ONTOLOGY-DRIVEN DATA EXCHANGE SYSTEM

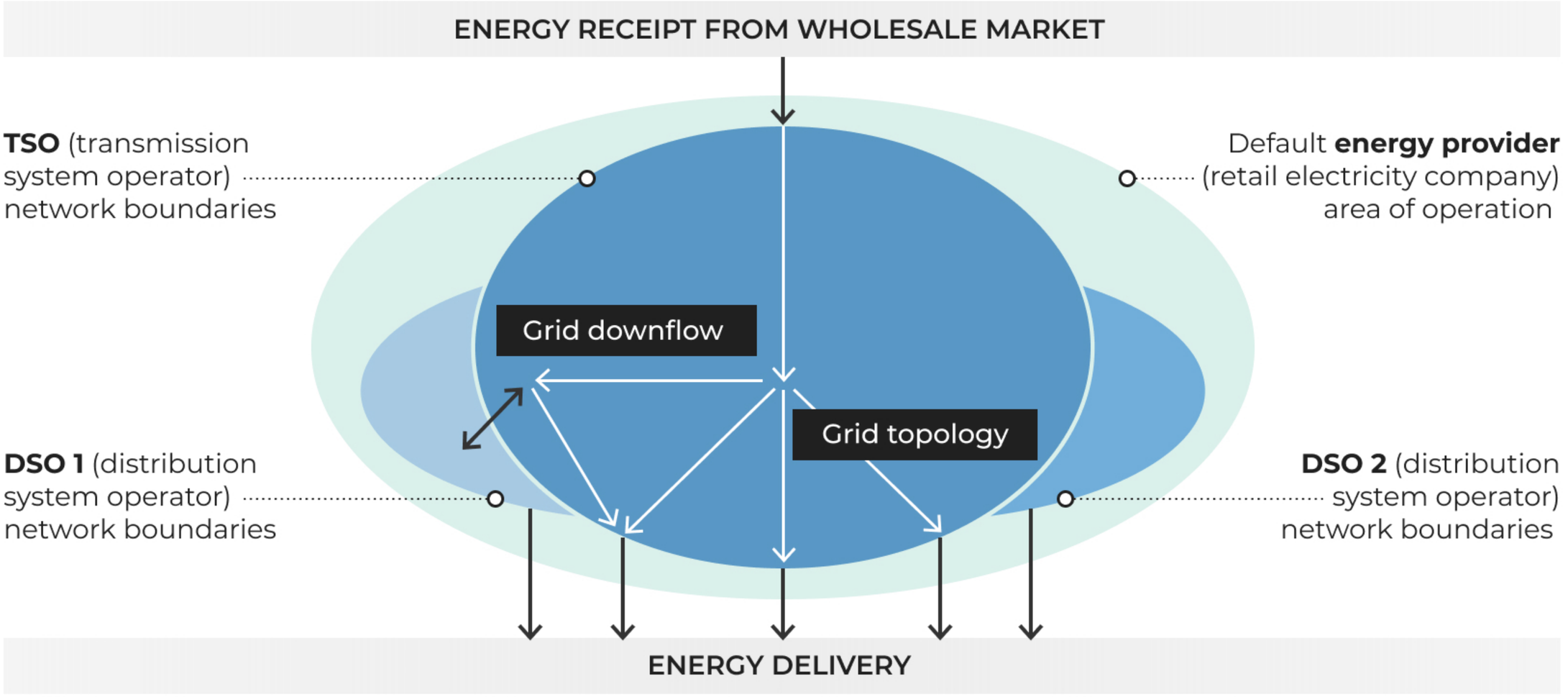
for Grid Companies

Sergey GORSHKOV
TriniData trinidata.com

Victor BELCHENKO
Sigma sigma-it.ru

Sergey ISAEV & Evgeny HLYZOV
DataFabric datafabric.cc

RETAIL ENERGY MARKET STRUCTURE



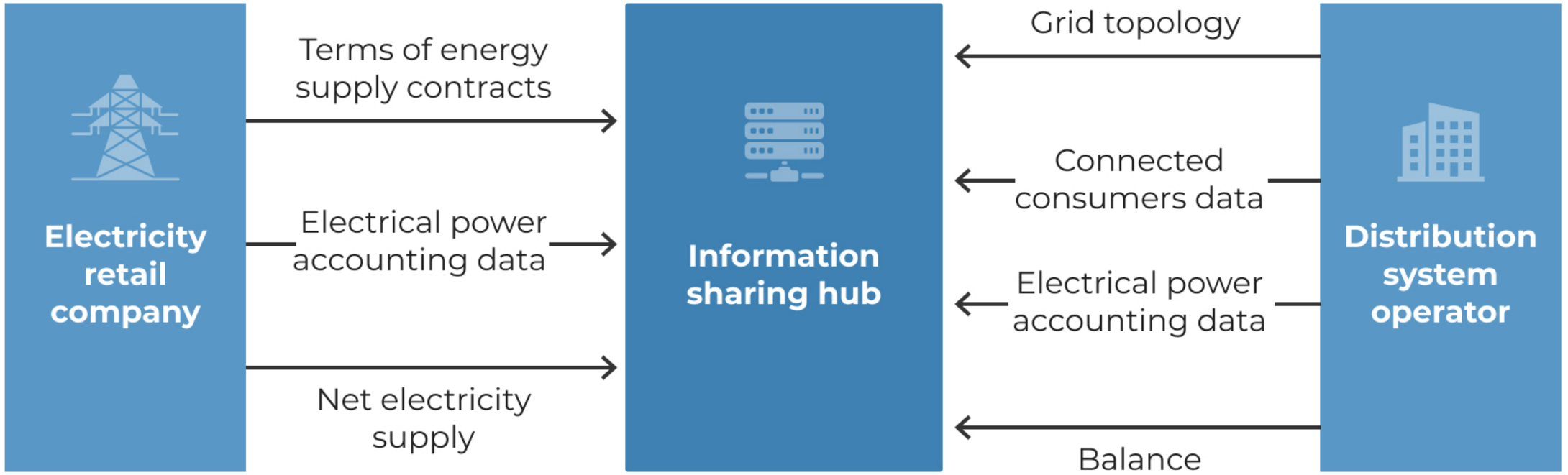
GOALS OF RETAIL ENERGY MARKET INFORMATION EXCHANGE

- **Default energy provider (retail electricity company) buys electrical energy** on the wholesale market and provides retail consumers with it.
- Transmission system operators (TSO) and Distribution system operators (DSO) **provide energy delivery services**.
- **Grid companies compensate transmission losses** by purchasing the necessary volumes of electrical energy from the upstream energy provider.

