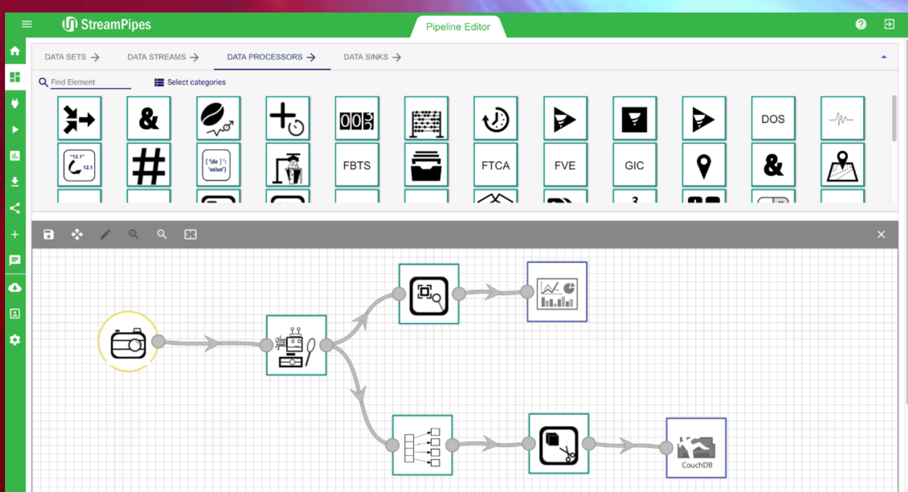
Building a Self-Service IoT Analytics Toolbox
Basics, Models and Lessons Learned

Rudi Studer, Dominik Riemer
IC3K 2018, Seville, September 20, 2018

Institute of Applied Informatics and Formal Description Methods (AIFB)

This talk

How to enable **application specialists** to **create** and **execute IoT analytics applications** based on distributed **stream processing** in a **self-service** manner?



Deploy

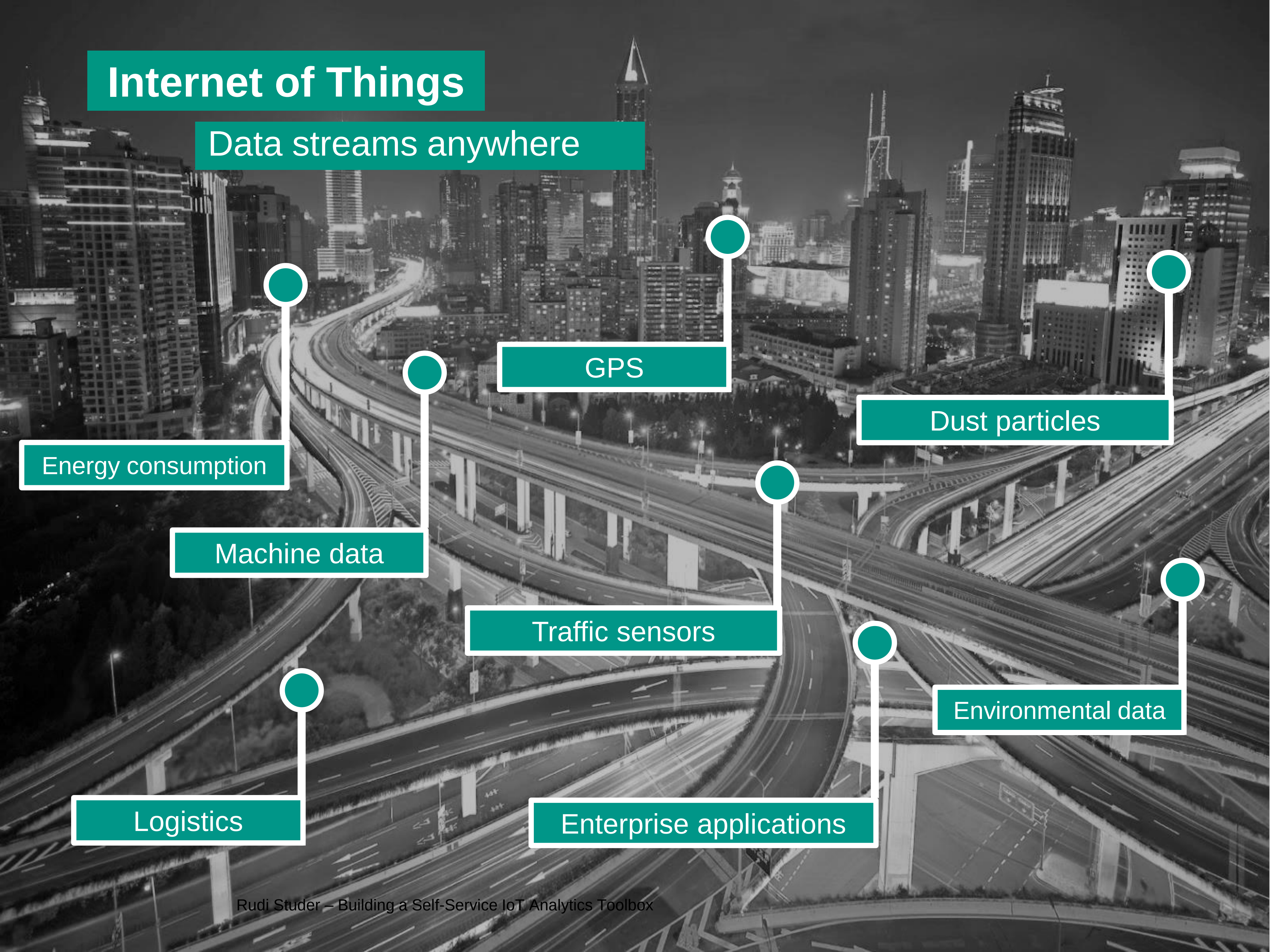
Distributed Stream Processing Architecture

Outline

- **Introduction & Motivation**
- Problems
- Development methodology for event-driven applications
  - Overview
  - Modeling of data streams and processing elements
  - Definition and execution of distributed processing pipelines
- Implementation and evaluation
- Conclusion

Internet of Things

Data streams anywhere



Events

GPS

Event [Luck02]

A record or an activity within a system

Event Type

Event Property

```
timestamp : long
vehicleId : string
latitude : double
longitude : double
```

Event Object

```
timestamp : 1453478150
vehicleId : ID5
latitude : 48.94
longitude : 8.40
```

Event Representation (JSON)

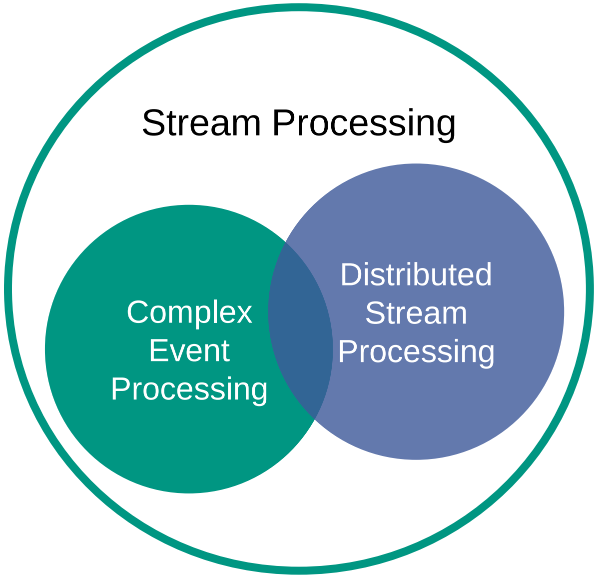
```
{
  "timestamp" : 1453478150
  "vehicleId" : "ID5"
  "latitude" : 48.94
  "longitude" : 8.40
```

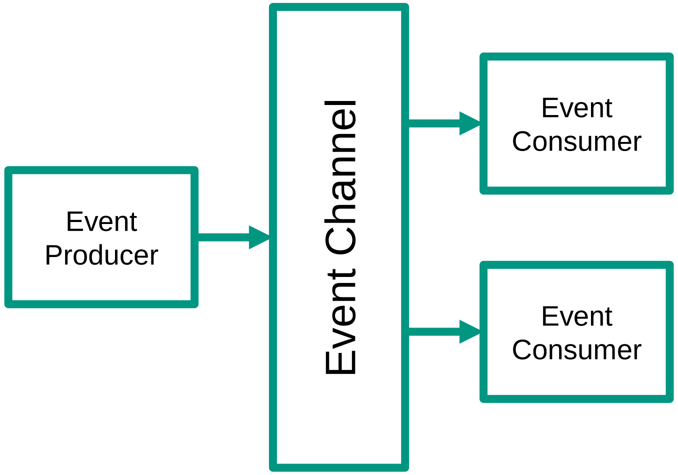
[Luck02]: Luckham, David: The power of events

Stream Processing

- **Event-Driven Architecture [BrDu10]**
  - Producers and consumers are completely decoupled
  - Publish/Subscribe [EtNi11]
    - Multiple consumers per event

  - **Push Interaction [EtNi11]**
    - One-way-communication
    - Producer does not expect that an event is processed by any consumer





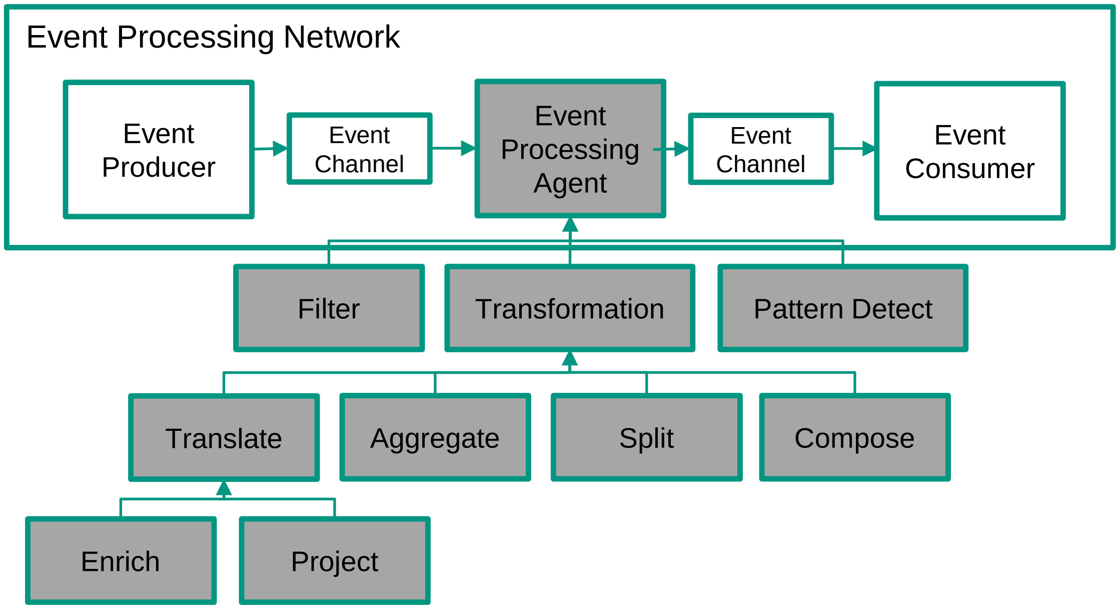
[BrDu10] Bruns, Dunkel: Event-Driven Architecture.
[EtNi11] Etzion, Niblett: Event Processing in Action

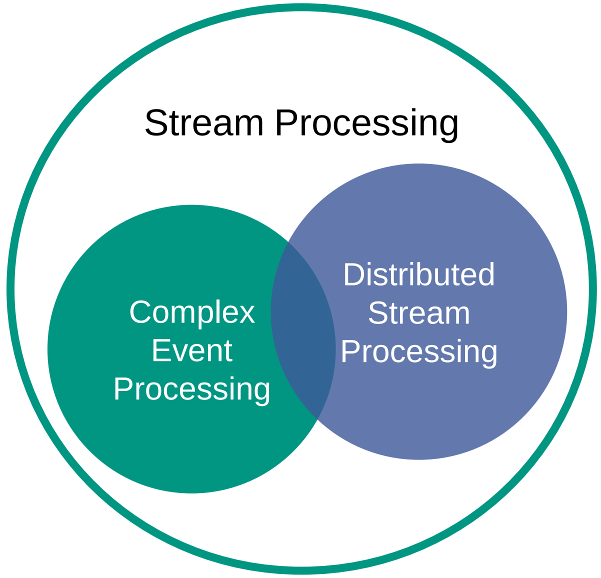
Complex Event Processing (CEP)

Focus on *Complex Pattern Detection* [WuDR06]
- e.g., absence of events, sequences, sliding windows

Event Processing Agent [EtNi11]
- Software component which processes events

Event Processing Network [EtNi11]
- Set of event producers, event processing agents and event consumers, connected through event channels





Tools: Examples

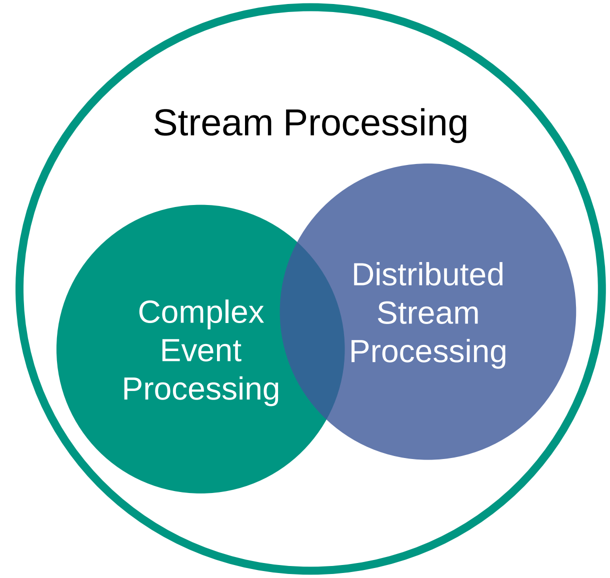
streambase
EsperTech
SAP EVENT STREAM PROCESSOR
APAMA

