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

Rudi Studer, Dominik Riemer
IC3K 2018, Seville, September 20, 2018
How to enable application specialists to create and execute IoT analytics applications based on distributed stream processing in a self-service manner?
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

- Energy consumption
- Machine data
- Logistics
- GPS
- Traffic sensors
- Dust particles
- Environmental data
- Enterprise applications

Rudi Studer – Building a Self-Service IoT Analytics Toolbox
Events

A record or an activity within a system

[Event [Luck02]]

GPS

Event Type

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

Event Object

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

Event Representation (JSON)

```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

[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

Event Processing Network

- Event Producer
- Event Channel
- Event Processing Agent
- Event Channel
- Event Consumer

- Filter
- Transformation
- Pattern Detect

- Translate
- Aggregate
- Split
- Compose

- Enrich
- Project

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

Tools: Examples

Stream Processing

Distributed Stream Processing

Complex Event Processing

Streambase

EsperTech

SAP

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

Vehicle Position

Detect Arrival at Supplier A

Detect Absence of Departure Event within 30 minutes

Notification

Vehicle Position

Detect Departure at Supplier A

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

Development Environment → Requirements → Change Requests → Business Analysts

Requirements:
- Change Requests
- Requirements

Deployment:
- Development Environment

Continuous Processing:
- Cluster/Engine
- Flow Control
- Dataflow

Results:
- External Systems
- Visualizations, Storage, Notifications

Developer

Rudi Studer – Building a Self-Service IoT Analytics Toolbox

Institute AIFB
Problem: Slow Development Cycles

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)

Detect Arrival

Detect Absence of Departure

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

Vehicle Position → Detect Arrival at Supplier A → Detect Absence of Departure Event within 30 minutes → Notification

Vehicle Position → Detect Departure at Supplier A
Requirement: Interoperability

Vehicle Position

MQTT

Detect Arrival at Supplier A

Kafka

Detect Absence of Departure Event within 30 minutes

AMQP

Detect Departure at Supplier A

Web socket

Notification
Requirement: Interoperability

Vehicle Position

MQTT

Detect Arrival at Supplier A

Kafka

Detect Absence of Departure Event within 30 minutes

AMQP

Detect Departure at Supplier A

Web socket

Notification

MQTT

MQTT
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.”
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

**Setup Phase**

- Development and grounding of re-usable event processing components

**Execution Phase**

- Definition and deployment of processing pipelines based on re-usable components
Methodology: Different Roles

Setup Phase

- Business Analysts
- Technical Experts

Execution Phase

- Business Analysts
- Pattern Engineers
Methodology: Two Tasks in Setup Phase

Setup Phase

- Domain Knowledge Modeling
  - Source Modeling
  - EPA Modeling
  - EC Modeling

Pipeline Element Modeling
- Stream Modeling
- EPA Implementation
- EC Implementation

Technical Experts

Business Analysts

Deployment

Execution Phase

- Pipeline Element Implementation
- Stream Implementation
- EPA Implementation
- EC Implementation
Methodology: Tasks

**Setup Phase**
- Domain Knowledge Modeling
- Pipeline Element Modeling
  - Source Modeling
  - EPA Modeling
  - EC Modeling
- Pipeline Element Implementation
  - Stream Modeling
  - EPA Implementation
  - EC Implementation

**Execution Phase**
- Business Analysts
  - Pipeline Identification
  - Pipeline Authoring
  - Pipeline Evolution
  - Pipeline Deployment
- Pattern Engineers
  - Pipeline Evolution
- Technical Experts
  - Pipeline Identification
  - Pipeline Authoring
  - Pipeline Deployment

**Expert-driven requirements**
- Domain Knowledge Modeling

**Evolution-driven requirements**
- Pipeline Element Modeling
- Pipeline Element Implementation

*Expert-driven requirements*

*Rudi Studer – Building a Self-Service IoT Analytics Toolbox*
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

Description Layer

Schema
- Data type, runtime name, semantics

Quality
- e.g., Frequency, Latency, Measurement Unit

Grounding
- Run-time format, run-time protocol

Semantic Description

Vehicle Position

Run-time layer

Format: JSON
Protocol: MQTT

```
{
    "timestamp": 1453478160,
    "vehicleId": "ID5",
    "latitude": 73.5,
    "longitude": 4.2
}
```

```
{
    "timestamp": 1453478170,
    "vehicleId": "ID5",
    "latitude": 73.5,
    "longitude": 4.2
}
```