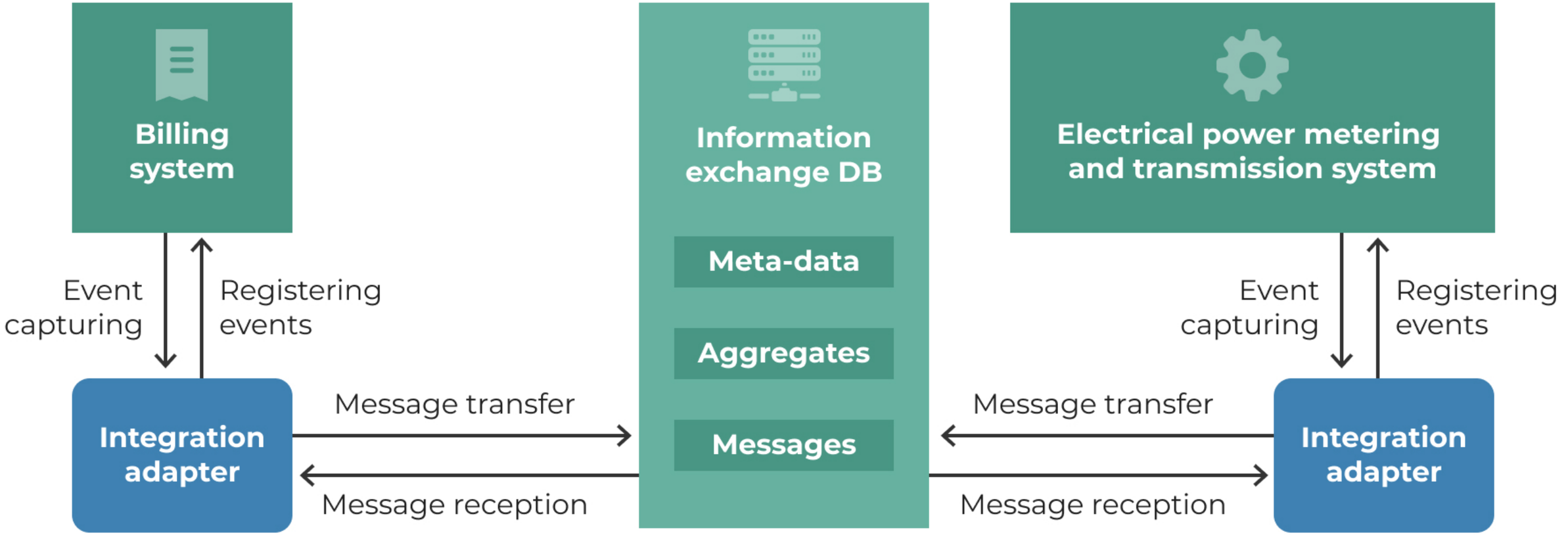
COMPLIANCE OF ENERGY AMOUNTS

1. Amount of energy delivery to consumers => Amount of energy transmission.
2. Amount of losses => Liability of grid operator to reimburse losses to retail companies.

INFORMATION SHARING SYSTEM OF RETAIL ENERGY MARKET



INFORMATION SHARING SYSTEM OF RETAIL ENERGY MARKET



MANAGEMENT BY MEANS OF SEMANTIC MODEL

CHALLENGES

- High degree of **complexity of a subject area** (hundreds of terms).
- **Subject area is changing dynamically** in Russia (3 regulatory acts, essential for the retail energy market model, were adapted in 2018).
- **Diversity of software** (information exchange participants are using software developed by various developers which had their own views of the subject area).
- **Business dynamics** (information exchange system implementation changes business processes iteratively).

ONTOLOGY-DRIVEN SOFTWARE

APPROACHES

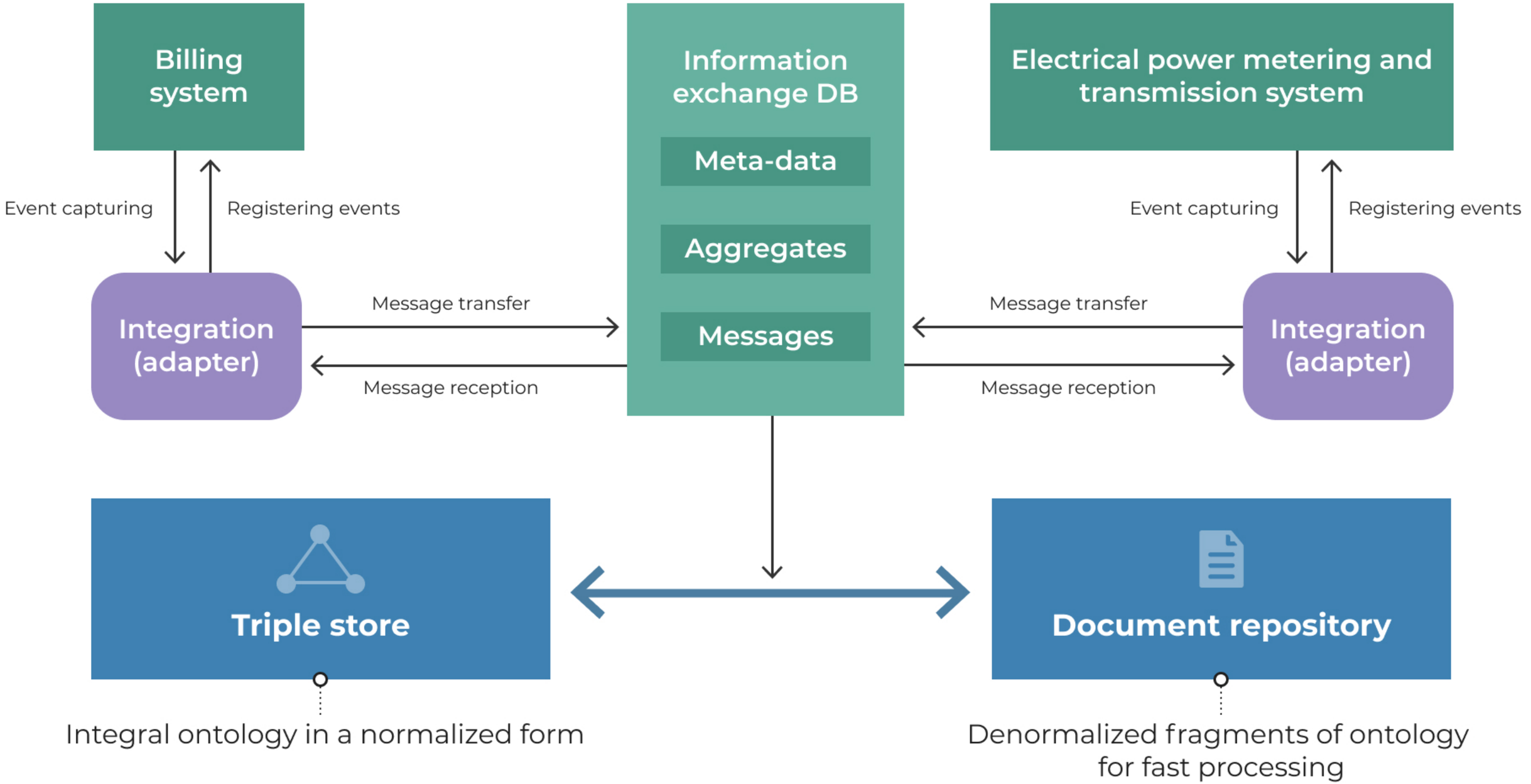
- The ontology systematizes key concepts used to describe subject area entities and business processes milestones.
- The ontology sets data structure (classes of the entities, their relations and attributes).
- The inference rules represent the logic of data messages validation and tranformation.

