„BABEȘ - BOLYAI" UNIVERSITY OF CLUJ-NAPOCA
DOCTORAL SCHOOL OF ECONOMICS AND BUSINESS ADMINISTRATION

DOCTORAL THESIS

-Summary-

Towards a Semantic Edge Processing of Sensor Data in Smart Environments: a Design Science Approach

Scientific Advisor:
Prof. PhD Nicolae Tomai

PhD Candidate:
Zălhan Paula-Georgiana

Cluj-Napoca

2020

Summary contents

Keywords

Internet of Things (IoT), Edge computing, Semantic stream processing, sensor data, semantic annotation, Semantic Sensor Network (SSN) ontology, Apache Kafka, Design Science Research methodology.

Thesis contents

Publications list

I. List of publications related to the thesis contents

1. **P.-G. Zălhan**, G. C. Silaghi and R. A. Buchmann. *Marrying Big Data with Smart Data in Sensor Stream Processing*. In A. Siarheyeva, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), Information Systems Development: Information Systems Beyond 2020 (ISD2019 Proceedings). ISD'19. Toulon, France: ISEN Yncréa Méditerranée: AIS eLibrary. URL: https://aisel.aisnet.org/isd2014/proceedings2019/ManagingISD/8/.
2. **P. -G. Zălhan**, *Transforming Big Data into knowledge using semantic stream processing technology: challenges and early progress*. In Proceedings of the 18th International Conference on INFORMATICS in ECONOMY (IE 2019) Education Research & Business Technologies, 2019, pp. 365-370, ISSN 2284-7472.

II. Other relevant publications

1. **P.-G. Zălhan**, *Building a LVCSR System for Romanian: methods and challenges*, Journal of Public Administration, Finance and Law (JoPAFL), no. 10/2016, pp. 181-191, 2016, ISSN 2285 – 2204.
2. C.-C. Osman, **P.-G. Zălhan**, *From Natural Language Text to visual models: A survey issues and approaches*, Informatică Economică Journal, vol. 20, no. 4, pp. 44-61, 2016, ISSN 1453-1305.

1. Introduction

In today's digital world, we are witnessing an exponential growth of Internet of Things (IoT) data which is generated with fast rate in smart environments. The growth of IoT data is caused by the large-scale IoT adoption, as the number of devices connected to the Internet, including sensors and machines, continues to grow at a steady pace. IoT data statistics provided by research organizations such as Cisco - which is the worldwide leader in IT, networking, and cybersecurity solutions - show that *5 quintillion bytes of data are produced every day* (Stack, 2018). Moreover, the dramatic acceleration of IoT generated data is also anticipated by Cisco which projects that *500 billion of IoT devices are expected to be connected to the Internet by 2030* (Cisco, 2018).

Despite the fact that the IoT domain is considered being a hot research topic, the complexity of this domain raise significant challenges that could forbid achieving its potential benefits. A recent survey (Noura, Atiquzzaman and Gaedke, 2019) on state-of-the-art solutions to facilitate IoT interoperability, shows that even though researchers have proposed numerous approaches and technologies to overcome IoT interoperability issues, research challenges still remain. The surveyed IoT solutions are focusing less on semantic interoperability. Moreover, the most popular IoT solutions do not consider the edge computing paradigm for speeding up and enhancing the efficiency of data analysis.

Large sets of complex data, known as Big Data, need to be processed, stored and presented into an efficient form with minimum time lag. If in the beginning, the Big Data ecosystem was just a collection of tools and techniques that were meant to manage large volumes of data, nowadays, the velocity property of IoT data must be also considered, by developing Big Data analytics solutions that are capable to collect and process streaming data in real-time. These solutions enhance the decision-making processes, bringing competitive advantages to businesses leaders. However, developing such solutions need to take into consideration the stream's characteristics that make the harvesting of data a challenging task. The heterogeneous nature of data originating from various sources leads to interoperability problems between IoT applications.

Integrating data from different sources using domain specific knowledge enables applications to better understand the environment where the data was generated. In this way, domain experts

or business leaders can take correct decisions, efficiently. Understanding the data context, potentially enables applications to take real-time decisions. The Semantic Web (SW) community channels its efforts to build models and techniques that transform data streams coming from various IoT devices (e.g. sensors) into machine-understandable descriptions. By means of semantic annotation, the sensor data are enriched with additional information and turned into machine-processable and machine-understandable descriptions that are easy to interpret, integrate and reuse by computers. The annotated sensor streams can be persisted into a triplestore for further inference using the SPARQL query language.