[WuDR06] Wu et.al.: High-Performance Complex Event Processing Over Streams
[EtNi11] Etzion, Niblett: Event Processing in Action

Distributed Stream Processing

Focus on *Scalable Processing of Events* [CBBC+03]

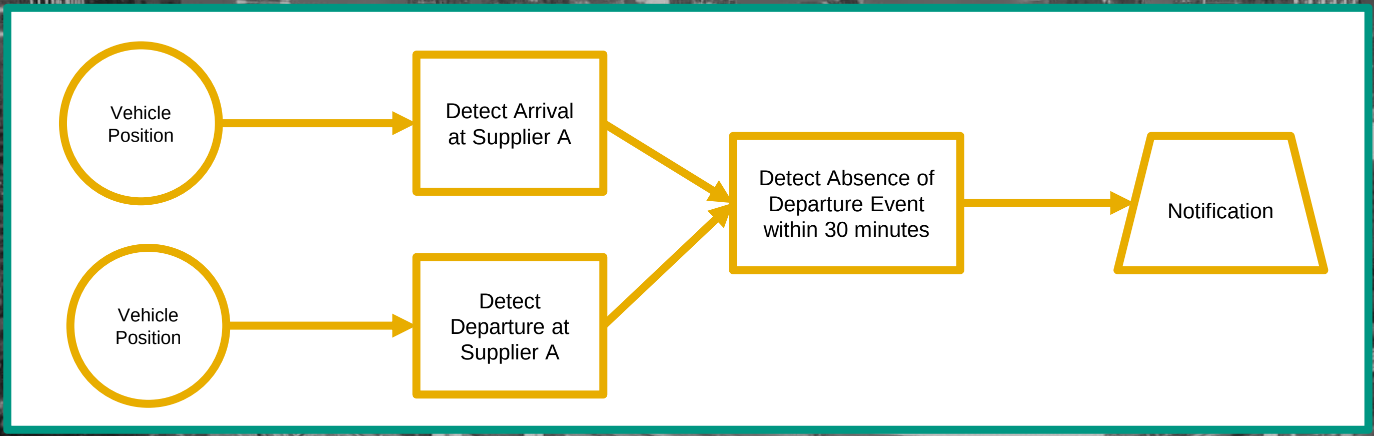
- CEP-Systems usually single-host systems, leading to scalability issues in case of very high throughput [AGDT14]
- High-Level programming APIs



Tools: Examples

[CBBC+03] Cherniack et.al.: Scalable Distributed Stream Processing.
[AGDT14] Andrade et.al.: Fundamentals of Stream Processing.

A building block of IoT analytics applications:
Flexible definition of real-time processing pipelines by application specialists



Integrated Monitoring

Flexible data integration from heterogeneous sources

Situational Awareness

Detection of situations and failures based on Complex Event Processing (CEP)

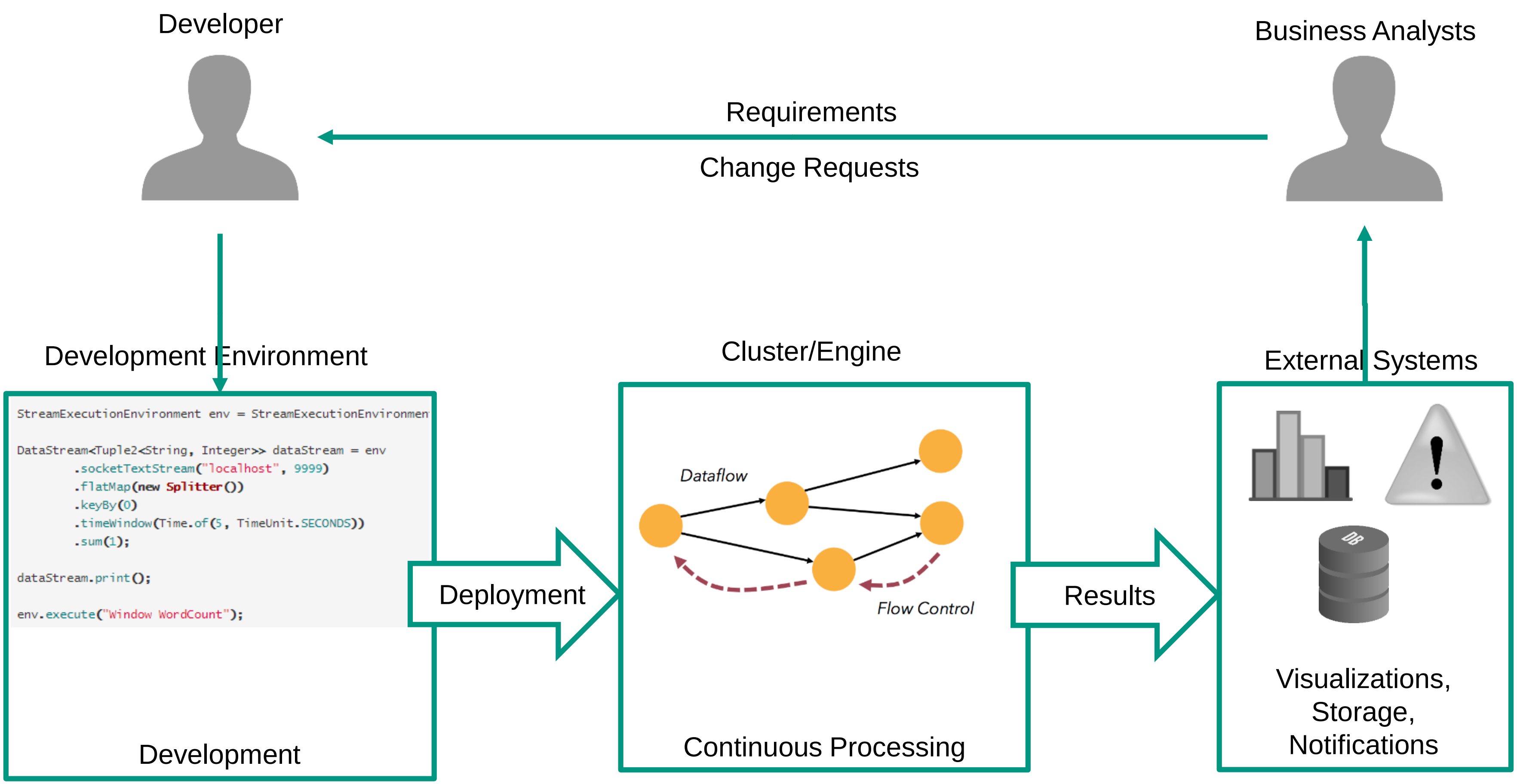
Continuous Data Harmonization

Continuous pre-processing and data harmonization for third-party systems

Outline

- Introduction & Motivation
- **Problems**
- Development methodology for event-driven applications
  - Overview
  - Modeling of data streams and processing elements
  - Definition and execution of distributed processing pipelines
- Implementation and evaluation
- Conclusion

Problem: Slow Development Cycles



Problem: Slow Development Cycles

Entwickler

Entwicklungsumgebung

```
StreamExecutionEnvironment env = StreamExecutionEnvironmen

DataStream<Tuple2<String, Integer>> dataStream = env
        .socketTextStream("localhost", 9999)
        .flatMap(new Splitter())
        .keyBy(0)
        .timeWindow(Time.of(5, TimeUnit.SECONDS))
        .sum(1);

dataStream.print();

env.execute("Window WordCount");
```

Detect Arrival

Detect Absence of Departure

Complex Event Processing

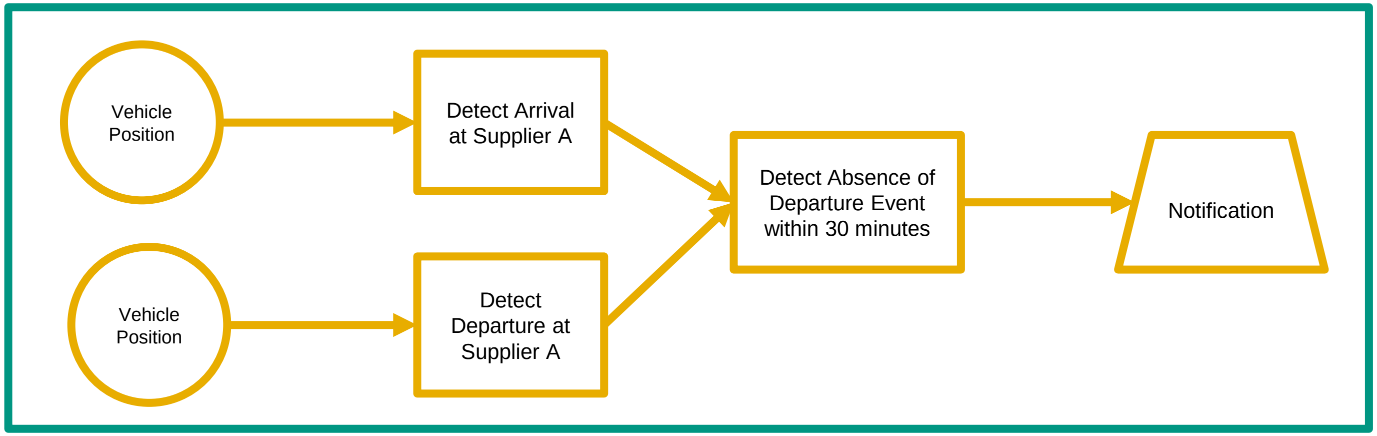
```
Register event types
```

```
insert into ArrivalEvent
select a.timestamp, a.vehicleId, getSupplier(b.latitude,
b.longitude) from pattern[every (a=PositionEvent ->
b=PositionEvent)]
where (!a.isInside(a.lat, a.lng, 49.06, 8.5, 500) and
b.isInside(b.lat, b.lng, 49.23, 4.27, 500) and
b.vehicleId=a.vehicleId)
```

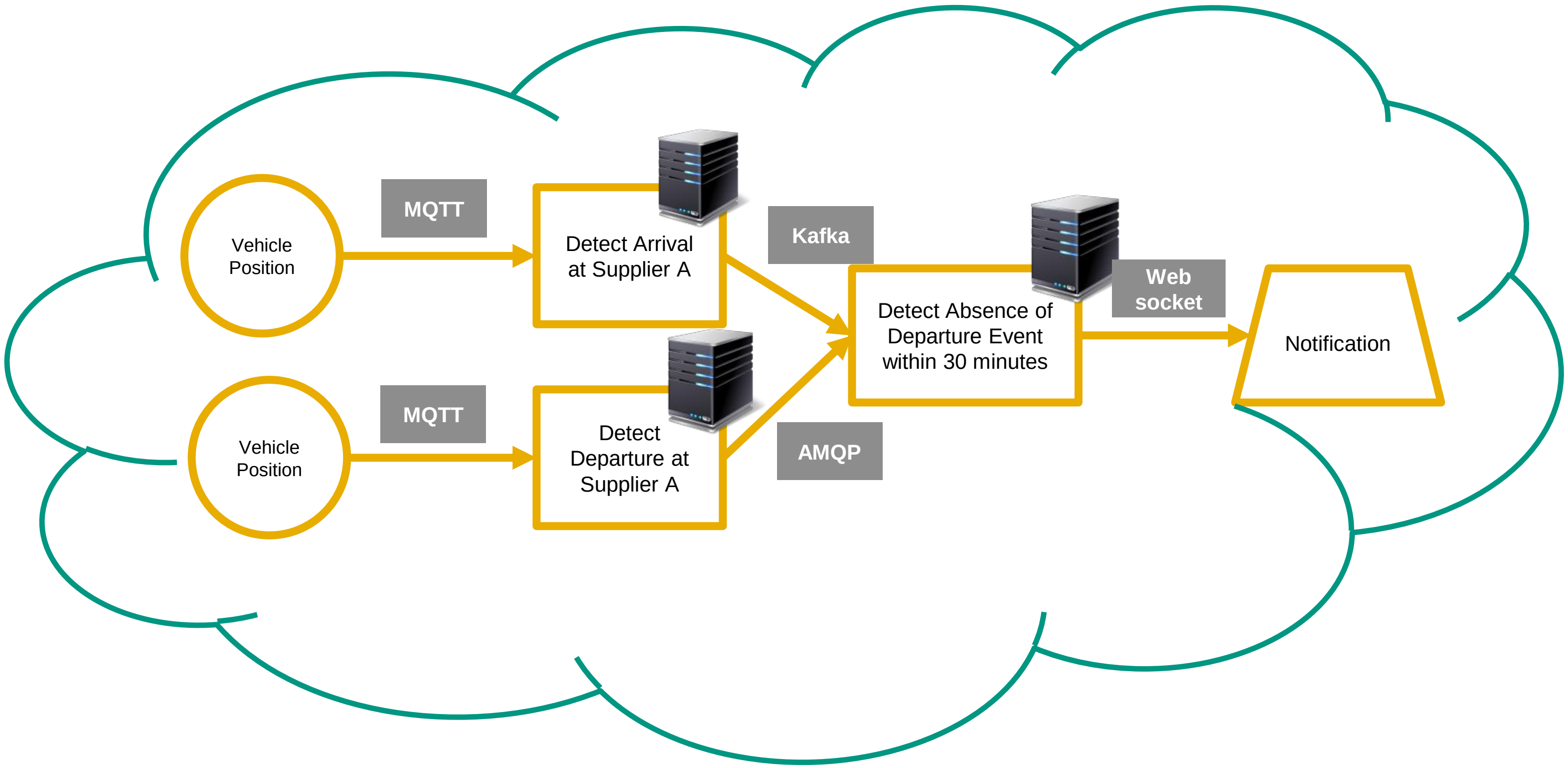
```
select a.vehicleId, (current_timestamp – a.timestamp) as time
from pattern[every a=ArrivalEvent -> timer:interval(30
minutes) and not b=DepartureEvent] where
a.vehicleId=b.vehicleId
```

```
Implement output adapter
```