PERFORMANCE ISSUES (CONDITIONS)

- Settlement day closing: several millions of events per 1-2 days in a bursty traffic. Single event reaction is no longer than 20 minutes.
- The data and their structure (model) should be placed in the triple store in order to use inference rules.
- Rules execution presupposes a complex graph traversal, which is a time-consuming processing.

PERFORMANCE ISSUES (SOLUTIONS)

- Speeding-up the access to triplestore (**indexing and caching**).
- Fragments of a semantic graph that are involved in high-speed computations are stored as **labeled graphs** which allows fast data transformation.
- Recording data representations as **JSON-LD documents** allows fast access for homogenous data visualization and analysis.
- As a result, a hybrid data storage architecture is implemented.


Billing system
Information exchange DB
Meta-data
Aggregates
Messages
Electrical power metering and transmission system
Event capturing
Registering events
Event capturing
Registering events
Integration (adapter)
Message transfer
Message reception
Message transfer
Message reception
Integration (adapter)
Triple store
Document repository
Integral ontology in a normalized form
Denormalized fragments of ontology for fast processing

MODEL OF EVENT STREAM PROCESSING

REGISTERING A NEW DELIVERY POINT

Identification of connection configuration elements	Record of configuration and condition of metering system	Record of a calculation scheme

RECORD OF A METER READINGS

Identification of connection configuration elements	Record of configuration and condition of metering system	Updating metering history

UPDATING READINGS HISTORY

Delivery point identification	Calculation scheme configuration check	Binding of consumption/ transmission volumes to calculation scheme	Meter readings and supply volumes consistency check

HIGH-LEVEL MODEL STRUCTURE

Energy market participant

- Organisation
- Organisational unit
- Person (subscriber)

Physical objects

- Substation
- Meter
- Register

Functional object (role)

- Delivery point
- Service point
- Usage location

Event

- Readings acquisition
- Meter installation
- Connection to the grid

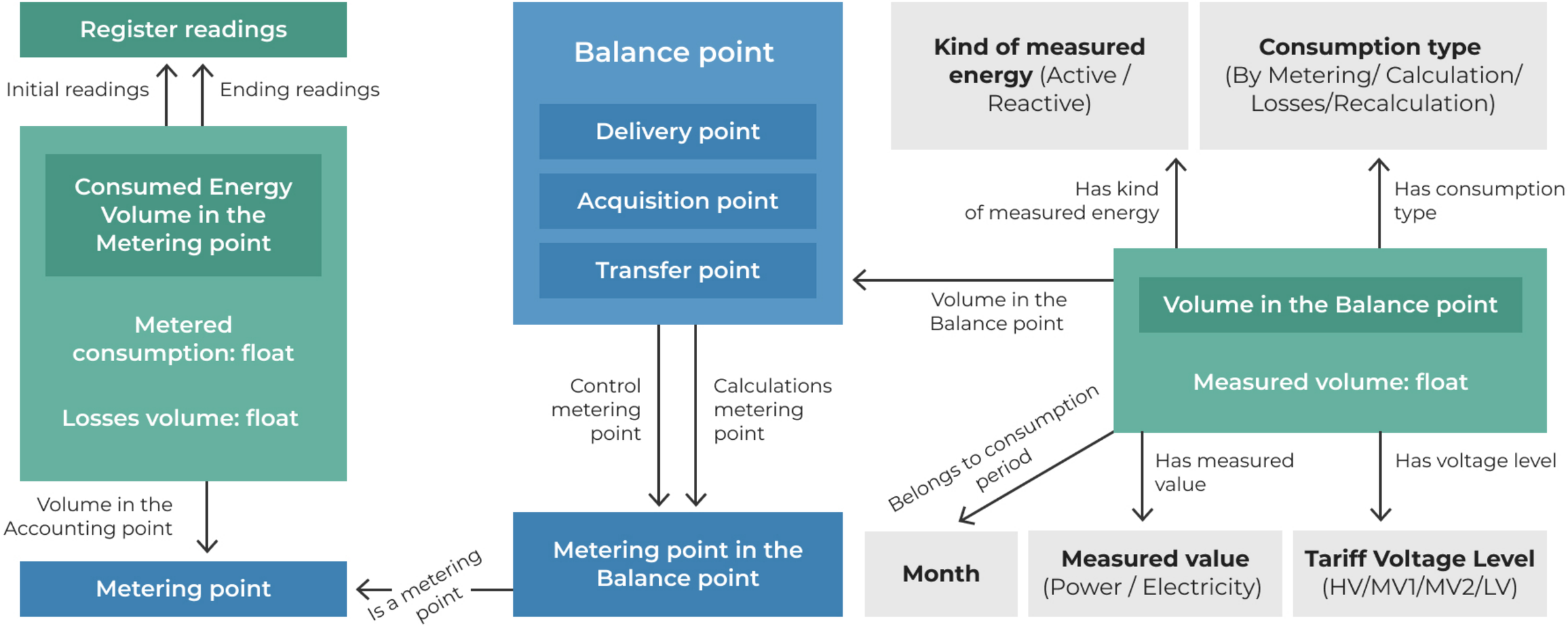
State

- Operation responsibility
- Balance
- Accounting scheme

Document

- Contract
- Meter installation act
- Unaccounted usage act

MODEL OF THE CONSUMED ENERGY VOLUME



CIM, EBIX AND OUR MODEL

CIM (Common Information Model) standards include:

- IEC 61970-301 contains basic definitions (mostly describing equipment and network structure)
- IEC 61968-11 covers energy distribution, including metering, billing, planning, etc.
- IEC 62325-301 covers market operations

