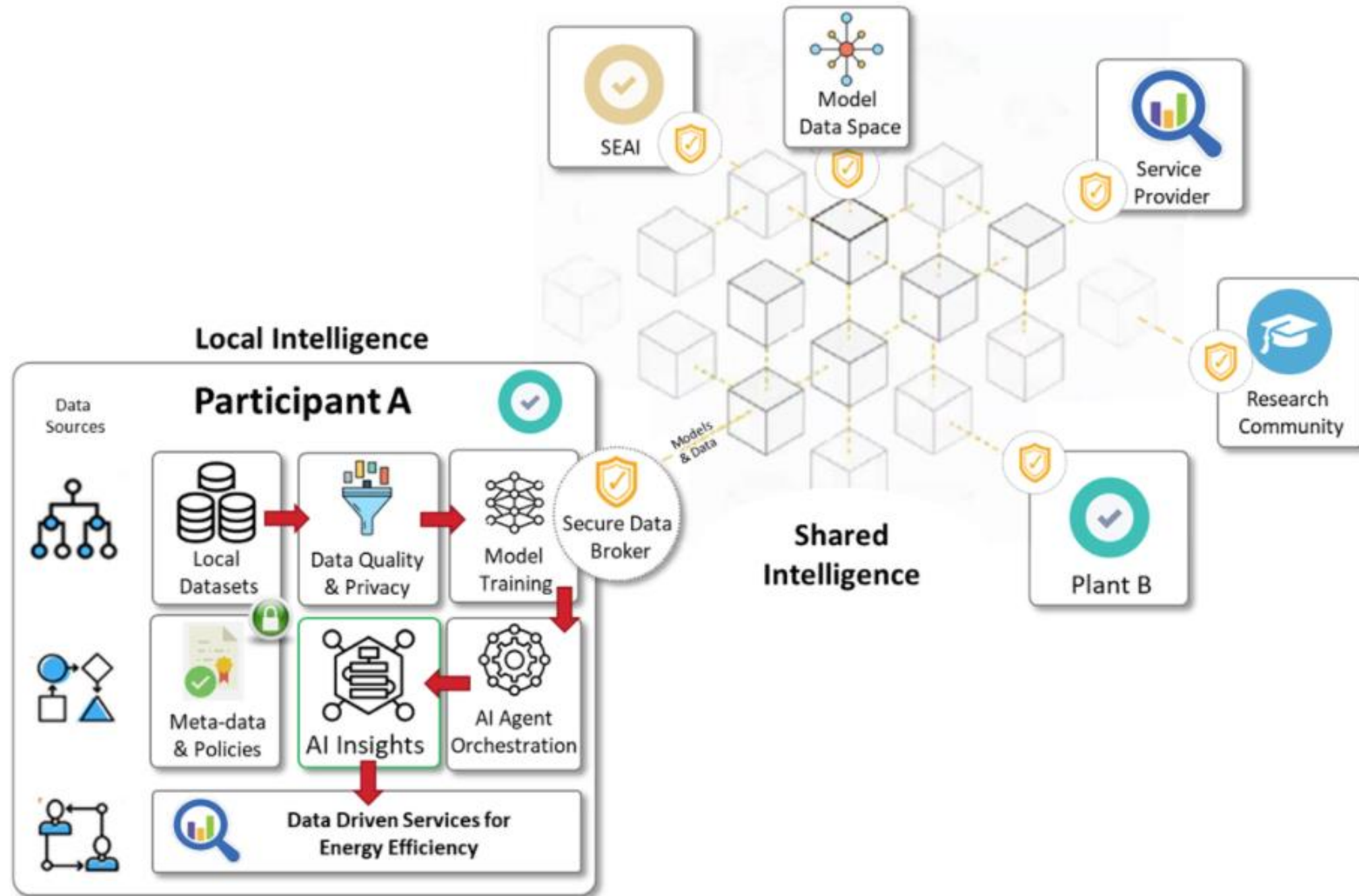# A Semantic Interoperability Toolkit for Sharing Energy Data and Models in a Manufacturing Data Space

Authors: Tharindu Ranathunga, Sourabh Bharti, Alan McGibney

Munster Technological University, Ireland

CORDS Idea

COllaboRative Data Space for distributed intelligence supporting energy performance management

