Big Data for everyone
Modeling of Data Processing Pipelines for the Industrial Internet of Things

Tutorial@IEEE BigData 2018

Ljiljana Stojanovic, Nenad Stojanovic, Philipp Zehnder, Dominik Riemer
Agenda

1. Industrial IoT
   + Demo!

2. Stream Processing & StreamPipes
   + Demo!

3. Analytics
   + Demo!

4. Future: Digital Twins
About us
Tutorial Presenters

Ljiljana Stojanovic
Nenad Stojanovic
Dominik Riemer
Philipp Zehnder
Fraunhofer IOSB
Nissatech
FZI Research Center for Information Technology
**Facts**

www.iosb.fraunhofer.de

- **Operational costs in 2017**: 56 Mio. €
- **Permanent employees**: 520
  - of which scientists and engineers: 330
- **Additional**: student interns 180

**Professorship**

»Vision and Fusion Laboratory«, KIT
»Optronik«, KIT
»Energy optimization«, TU Ilmenau
»Kognitive Automation«, Universität Bielefeld (*in progress*)
Industrial IoT
IloT – Industrial Internet of Things

Dr. Ljiljana Stojanovic (Fraunhofer IOSB)
The Industrial Internet of Things (IIoT) is the use of the Internet of Things (IoT) in the industrial domain with the goal of monitoring and controlling production processes.

The manufacturing sector is expected to be one of the top adopters of IoT technologies.

Leveraging the power of advanced sensing technologies, applications, such as remote monitoring, anomaly detection, diagnosis, and control of processes and assets have already gained rapid popularity in manufacturing industries.

While huge progress has been made on making assets "smarter" and production more efficient over recent years, the full potential of the IIoT has not yet been exploited sufficiently.
IIOT CHALLENGES

- The available **bandwidth** for the data transmission is usually not sufficient for the vast amount and frequency of data created by IIoT devices.
  - For instance, a factory can produce an 1 terabyte of data per day.
  - These numbers will only grow with the continued digitalization of factories.

- Manufacturing domain has very demanding ultra-low **latency** requirements for processing IIoT data.
  - As IIoT deals with physical entities, reaction time cannot be arbitrary.
  - Failing to react with the proper latency can lead to system instability, infrastructure damage, or even risk to human operators.
IIOT CHALLENGES

- **Connectivity** (and consequently the availability of services) is hard to guarantee for several IIoT applications
  - Industrial facilities for process manufacturing, oil and gas, and power generation are run around-the-clock
  - They have to ensure that their lines are always up and running efficiently 24x7x365
  - However, IIoT systems connected to the public cloud may be affected by unreliable connectivity, resulting in loss of availability of data

- **Security** plays very important role
  - Existing legacy operational technology (OT) is often designed with no thought for the modern day information technology (IT) connectivity and security requirements
IIOT ECOSYSTEM

Manufacturing in the Cloud – the dawn of digital services, Frost & Sullivan, October 2016
**BARRIERS**

- **Lack of expertise** (e.g. the future of IIoT will require a skilled workforce who are not just experts in industrial processes but also have IT know-how)
- **Large capital investments** (e.g. to retrofit existing machines so that they can be sensor enabled) etc.
- **Lack of common protocols or standards**, which are needed for diverse devices and machines to inter-operate seamlessly, sharing data and information
- **Security and availability of services** will be additional barriers to adoption of IIoT

The future of industrial Internet will require a skilled workforce who are not just experts in industrial processes but also have IT know-how.
OUR VISION - BIG DATA FOR EVERYONE:
MODELING OF DATA PROCESSING PIPELINES FOR THE INDUSTRIAL INTERNET OF THINGS

- Data scientist – person with ability to find and interpret rich data sources; create visualizations to aid in understanding data; build mathematical models using the data; and present and communicate the data insights
- People with such a combination skills are quite rare
- Overcoming this shortage will be the critical success factor not just for new big data technologies but also for a broad application and uptake of big data analytics
- By allowing people to describe the desired analytics in a high level declarative language, we will enable the domain experts to perform the analysis of big data on their own
  - More intuitive solutions supporting the development of real-time applications
  - Methods and tools enabling flexible modeling of real-time and batch processing pipelines by domain experts
AGENDA

- Generic IIoT platform capabilities
  - Monitor IIoT event streams
  - Enable data aggregation
  - Enable specialized analysis
- Engage back-end IT systems/services
- Control (and even change) the end points to support IoT solutions
- Support application development

THE PROBLEM: BOTTLENECK IN TODAY’S INDUSTRIAL IOT SOLUTIONS
WHAT DO WE OFFER TO THE CUSTOMERS?