The main objective of our research will be to build a system that is capable to collect, process, store and present IoT large-scale data in an efficient and easily interpretable form. A stream data processing pipeline covers all aspects of an integral data processing chain, from the moment when a newly data was generated up to the moment when the data is prepared for final consumption. Moving raw data from the real-time sources to a target data store for further analysis and visualization must be achieved in a reliable and scalable way. In retrospect, this research aims to design a scalable, high-throughput, low-latency data pipeline which includes sensor data collection, semantic annotation of sensor data, data storage and query processing.

To achieve the research aim, this research follows the Design Science Research (DSR) paradigm because this research paradigm is considered appropriate when research aims to develop solutions for the needs of the ’business’ and its environment in the field of Information Systems (IS). The design problem is to overcome the semantic IoT interoperability issue by delivering a novel data pipeline that fulfills the low-latency requirement in order to provide timely and valuable insights from the underlying data to decision makers.

In order to fulfill the thesis aim, we address the following research questions:

1. How a low-latency stream processing pipeline can be devised in order to sustain the semantic enrichment of the large-scale sensor data generated in a smart environment?
2. Can we identify the proper place in a data pipeline where to execute the semantic sensor data annotation for faster data analysis?
3. How a semantic edge environment can be designed and managed to support real-time data processing and workflow execution of sensor data?

Given the above-described research questions, we pursued to study the following:

1. To identify requirements and gaps in the field of semantic processing of stream data.
2. To achieve semantic interoperability with IS that rely on the annotated streams.
3. To evaluate the performance of the proposed pipeline through real-world use cases from the IoT domain.
4. To propose possible future work and enhancements that are meaningful to continue improving the proposed pipeline.

The present thesis enlarges the state-of-the-art with respect to the topic of semantic stream processing with the following contributions:

1. Provides a literature review on semantic stream processing highlighting the scientific efforts that were made at the intersection of IoT stream processing and semantic technology.
2. Extends the Semantic Sensor Network (SSN) ontology identified in the literature with additional classes and properties, in which some are scenario-independent and other are targeting specific use cases, to create a hierarchy of sensor, observation, feature-of-interest and platform classes, as well as relationships between sensors and their critical observations, between sensors and their features-of-interest, and between different features-of-interest.
1. Designs an architecture of the proposed artifact and the main components of the architecture include sensor data collection, semantic annotation, data storage and query processing.
2. Develops a Proof of Concept (PoC) for the proposed architecture demonstrating that the semantic edge processing of sensor streams can be achieved in the development phase of the DSR process iteration.
3. Validates the implemented prototypes according to the Design Science principles using two different business scenarios.

The content of the thesis is organized as follows:

Chapter 2 presents the stream processing concepts with their application in the IoT domain. We explain the meaning of a stream and its essential characteristics, then we describe the stream processing paradigm and the existing stream processing architectures devised for real-time

processing systems. This chapter also provides the list of IoT protocols and standards, as well as the general architecture of a Big Data processing pipeline.

Chapter 3 provides the theoretical aspects regarding the SW standards and models for semantic annotation, focusing more on the concepts provided by the SSN ontology.

In Chapter 4 we provide a structured literature review which highlights the adopted approaches and the proposed solutions in building a semantic stream processing system.

In Chapter 5 we describe the adopted methodology, the proposed solution, the performance metrics taken into consideration when validating our solution, and the proposed extensions of the SSN ontology.

In Chapter 6 we present the design decisions of a smart airport and a smart factory, and discuss the experimental results obtained when evaluating the performance of the proposed semantic stream processing pipeline.

Chapter 7 comprises the conclusions of our research effort and includes possible future work and enhancements.

2. Stream Processing in IoT

This section presents the theoretical foundation regarding the stream processing paradigm. We will use the concepts defined here to build a semantic stream processing pipeline architecture in Chapter 5. It is important to choose the right data processing architecture and to know which the main stages are involved when building a stream processing pipeline.

2.1. Theoretical foundation of stream processing in IoT

Stream processing paradigm (Liu, Dastjerdi and Buyya, 2016) was proposed by research communities as a Big Data solution to process large volumes of data within a time-constraint using distributed resources. Stream processing is the opposite for batch processing, which is not capable of processing dynamic data generated at fast rate. Stream processing is the ideal solution for managing IoT ecosystems as it has the ability to analyze massive amounts of data from multiple sources in real-time.

However, stream data generated in IoT environments has complex features which make handling of it challenging. The characteristics of data streams such as dynamicity, continuousness, heterogeneity, volatility have to be taken into consideration along with their implications when designing a stream processing system. The streaming architecture must be able to process the data in motion, as soon as it becomes available, in order to respond to-the-second at the dynamicity of the IoT environment. Since the data streams are processed in a timely fashion, the infrastructure must handle stream imperfections on the fly including delayed, missing or out-of-order data (Stonebraker, Cetintemel and Zdonik, 2005). The stream, which is a never-ending sequence of time-varying data elements, is continuously carried through the processing pipeline and is periodically updated as new data becomes available. Furthermore, data streams are collected from heterogeneous data sources and exist in different formats such as photos, videos, audio, text, etc.