```
{
    "timestamp": 1453478180,
    "vehicleId": "ID5",
    "latitude": 73.3,
    "longitude": 4.1
}
```
Stream Modeling: Vocabulary

Event Schema — Event Property — rdf:Property

Property Quality — so:DataType — Property Quality

Event Stream — Event Property — so:Text

Stream Quality — ssn:Frequency — ssn:Accuracy

Event Property — runtimeName — valueSpecification

Value Specification — so:Text

Semantic Event Producer — produces — ssn:Platform

Event Stream — produces — ssn:Platform

Stream Grounding — hasGrounding — Event Stream

Stream Quality — hasStreamQuality — Event Stream

Stream Grounding — hasTransportProtocol — TransportFormat

TransportFormat — KafkaProtocol — MQTPProtocol

Binary Format — ThriftFormat — AvroFormat

Value Specification — so:QuantitativeValue

Enumeration — so:Text

ssn:Platform — hasEventProperty — Event Stream

hasSchema — Event Stream — Event Property

so:DataType — hasGrounding — Event Stream

transportFormat — Event Stream — Event Property

ssn:Property — domainProperty — rdf:Property

runtimeName — Event Property — rdf:Property

valueSpecification — so:Property

ssn:Platform — produces — ssn:Platform

ssn:Platform — hasGrounding — Event Stream

ssn:Platform — hasTransportProtocol — TransportFormat

ssn:Platform — hasTransportFormat — Binary Format

Rudi Studer – Building a Self-Service IoT Analytics Toolbox

Institute AIFB
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**

```
Position Schema
  +-------------------------------------+
  | Timestamp Property                  |
  +-------------------------------------+
  | Latitude Property                   |
  +-------------------------------------+

  Timestamp Property
  +---------------------+
  | so:Number           |
  +---------------------+
  | sao:Timestamp       |
  +---------------------+

  Latitude Property
  +---------------------+
  | so:Float             |
  +---------------------+
  | geo:lat              |
  +---------------------+