Integration Tools, IoT platforms and solutions for manufacturing

Fraunhofer PLUGandWORK™ allows easier manufacturing integration by translating machine language into open international standards of IEC.

Common communication level: transferring language into OPC UA protocol

Common language level: an enormous database for translating proprietary system languages

Intelligent services
SEMANTIC EDGE

- Plug-in/out management layer enables on demand plug-in/out based on out-of-the-box connectivity
- Knowledge extraction layer creates machine-understandable representation of the raw sensor data
- Smart integration layer ensures data fusion in real-time near to the data sources by dealing with both syntactic heterogeneity and semantic heterogeneity
- Intelligent service layer consists of many services for building advanced applications suitable for the edge
- Semantic models ensure to achieve common, shared understanding across the existing OT and IT systems
WITHIN A FACTORY: PLUG & WORK BASED ON USB-LIKE INTERFACES

- (Self-)Description via AutomationML (IEC 62714)
- Communication/Data-Management/ Validation via OPC UA (IEC 62541)
- AML2UA: https://aml2ua.iosb.fraunhofer.de/
  - The AML-UA-Converter compiles AutomationML models in UA-XML-Nodesets, according to the rules of the [DIN SPEC 16592 – Combining OPC Unified Architecture and Automation Markup Language](http://example.com)
  - The converted UA-XML-Nodeset is used for creating C-Code for an aggregating OPC UA Server based on the [open62541](http://example.com) toolkit
  - For DataVariables of type OPC UA available in AutomationML an integrated OPC UA Client with subscriptions to the underlying OPC UA Server is integrated in the aggregating server automatically
REAL-TIME ANALYTICS

- An event is defined as “something that occurs, happens or changes the current state of affairs”
- Complex event processing looks for a combination of certain types of events (i.e. patterns or situation of interest) to create a higher-level business event
- A pattern is defined as a template which specifies a combination of event
- At runtime, this template is matched against the event stream and outputs an event once the template satisfies the event sequence
REAL-TIME ANALYTICS

The client (Android) app offering an intuitive graphical editor for creating and editing the patterns.

A server part based on the WSO2 Siddhi to discover patterns in real-time and send notification or even perform an action.

App for modelling, management and notification.

The server manages all users, sensors, patterns and hosts the Siddhi engine for the actual pattern recognition.
IMPLEMENTATION DETAILS

- **Model**: Contains the representation of CEP patterns
- **API**: Contains interfaces used for communication between the client and the server. It is based on a reactive event flow by being designed around the RxJava library and is used by the client module.
- **Compiler**: Contains interfaces describing a compiler which translates CEP patterns to a CEP language. It contains a Siddhi-specific implementation of the compiler as well.
- **JSON-Converter**: Responsible for converting Java objects to the JSON format and vice-versa. This is used for transferring data (e.g. patterns) between the client and the server.
- **Server**: Contains the server which stores and manages the patterns (and their deployment), users, sensors.
- **Client**: Contains the client Android app. The app uses the model, compiler and JSON converter as well as the API.
AN EXAMPLE

```python
from every(e1=TempStream) -> e2=TempStream[e1.roomNo==roomNo and (e1.temp + 5) <= temp] within 10 min
select e1.roomNo, e1.temp as initialTemp, e2.temp as finalTemp
insert into AlertStream;
```
PRODUCTION CONTROL SYSTEMS – EXAMPLE
DAIMLER BREMEN

- System for managing and monitoring the entire production of the passenger car ‘C-Class’ factory in Bremen
- It monitors and controls about 450 PLCs and 2,000 pieces of equipment and provides information to more than 1,560 users in body shop, paint shop and assembly
STANDARDISATION