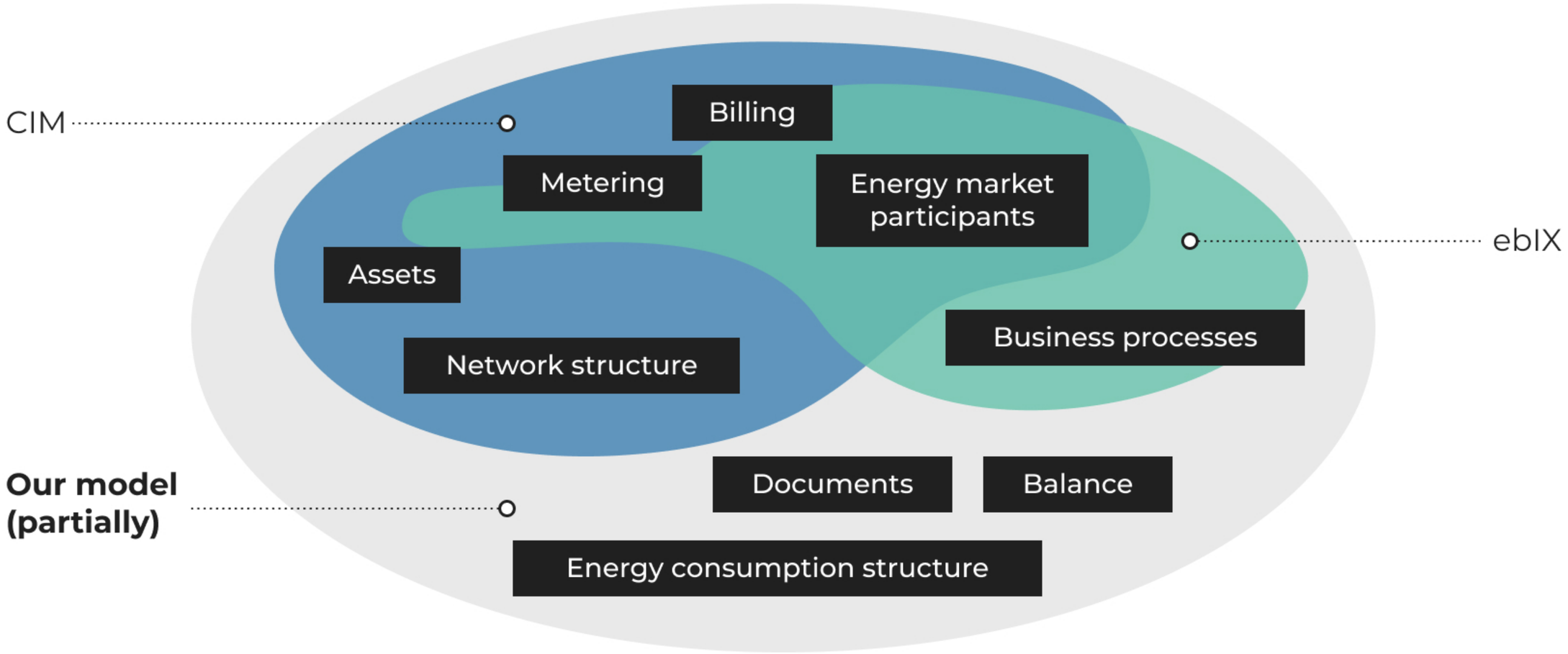
ebIX (European forum for energy Business Information eXchange) is an organization offering a Harmonized Role Model and a set of models describing Business requirements, Process models and Information models for several processes, such as “Change of supplier”, “End of supply”, etc.

Both CIM and ebIX models have UML representations, and CIM has a RDF projection. Their classes, attributes and definitions may be used as a base for ontology development for the energy market.

CIM is widely used by the Russian grid companies.

MODELS COVERAGE

COULD WE USE ONE OF THESE MODELS AND NOT “REINVENT THE WHEEL”?



CONCEPTUAL DEFINITIONS

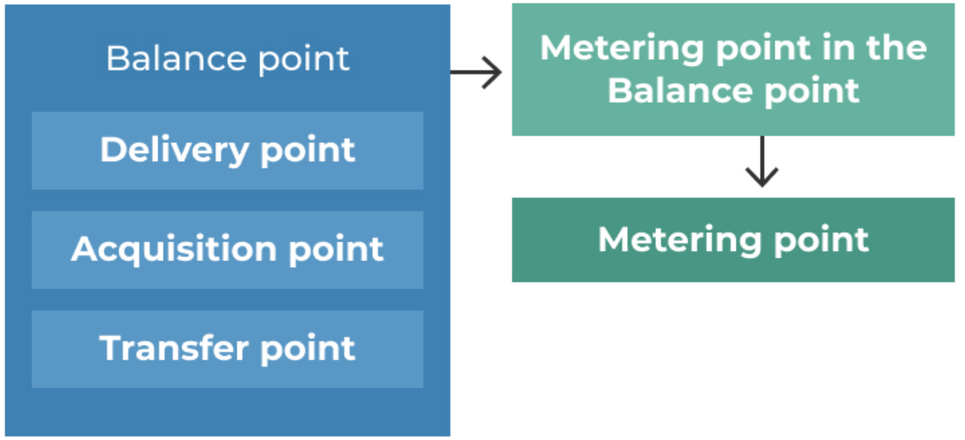
Balance point – a superclass of points used in the balance model

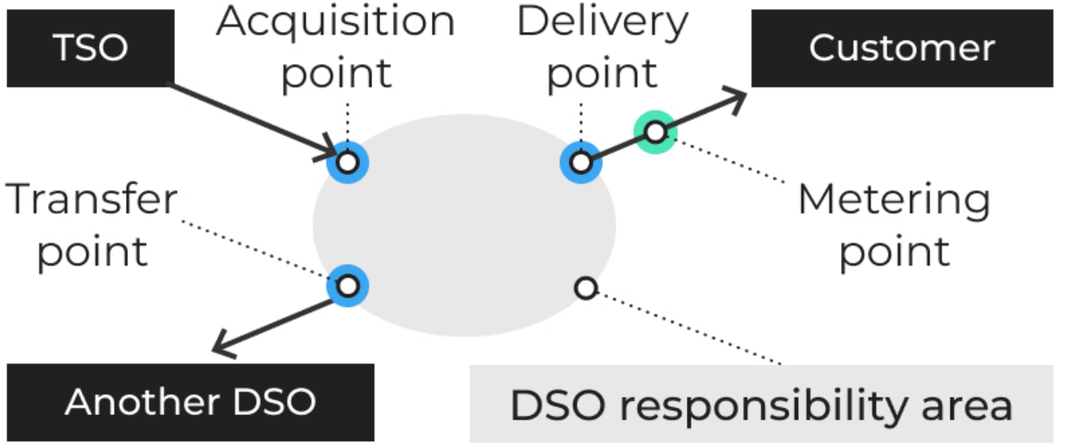
Delivery point – a point of energy supply contract fulfillment *(Russian govt. decree N 442 of 2012-05-04)*

Acquisition point – a point of energy acquisition from TSO or generation facility