2.2. Event-driven patterns for real-time Big Data analytics

In contrast to the traditional service-oriented architecture (SOA) that restricts the flexibility and the scalability of a system because of its synchronous request-response mechanism (Jerry, 2019),

the event-driven architecture (EDA) is used to build highly scalable and highly adaptable system because it can handle the data streams into an asynchronous way. In EDA, an event represents a real-world observation that describes a fact that physically happened in an environment. There are two types of event-driven architectures: Lambda architecture and Kappa architecture. The Lambda architecture (Marz and Warren, 2015) integrates real-time and batch data analysis using three distinct layers of processing, while the Kappa architecture (Kreps, 2014), which is a simplified alternative to the Lambda architecture, eliminates the batch layer. Usually, the Kappa architecture is used to design systems that need to handle distinct and complex events in real-time.

2.3. IoT data protocols and standards

IoT protocols that were especially developed to ensure successful data communication between connected devices are the following:

- *Message Queuing Telemetry Transport* (MQTT) – lightweight protocol for machine-to-machine communication (M2M) of constrained devices in edge environments.
- *Message Queuing Telemetry Transport-Sensor Network* (MQTT-SN) – specialized version of MQTT designed for Wireless Sensor Networks, with low cost, battery-operated devices (Govindan and Azad, 2015; Stanford-Clark and Truong, 2013).
- *Constrained Application Protocol* (CoAP) – the most recent application protocol which was proposed for use in resource-constrained devices
- *Advanced Message Queuing Protocol* (AMQP) – is an open standard application protocol for message-oriented middleware environments (Firouzi, Chakrabarty and Nassif, 2020).

2.4. The general Big Data processing pipeline in IoT

A Big Data processing pipeline must be able to capture, transform, store and present relevant data so that it can be used to gain real-time insights. Figure 1 presents our general data processing pipeline which comprises several stages including data ingestion, data processing, data storage, and data visualization.

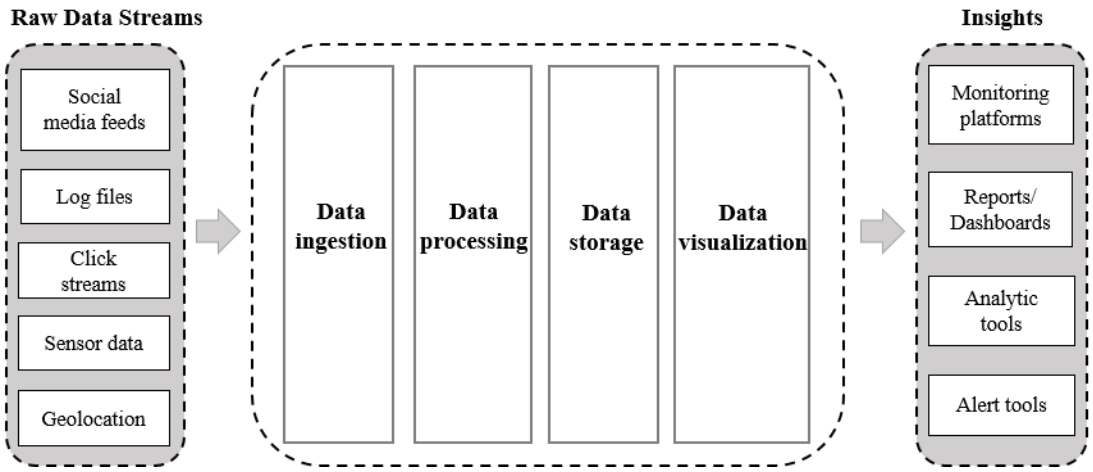

Figure 1. The general Big Data processing pipeline (Zălhan, 2019)

In the ingestion stage, raw data-streams are collected from heterogeneous sources, such as social media platforms, wireless sensor networks or other connected IoT devices. In the data processing stage, complex data transformations are applied on multiple input streams and multiple stream records using streaming operators such as filter, aggregation, and join. These operations can be restricted on a finite portion of a stream called window. To ensure real-time processing with very quick response time, the data streams must be processed in parallel by multiple machines in a cluster. The purpose of the data storage stage is to persist relevant data into a highly-scalable distributed database for later and complex data analysis. The data visualization or presentation stage has the role to create an interface of the stream processing system, where advanced analytics and monitoring platforms can be used to gain meaningful insights over the stored data. This interface can be a dashboard for continuous real-time data monitoring, or an alert tool for notifying consumers to take rapid actions when critical patterns are discovered in the underlying data.