Challenges Addressed - Breaking Data Silos

- **Disparate Energy Data Sources**
- **Data Consolidation**
- **Extracting Meta Data**
- **Local intelligence to shared intelligence**
- Semantic Interoperability
  - **Energy Data**
  - **ML Model Reusability**
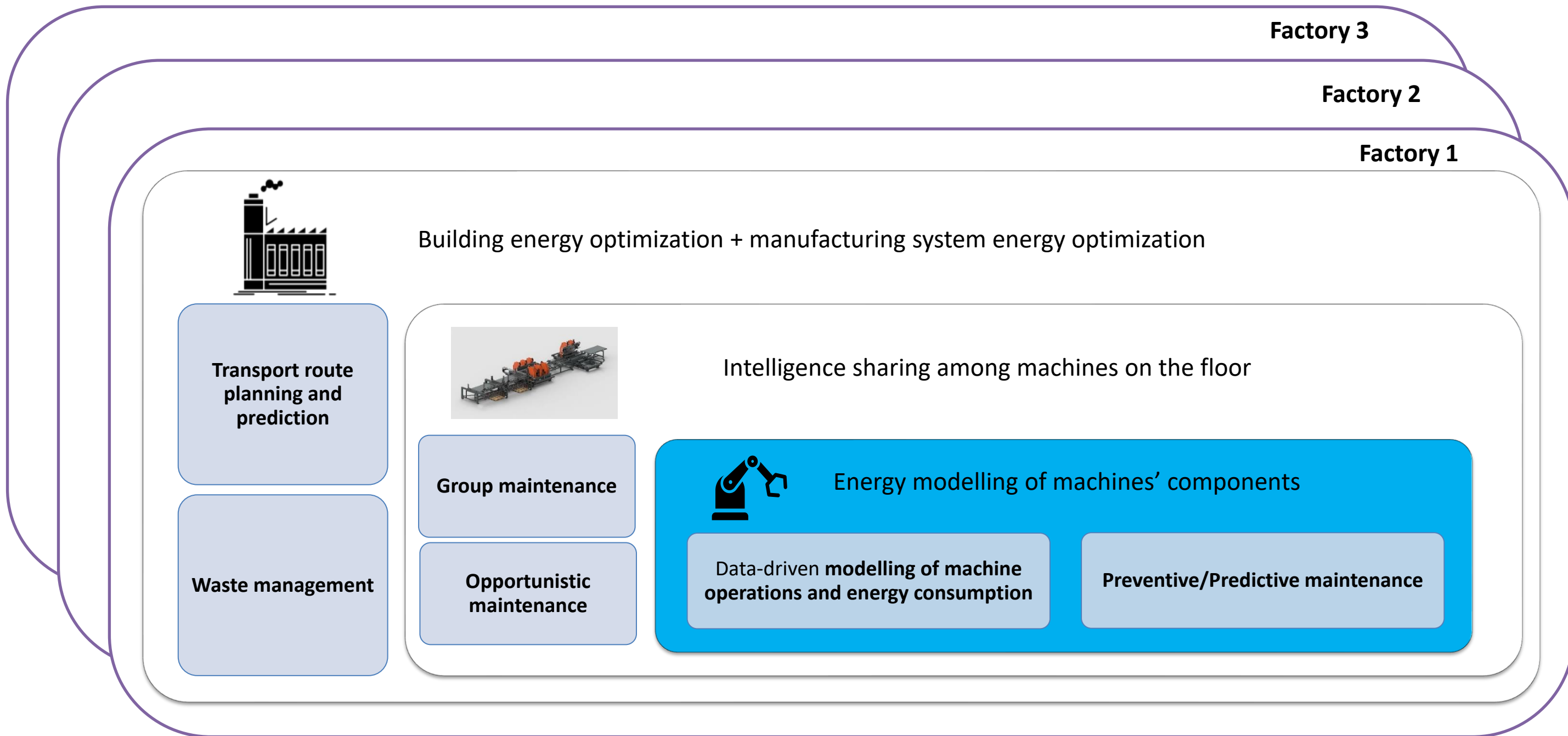
Project Impact
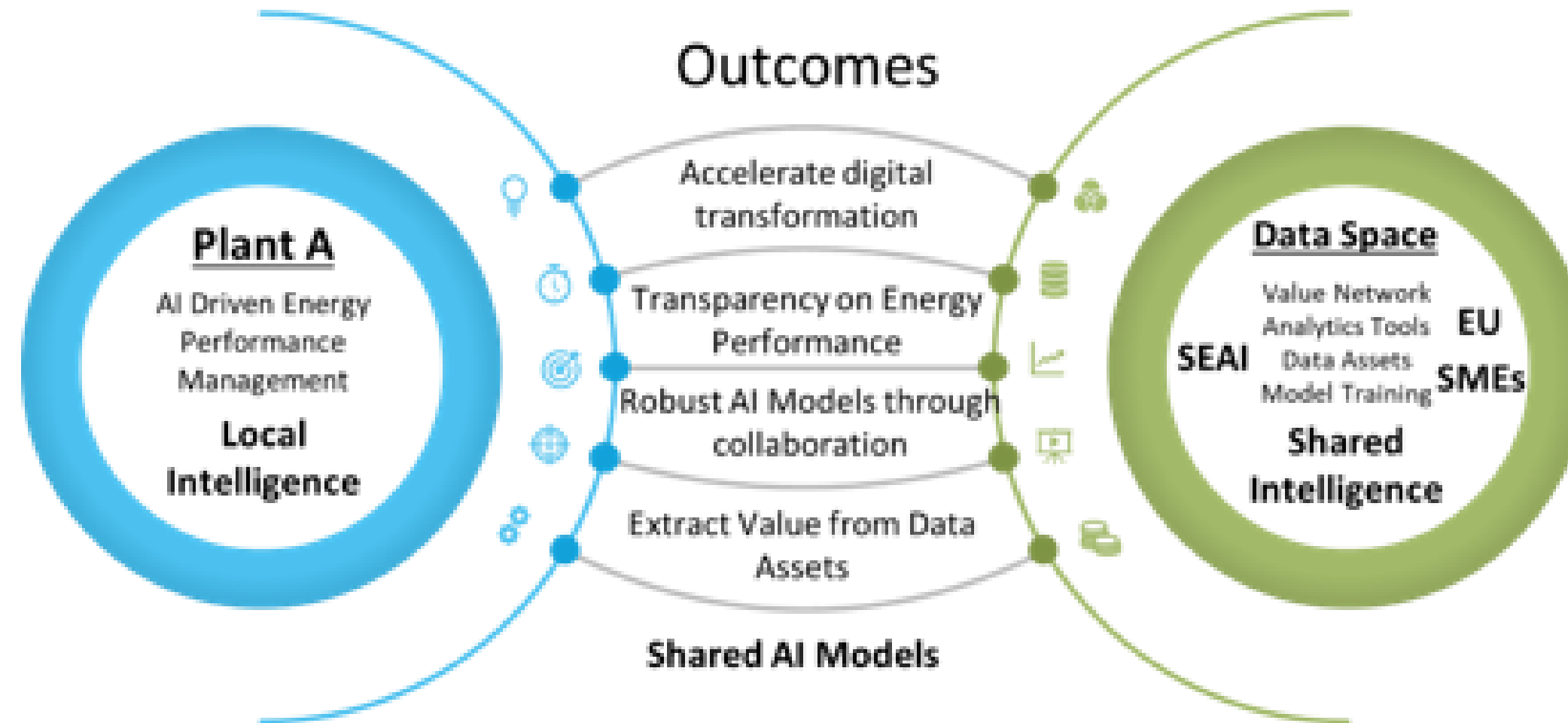
Strategic Goal 1:
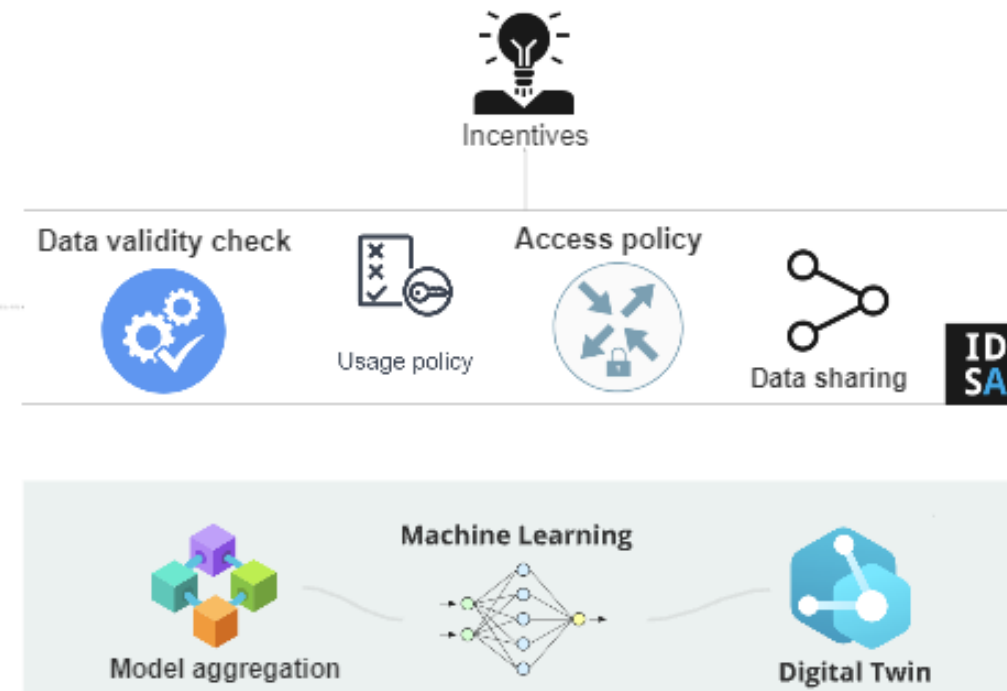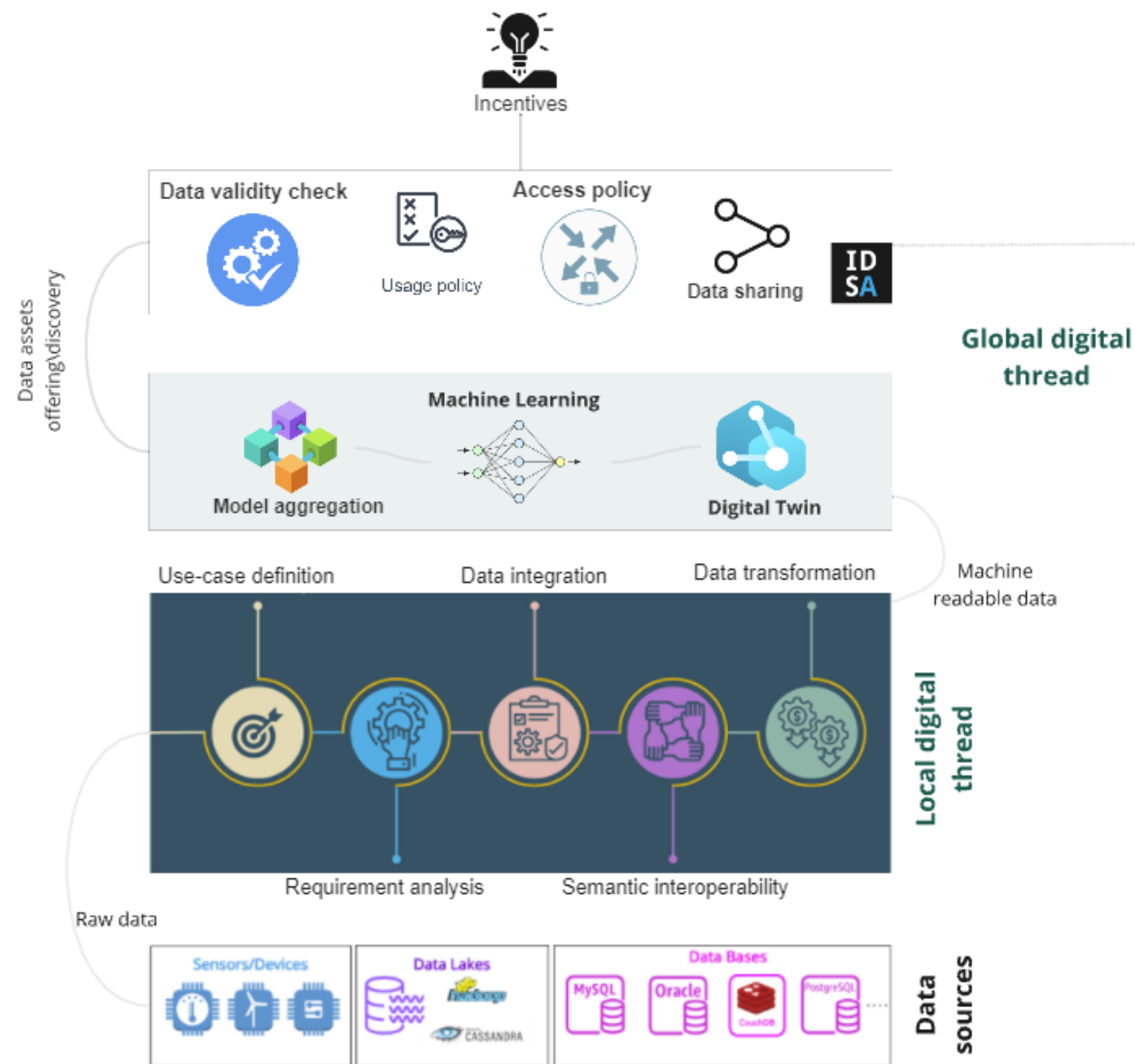Develop and open collaborative data space.