Transfer point – a point of energy transfer to other DSOs or their customers

Metering point – a point where energy transfer metering is performed



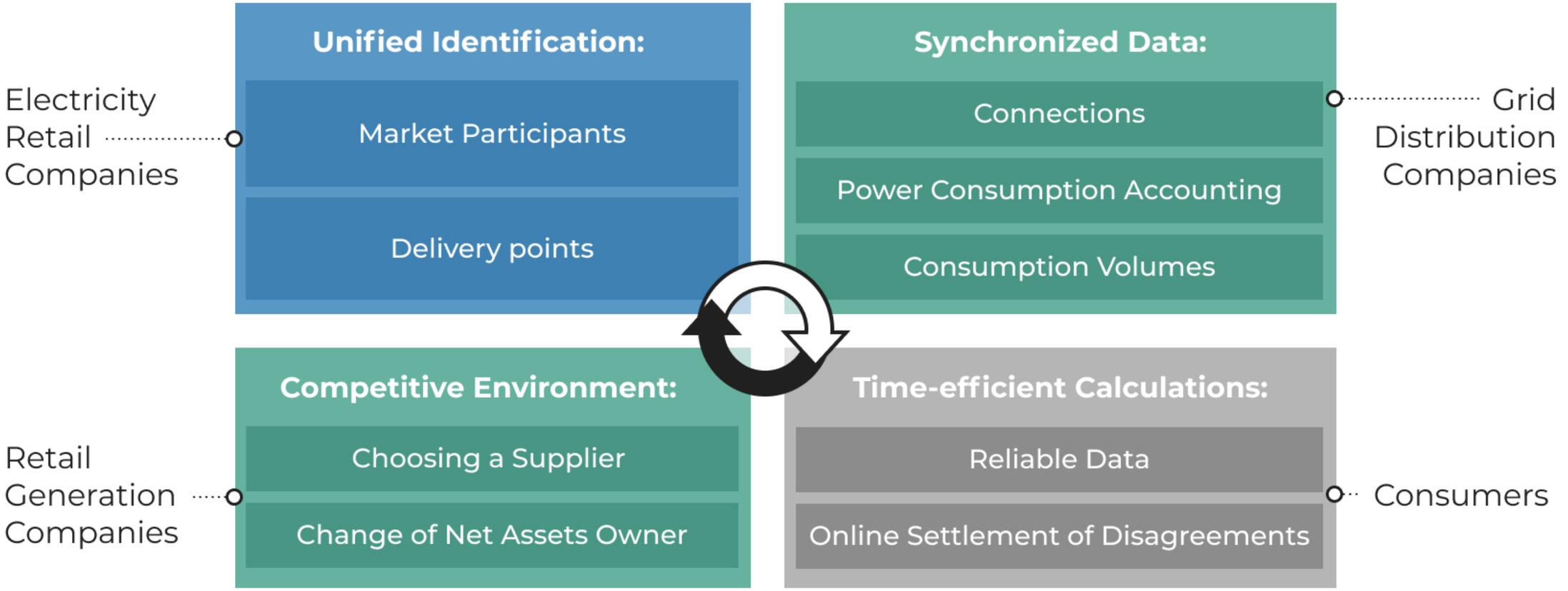


FORMALIZATION OF THE BALANCE FORM STRUCTURE

	Aggregation method		Sum	
	Attribute identifier		Volume according to Meters Reading	Measured Volume
	Class and query condition		Volume at the Metering point	Volume at the Metering point having Voltage level = Hight Voltage
Class and query condition			FACT	
	#	INDICATORS	METERED VOLUME	HV
	1	Input to the TSO grid		
	1.1	Input to the TSO grid from XYZ supplier		
Objects of the “**Volume in the Balance point**” class, related with the **Balance Point**, wich has “**Contained in the Balance group**” relation with the balance group, which has the “**Direction” property** = “**Ingoing**” and is reference by the property “**Included in the grid responsibility area**” from the object “**ABC branch**” belonging to the “**XYZ supplier**”	1.1.1	Input to the TSO grid from XYZ supplier in the ABC branch		
	2	Total power output		

ADVANTAGES FOR RETAIL ENERGY MARKETS

1. Rapid and reliable market data exchange within the common information context



ADVANTAGES FOR RETAIL ENERGY MARKETS

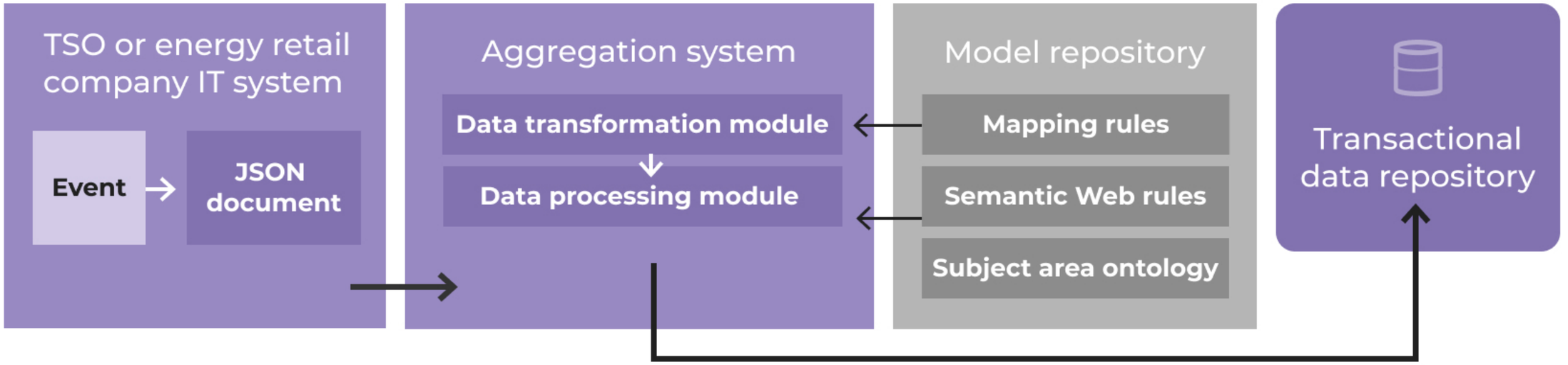
2. Energy balances coherence of all the retail market participants

Electric energy purchase and sales to consumers	Power receipt and power delivery to consumers and to allied grid companies	Power reception and power transfer within the segment of a grid
Balance of retail market in a region	**Balance of local grid company**	**Balance on energy distribution centers**
Balance of energy supplier	**Balance of grid distribution company**	**Balance on grid assets**