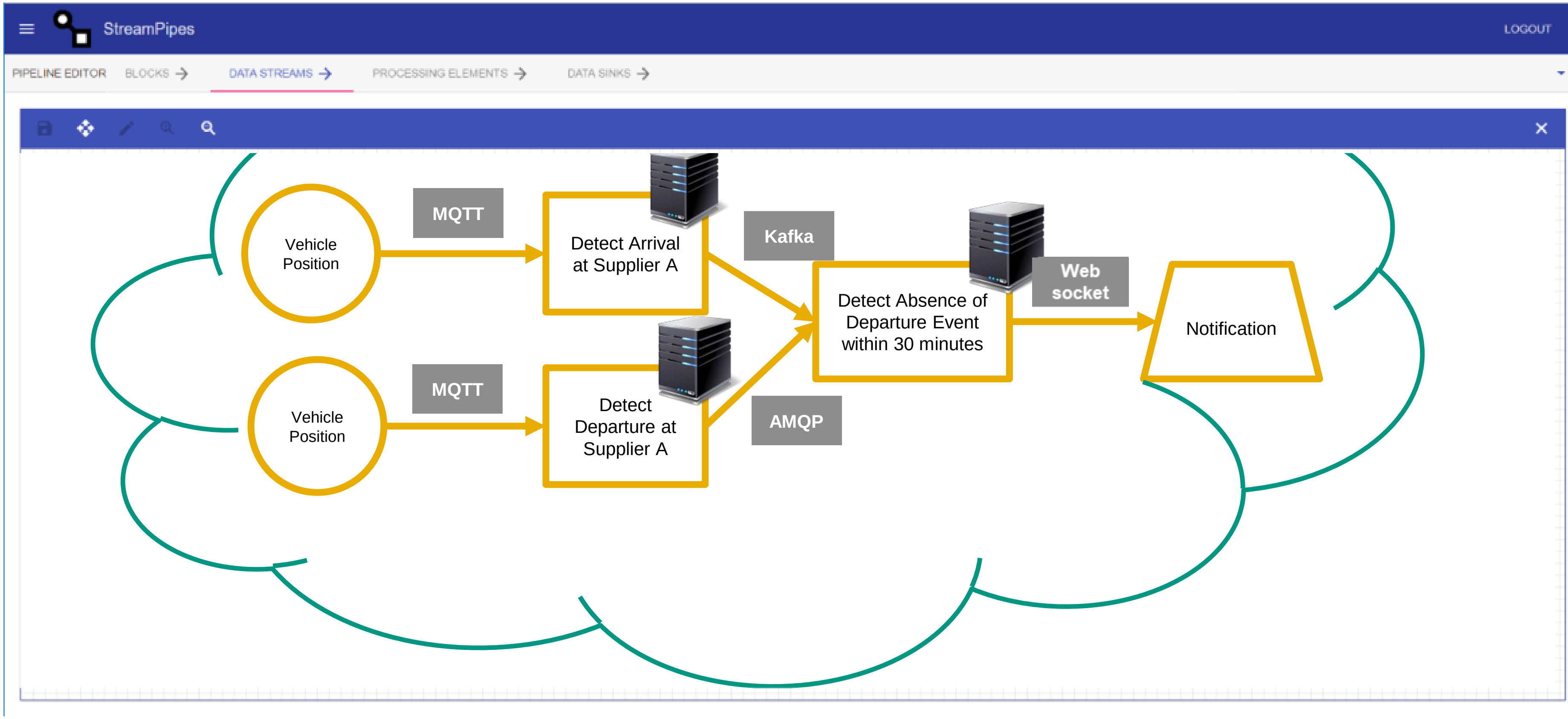
Requirement: Interoperability



Requirement: Interoperability



Requirement: Interoperability



Problems: Summary

Observation

- frequent changes of applications required due to:
  - semantic/syntactic changes of event producers
  - new/changing requirements of application specialists
- high effort needed due to slow development cycles
- demand for„**Self-Service Data Analysis**"

*"**Analysts** should be able to process streaming data to gain insight - and once insight has been gained, to **easily refine the processing pipelines for even more insight** or to switch their focus of attention completely **without much latency.**"*

otto group

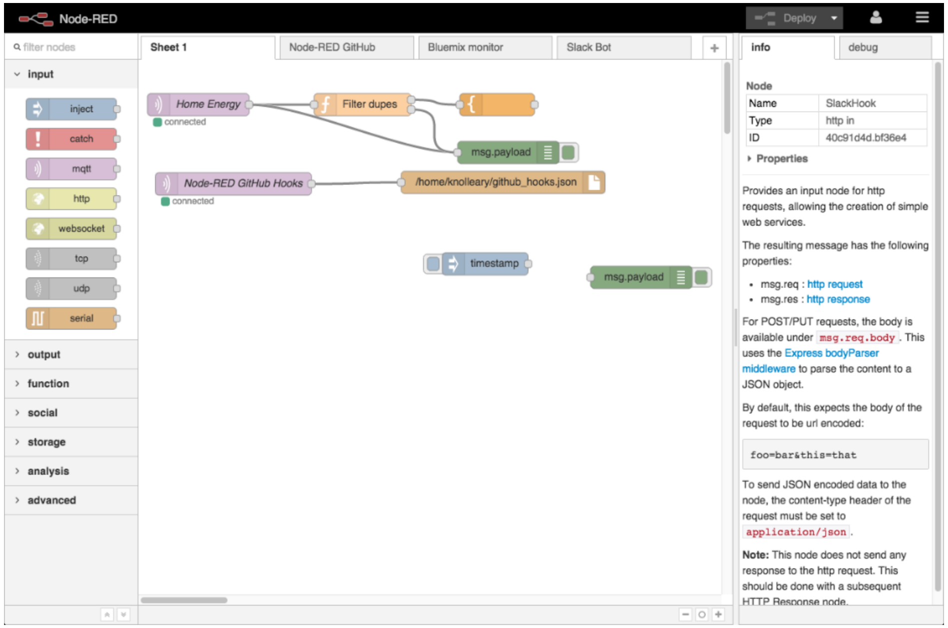
Example: Node-Red

Node-Red

- "Wiring the Internet of Things", initially developed by IBM Research

Differences

- Single-Host system (no distribution of operators)
- Basic consistency checking (based on datatypes only)
- HTML-based configuration
- Relies on JavaScript runtime



Outline

- Introduction & Motivation
- Problems
- **Development methodology for event-driven applications**
  - **Overview**
  - Modeling of data streams and processing elements
  - Definition and execution of distributed processing pipelines
- Implementation and evaluation
- Conclusion

Methodology: Phases

Preparation

Development and grounding of re-usable event processing components

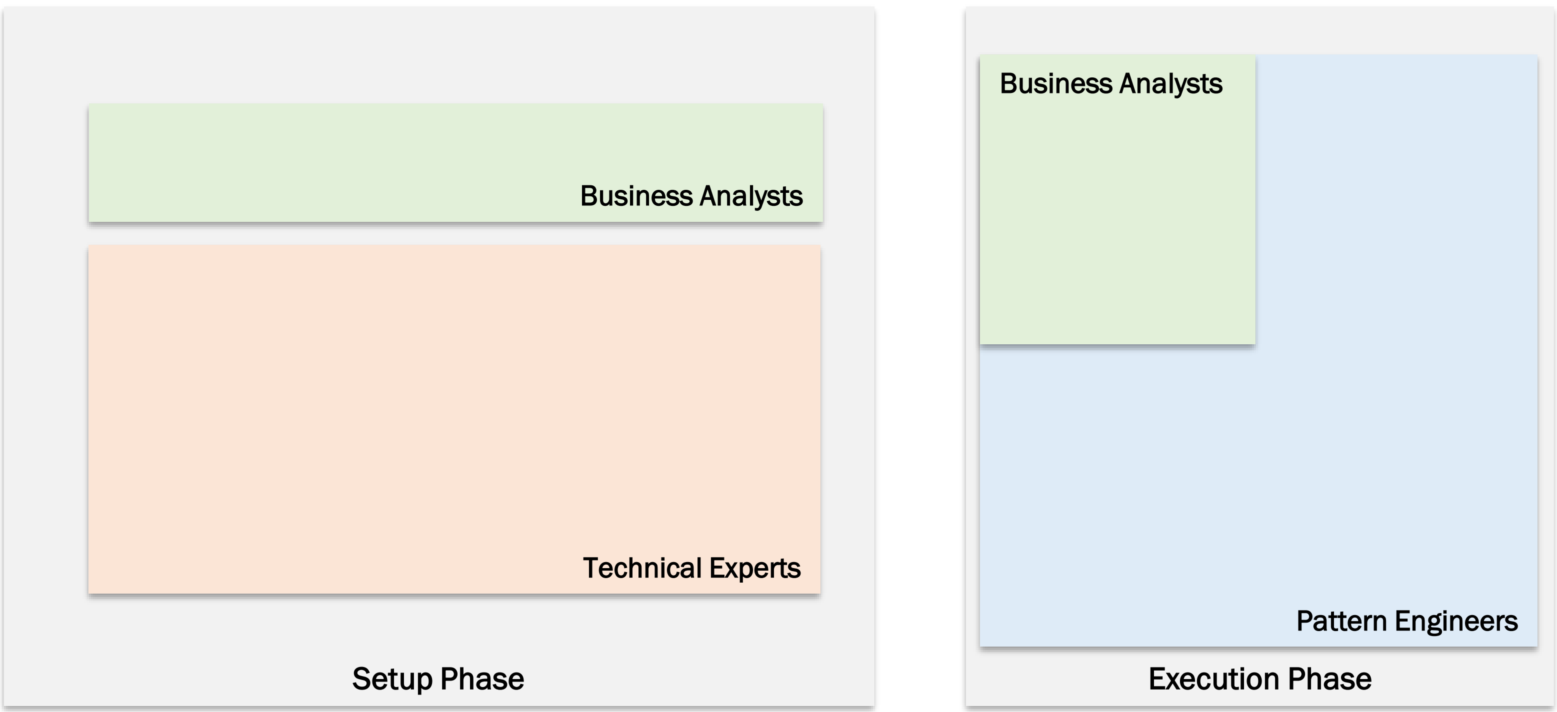
Setup Phase

Execution

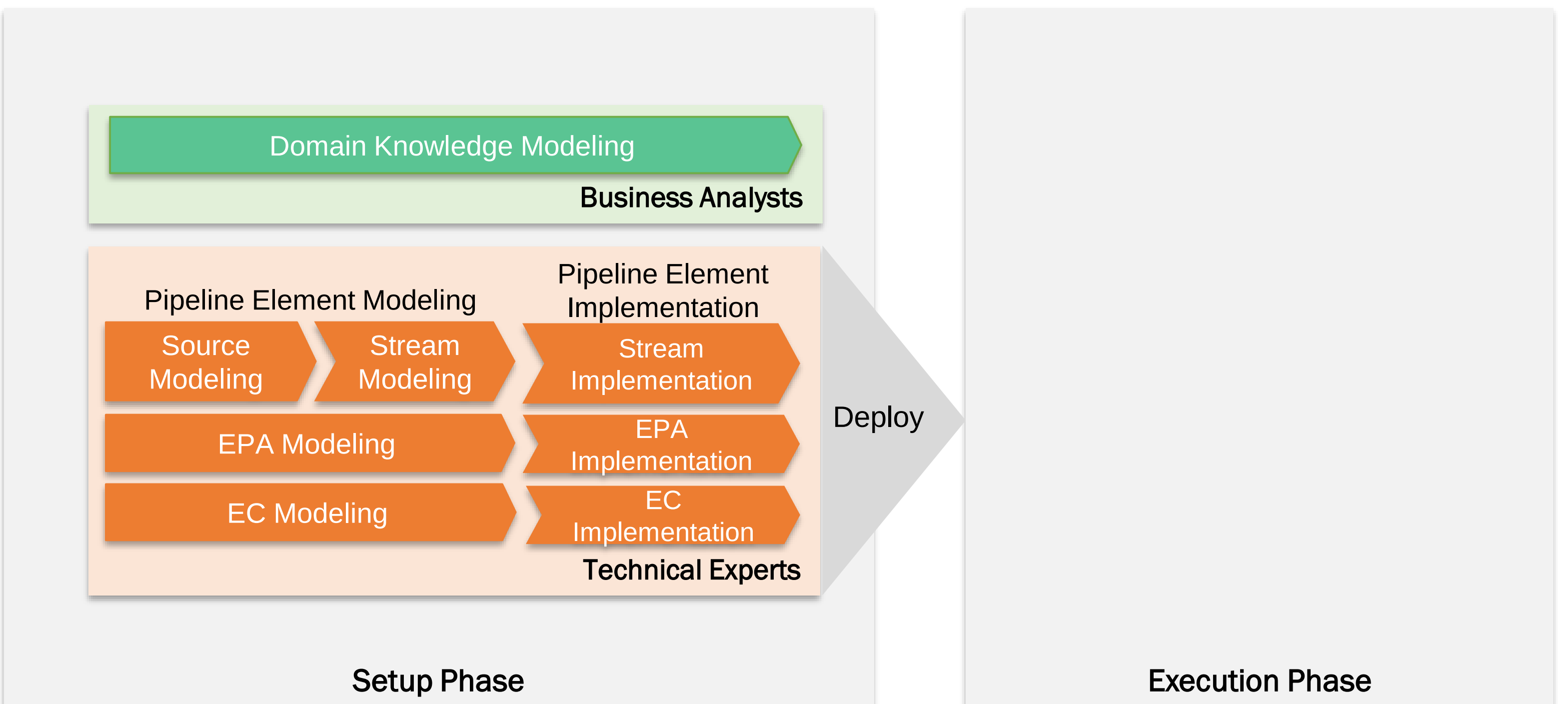
Definition and deployment of processing pipelines based on re-usable components