3. Semantic Technologies for IoT

This chapter presents the theoretical framework regarding the SW standards and models. We will use the concepts defined here to achieve the semantic modeling task by annotating the raw stream data that traverses the semantic stream processing pipeline presented in Chapter 5. The semantic enrichment of device data with additional data from external sources, such as vocabularies and ontologies, help developers to design and implement interoperable IoT applications. Providing explicit descriptions of sensor measurements helps users and machines to understand in a unified way the context where the heterogeneous data was generated.

3.1. Resource Description Framework

The Resource Description Framework (RDF) model (Klyne, Carroll and McBride, 2009) helps to create graph-based representations about Web resources, using a variety of syntax notations and data serialization formats. The underlying structure of an RDF data model consists of expressions in the form of subject-predicate-object, known as triples (Domingue, Fensel and Hendler, 2011) where the predicate describes the relationship between the subject and the object. A collection of triples forms a directed, labeled graph, where the RDF nodes are its subjects and objects. For storing RDF graphs, different serialization formats exist, such as Turtle. The basic RDF vocabulary contains a predefined set of classes and properties that can be used to construct RDF statements about relationships between different resources.

3.2. RDF Schema

RDF Schema (RDFS) is a data-modeling vocabulary that extends the basic RDF with additional classes and properties to allow describing the meaning of resources and the relationship between them. These resources are used to determine characteristics of other resources, such as domains and ranges of properties. The standard properties named "rdfs:domain" (domain) and "rdfs:range" (range) can be used to specify restrictions on the subjects and objects of RDF triples. The "rdfs:subClassOf" and "rdfs:subPropertyOf" properties are used to create class and property hierarchies. There are other RDF vocabularies which are frequently used when building SW

applications. For example, Schema.org vocabulary contains a list of predefined terms to describe concepts such as Person, Employee, Organization, etc.

3.3. Web Ontology Language

RDFS vocabulary provides simple semantics for IoT applications by allowing users to define classes that may have multiple subclasses and super classes, and properties that may have multiple subproperties, domains and ranges. Web Ontology Language (OWL) (Motik et al., 2009) extends RDFS with richer semantics which are needed to achieve interoperability between various components of a smart system (Li and Zhong, 2004). The OWL and RDFS standard terms can be used to build class or property axioms (i.e. statements whose subject is a class or a property). For example, the "owl:inverseOf" standard property provided by OWL used to define inverse relations between properties, i.e. define relations in both directions.

3.4. Semantic Sensor Network Ontology

The Semantic Sensor Network (SSN) ontology (Haller et al., 2017) is a key enabling SW technology for sensor networks, as it facilitates semantic interoperability, integration and reasoning upon sensor data. SSN ontology focuses on describing physical sensor networks, such as sensors, observations that result from sensing, and deployments in which sensors are used. The SSN ontology includes a self-contained core ontology called SOSA (Sensor, Observation, Sample, and Actuator) (Janowicz, Haller, Cox, Le Phuoc, and Lefrançois, 2019) for its basic classes and properties. There are several SOSA terms related to modeling observations and other concepts:

- *sosa:Sensor* – the Sensor class represents the concept of a sensing device.
- *sosa:Platform* –a platform represents an entity that hosts other entities, such as sensors.
- *sosa:FeatureOfInterest* – the thing whose property is being calculated when an observation result has arrived in the environment.
- *sosa:ObservableProperty* – an observable characteristic of a feature-of-interest.
- *sosa:Observation* – the Observation concept represents the activity of estimating or calculating a value of a propriety of a feature-of-interest.
- *sosa:Result* – the Result concept represents the result of an observation.

- *sosa:Procedure* – the notion of Procedure can describe a workflow, an algorithm, or a computational method that specifies how to make an observation; it also specifies the steps that need to be executed in order to arrive at a result.

SOSA properties are defined in both ways, from domain to range. For example, the "sosa:madeObservation" core property specifies the relationship between a sensor and an observation made by that sensor. In contrast, the "sosa:madeBySensor" inverse property connects observations to sensors. To model the observation value generated in an IoT environment, the "sosa:hasResult" property links an observation with the result which contains the sensed value. The "sosa:hasResult" is used to model complex information about an observation, namely what the physical sensor measures in the environment (domain phenomena), which is the unit of measure of the sensor value or where was the sensor value being captured (sensor location).

The SSN ontology is built on top of SOSA and introduces additional terms related to modeling observations: "ssn:Stimulus", "ssn:Input", "ssn:Output", "ssn:System", "ssn:Deployment", for describing a real-world event that 'triggers' the sensor, information that is provided as input or is obtained as output of a procedure, pieces of infrastructure that implement procedures, deployment of one or more systems.

Regardless the efforts being spent to enrich the existing SSN ontology, according to the W3C's final report (Vadivel and Subramanian, 2017), this ontology has some limitations:

- The SSN ontology excludes a hierarchy of sensor types, or domain types (i.e. subclasses of ssn:FeatureOfInterest).
- Common terms to represent sensor measurements, unit types, or location are not included in the SSN ontology.