Strategic Goal 2:
Demonstrate the value of 'untapped' data to unlock potential energy savings
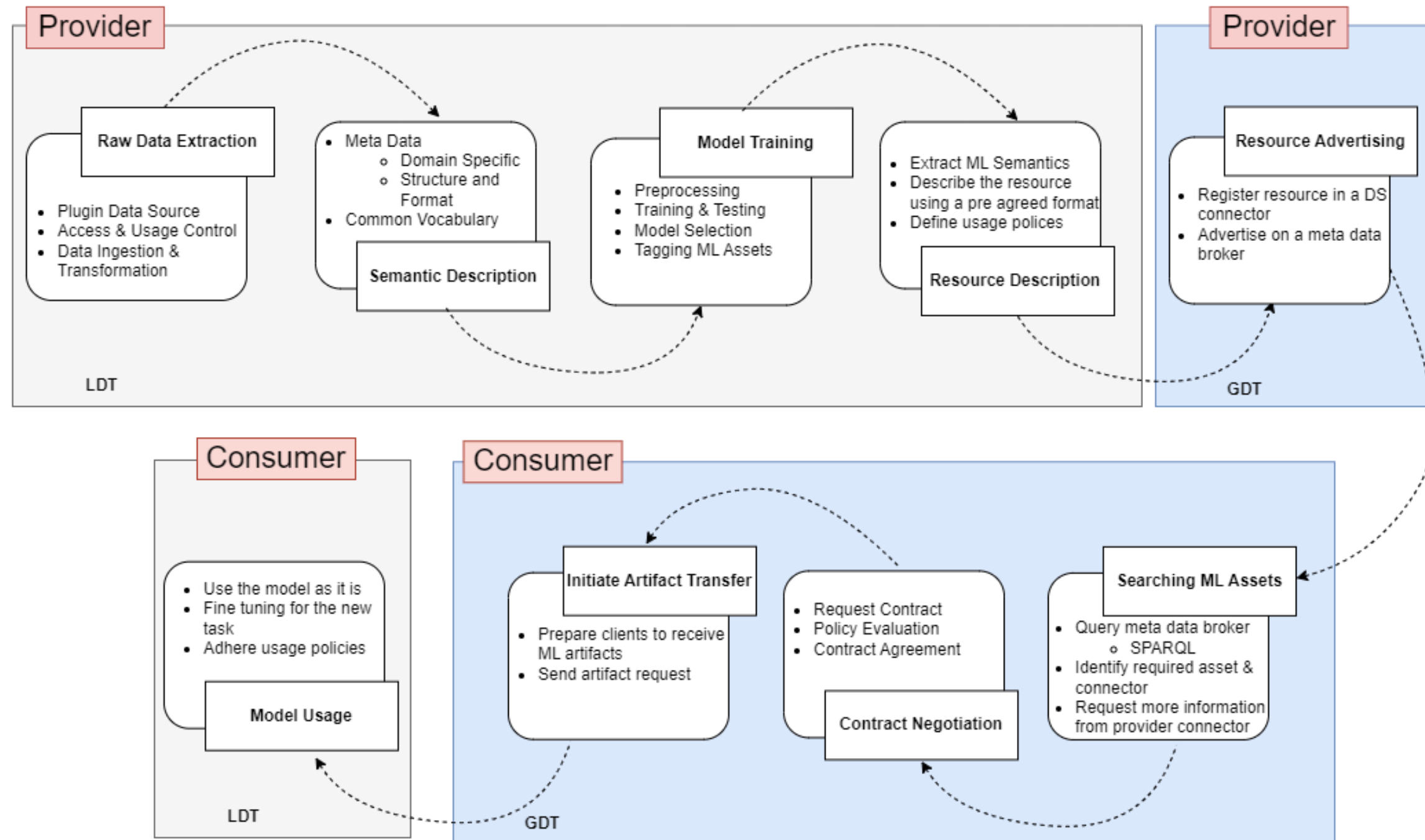
Enhancing decision making

