatible only for consumers that handle unknown values gracefully; the contract therefore requires consumers to tolerate unrecognized enumeration values, and a new value is announced before it is produced. Removing a field, changing a type, or changing a semantic definition is a major change. A major version is published alongside the previous version for a stated migration window, during which both are produced from the same normalized stream. Consumers declare the version they read, and the platform reports which consumers are still on a deprecated version so that migration can be managed rather than forced.

5.4 Access policy

Access is granted per column class and per purpose. Column classes are analytics grade, which contains coarsened fields safe for aggregate work, and feature grade, which contains full retained precision. Purposes are enumerated in the datasheet and each consumer registration names the purpose under which it reads. Policy is enforced at the storage layer through column and row level controls so that the contract and the enforcement cannot diverge.

6 INTERACTION ANALYTICS AND AI FEATURE GENERATION

6.1 Interaction analytics

Analytics consumers work primarily with aggregates. From the event table they derive an interaction edge table keyed by actor, target, channel, and time bucket, with counts and latency statistics. Standard network measures such as degree, betweenness, and community assignment are then computed over the edge table using established methods [16]. Responsiveness is expressed as latency distributions per group and channel rather than per individual, and reports are released only at a minimum group size defined in policy. Where finer disaggregation is required for a legitimate purpose, aggregate release can be protected with k-anonymity style suppression [18] or with a differentially private mechanism [19]; the choice depends on the sensitivity of the query and is recorded as a policy decision, not left to the analyst.

6.2 AI feature generation

In this design, feature entities are non-person entities such as requests, threads, channels, queues, or approved groups; person level feature generation is outside the scope of the shared product. Feature consumers require record level features, keyed by such entities, that are correct as of a specific point in time. The platform provides a feature definition registry in which each feature is expressed as a declarative transformation over the product, with an entity key, an aggregation window, and a version. Feature values are materialized into an offline store for training and into an online store for inference, following the training and serving consistency pattern established by production machine learning platforms [15]. Table 2 lists example feature families that can be derived from the product without any access to content.

Table 2: Feature families derivable from communication metadata

Feature family	Example features	Typical use
Volume	event volume per interval, active thread count, request volume by channel	demand forecasting, capacity planning
Responsiveness	median reply latency per queue, fraction of requests acknowledged within one hour	service level monitoring, request routing
Breadth	distinct counterparties per group over 30 days, cross-group ratio	interaction structure analysis
Rhythm	share of channel activity by hour of day, share by day of week	demand forecasting, scheduling of capacity
Thread structure	mean thread length, reply depth, reopen rate	issue escalation prediction
Channel mix	email versus chat ratio, call share	channel migration studies

6.3 Point in time correctness

Training data must not leak future information. Because the product records both event time and ingest time, a training set can be built with an as-of semantic: for each training example at time t , the feature value is computed only from events with event_time strictly before t and ingest_time at or before t , unless the feature definition records an explicit historical availability cutoff. The ingest_time condition is what prevents leakage: an event that occurred before t but arrived after t was not available to a system making a decision at t , and it must not be available to the training example either. This requires the product to preserve immutable event and ingest timestamps and to support versioned historical snapshots, which open table formats with transactional logs provide [10]. Time travel is available only within the reproducibility horizon. A training set whose examples fall inside the horizon can be rebuilt exactly; one whose examples fall outside it cannot, and models trained on it should therefore record the product version and horizon in force at training time so that the limitation is visible at audit. The same feature definitions are executed in the online path over a bounded window of recent events, and consistency between the two paths is verified by periodically replaying online inputs through the offline path and comparing outputs [13].

6.4 Feature quality and drift

Each feature carries the schema version of the product it was computed from. When a major contract version changes a semantic definition, features that depend on the affected field are flagged for recomputation and models trained on them are flagged for review. Distribution statistics for each feature are computed on every materialization and compared against a reference profile; deviations beyond a configured tolerance raise an alert to the feature owner. This applies the data validation discipline of Schelter et al. [14] and the ML test score checks of Breck et al. [13] at the feature level.

7 GOVERNANCE AND PRIVACY CONTROLS

The governance model rests on three roles. The product owner is accountable for the contract, the quality objectives, and the datasheet. The data protection function approves purposes, reviews minimization decisions, and holds the re-identification process. Consumers register their use, declare a purpose, and accept the contract. Changes to any of the contract, the purpose list, or the policy pass through a lightweight review in which the three roles participate.