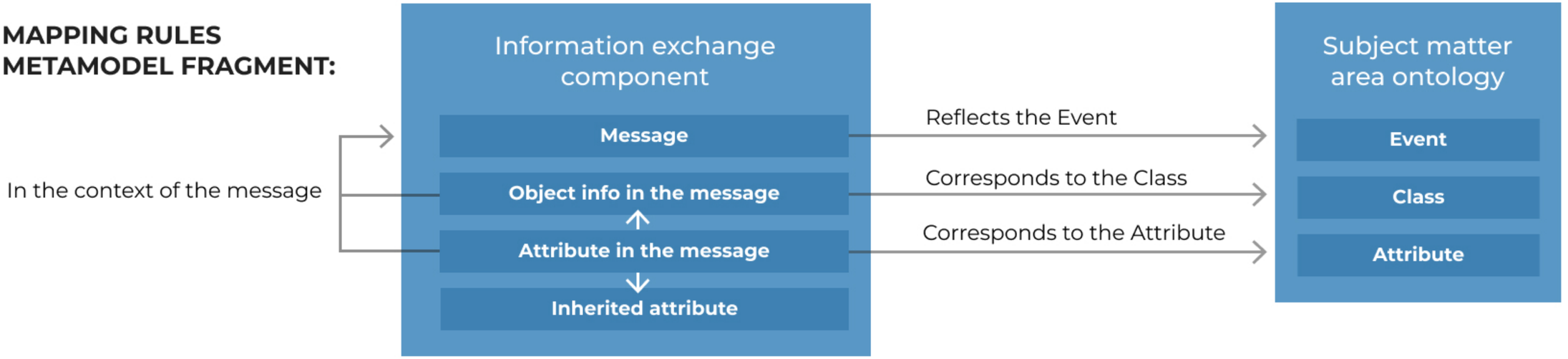
Software Implementation

VERSION 1

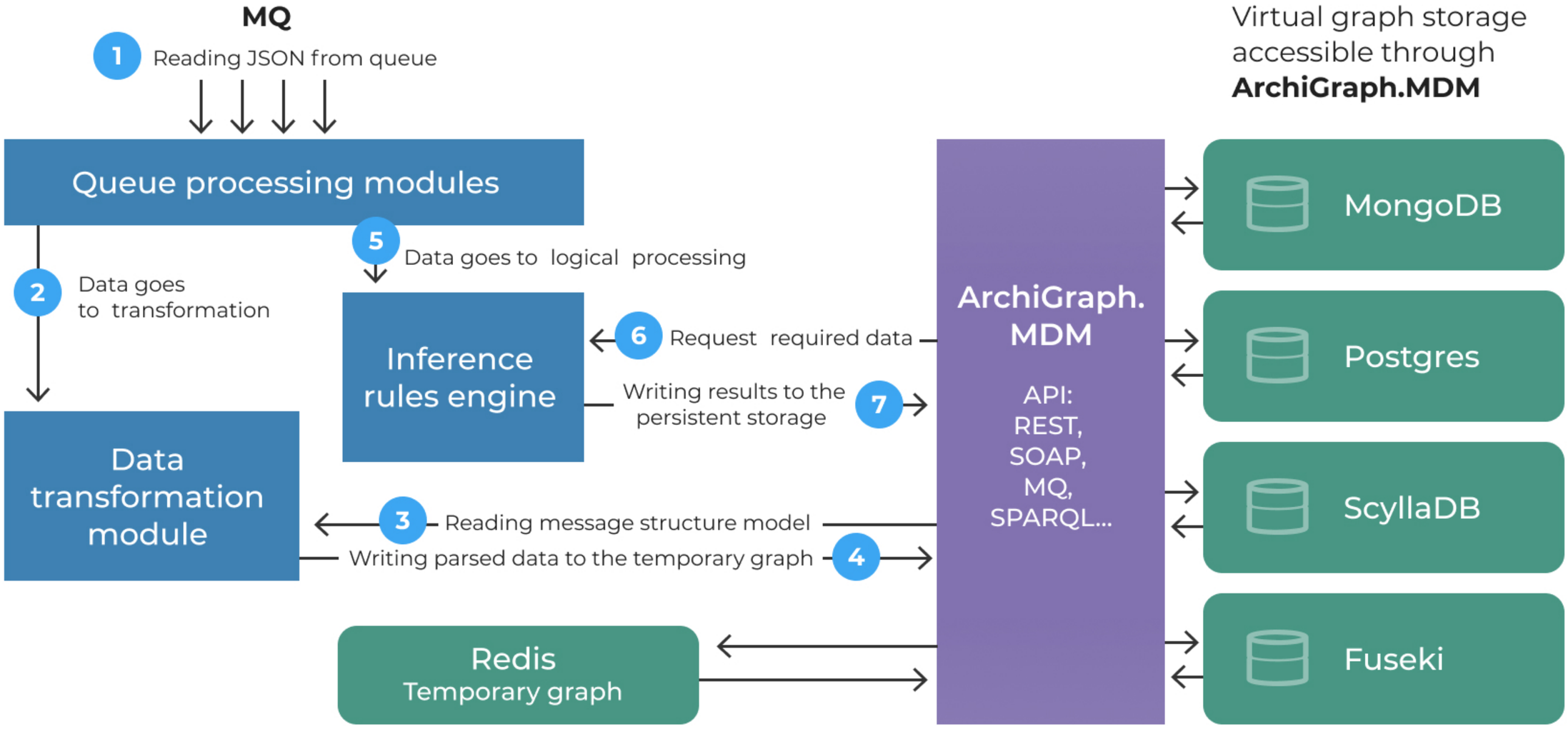
DATA PROCESSING PIPELINE



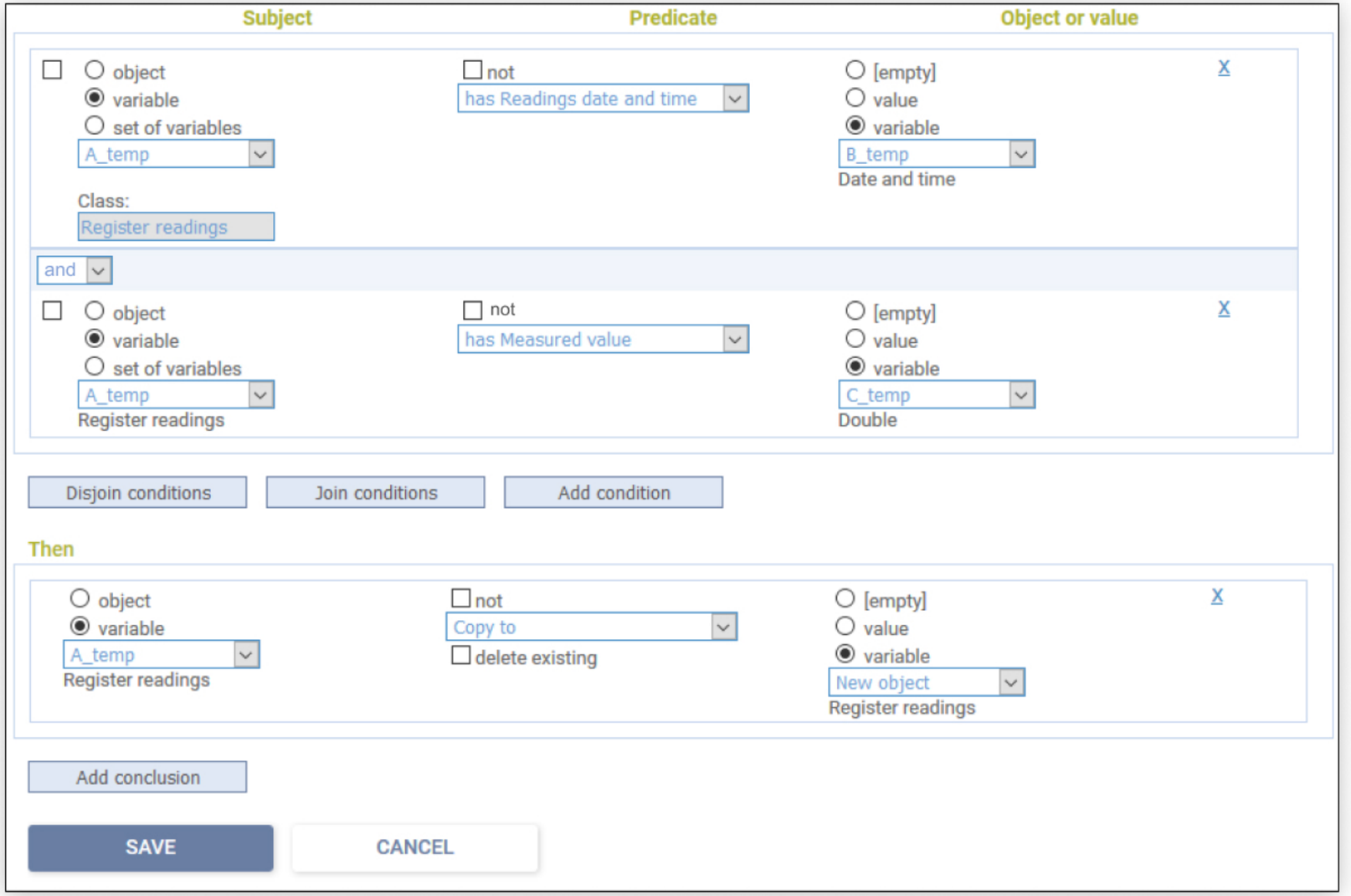
MAPPING RULES METAMODEL FRAGMENT:



DATA TRANSFORMATION AND PROCESSING



DATA TRANSFORMATION RULES EDITOR



Software Implementation

VERSION 2

KEY RUNTIME REQUIREMENTS

- **Fault tolerance**. The service should remain consistent even after a crash of one node without a data loss
- **Horizontal scalability**. The service should increase its throughput after adding a new node to the cluster
- The service should be able to process at least **2 million messages** that arrive irregularly within two days. The maximum latency for processing each message is 20 minutes

KEY DESIGN REQUIREMENTS

- The data is to be represented in RDF form, complying with the ontology.
- The rules are to be stored in a triplestore DB and can be modified by the analytics.
- It is possible to use intermediate stores to improve the data throughput but eventually the state should be written into a triplestore DB.