Execution Phase

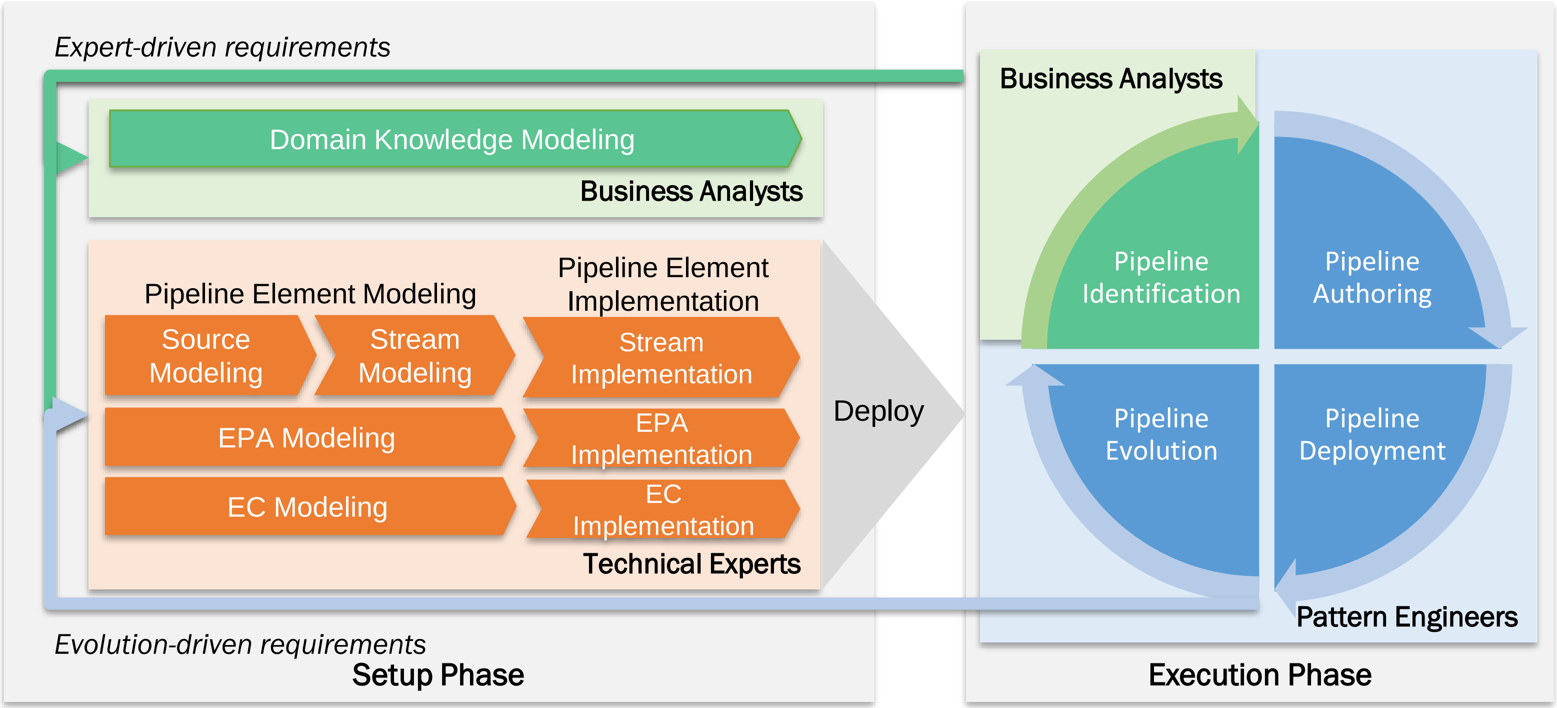
Methodology: Different Roles



Methodology: Two Tasks in Setup Phase

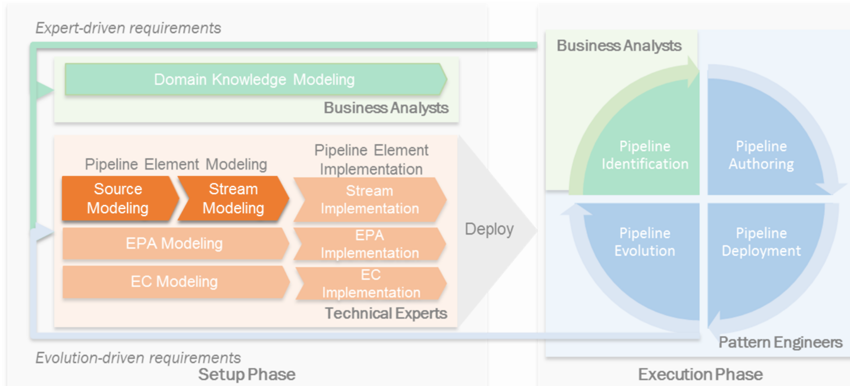


Methodology: Tasks

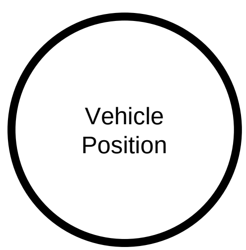


Outline

- Introduction & Motivation
- Problems
- **Development methodology for event-driven applications**
  - Overview
  - **Modeling of data streams and processing elements**
  - Definition and execution of distributed processing pipelines
- Implementation and evaluation
- Conclusion



Semantic Modeling of Data Streams



Related Work: Semantic Sensor Network Ontology (SSN)

Scope

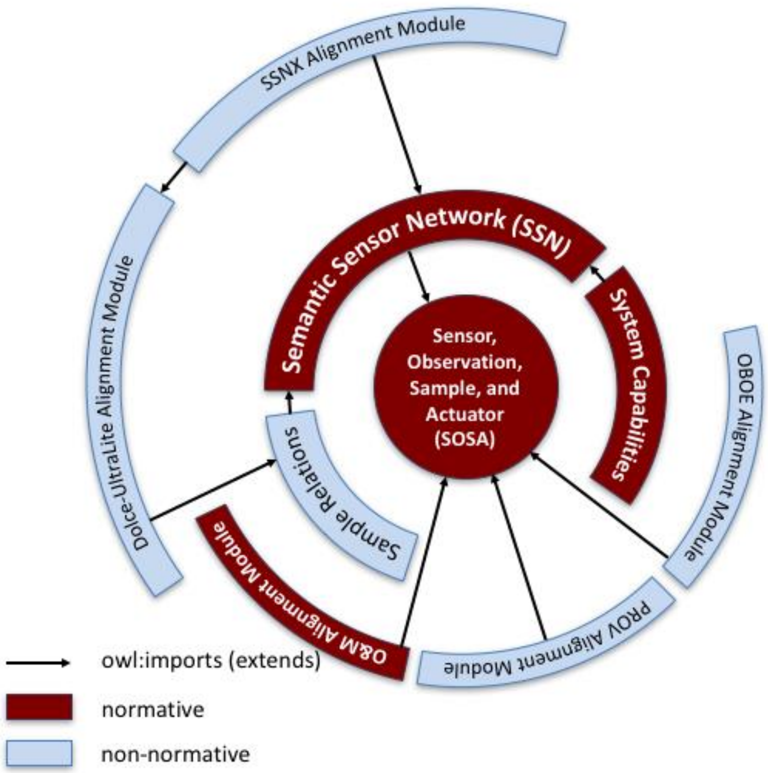
Define capabilities of sensors and sensor networks

Vocabulary

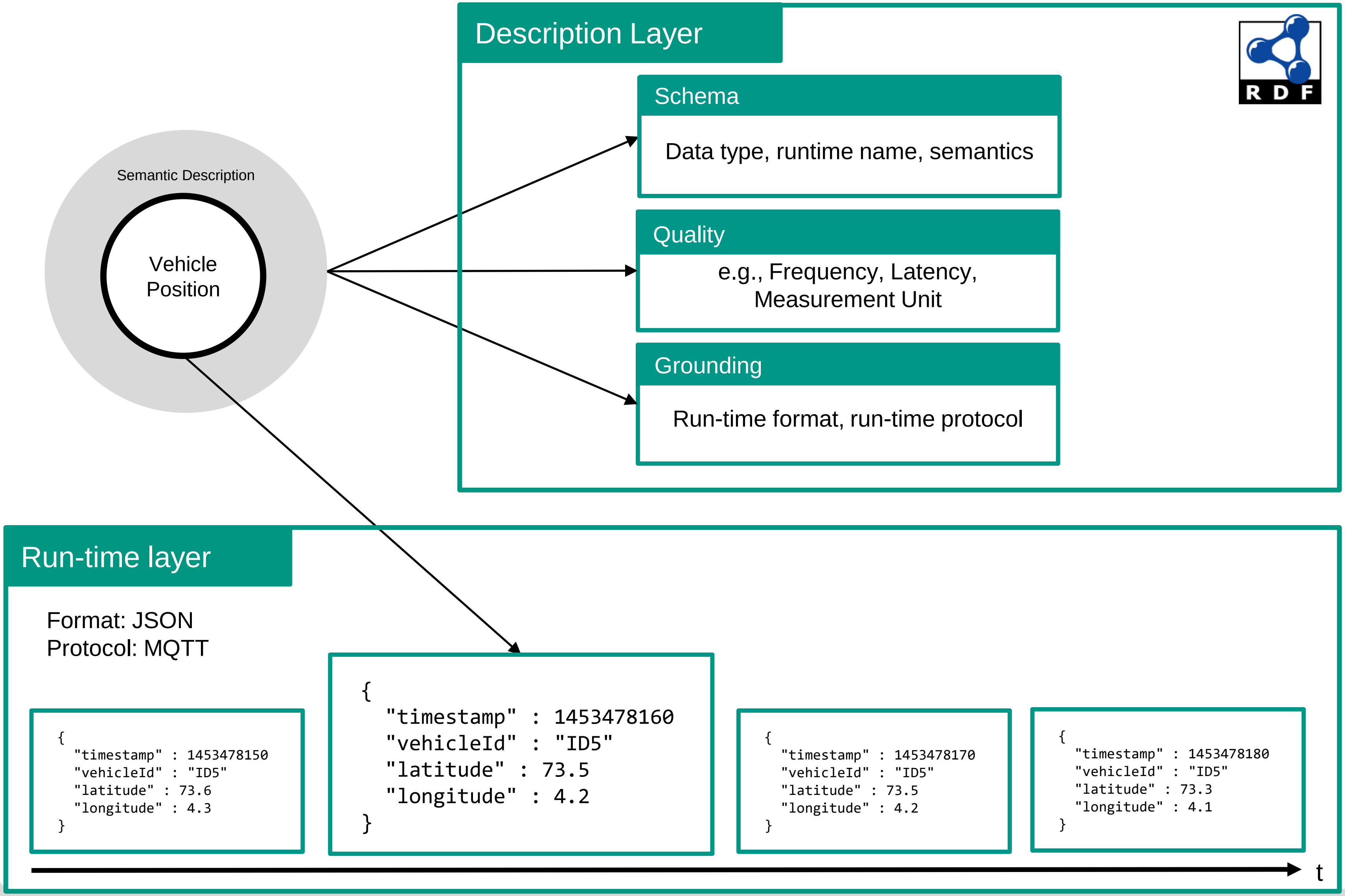
- Provides ways to model platforms, sensors and observations
- Aligned with upper ontologies
- modular (only observations, only sensors)
- Can be extended (e.g., with measurement units)

Our approach

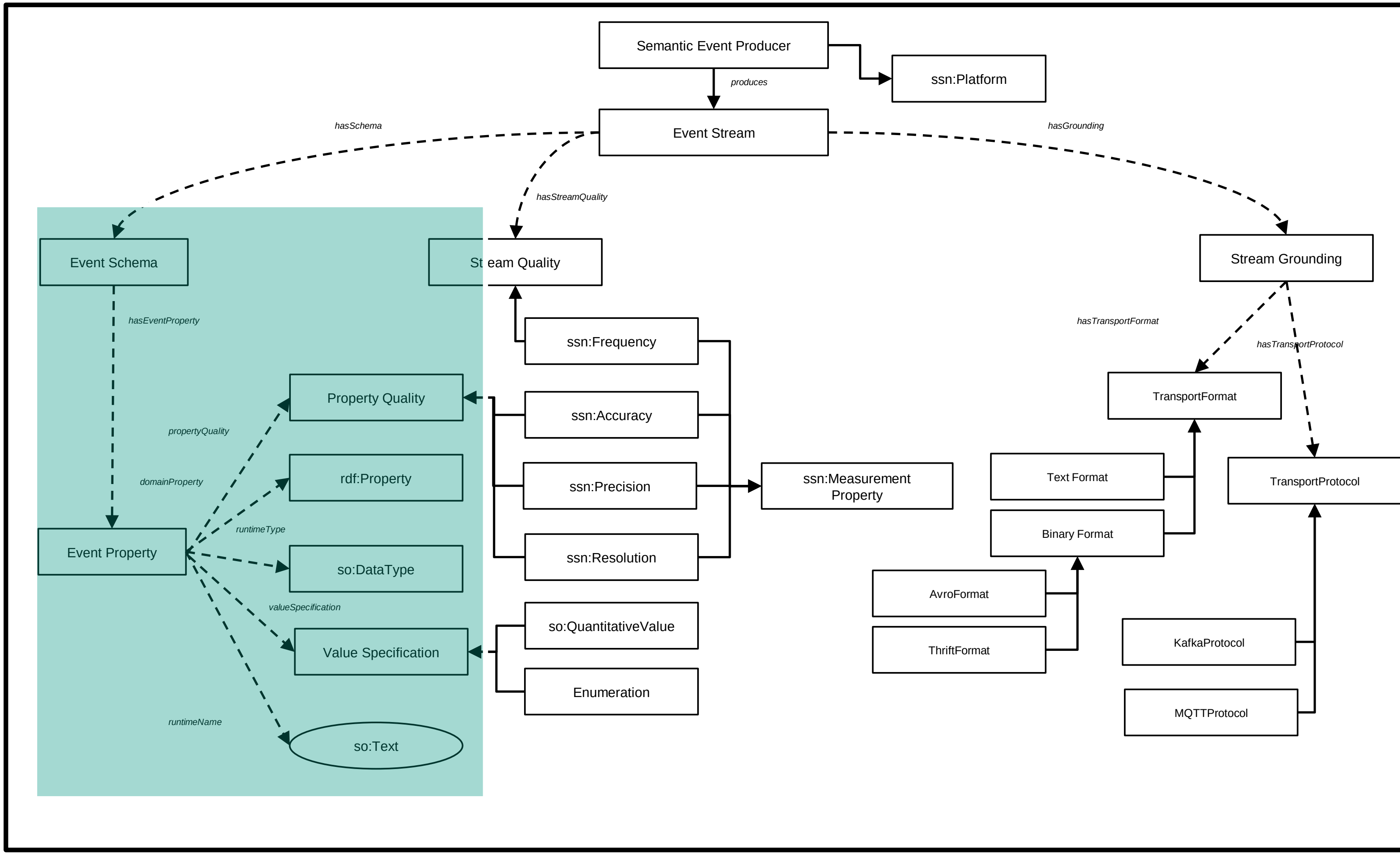
- Reuse parts of SSN (e.g., qualities)
- Use RDF only as metadata description to streams
- Use existing serialization formats for event transmission