Several controls are worth stating explicitly because they are frequently omitted in ad hoc pipelines. Pseudonymization keys are rotated on a schedule, and each product version records the key epoch, so that deterministic identifier based joins across epochs are not possible without the audited re-identification service. Rotation does not eliminate linkage risk: interaction structure, timing patterns, group membership, and activity volume are quasi-identifiers that can support probabilistic linking across epochs even when pseudonyms change. Rotation should therefore be understood as removing the trivial join path, with the minimum group size, aggregate release policies, and access logging of this section providing the remaining protection. Retention is enforced per field class; latency and volume aggregates may be retained longer than event level records, and the event level retention period is published as the reproducibility horizon. Every read of feature grade columns is logged with the consumer identity and declared purpose, and these logs are reviewed periodically. Individual level reporting is prohibited by policy for analytics consumers, and the minimum group size for aggregate release is stated in the contract. Finally, the datasheet documents known limitations of the data, such as sources that do not report delivery state, so that consumers do not draw conclusions the data cannot support [5].

These controls are designed to satisfy the purpose limitation, data minimization, storage limitation, and accountability principles that appear in the General Data Protection Regulation [20] and in comparable regimes. The paper does not offer legal advice; the intent is to show that the principles can be implemented as platform mechanisms rather than as documentation alone.

8 EVALUATION FRAMEWORK

Because organizations differ in sources, scale, and regulatory setting, we propose an evaluation framework rather than reporting figures from a single deployment. The framework has four dimensions, each with a measurement method that can be automated.

Contract conformance. The fraction of published records that pass schema validation and the semantic checks defined in Section 5.1, computed per product version and per source system.

Freshness and completeness. Lag from event time to availability, measured as a distribution, and the completeness ratio against source counts after the freshness window.

Reproducibility. For a sample of historical feature values, recompute from the product as of the original event time and report the fraction that match exactly. A mismatch may indicate a leak of ingest time into feature logic or a non-idempotent transformation, and each mismatch should be traced to its cause.

Consumer consolidation. The number of independent extraction pipelines retired in favor of the product, and the number of distinct metric definitions reduced to one. This is an organizational measure, but it is the most direct evidence that the product is fulfilling its purpose.

The privacy controls of Sections 4.3 and 7 are evaluated as policy test cases rather than as metrics. Table 3 lists the minimum set; each is executed against the published product on every contract release and any failure blocks the release.

Table 3: Privacy policy test cases executed on every contract release

Test case	Procedure	Expected result
Content exclusion	Scan every published column for free text fields or body fragments	None present
Analytics grade precision	Read event_time and latency_ms through an analytics grade role	Coarsened values only
Instance token exposure	Read source_instance_id through an analytics grade role	Access denied
Minimum group size	Request an aggregate over a group smaller than the published minimum	Suppressed or noised per policy
Epoch join	Attempt an equality join on actor_pid across two key epochs	Zero matches
Horizon enforcement	Query event level records older than the reproducibility horizon	No records returned
Purpose binding	Read feature grade columns under a consumer registration with no approved purpose	Access denied and logged

For quantitative reporting, an implementation should publish, per product version: event throughput, freshness at the median and at the upper percentiles stated in its contract, completeness ratio, conformance ratio, duplicate rate before and after deduplication, feature replay agreement, the number of consumer pipelines retired, and the pass rate of the Table 3 test cases. This paper does not report such figures. The architecture is a synthesis of the cited patterns; it has not been evaluated as an integrated deployment, and we state this plainly rather than present illustrative numbers that could be mistaken for measurements.

An implementation should be considered mature when conformance and reproducibility are close to complete, when freshness and completeness meet the objectives stated in its contract, and when no consumer reads communication metadata from any path other than the product. A worked evaluation on a specific deployment is left as future work; the framework is provided so that such an evaluation can be reported consistently.

9 DISCUSSION AND LIMITATIONS

The design makes several trade-offs. Excluding content limits the range of analytics possible; sentiment, topic, and intent analysis are out of scope by construction. We regard this as the correct default for a shared product, with content analysis handled as a separate, more tightly scoped product where justified. Pseudonymization with rotating keys complicates longitudinal analysis across key epochs; organizations that need long baselines should choose epoch lengths accordingly and route cross-epoch joins through the audited service. As noted in Section 7, rotation removes the deterministic join path but not the possibility of probabilistic linkage through quasi-identifiers, and organizations with a strong adversarial threat model should evaluate the aggregate release policy with that in mind. The reproducibility horizon is a second trade-off: a longer horizon supports longer training windows and more complete backfills, while a shorter one reduces the volume of event level personal data held; the contract makes the choice explicit but cannot make it for the organization. Identity resolution across sources is inherently imperfect, and the quality of the mapping table bounds the quality of every downstream measure; the mapping table therefore deserves the same contract discipline as the product itself.

The framework is technology neutral, but it assumes a durable log for ingestion, a table format with transactional history for the product, and column level access control at the storage layer. Organizations lacking any of these will need to introduce them before the reproducibility and access guarantees can be met. Finally, the paper does not present quantitative results from a deployment. The evaluation framework in Section 8 is intended to make such results comparable across organizations when they are reported.