4. State-of-the-art in Semantic Stream Processing

Based on a structured literature review on semantic stream processing, in this chapter we develop a theoretical framework that will be helpful in understanding and analyzing the scientific efforts made in the domain of Semantic Stream Processing. This analysis will constitute the foundation for further identifying the issues and gaps in the current research literature. This systematic review includes high-rated scientific conferences and journals.

Semantic Stream Processing (SSP) is defined by (Le Phuoc and Hauswirth, 2019) as a set of models, principles and techniques used to analyze and process data streams by exploiting the meaning of the stream data elements. Focusing on the progress that has been made in the field of SSP to deal with highly dynamic data that needs to be processed in a timely fashion, the most consolidated research findings include: (1) extensions of the RDF model and the SPARQL query language, which are produced and implemented by a corresponding number of RDF Stream Processing engines, and (2) several inference techniques proposed under the Stream Reasoning label to gain new insights over the streaming data.

In the following we will point to some of the limitations identified in the above-studied literature. The main scientific contribution of our thesis will be to address these limitations and propose an alternative solution.

4.1. RDF Stream Processing

The RDF Stream Processing (RSP) research trend or Linked Data Stream Processing (Le-Phuoc, Parreira and Hauswirth, 2012) proposes several RSP engines that use the RDF model in representing stream data and execute continuous SPARQL queries over the RDF streams, using various approaches.

The first RSP engines were centralized systems designed to run on a single machine by applying continuous queries over RDF streams using SPARQL extensions. All these engines extend the standard SPARQL grammar with additional features such as aggregation capabilities, window operators, and the possibility to combine multiple streams. These systems are built on top of existing Data Stream Management Systems (DSMS) to process the dynamic part of the query,

and RDF triple storage to evaluate the static part of the query. However, these systems are unable to handle large volumes of data coming from heterogeneous sources and to execute multiple continuous queries over RDF data streams in a distributed environment.

To remedy the limitations of centralized RSP engines and improve the performance of existing RSP systems, distributed RSP engines were designed to enable concurrent queries over the incoming data. Distributed RSP engines are based on a cluster computing infrastructure to perform parallel processing over streams. However, some of the proposed system focuses only on dynamic data.

Meanwhile, were proposed several solutions that collect data streams in real-time from different sources, process and send data to various destination systems using a middleware component. Some of the middleware solutions support semantic annotation of sensor data using the SSN ontology which is extended with constructs for modeling the measurements of temperature, humidity, light, and voltage sensors. Other solutions use a cloud infrastructure to perform the semantic modeling and to execute SPARQL queries over the annotated streams. However, none of the existing systems do not consider the edge computing paradigm for speeding up and enhancing the efficiency of data analysis.

4.2. Stream Reasoning

Stream reasoners extend the traditional RSP engines with rule-based, logical, reasoning capabilities. Some of the systems use reasoning languages that extend the first and the second order logics with generic window operators. Some authors use approaches based on stream reasoning models and techniques to process semantically-enriched data streams for supporting decision making in smart environments (e.g. a smart city). Stream reasoning is combined with machine learning techniques, such as supervised stream learning for stream data classification.

5. Design and Implementation of the Semantic Stream Processing pipeline

In this chapter we describe the adopted methodology when building the semantic stream processing pipeline. We present the proposed system architecture by highlighting its main components and the technologies that were used in the implementation process. We describe the proposed extensions of the SSN ontology that were introduced in this work to model scenario independent and scenario-specific concepts related to sensor networks. We briefly describe the system configurations, and the alternative pipeline architectures that we consider in Chapter 6 when evaluation where is preferable to execute the semantic annotation task. Lastly, we discuss the performance metrics that were chosen to validate or SSP pipeline.

5.1. Methodology

When building the Semantic Stream Processing (SSP) pipeline, the Design Science Research (DSR) paradigm (Wieringa, 2014) was adopted, following an iterative development cycle with the explicit intention of improving the pipeline's performance which is empirically investigated taking into consideration two IoT application scenarios: a smart airport and a smart factory. The stakeholders are the emergency response teams which need to rapidly react for the rescue of persons in case of critical observations have been sensed (and reasoned upon) within the smart environment.

According to (Peffers, Tuunanen, Rothenberger, and Chatterjee, 2007), after identifying the research problem, stating the motivations and defining the objectives, the next step in the DSR process is to design and develop the artifact. The implemented SSP pipeline using Apache Kafka, as a distributed streaming platform, and GraphDB, as a semantic database server, represents an instantiation artifact (March and Smith, 1995; Hevner, March, Park, and Ram, 2004).