Stream Modeling: Requirements



Stream Modeling: Vocabulary



Stream Modeling: Event Schema

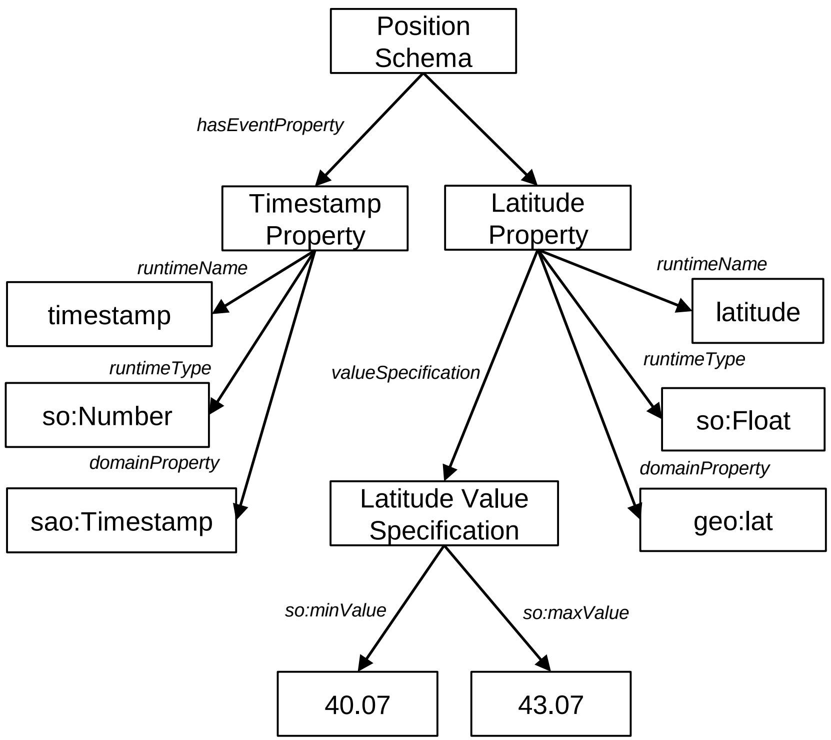
Model

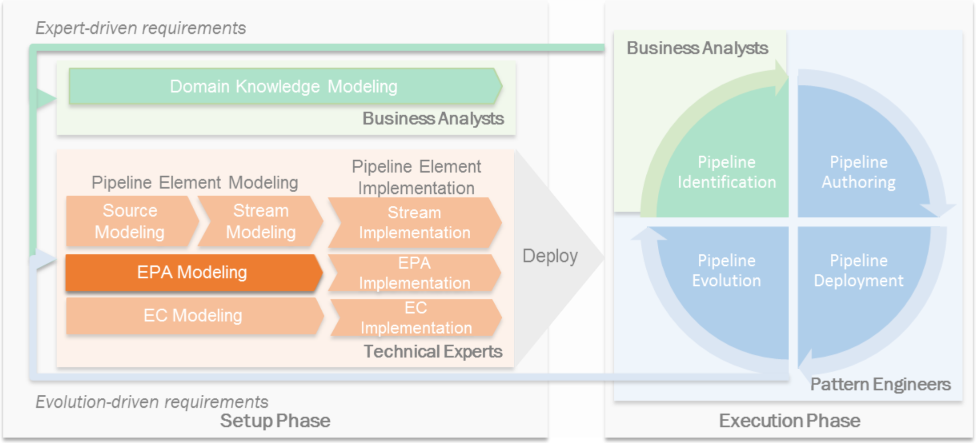
Event Schema:

Description of the structure of the event at run-time

- **runtimeName**
  - Identifier of event property name in the run-time format
- **runtimeType**
  - Data type of the event property at run-time
- **domainProperty**
  - additional semantic description used for matching during pipeline definition
- **valueSpecification**
  - value specification of the event property
- **measurementUnit**
  - Measurement unit of the event property
- **propertyQuality**
  - Property-specific quality attributes,
  - e.g., accuracy

Example

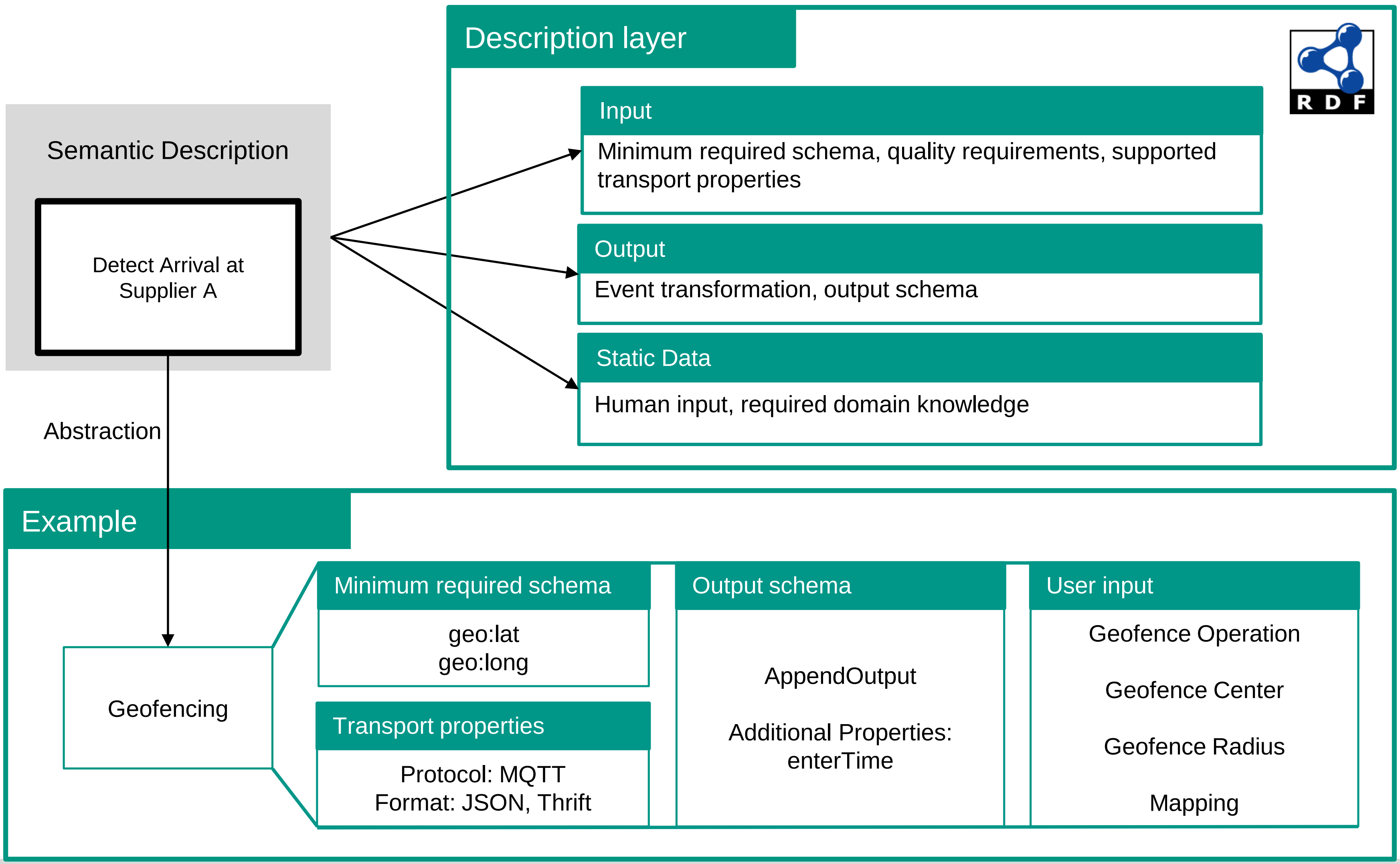




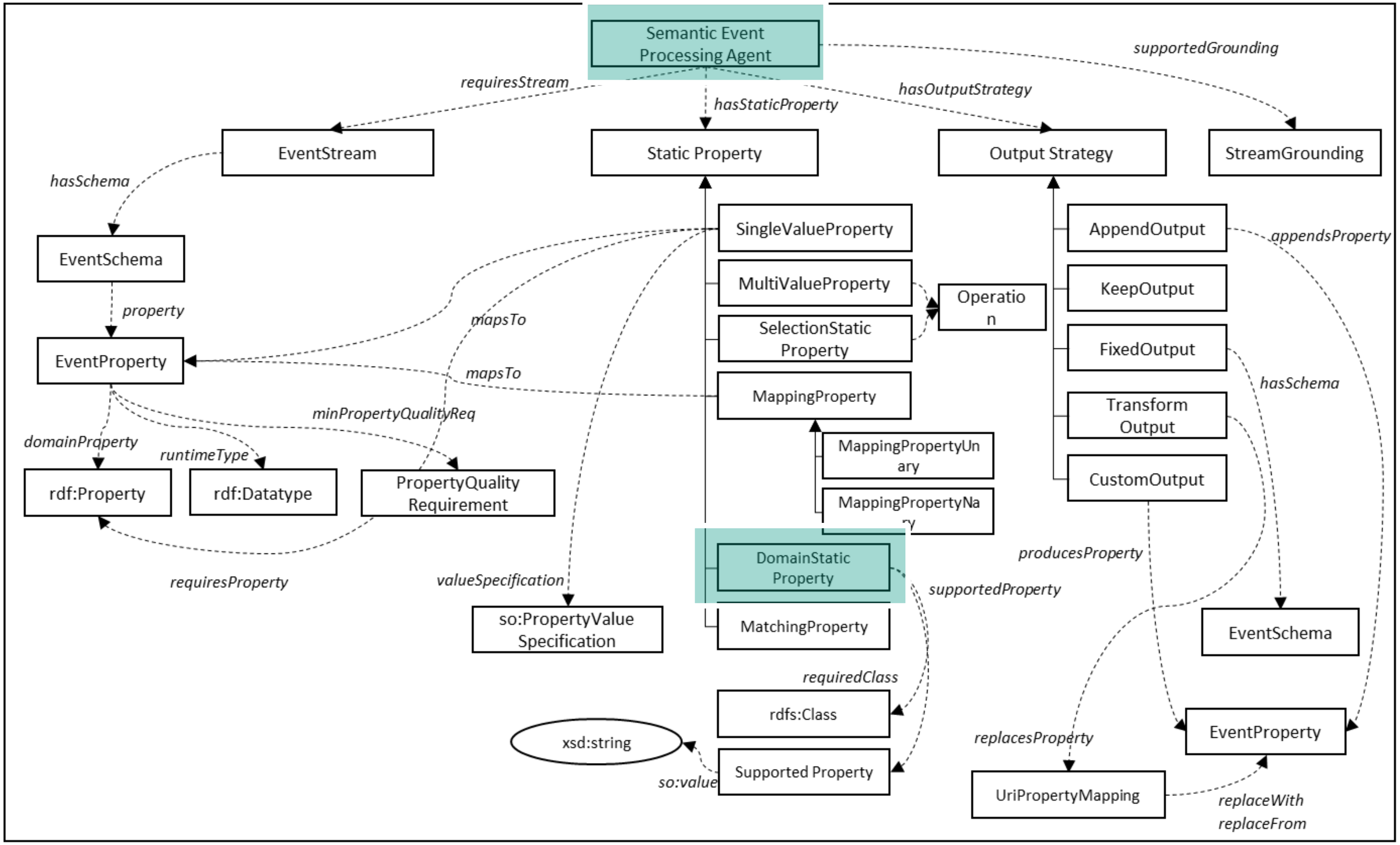
Semantic Modeling of Event Processing Agents