  Latitude Value Specification
  +---------------------+
  | 40.07                |
  +---------------------+
  | 43.07                |
```
Semantic Modeling of Event Processing Agents

Detect Arrival at Supplier A
EPA Modeling: Requirements

Semantic Description
- Detect Arrival at Supplier A

Description layer
- **Input**
  - Minimum required schema, quality requirements, supported transport properties
- **Output**
  - Event transformation, output schema
- **Static Data**
  - Human input, required domain knowledge

Abstraction

Example
- **Minimum required schema**
  - geo:lat
  - geo:long
- **Transport properties**
  - Protocol: MQTT
  - Format: JSON, Thrift
- **Output schema**
  - AppendOutput
  - Additional Properties: enterTime
- **User input**
  - Geofence Operation
  - Geofence Center
  - Geofence Radius
  - Mapping

Example:
- Geofencing

Rudi Studer – Building a Self-Service IoT Analytics Toolbox
EPA Modeling: Vocabulary

Semantic Event Processing Agent

EventStream

Static Property

Output Strategy

StreamGrounding

Supported Property

EventProperty

Mapping Property

Domain Static Property

Single Value Property

Multi Value Property

Selection Static Property

Mapping Property On

Mapping Property Not

Operatio

Append Output

Keep Output

Fixed Output

Transform Output

Custom Output

Supported Property

Uri Property Mapping

replaceWith replaceFrom

requiresStream

hasStaticProperty

hasOutputStrategy

supportedGrounding

hasSchema

mapsTo

minPropertyQualityReq

runtimeType

property

domainProperty

rdf:Property

rdf:Datatype

Property Quality Requirement

requiredClass

xsd:string

xsd:propertyValue Specification

valueSpecification

replaceProperty

supportedProperty

EventSchema

EventProperty
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

Detect Arrival at Supplier A

Required static data (e.g., location of suppliers)

Knowledge Base

Model

DomainStaticProperty

requiredClass

supportedProperty

rdfs:Class

SupportedProperty

supportedProperty

so:value

rdf:Property

xsd:string

Example

LocationDomainProperty

requiredClass

supportedProperty

geo:Location

LatitudeProperty

LongitudeProperty

geo:lat

so:value

"43"

geo:long

"6"

---

Rudi Studer – Building a Self-Service IoT Analytics Toolbox
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
System

Vehicle Position

Detect Arrival at Supplier A
Geo-distributed processing pipelines

Modeling Layer

- Vehicle Position
  - Detect Arrival at Supplier A
  - Detect Absence of Departure Event within 30 minutes
  - Notification

Management Layer

- Matching & Execution Management

Application Layer

- Description Layer
- Esper Controller
  - Runtime
  - Absence
- Spark Controller
  - Runtime
  - Geofencing
- Flink Controller
  - Runtime
  - Notication

Event Channel (e.g., Kafka)

- JSON, XML, Thrift
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

1. Knowledge Editor
2. Description Model Editor
3. SDK
4. Runtime Wrapper
5. Pipeline Authoring Tool
6. Integration & Execution Engine

Rudi Studer – Building a Self-Service IoT Analytics Toolbox
Pipeline Authoring Tool

Data Sets ➔ Data Streams ➔ Data Processors ➔ Data Sinks ➔ Pipeline Elements

Pipeline Assembly

Rudi Studer – Building a Self-Service IoT Analytics Toolbox
Model Editor

Selection of pipeline element type

Assisted ontology instantiation

Runtime wrapper selection
Institute AIFB

Rudi Studer – Building a Self-Service IoT Analytics Toolbox

Model Editor

Generated skeleton

Only the application logic needs to be added to the generated code!
Adapter Library: Access to existing data sources

Data Marketplace

- Historic
- Real-Time

Filter

Create New Adapter

All Running Adapters

- Generic
  - Generic Adapter for open data

- CouchDB
  - Connect to CouchDB

- NYC Taxi
  - Taxi data of New York

- Emergency Incidents
  - New York response to Emergency Incidents

- KA Feedback
  - Karlsruhe Radio

- LUBW Air Data
  - Stations German Highay

- openSenseMap
  - Connect to a ROS Broker

- ROS
  - Connect to a ROS Broker

- MySQL
  - Connect to a MySQL DB

- Twitter
  - Follow Hashtag

- Twitter
  - Random

- GDELT
  - News of the World
### Self-Service Analytics with StreamPipes

**Example pipeline elements**

<table>
<thead>
<tr>
<th>Data Streams</th>
<th>Data Processors</th>
<th>Data Sinks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sensors</strong></td>
<td><strong>Complex Event Processing</strong></td>
<td><strong>Visualization</strong></td>
</tr>
<tr>
<td>- Machines (e.g., OPC)</td>
<td>- Pattern Detection</td>
<td>- Charts</td>
</tr>
<tr>
<td>- MES</td>
<td>- Aggregation</td>
<td>- Geo</td>
</tr>
<tr>
<td>- Environmental Data</td>
<td>- Filter</td>
<td>- Tables</td>
</tr>
<tr>
<td><strong>Enterprise Applications</strong></td>
<td><strong>Advanced Analytics</strong></td>
<td><strong>Notifications</strong></td>
</tr>
<tr>
<td>- Production plans</td>
<td>- Vibration Detection</td>
<td>- E-Mail</td>
</tr>
<tr>
<td>- Databases</td>
<td>- Predictions</td>
<td>- Dashboard</td>
</tr>
<tr>
<td>- Master data</td>
<td>- Online Classification</td>
<td><strong>Storage / Messaging</strong></td>
</tr>
<tr>
<td><strong>Human Sensors</strong></td>
<td><strong>Transformations</strong></td>
<td>- Elasticsearch</td>
</tr>
<tr>
<td>- E.g., mobile applications</td>
<td>- Enrichment</td>
<td>- Kafka</td>
</tr>
<tr>
<td>-</td>
<td>- Conversion</td>
<td>- HDFS</td>
</tr>
<tr>
<td>-</td>
<td>- Replacement</td>
<td><strong>Actuators</strong></td>
</tr>
</tbody>
</table>

- **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)

<table>
<thead>
<tr>
<th>Task</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Quality monitoring of industrial-grade 3D printers with a large German manufacturer</td>
<td>• 3D printers</td>
</tr>
<tr>
<td></td>
<td>• Humidity</td>
</tr>
<tr>
<td></td>
<td>• Temperature</td>
</tr>
<tr>
<td></td>
<td>• Machine settings</td>
</tr>
<tr>
<td>• Correlate environment data to product quality</td>
<td>• Environmental Sensors</td>
</tr>
<tr>
<td></td>
<td>• Temperature</td>
</tr>
<tr>
<td>• Constantly monitor 3D printing parameters from different machines</td>
<td>• Humidity</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>config</th>
<th>p=1</th>
<th>p=2</th>
<th>p=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>esper</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>semantic-epa-esper</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>semantic-epa-flink</td>
<td>2</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

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
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


Thank you!

Questions?

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

streampipes.org

docs.streampipes.org

github.com/streampipes/streampipes-ce