- Fraunhofer is a member of expert panel of the Standardization Council Industrie 4.0 (SCI 4.0)
- Fraunhofer contributes to several working groups of the German Platform Industrie 4.0, e.g. on reference architectures, semantics, security and interoperability
- Fraunhofer co-chairs the IIC TG Group 'Distributed Data & Interoperability Management' working on semantic interoperability in IIoT systems
- Fraunhofer participates in the IIC TG 'Digital Twin Interoperability'
- Fraunhofer leads the approved IIC testbed Smart Factory Web
- Fraunhofer is a founding member of the IDS
Thank you for your attention!

Dr. Ljiljana Stojanović
Head of “Smart Factory Systems”
Fraunhofer IOSB, Fraunhoferstr. 1
76131 Karlsruhe, Germany
Tel.: +49-721-6091-287
Email: ljiljana.stojanovic@iosb.fraunhofer.de
Stream Processing & StreamPipes
How to enable **application specialists** to **create** and **execute** distributed **stream processing applications** 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
Event [Luck02]
A record or an activity within a system

Event Representation (JSON)
{
  "timestamp" : 1453478150,
  "vehicleId" : "ID5",
  "latitude" : 48.94,
  "longitude" : 8.40
}
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

[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-Systeme 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

- Developer
- Business Analysts
- Development Environment
- Cluster/Engine
- Deployment
- Continuous Processing
- Results
- External Systems
  - Visualizations, Storage, Notifications
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
Objective
Less development effort for Big Data Analytics

Development process today
Developer

Development process with StreamPipes
Application Specialist

Real-Time Big Data Infrastructure
Flink, Spark, Kafka

manual deployment
automatic deployment
Methodology: Phases

**Preparation**
Development and grounding of re-usable event processing components

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

**Setup Phase**

**Execution Phase**
Methodology: Different Roles

Setup Phase:
- Technical Experts

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

- **Domain Knowledge Modeling**
  - Business Analysts

- **Pipeline Element Modeling**
  - Source Modeling
  - Stream Modeling
  - EPA Modeling
  - EC Modeling

- **Pipeline Element Implementation**
  - Stream Implementation
  - EPA Implementation
  - EC Implementation

- **Technical Experts**

- **Setup Phase**

- **Execution Phase**

- **Deploy**
Methodology: Tasks

Expert-driven requirements

- Domain Knowledge Modeling
- Business Analysts
- Pipeline Element Modeling
  - Source Modeling
  - EPA Modeling
  - EC Modeling
- Pipeline Element Implementation
  - Stream Modeling
  - EPA Implementation
  - EC Implementation
- Technical Experts

Evolution-driven requirements

Setup Phase

- Expert-driven requirements
- Business Analysts
- Pipeline Element Modeling
  - Source Modeling
  - EPA Modeling
  - EC Modeling
- Pipeline Element Implementation
  - Stream Modeling
  - EPA Implementation
  - EC Implementation
- Technical Experts

Execution Phase

- Business Analysts
  - Pipeline Identification
  - Pipeline Authoring
  - Pipeline Evolution
  - Pipeline Deployment
- Pattern Engineers
  - Pipeline Identification
  - Pipeline Authoring
  - Pipeline Evolution
  - Pipeline Deployment

Deploy
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
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

**Semantic Description**

- **Vehicle Position**

**Description Layer**

- **Schema**
  - Data type, runtime name, semantics
- **Quality**
  - e.g., Frequency, Latency, Measurement Unit
- **Grounding**
  - Run-time format, run-time protocol

**Run-time layer**

- **Format:** JSON
- **Protocol:** MQTT

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

{  
  "timestamp" : 1453478150  
  "vehicleId" : "ID5"  
  "latitude" : 73.6  
  "longitude" : 4.3  
}

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

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

- Semantic Event Producer
- Event Stream
- Event Schema
- Stream Grounding
- Stream Quality
  - rdf:Property
  - so:DataType
  - so:Text
  - TransportFormat
  - TransportProtocol