Detect Arrival at Supplier A

EPA Modeling: Requirements



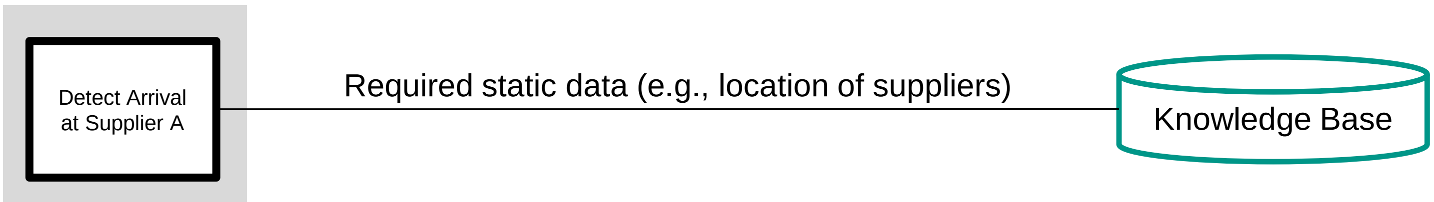
EPA Modeling: Vocabulary

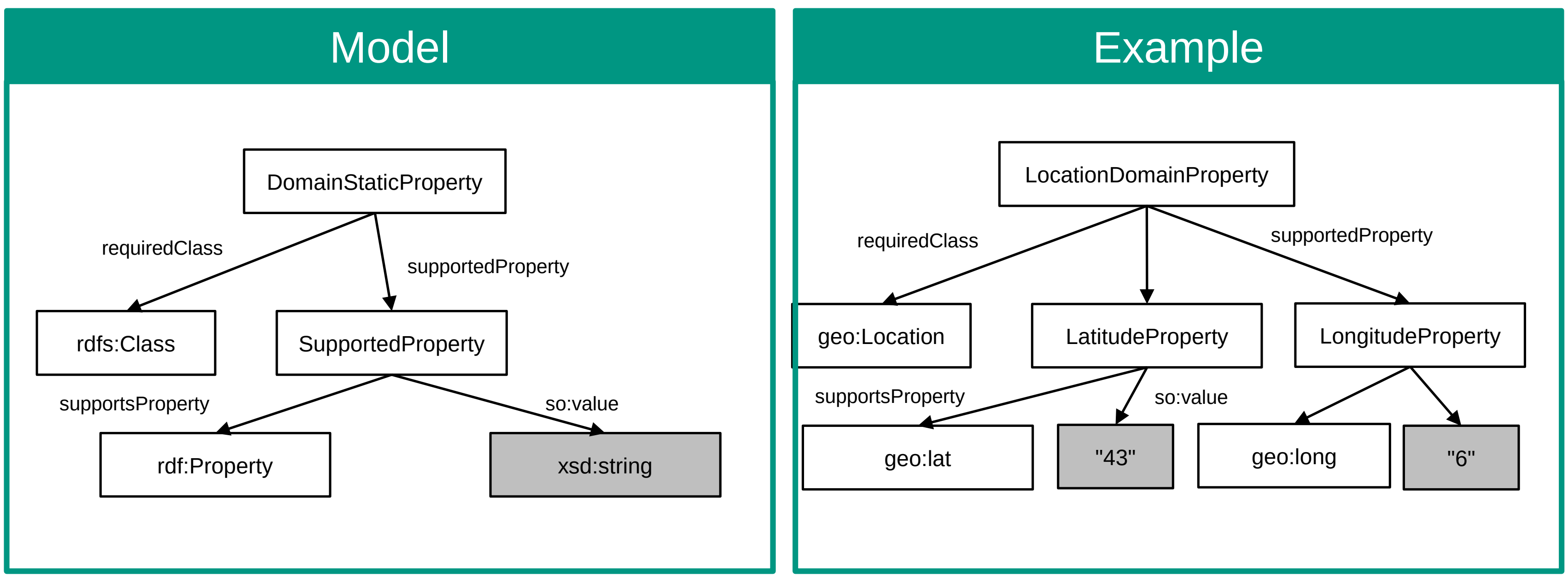


EPA Modeling: Domain Knowledge

Example:

- The geofence EPA requires a coordinate acting as the geofence center
- Locations of suppliers are stored separately as domain knowledge

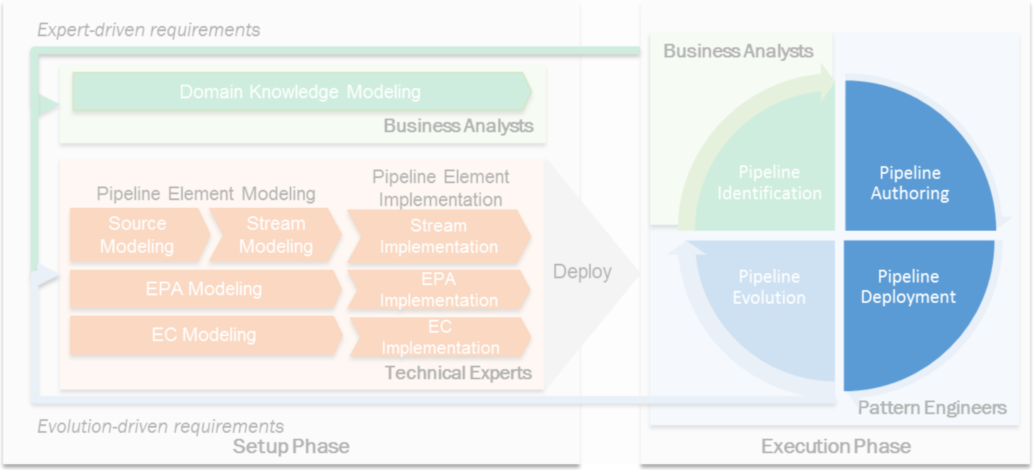




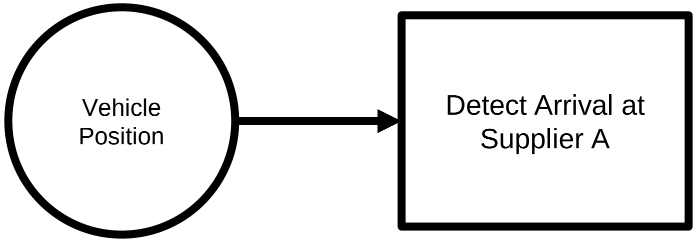
Provided in the setup phase | Provided in the execution phase

Outline

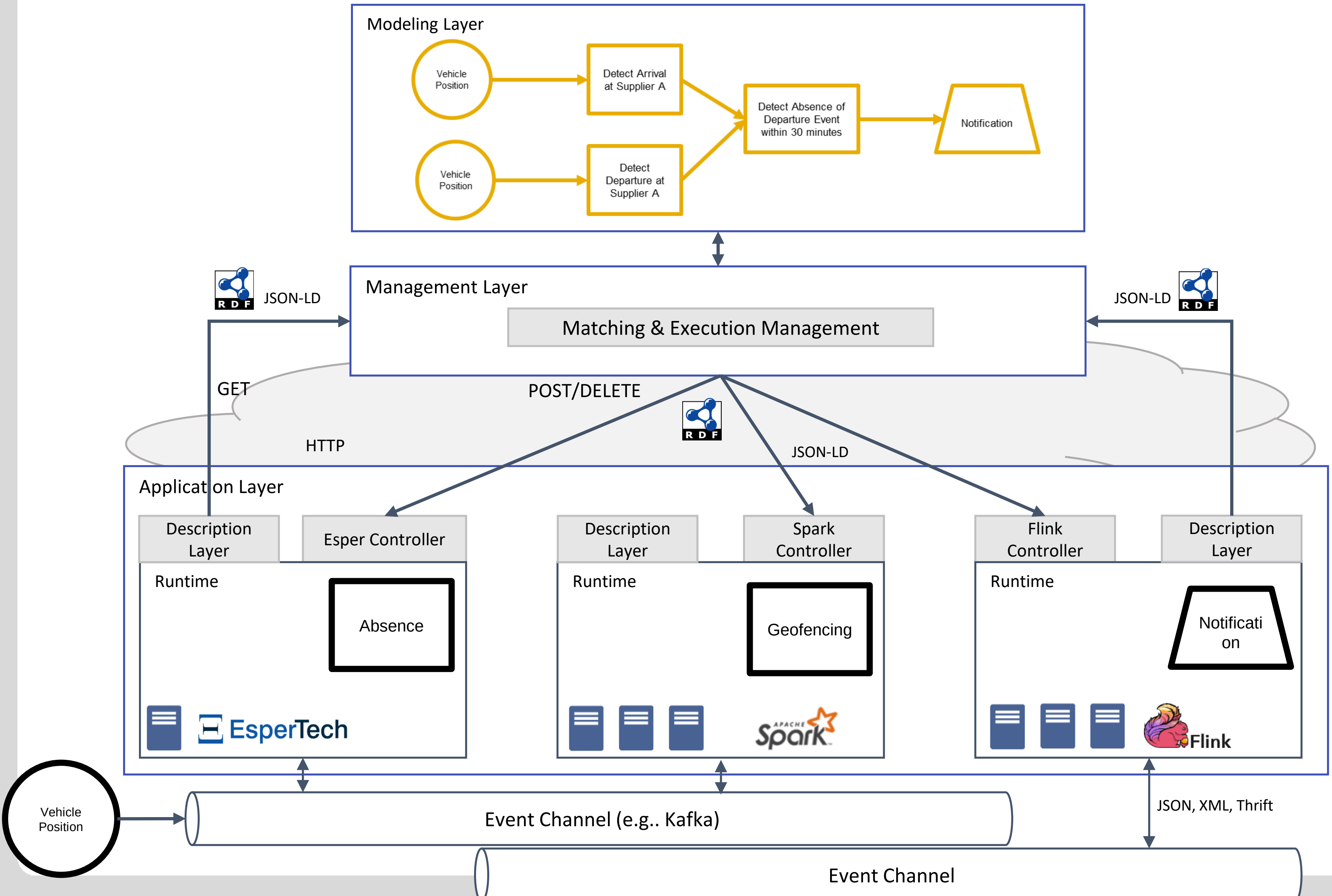
- Introduction & Motivation
- Problems
- **Development methodology for event-driven applications**
  - Overview
  - Modeling of data streams and processing elements
  - **Definition and execution of distributed processing pipelines**
- Implementation and evaluation
- Conclusion


Expert-driven requirements
Domain Knowledge Modeling
Business Analysts
Pipeline Element Modeling
Pipeline Element Implementation
Source Modeling
Stream Modeling
Stream Implementation
EPA Modeling
EPA Implementation
EC Modeling
EC Implementation
Technical Experts
Deploy
Evolution-driven requirements
Setup Phase
Business Analysts
Pipeline Identification
Pipeline Authoring
Pipeline Evolution
Pipeline Deployment
Pattern Engineers
Execution Phase

System


Vehicle Position
Detect Arrival at Supplier A

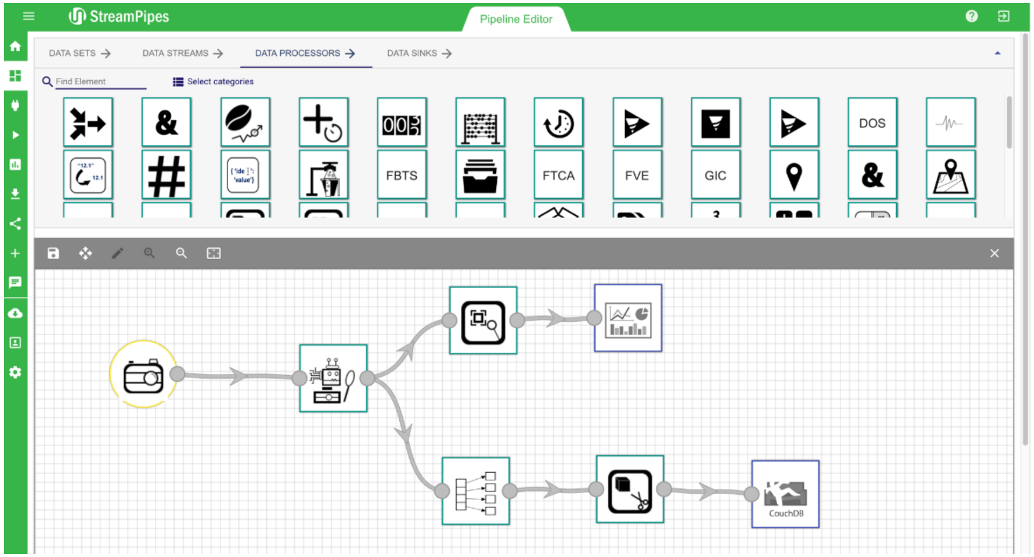
Geo-distributed processing pipelines



Outline

- Introduction & Motivation
- Problems
- Development methodology for event-driven applications
  - Overview
  - Modeling of data streams and processing elements
  - Definition and execution of distributed processing pipelines
- **Implementation and evaluation**
- Conclusion