- All the new messages should be stored in a backup storage.
- The information about how the message is processed by the modules should be stored in the metadata repository.

ARCHITECTURAL CHOICES. INFRASTRUCTURE

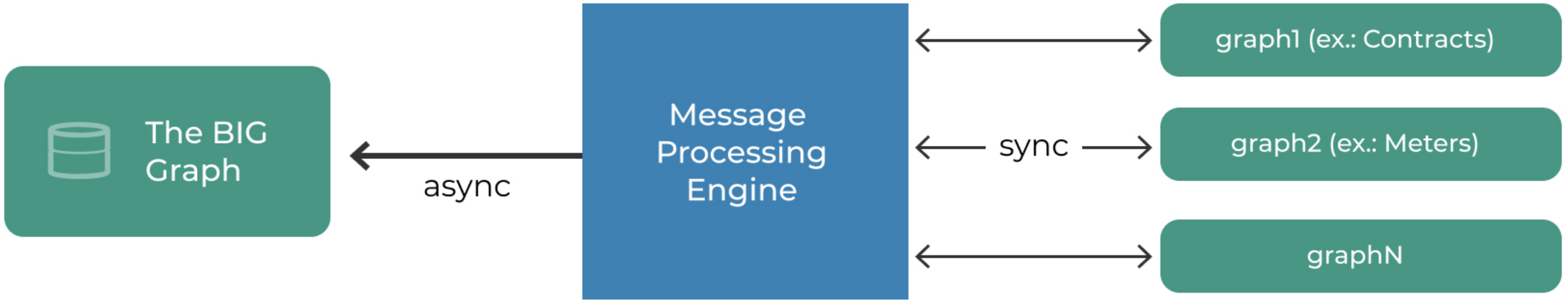
Two viable options:

1. Kafka + Kafka Stream
2. RabbitMQ + Apache Beam (with Apache Flink as a backend)

We choose **Kafka** because of a relatively small number of the required components, better out-of-box scalability and durability.

ARCHITECTURAL CHOICES. STORAGES – 1

To be able to process the messages really fast we decide not to use one BIG graph, but multiple mini-graphs and store them in a KV database (key = graph IRI, value = serialized Jena model).



We use ScyllaDB as a KV storage for its amazing performance and scalability.

ARCHITECTURAL CHOICES. STORAGES – 2

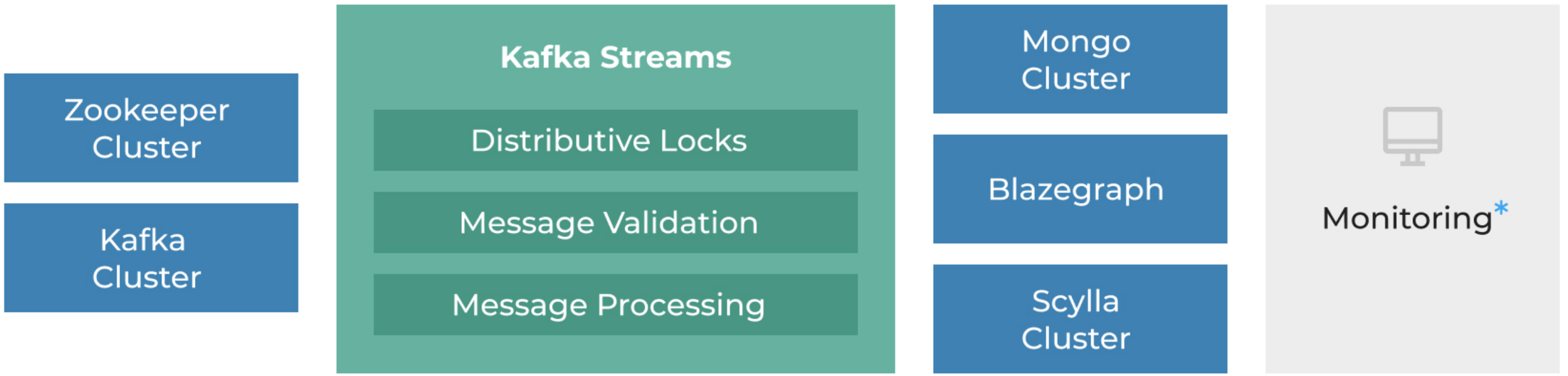
The choice of a triplestore was a bit tricky. We intended to find a solution which can operate in a cluster mode. Such a solution should be used with Apache Rya and our own propriate triplestore* with ScyllaDB as a backend.