In order to achieve a suitable solution, it is necessary that the solution is developed iteratively and is validated. This research utilizes two iterations prior to the construction of the final solution pipeline. In each iteration, a prototype is implemented and evaluated based on some criteria. Iteration one uses a smart airport use case to develop a scalable SSP pipeline following the Big Data reference architecture. We assess the viability of the pipeline conducting several experiments

under various Kafka cluster configurations. The result of the validation process on the performance of the first prototype using the scalability metric shows that the distributed Kafka cluster configuration manages to support the semantic enrichment of sensor data. This Kafka cluster configuration is used as input for the second iteration which utilizes a smart factory use case to develop a low-latency SSP pipeline. To investigate where is it preferable to execute the semantic annotation in order to deliver faster insights of the underlying sensor data, two alternative architectures were implemented: semantic annotation after the raw sensor data is ingested into the system, and semantic annotation directly on the sensing devices output by means of semantic edge computing. We evaluate the performance of the alternative pipeline architectures using throughput and response time metrics. Iteration two improves the developed pipeline in terms of speed and data analysis demonstrating the pipeline's efficiency. Iteration two represents the final artifact of this research.

5.2. Proposed solution overview: a semantic stream processing pipeline

The main components of the proposed SSP pipeline are illustrated in Figure 2. Data sources are represented by various sensors installed in a smart environment. In this thesis, wearable sensors are used in combination with ambient sensors to support the target IoT application cases. Geolocated audio sensors are taken into consideration to support the smart airport use case, while the heartbeat, location and proximity sensors are used to build a smart factory ecosystem.

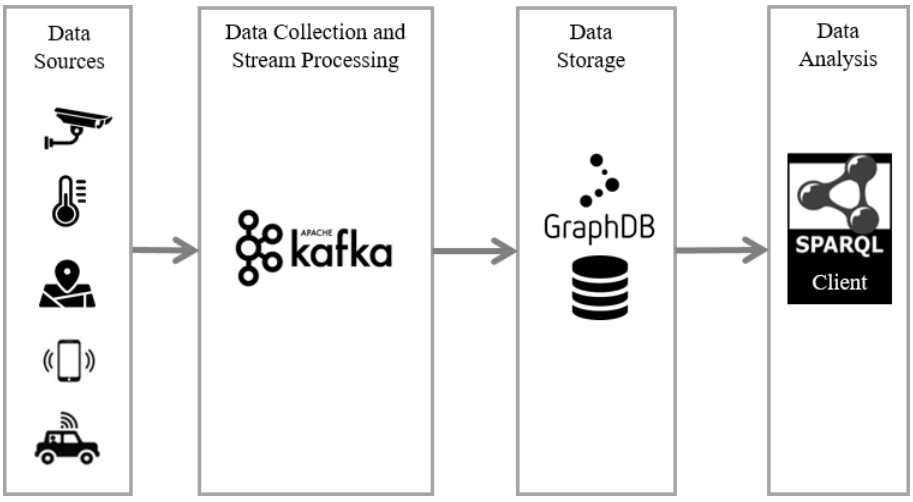

Figure 2. Proposed semantic sensor stream processing pipeline (Zălhan, Silaghi, & Buchmann, 2019)

The continuous data gathered from sensor data sources is collected and processed by a distributed data ingestion system for later analysis. In this thesis, Apache Kafka (Narkhede, Shapira, and Palino, 2017) is employed for ingesting the sensor streams because it is able to handle large-scale data in real-time. Also, Kafka stores data streams in a fault-tolerant manner using the replication mechanism guarding against data loss.

The basic architecture of Kafka is organized around a few key terms: topics, producers, consumers, and brokers. The message is the unit of data within Kafka. All Kafka messages are organized into topics. Topics can be broken down into a number of partitions containing sequences of messages that are spread over multiple servers working in parallel. Each partition can be optionally replicated across a configurable number of servers to guarantee fault tolerance. Thus, partitions are the modality in which Kafka provides scalability and replication.

Apache Kafka, as high-throughput distributed messaging system, is based on the publish/subscribe model, where several producers write messages into topics, and several consumers read the messages that were previously published by producers. Internally, Kafka runs as a cluster which connects multiple producers and consumers to one or more servers, known as brokers which receive messages from producers and then commits the messages to storage on disk. Kafka uses Apache Zookeeper to store metadata about the Kafka brokers and consumer details.

To provide machine-readable and machine-interpretable descriptions of the ingested data, the streaming data previously collected are turned into useful information using semantic data annotations with the extended SSN ontology. The semantic descriptions for sensor streams are persisted into a semantic graph database for reasoning, later analysis or integration with a legacy IS (e.g., a notification system). In this thesis, we chose the GraphDB semantic database server to store the annotated streams. According to recent survey (Bellini and Nesi, 2008) on performance evaluation of currently available RDF stores, GraphDB outperforms existing semantic database servers when considering the loading and indexing time for triplestore initialization. Load times may increase causing a delay in deliver data at the expected time.