Solution: Digital Threads

Local Digital Thread
- Consolidate Data Sources
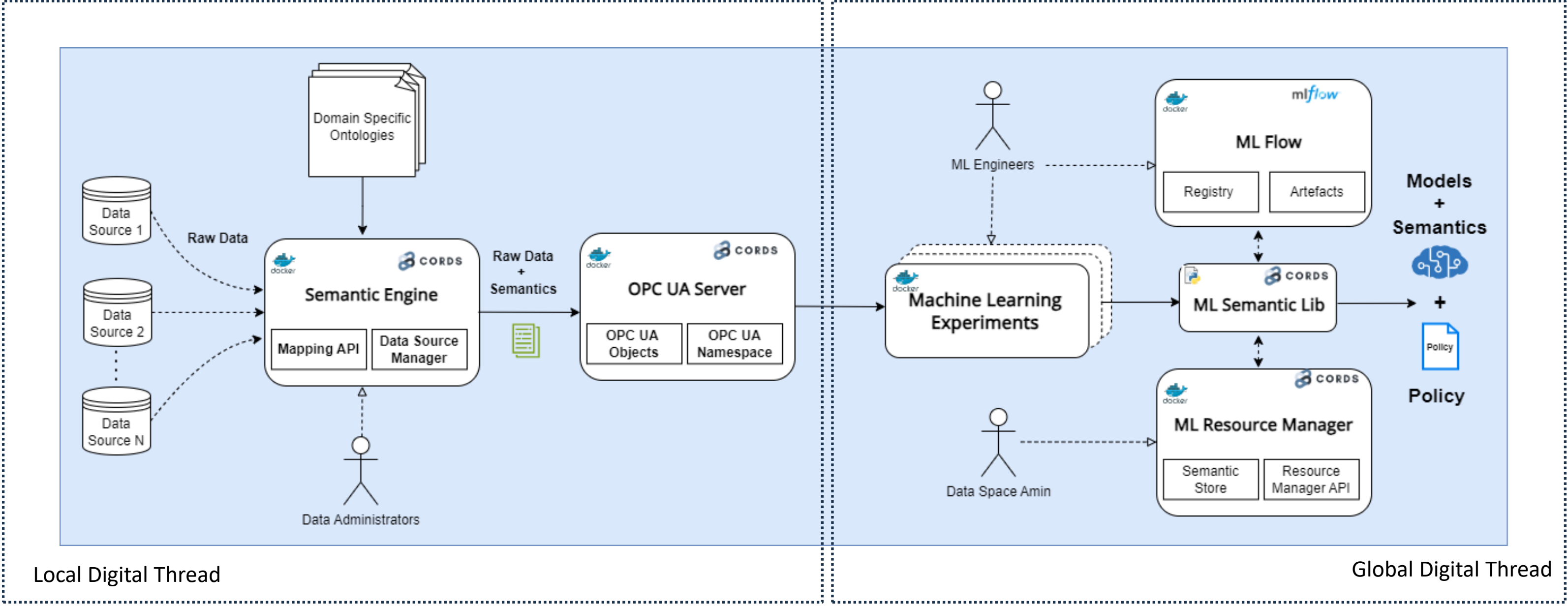- Prepare data for ML
- Preserve semantic interoperability