StreamPipes: Open Source Self-Service Analytics



Semantics-based modeling layer

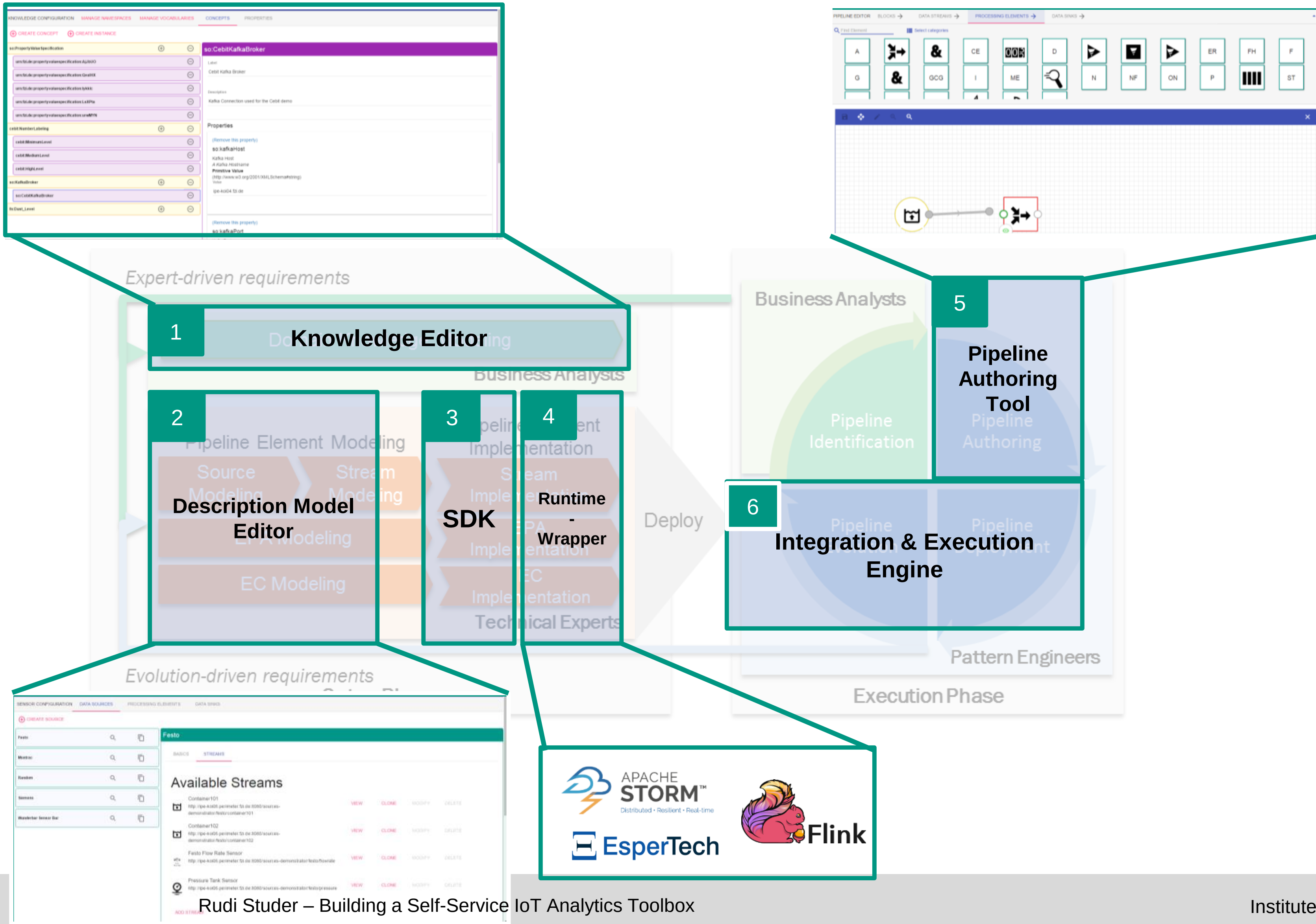
Arbitrary data formats and protocols

Geo-distributed execution

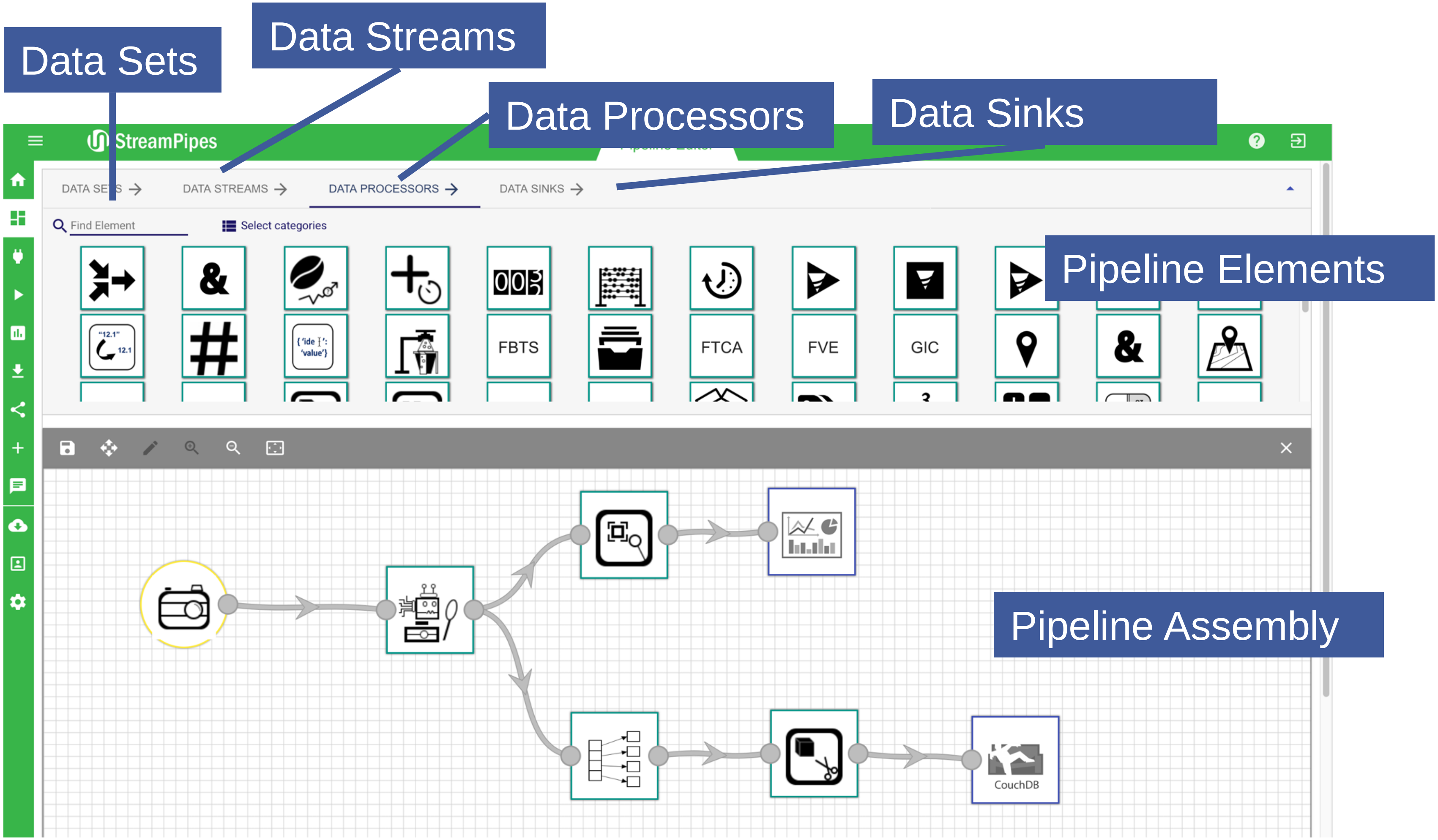
Exchangeable runtime wrappers (Single host or distributed)