Finally we decided to use a standalone **Blazegraph** due to its better performance. It implies single point of processing errors, but it can be tolerated as far as it doesn't stop message processing.

We use **MongoDB** as a metadata repository and a backup storage, mainly because of a positive experience with it.

* https://github.com/DataFabricRus/scylla-rdf, written in Kotlin

ARCHITECTURAL OVERVIEW



* Prometheus + Grafana. We collect and display stats from Kafka, ScyllaDB, Mongo and application metrics

PERFORMANCE

Hardware (Google Cloud):

- Infrastructure: 3 x n1-standard-8 (8 vCPUs, 30 GB memory)
- Scylla: 3 x n1-standard-2 (2 vCPUs, 7.5 GB memory)
- Streams: 3 x n1-standard-4 (4 CPU, 15 GB memory)

We reached 500 messages (meter readings), ~7 rules applied for each message per second (~1mb/s) by means of this hardware. The time of calculation of all the rules varied from 20ms to 100ms depending on the type of a message. It took about 70 minutes to process 2 million messages which fulfilled the requirements.

The speed of processing depends on the complexity of rules and the amount of graphs we need to process a message. But the production cluster will be much more effective, so we are ready to challenge an increasing complexity and stricter performance requirements.

RULE ENGINE FUNCTIONS AND ALGORITHMS

Rule Engine functions:

- Extract graph names from the processed messages
  Names are used: to prevent simultaneous data modification (locking), to load data relevant for transformation from Record Storage, to write transformation results to Record Storage and triplestore.
- Check message data and transform it into the form conforming the domain model

Rule Engine algorithm:

1. Initialize Rule Engine (at the start of the system)

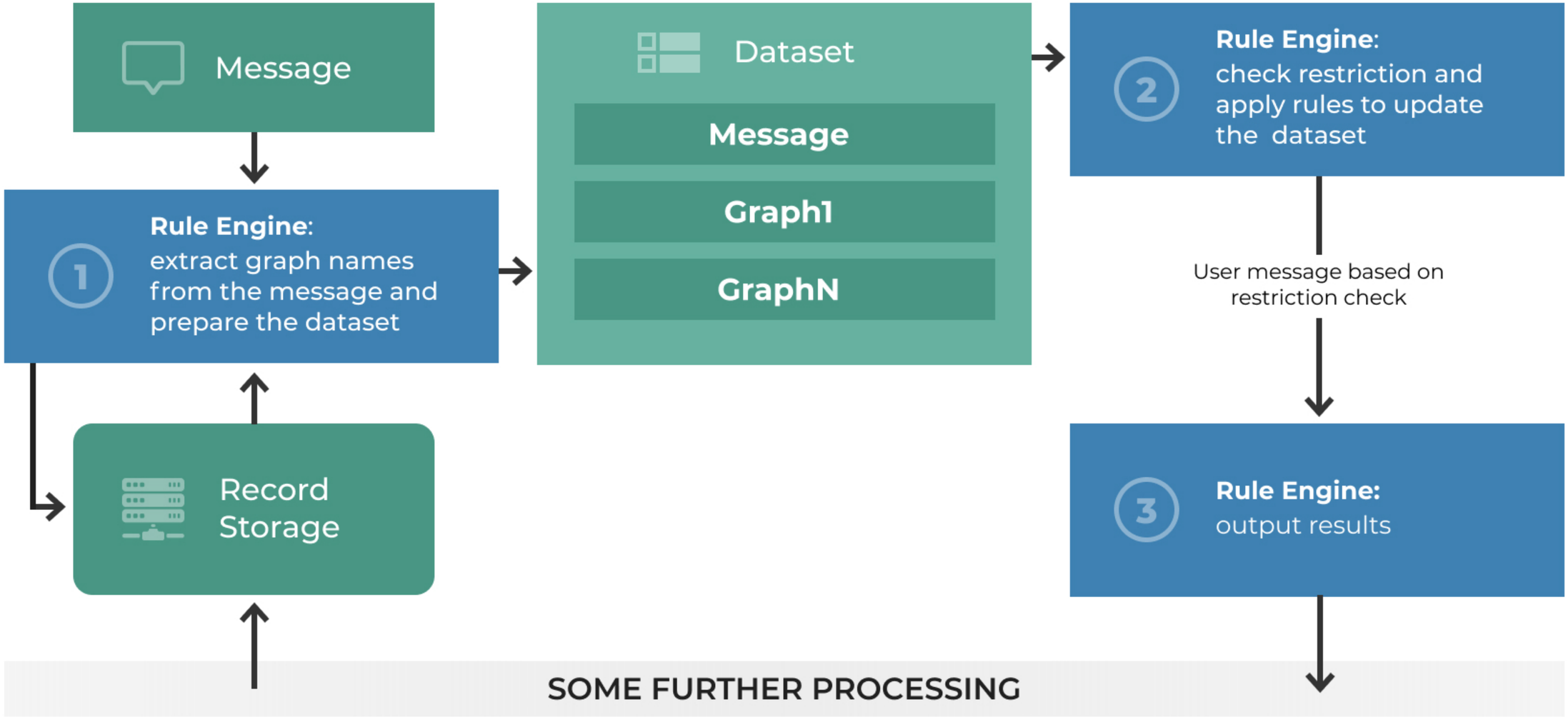
2. Prepare dataset containing data relevant for the transformation (including a message itself)

3. Check if the received message meets the user-defined restrictions

4. Update dataset based on the message through a transformation rules application

5. Output updated dataset

RULE ENGINE ALGORITHM (STEPS 2-5)



SPECIFICATION OF RULES AND RULE ENGINE INITIALIZATION

RULE ONTOLOGY

Domain Ontology Event Class
(described in the message)

SPARQL-queries to extract graph names

SPARQL-queries to check user-defined restrictions

- violation category
- user message template
- SPARQL SELECT query

SPARQL-queries to update the in-memory dataset

- restrictions
- SPARQL INSERT query

RULE ENGINE

Event Rule Set
(generated on Rule Engine Initialization)

Jena (including parameters and values from the message) queries