Global Digital Thread
- Manage model training
- Model advertising and discovery
- Transfer Learning/Federated Learning

CORDS Information Flow

Interoperability Toolkit

Local Digital Thread

Semantic Interoperability

- Common Information Model (CIM) for Energy Data
- Semantic Engine/API
  - Connecting to data sources
  - Mapping existing schema to CIM
- OPC UA Server
  - Data consolidation
  - Data Access through pub/sub model
  - OPC UA Namespace based on the CIM
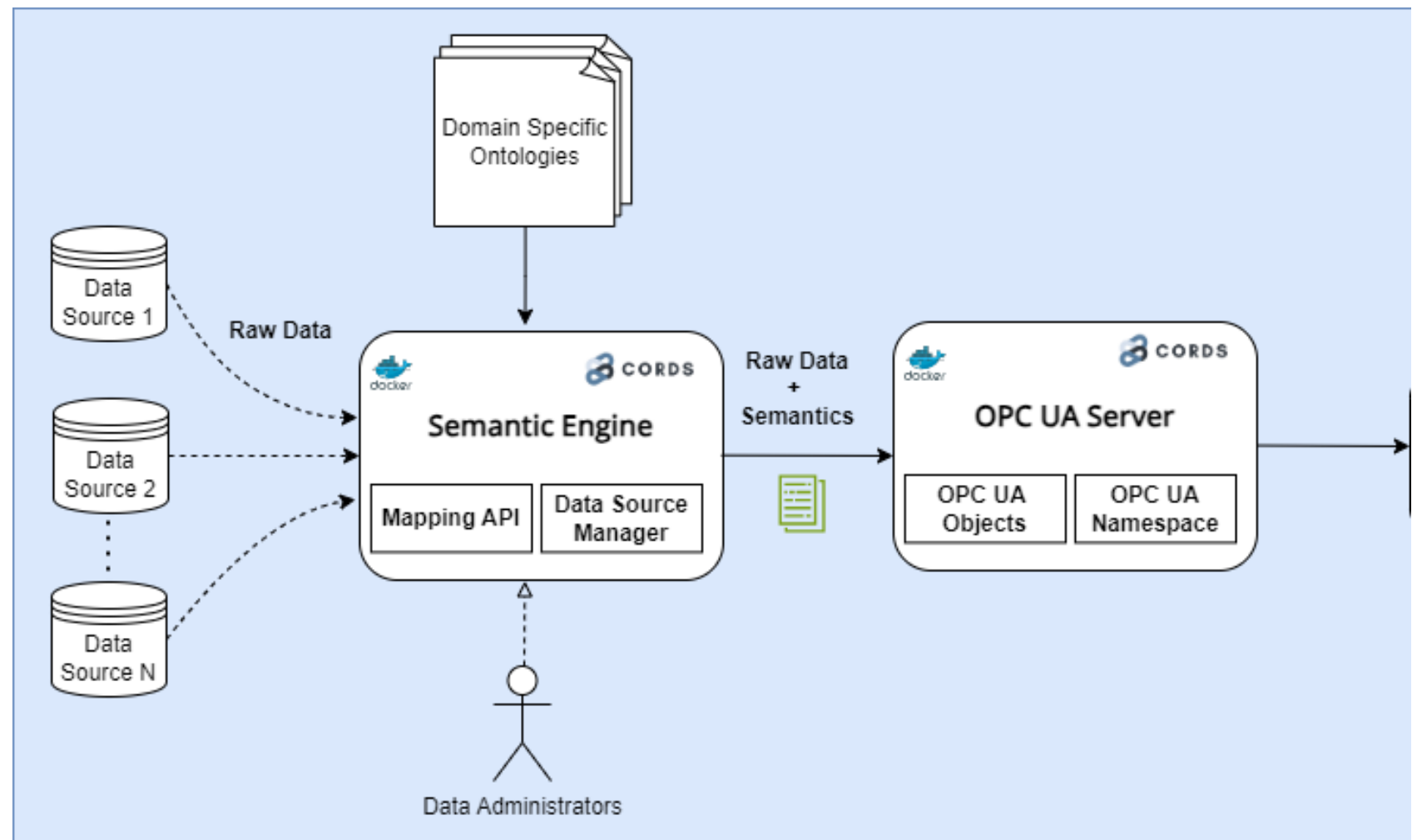
Toolkit: Local Digital Thread
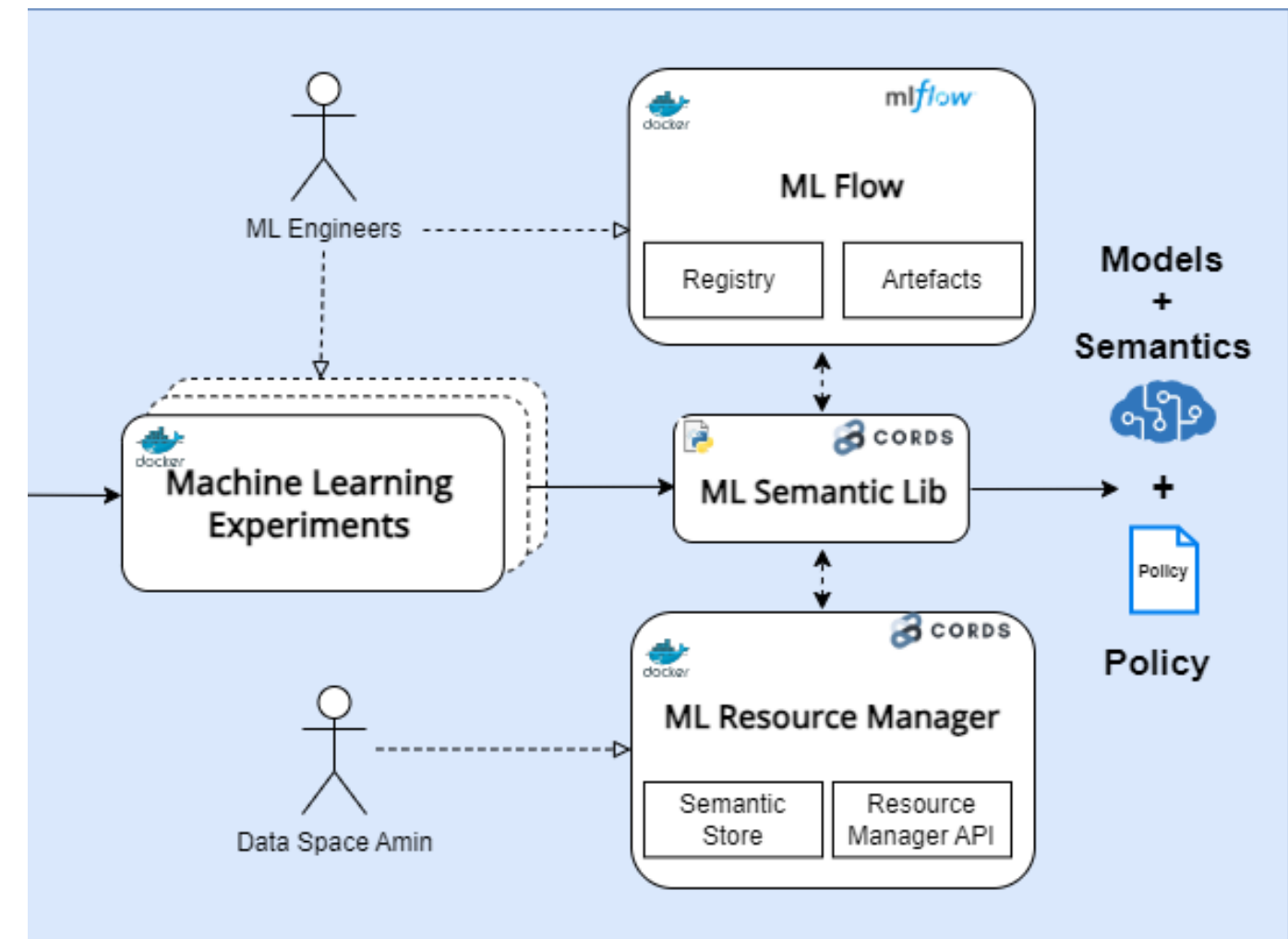
CIM for Energy Data