To access the sensor data and the legacy data stored as triples, the SPARQL client allows querying the RDF descriptions of the large integrated dataset. The semantic streams stored in the RDF graph database are retrieved and manipulated using SPARQL query language and SPARQL-based rules to further explore the semantic descriptions in order to infer new knowledge and useful insights.

5.3. Proposed SSN ontology extensions

Because the SSN ontology does not model a hierarchy of sensor types, nor domain types and does not differentiate between sensor observations that have critical values and sensor observation values that reveal normal observed ranges, we propose some extensions to the existing SSN ontology to enable:

- hierarchy of sensor, observation, feature-of-interest and platform classes
- linking directly between sensors and features-of-interest, relationships between sensors and critical observations, as well as relationships between different features-of-interest.

To overcome the limitations of the existing SSN ontology, we extend its syntax with additional classes and properties in which some are scenario-independent and other are targeting specific use cases. The knowledge base is enriched with a set of axioms, defining RDF statements built with standard terms to establish machine-readable meaning for SSN properties and classes.

We create multiple axioms using the "rdfs:subClassOf" standard property provided by the RDFS vocabulary to build a hierarchy of sensor classes, defining the Wearable Sensor concept which represents sensors attached to human body, and the Ambient Sensor concept to model sensors deployed into the physical environment. Also, more specialized sensor types are defined to support specific IoT use cases.

In order to detect critical patterns in a smart environment, the sensor readings are compared with threshold limits that define the normal observed ranges. Thus, we create a taxonomy of observations by categorizing sensor observations into lower critical observations or upper critical observations based on the sensor values generated into the environment.

We extend the "sosa:FeatureOfInterest" standard class to create a hierarchy of features-of-interest in order to distinguish between active and passive things whose properties are being observed. We model living, breathing entities such as actors and employees. In this process, we use the Person concept provided by the Schema.org vocabulary. Similarly, we model passive entities such as fixed equipment, buildings and indoor environments. For each active and passive things we introduce more specialized entities to support different IoT uses cases.

We also define new relationships between proposed concepts with the help of the "rdfs:subPropertyOf" standard property. Each proposed property is defined from domain to range. We introduce a new property to describe the relationship between a sensor and its critical observation (upper or lower critical). Also, we introduce new property describing relationships

between various features-of-interest (active or passive) and sensors. Some of proposed properties have inverse properties to facilitate graph navigation. To define these relations in both directions we use the "owl:inverseOf" provided by the OWL standard terminology.

Furthermore, we define relationships between fixed equipment and the environment in which they are stored, between an actor and the environment in which it acts. Also, we define a hierarchy of observable characteristics of features-of-interest. These observable characteristics link sensor observations to observation results.

5.4. Solution implementation

Kafka cluster configurations are made using the Confluent[1] platform which is used to build real-time data pipelines and streaming applications by integrating data from multiple data sources. We implement two different Kafka cluster setups: (1) *single node – single broker configuration*; and (2) *single node – multiple brokers configuration.* These configurations will be validated later, in chapter 6, where we investigate the most suitable Kafka cluster configuration to support the semantic annotation of large-scale sensor data.

For identifying the place in the pipeline where is preferable to execute the semantic modeling task, we have implemented two alternative pipeline architectures:

(1) *Semantic modeling on consumer side* – where Kafka producers generate and publish sensor streams into topics, while Kafka consumers subscribe to these topics, then read, annotate the sensor streams and store the corresponding RDF streams triplestore.

(2) Semantic modeling on producer side – where Kafka producers generate, annotate sensor streams and publish the corresponding RDF streams into topics, while Kafka consumers subscribe to these topics, then read and persist the RDF streams in triplestore.

In both pipeline architectures, sensor streams conform to a schema which contains several attributes: sensor identifier, sensor type, the number of the platform that host various sensors, stream identifier, sensor value, as well as the timestamp marking the time when the sensor value was generated into the smart environment.

[1] https://www.confluent.io/product/confluent-platform

To simulate the streaming data coming from various sensors deployed in a smart environment, we have defined Kafka producers and consumers using the main classes exposed by the Kafka-python[2] library.

5.5. Solution validation

The proposed pipeline's performance is evaluated using various metrics: throughput, scalability and execution time. The ingestion system that feeds continuously the pipeline with sensor streams must be capable to process high-throughput data. Furthermore, the processing pipeline should scale to very high throughput. Apache Kafka, as a data collection and stream processing component of our data pipeline, was designed for Big Data use cases which need horizontal scalability for both message senders and message receivers. Also, we are interested to determine the necessarily amount of time to generate a message in dependence to where the semantic modeling task is achieved.