Open Source: https://docs.streampipes.org

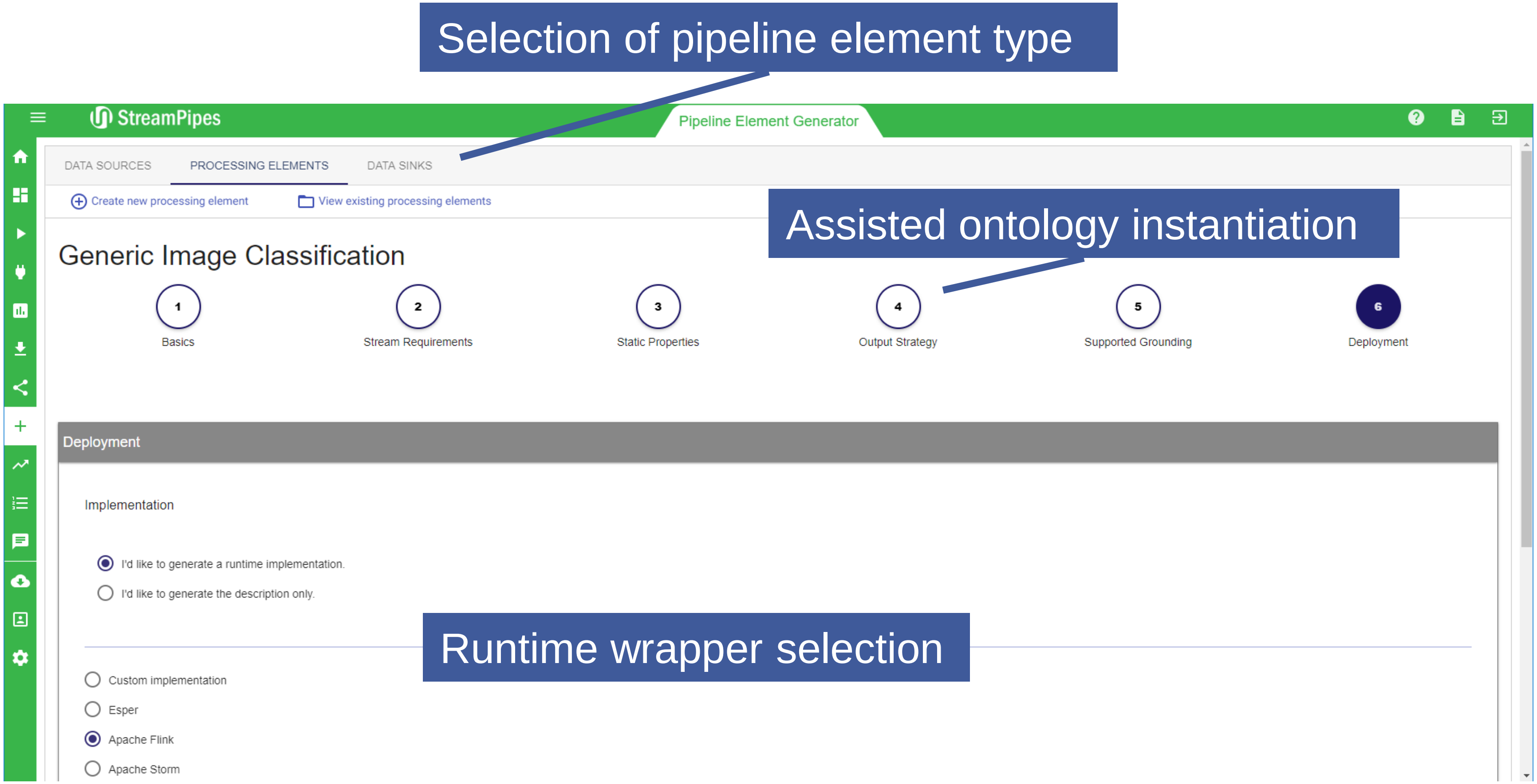
Tool Support: StreamPipes



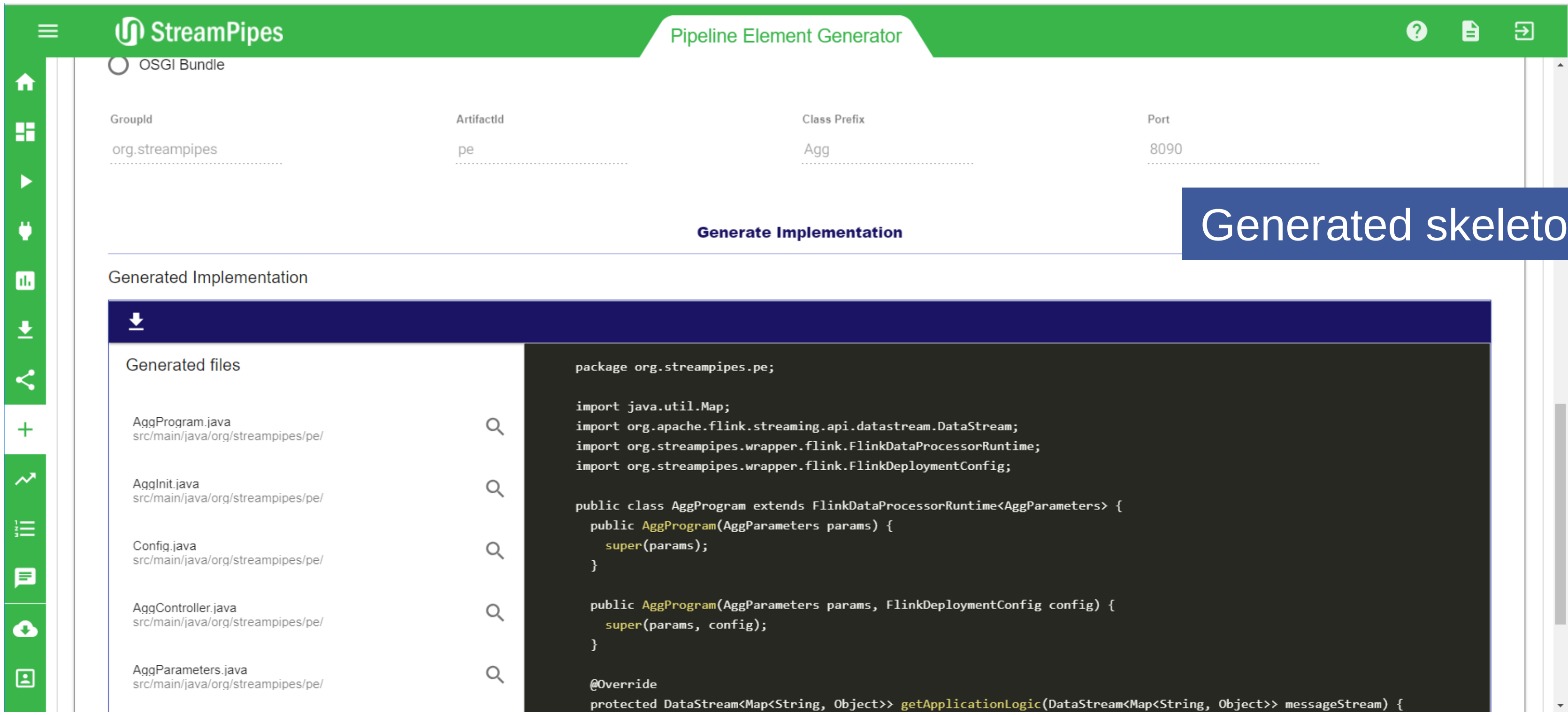
Pipeline Authoring Tool



Model Editor



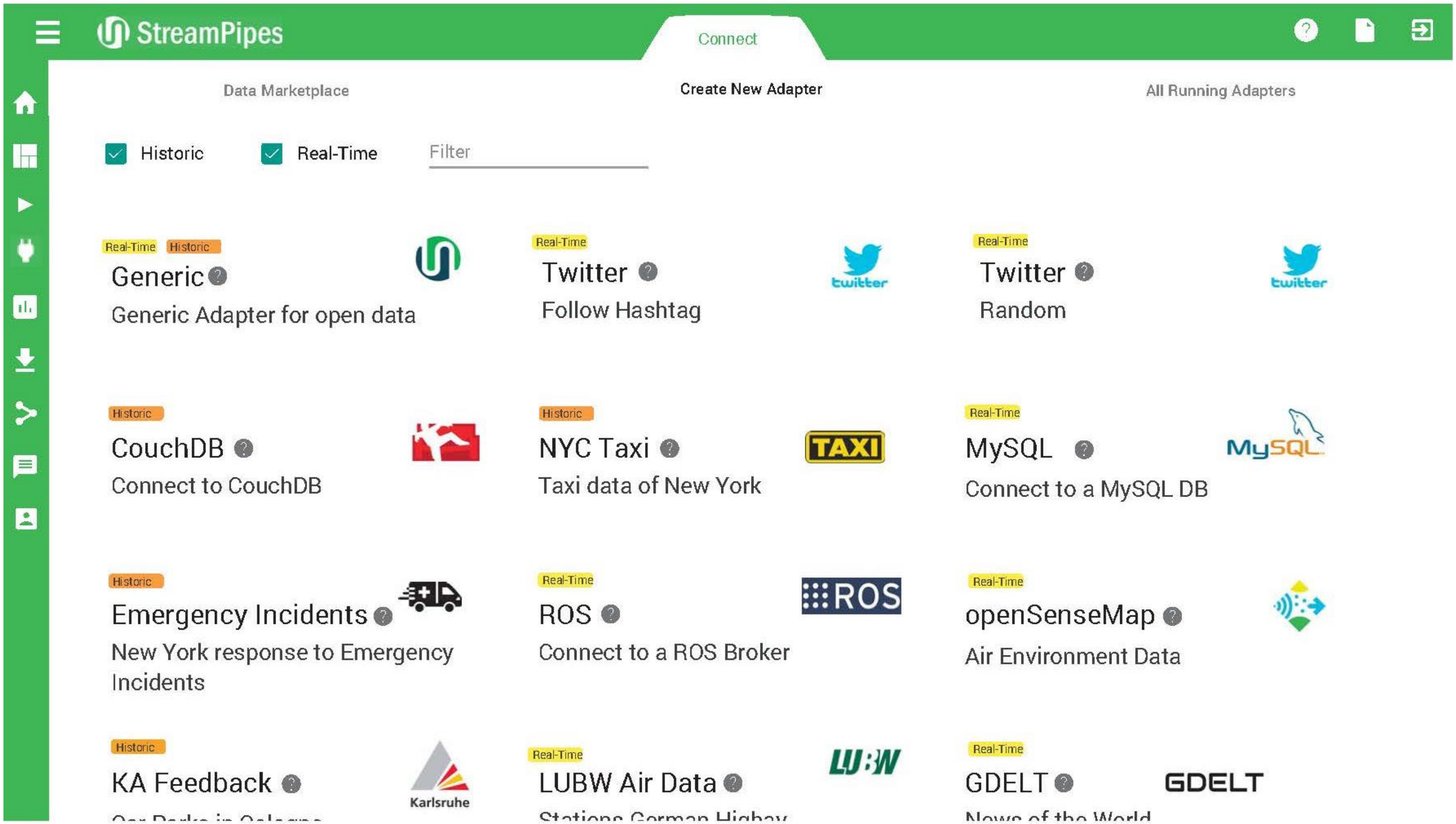
Model Editor



Generated skeleton

Only the application logic needs to be added to the generated code!

Adapter Libary: Access to existing data sources

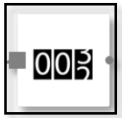


Self-Service Analytics with StreamPipes
Example pipeline elements

Data Streams

- **Sensors**
  - Machines (e.g., OPC)
  - MES
  - Environmental Data
- **Enterprise Applications**
  - Production plans
  - Databases
  - Master data
- **Human Sensors**
  - E.g., mobile applications

Data Processors



- **Complex Event Processing**
  - Pattern Detection
  - Aggregation
  - Filter
- **Advanced Analytics**
  - Vibration Detection
  - Predictions
  - Online Classification
- **Transformations**
  - Enrichment
  - Conversion
  - Replacement

Data Sinks

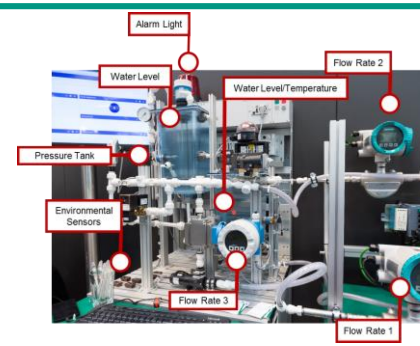
- **Visualization**
  - Charts
  - Geo
  - Tables
- **Notifications**
  - E-Mail
  - Dashboard
- **Storage / Messaging**
  - Elasticsearch
  - Kafka
  - HDFS
- **Actuators**

Case Studies

Overview

Production

ACM DEBS



smartAutomation

> 100 implemented pipeline elements

DEBS Grand Challenge

- Common evaluation testbed for event-based systems
- Dataset: New York City Taxi Data (130 M Events)
- ask: geospatial real-time analytics (profitable areas, frequent routes)

Results

- Re-usable pipeline elements
  - Distance
  - Spatial Aggregations
  - Ranking
- Solved Grand Challenge based on re-usable pipeline elements and pipelines
- Compared performance to baseline (single host and distributed systems)

Use Case: Image Processing for Logistics

Task

- Flexible quality tracking for inbound logistics
- Detect quality problems based on
  - Cheap IoT sensors (e.g., vibration)
  - Image data from stationary cameras
- Provide transport planners with a flexible solution to analyse various KPIs

Sensors

- Cameras
- IoT sensors (Bosch XDK, proprietary sensors)
  - Temperature
  - Light
  - Acceleration

Pipelines

- Take pictures of incoming products, classify them using deep neural networks and recognize quantity deviations
- Automatically read and recognize parcel labels
- Get insights on proper handling of sensitive products during transport

Use Case: Production Monitoring (3D Printing)

Task

- Quality monitoring of industrial-grade 3D printers with a large German manufacturer
- Correlate environment data to product quality
- Constantly monitor 3D printing parameters from different machines

Sensors

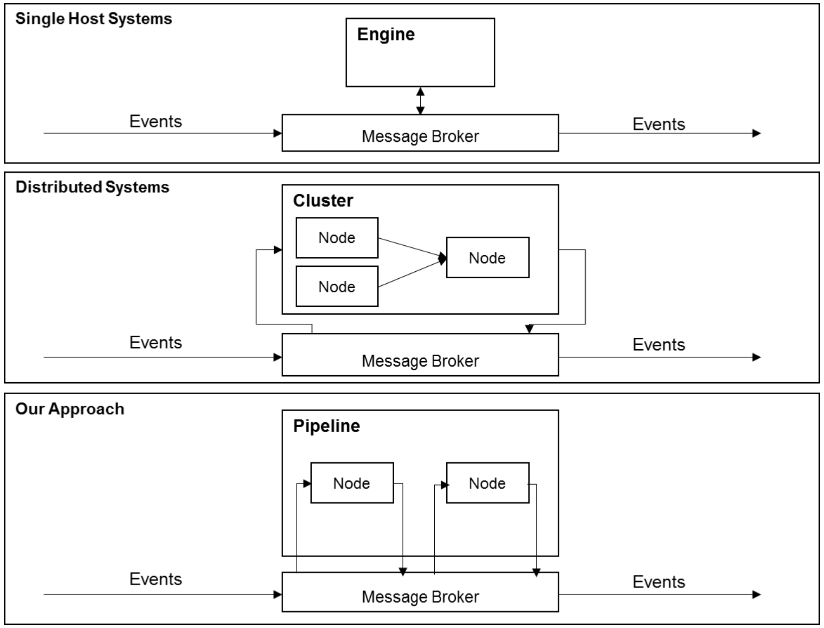
- 3D printers
  - Humidity
  - Temperature
  - Machine settings
- Environmental Sensors
  - Temperature
  - Humidity

Pipelines

- Constant monitoring: Environmental parameters are very critical for production outcome
- Early detection of potential quality issues based on correlations of internal & external sensors
- Benchmarking (compare outcome to other plants/facilities)

Performance

Performance compared to single host systems

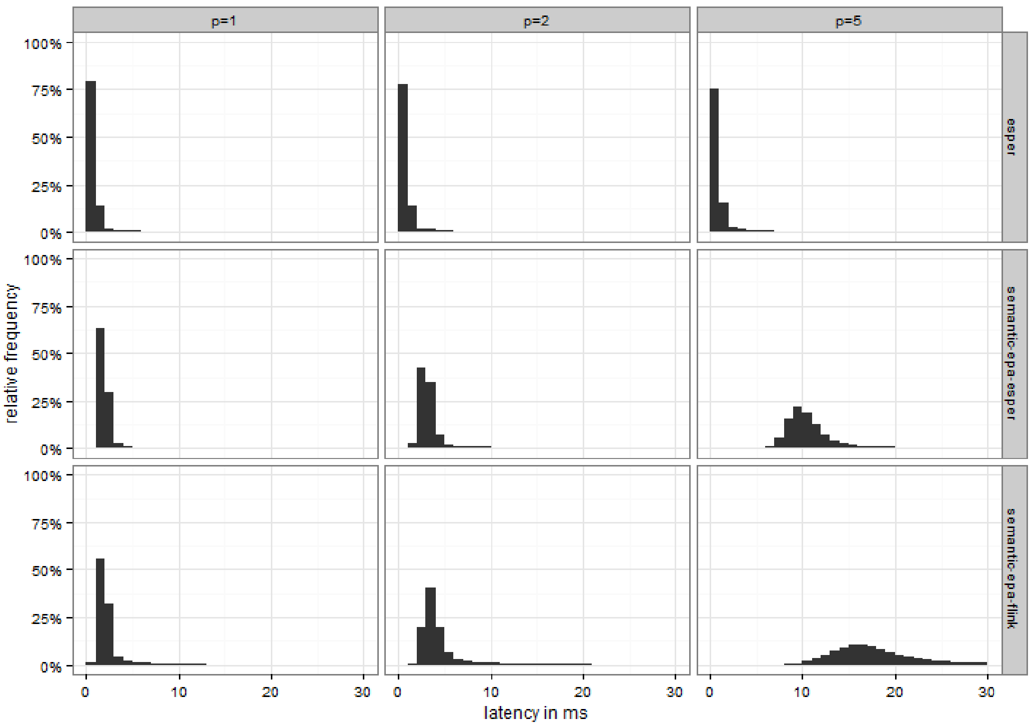


Setting

- **Hard- and software**
  - 3 Servers (24GB RAM, 12GB, 8GB)
  - CPU: 4x 2,3 Ghz
  - Kafka, Flink, Esper, ActiveMQ
- **Configurations**
  - esper: Single-host system
  - semantic-epa-esper: Distributed system (esper nodes)
  - semantic-epa-flink: Distributed system (flink nodes)
- **Experiment**
  - Pipeline size (p) 1, 2, 5
  - 1000 events/sec, 5000 events/sec
  - 100.000 events

Performance

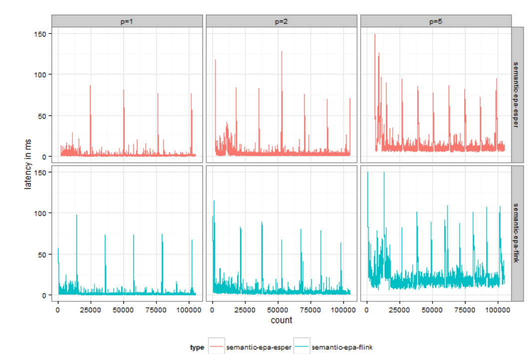
Performance: Latency



config	p=1	p=2	p=5
esper	1	1	1
semantic-epa-esper	2	3	10
semantic-epa-flink	2	4	16

latency [ms]

Performance: Latency/Time



Outline

- Introduction & Motivation
- Problems
- Development methodology for event-driven applications
  - Overview
  - Modeling of data streams and processing elements
  - Definition and execution of distributed processing pipelines
- Implementation and evaluation
- **Conclusion**

Conclusion

Summary

- **Self-Service Analytics Toolbox for the IoT**
  - **Methodology**: novel 2-phase development methodology for distributed event processing applications
  - **Semantic Models** and **vocabulary** for run-time independent description of event producers, processing agents and consumers
  - **System**: to define and execute distributed event processing pipelines
  - **Open source software artifact (StreamPipes)** as tool support

Outlook

StreamPipes

Intelligent IoT Analytics

Ongoing research directions

- **Semi-automatic pipeline generation (based on model repository & runtime data)**
- **Dynamic Edge Processing (automatically distribute nodes on edge units)**

Industrial-grade open source product

- **Extend pipeline element repository with reusable analytics operators**
- **Pipeline monitoring, logging, app system on top of pipelines**

Bibliography

- Dominik Riemer, Nenad Stojanovic, Ljiljana Stojanovic.
  A Methodology for Designing Events and Patterns in Fast Data Processing. In: *Proceedings of the 25th International Conference on Advanced Information Systems Engineering (CAiSE)*. Valencia, Spain, 2013.

- Dominik Riemer, Ljiljana Stojanovic, Nenad Stojanovic.
  SEPP: Semantics-Based Management of Fast Data Streams. In: *Proceedings of the 7th IEEE International Conference On Service-Oriented Computing and Applications (SOCA)*. Matsue, Japan, 2014.

- Dominik Riemer, Florian Kaulfersch, Robin Hutmacher, Ljiljana Stojanovic.
  StreamPipes: Solving the DEBS Grand Challenge with Semantic Stream Processing Pipelines. In: *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems (DEBS).* Oslo, Norway, 2015.

- Philipp Zehnder, Dominik Riemer:
  Modeling self-service machine learning agents for distributed stream processing. In: *Proceedings of the 2017 IEEE Conference on Big Data (IEEE BigData)*. Boston, USA, 2017.

Thank you!

Questions?

riemer@fzi.de
rudi.studer@kit.edu

streampipes.org

docs.streampipes.org

github.com/streampipes/streampipes-ce