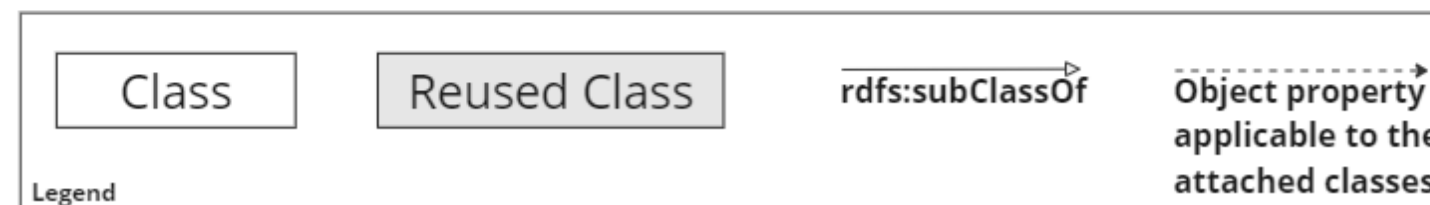- Reused ontologies:
  - SAREF
  - CIM
  - SEAS
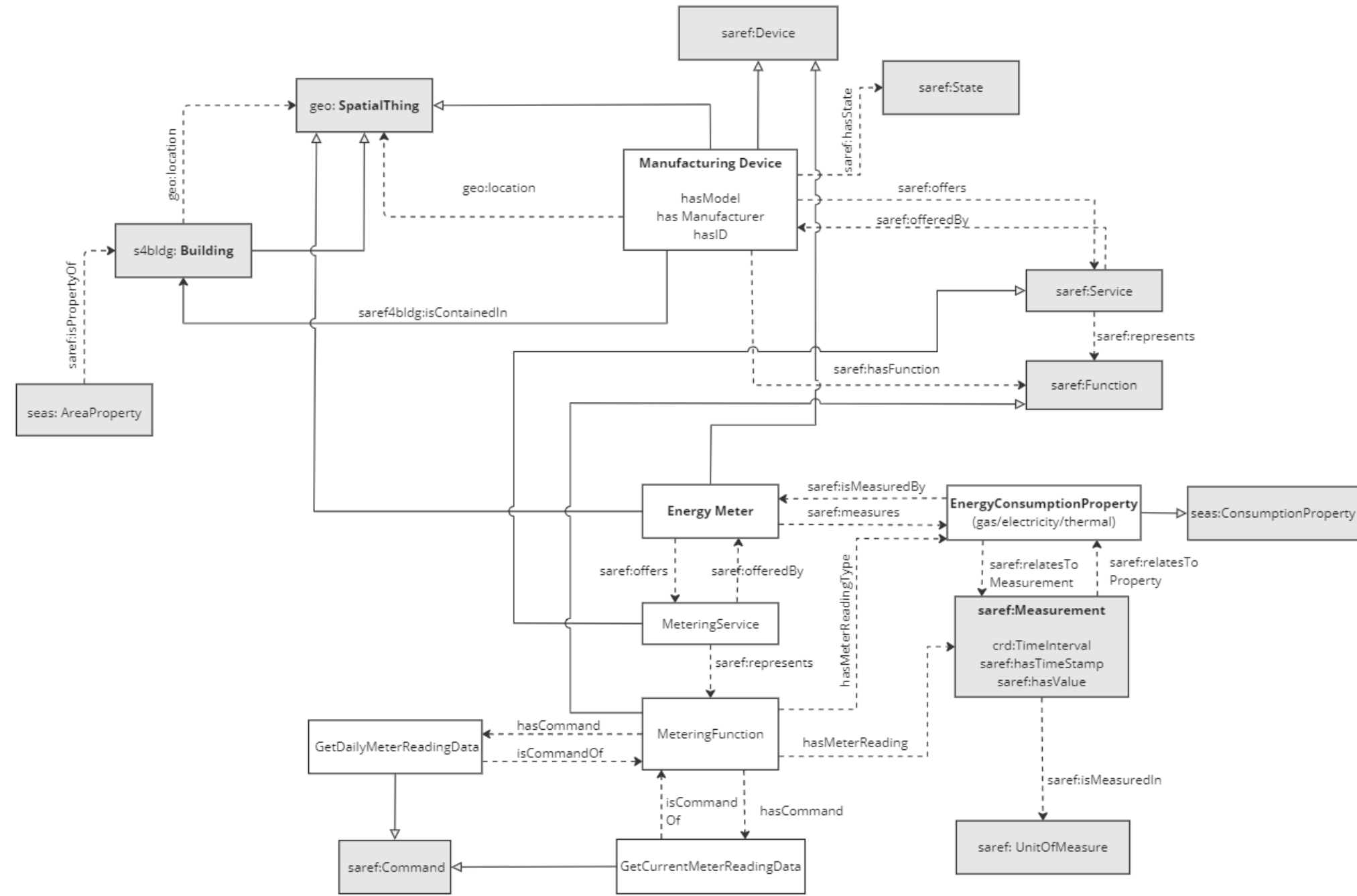
Global Digital Thread