- Event Property
  - propertyQuality
  - domainProperty
  - rangeProperty
  - rdf:Property
  - so:DataType
  - Value Specification
  - ssn:Text

- Property Quality
  - ssn:Frequency
  - ssn:Accuracy
  - ssn:Precision
  - ssn:Resolution
  - so:QuantitativeValue
  - Enumeration

- Stream Grounding
  - TransportFormat
  - Text Format
  - Binary Format
  - AvroFormat
  - ThriftFormat
  - KafkaProtocol
  - MQTTProtocol
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**

Example Position Schema:

<table>
<thead>
<tr>
<th>Property</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timestamp</td>
<td></td>
</tr>
<tr>
<td>Latitude</td>
<td></td>
</tr>
<tr>
<td>Latitude Value</td>
<td></td>
</tr>
</tbody>
</table>

Example Diagram:

```
Position Schema
    hasEventProperty
    └── Timestamp Property
        runtimeName: timestamp
        runtimeType: so:Number
    └── Latitude Property
        runtimeName: latitude
        runtimeType: so:Float
    Latitudes Value Specification
       (so:minValue: 40.07) (so:maxValue: 43.07)
```

Diagram shows the relationship between different properties and their specifications in a position schema.
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

Example
- Geofencing
  - 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
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

![Diagram](image)

Model

- **DomainStaticProperty**
- **SupportedProperty**
  - `rdfs:Class`
  - `supportedProperty`
  - `requiredClass`
  - `supportsProperty`
  - `rdf:Property`
  - `xsd:string`

Example

- **LocationDomainProperty**
  - `geo:Location`
  - `LatitudeProperty`
  - `LongitudeProperty`
  - `requiredClass`
  - `supportedProperty`
  - `geo:lat`
  - `"43"`
  - `geo:long`
  - `"6"`

Diagram:

- **Detect Arrival at Supplier A**
- **Required static data (e.g., location of suppliers)**
- **Knowledge Base**

Legend:

- 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
Vehicle Position

Detect Arrival at Supplier A
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
Pipeline Authoring Tool

Data Sets

Data Streams

Data Processors

Data Sinks

Pipeline Elements

Pipeline Assembly
Model Editor

1. Selection of pipeline element type
2. Assisted ontology instantiation
3. Runtime wrapper selection
Model Editor

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

StreamPipes

Data Marketplace

Create New Adapter

Historic  Real-Time

Filter

Real-Time

Generic
Generic Adapter for open data

Real-Time

Twitter
Follow Hashtag

Real-Time

Twitter
Random

Real-Time

CouchDB
Connect to CouchDB

Real-Time

NYC Taxi
Taxi data of New York

Real-Time

MySQL
Connect to a MySQL DB

Real-Time

Emergency Incidents
New York response to Emergency Incidents

Real-Time

ROS
Connect to a ROS Broker

Real-Time

openSenseMap
Air Environment Data

Historic

KA Feedback
Fuel Data in Karlsruhe

Real-Time

LUBW Air Data
Station in German Highway

Real-Time

GDELT
News of the World
### 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**
Use Case: Image Processing for Logistics

<table>
<thead>
<tr>
<th>Task</th>
<th>Sensors</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Flexible quality tracking for inbound logistics</td>
<td>▪ Cameras</td>
</tr>
<tr>
<td>▪ Detect quality problems based on</td>
<td>▪ IoT sensors (Bosch XDK, proprietary sensors)</td>
</tr>
<tr>
<td>▪ Cheap IoT sensors (e.g., vibration)</td>
<td>▪ Temperature</td>
</tr>
<tr>
<td>▪ Image data from stationary cameras</td>
<td>▪ Light</td>
</tr>
<tr>
<td>▪ Provide transport planners with a flexible solution to analyse various KPIs</td>
<td>▪ Acceleration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pipelines</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Take pictures of incoming products, classify them using deep neural networks and recognize quantity deviations</td>
<td></td>
</tr>
<tr>
<td>▪ Automatically read and recognize parcel labels</td>
<td></td>
</tr>
<tr>
<td>▪ Get insights on proper handling of sensitive products during transport</td>
<td></td>
</tr>
</tbody>
</table>
# 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)
Discussion