10 CONCLUSION

Communication metadata is one of the most informative and one of the most sensitive datasets an organization holds. Treating it as a data product, with a canonical model, a versioned contract, privacy controls applied before publication, and a single governed path into both analytics and feature generation, resolves the duplication, inconsistency, and governance problems that arise when each consumer builds its own extraction. The reference architecture and contract presented here can be implemented on common streaming and lakehouse infrastructure and adapted to local regulatory requirements. Future work includes a quantitative evaluation of the framework on a production deployment, extension of the canonical model to additional channels, and formal analysis of the privacy guarantees offered by the aggregate release policies.

STATEMENTS ON COMPLIANCE WITH ETHICAL STANDARDS

Acknowledgments

The author would like to thank industry peers and reviewers for their constructive feedback on earlier drafts of this work.

Disclosure of conflict of interest

The author declares no conflict of interest.

REFERENCES

- [1] D. Sculley et al., "Hidden Technical Debt in Machine Learning Systems," in *Advances in Neural Information Processing Systems* 28, 2015, pp. 2503-2511.
- [2] N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data Lifecycle Challenges in Production Machine Learning: A Survey," *ACM SIGMOD Record*, vol. 47, no. 2, pp. 17-28, 2018.
- [3] I. A. Machado, C. Costa, and M. Y. Santos, "Data Mesh: Concepts and Principles of a Paradigm Shift in Data Architectures," *Procedia Computer Science*, vol. 196, pp. 263-271, 2022.
- [4] Z. Dehghani, *Data Mesh: Delivering Data-Driven Value at Scale*. Sebastopol, CA, USA: O'Reilly Media, 2022.
- [5] T. Gebru et al., "Datasheets for Datasets," *Communications of the ACM*, vol. 64, no. 12, pp. 86-92, 2021.
- [6] J. Kreps, N. Narkhede, and J. Rao, "Kafka: A Distributed Messaging System for Log Processing," in *Proc. NetDB Workshop*, Athens, Greece, 2011.
- [7] T. Akidau et al., "The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing," *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1792-1803, 2015.
- [8] M. Kleppmann, *Designing Data-Intensive Applications*. Sebastopol, CA, USA: O'Reilly Media, 2017.
- [9] M. Armbrust, A. Ghodsi, R. Xin, and M. Zaharia, "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," in *Proc. 11th Conf. Innovative Data Systems Research (CIDR)*, 2021.
- [10] M. Armbrust et al., "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3411-3424, 2020.
- [11] M. Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56-65, 2016.
- [12] R. Kimball and M. Ross, *The Data Warehouse Toolkit*, 3rd ed. Indianapolis, IN, USA: Wiley, 2013.
- [13] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction," in *Proc. IEEE Int. Conf. Big Data*, 2017, pp. 1123-1132.

- [14] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, "Automating Large-Scale Data Quality Verification," *Proc. VLDB Endowment*, vol. 11, no. 12, pp. 1781-1794, 2018.
- [15] D. Baylor et al., "TFX: A TensorFlow-Based Production-Scale Machine Learning Platform," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2017, pp. 1387-1395.
- [16] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge, U.K.: Cambridge University Press, 1994.
- [17] N. Eagle and A. Pentland, "Reality Mining: Sensing Complex Social Systems," *Personal and Ubiquitous Computing*, vol. 10, no. 4, pp. 255-268, 2006.
- [18] L. Sweeney, "k-Anonymity: A Model for Protecting Privacy," *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 5, pp. 557-570, 2002.
- [19] C. Dwork, "Differential Privacy," in *Proc. 33rd Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2006, pp. 1-12.
- [20] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)," *Official Journal of the European Union*, L 119, 2016.
- [21] A. Wider, S. Verma, and A. Akhtar, "Decentralized Data Governance as Part of a Data Mesh Platform: Concepts and Approaches," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2023, pp. 746-754.
- [22] A. Jones, *Driving Data Quality with Data Contracts*. Birmingham, U.K.: Packt Publishing, 2023.
- [23] P. Jain et al., "Analyzing and Comparing Lakehouse Storage Systems," in *Proc. 13th Conf. Innovative Data Systems Research (CIDR)*, 2023.
- [24] J. de la Rua Martinez et al., "The Hopsworks Feature Store for Machine Learning," in *Companion of the 2024 Int. Conf. Management of Data (SIGMOD Companion)*, 2024, pp. 135-147.
- [25] N. Ponomareva et al., "How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy," *Journal of Artificial Intelligence Research*, vol. 77, pp. 1113-1201, 2023.
- [26] J. Mayer, P. Mutchler, and J. C. Mitchell, "Evaluating the Privacy Properties of Telephone Metadata," *Proceedings of the National Academy of Sciences*, vol. 113, no. 20, pp. 5536-5541, 2016.