Semantic Interoperability

- ML Semantic Library
  - Python based
  - Facilitate ML engineers to describe experiments
  - CIM for ML assets
- ML Flow Integration
  - Model tracking and repository management
  - Meta data visualization
- ML Resource Manager
  - Extracting metadata
  - Translating to RDF/IDSA compatible format
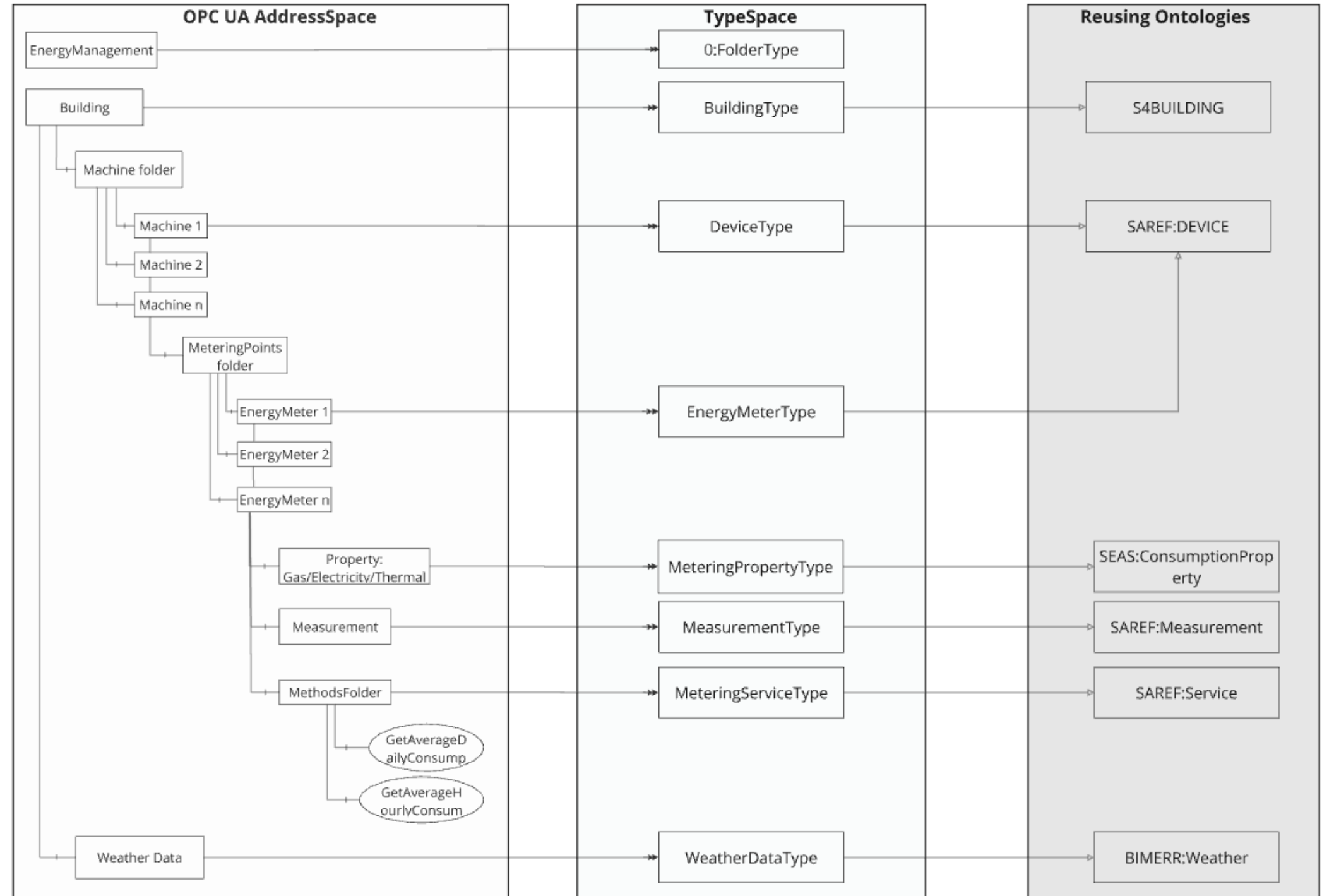 - Policy management

CIM for ML

- Reused ontologies:
  - ML Schema
  - DCAT

Toolkit in action

Mapping API

OPC UA Server Abstract View

Toolkit in action


```python
#Importing CORDS Tags
import cords_semantics.tags as cords_tags


with mlflow.start_run(run_name = run_name) as mlflow_run:
    mlflow_run_id = mlflow_run.info.run_id

    mlflow.set_experiment_tag("second best_model", "K-Neighbors Regressor")
    mlflow.set_tag("tag2", "K-Neighbors Regressor")
    mlflow.set_tag(cords_tags.CORDS_RUN, mlflow_run_id)
    mlflow.set_tag(cords_tags.CORDS_RUN_EXECUTES, "K-Neighbors Regressor")
    mlflow.set_tag(cords_tags.CORDS_IMPLEMENTATION, "python")
    mlflow.set_tag(cords_tags.CORDS_SOFTWARE, "sklearn")

    mlflow.sklearn.log_model(models['K-Neighbors Regressor'], "knnmodel")

    mlflow.log_metric("test_RMSE", rmse_scores.loc[rmse_scores['Model Name'] == 'K-Neighbors Regressor'
    mlflow.log_metric("test_MAE", mae_scores.loc[mae_scores['Model Name'] == 'K-Neighbors Regressor', '
    mlflow.log_metric("test_R2_Score", r2_scores.loc[r2_scores['Model Name'] == 'K-Neighbors Regressor'

    mlflow.log_input(dataset, context="training")


    print("MLFlow Run ID: ", mlflow_run_id)
```

Tagging in ML Experiments

ML FLow

Toolkit in action

Resource Description Generation

IDS Compatible Resource Description

Further Research Directions

- Integration to CORDS Minimum Viable Data Space.
- Explore automated tagging
- Usability improvements
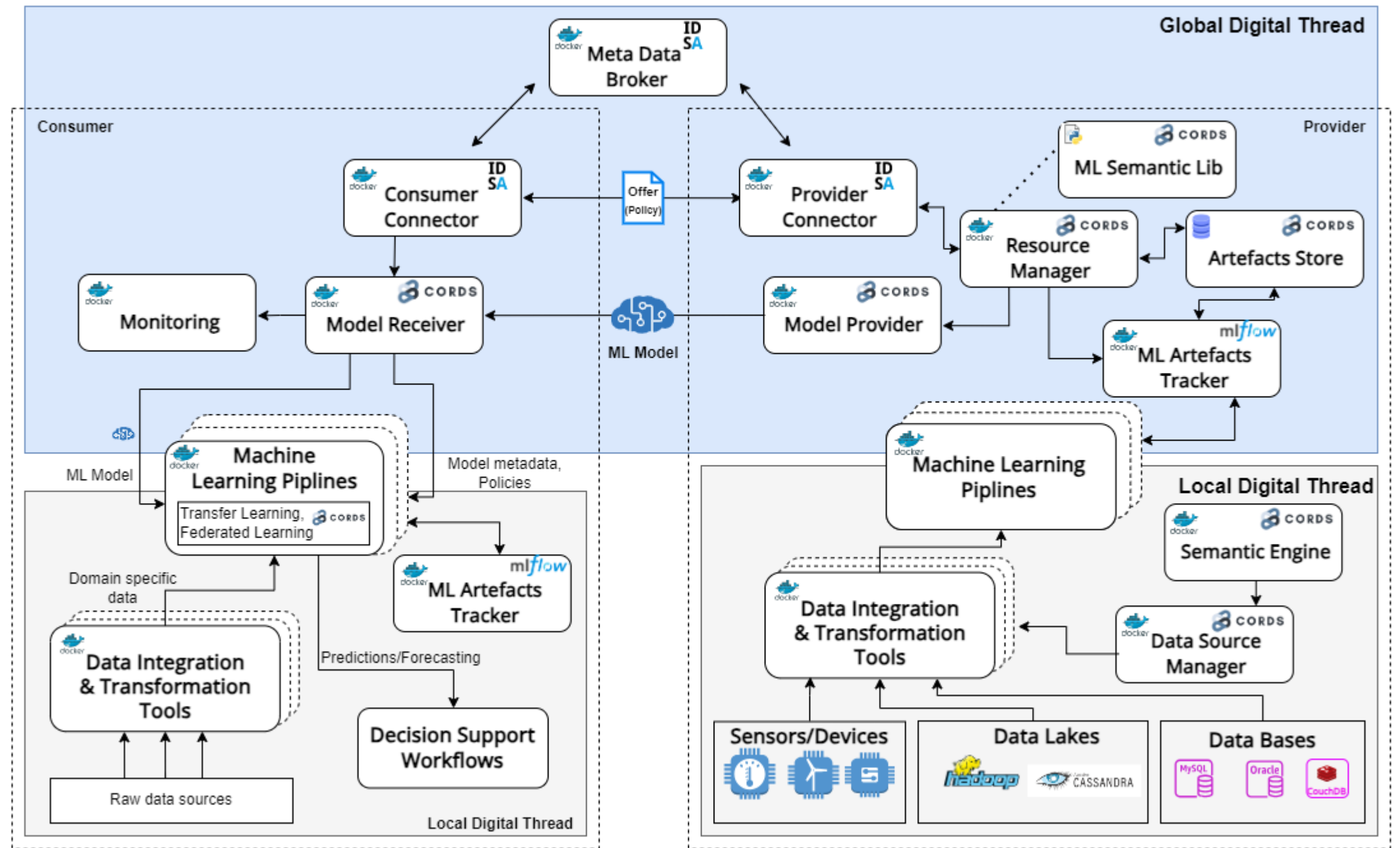
Acknowledgement

Website: cords.ie