[2] http://github.com/dpkp/kafka-python

6. Use Cases in IoT Smart Environments

In this chapter, we present two IoT application cases to illustrate how the SSP pipeline introduced in Chapter 5 can be used in practice. The IoT use cases chosen in thesis belong to different categories of smart cases of IoT. The smart airport application case falls into the governance application category of IoT use cases, while the smart factory application case belongs to the class of industrial use cases of IoT. First we describe the scenario of each IoT application case, then we present the corresponding design decisions, and lastly we discuss the experimental results.

6.1. Smart Airport

In order to effectively prevent future terrorist attacks, the infrastructure of a smart airport is designed with audio sensors and modern technologies, such as Automatic Speech Recognition (ASR) and geo-localization technology. The purpose of ASR technology in the smart airport context is to detect suspicious conversations of passengers and alert security operators to take rapid actions in case critical patterns are detected into the underlying uttered discourse. The speech recognition component is out of scope for this research study, as we focus on the semantic integration architecture and parallelization of the semantic annotation effort. We use previous project experience (Zălhan, Stan, Teodorescu, Saupe, & Duma, 2016) regarding the building of an ASR system and the development of such a component is part of the future work.

To simulate the streaming data from the geolocated audio sensors deployed in a smart airport, we have used Producer and Consumer APIs that support custom implementations to write and read streams of data in the Kafka cluster. The published messages are written into a topic. In Figure 3, we present the semantic model of a raw audio sensor stream originally written in JSON format. Because some aspects such as sensor value units and measurements, or dynamic behaviors such as sensor location, are not tackled by the existing SSN ontology, in the semantic annotation process of sensor data we use the „Quantities, Units, Dimensions and Data Types" (QUDT)[3] vocabulary

[3] http://www.qudt.org/

to model acoustic units, and geospatial vocabularies such as GeoSPARQL[4] to model the location of an audio sensor in the smart airport environment.

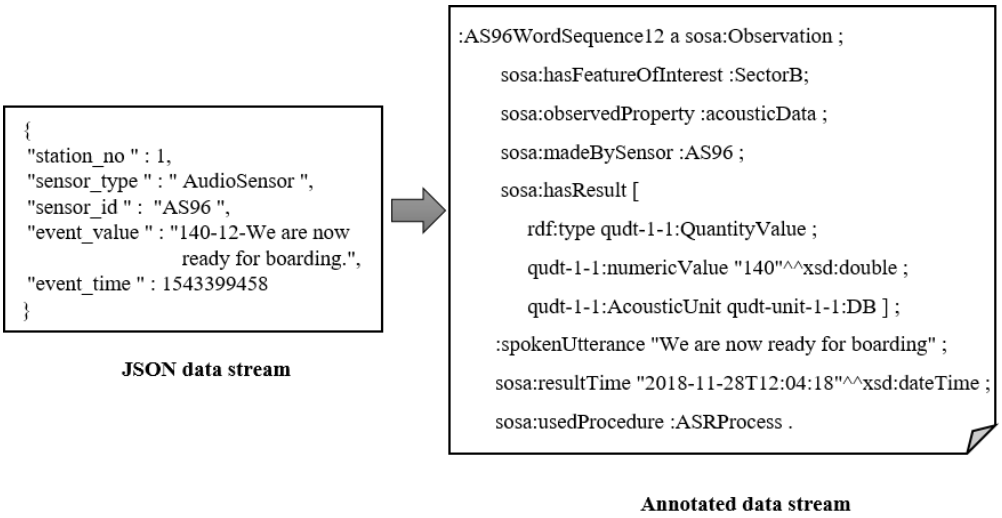

Figure 3. Description of an audio sensor observation (Zălhan, Silaghi, și Buchmann, 2019)

The resulted RDF descriptions are published into GraphDB for later analysis and querying. The data stored in the RDF graph database is retrieved and manipulated using SPARQL query language which is used to infer new information and knowledge from the existing one by executing various queries over the RDF statements. Using a more complex query, the system notifies the security operators responsible with the sector where an upper critical observation was made by sending them a message on their telephone. When defining this query, we semantically lift the airport legacy database to connect it with the graph that describes sensor data.

We use the smart airport ecosystem to decide which is the most suitable Kafka-based scenario for achieving the semantic modeling task. Two different Kafka cluster configurations were set up: (1) single node – single broker Kafka cluster configuration; (2) single node – multiple brokers Kafka cluster configuration. The experimental results from Figure 4 show that in the second Kafka scenario, the Kafka cluster manages to cope with ingesting large volumes of sensor data by distributing the write load over brokers that are working in parallel. Moreover, the distributed configuration of the Kafka cluster sustains the semantic annotation of massive amounts of sensor data using multiple consumer instances that concurrently read the topic messages.

[4]https://www.opengeospatial.org/standards/geosparql

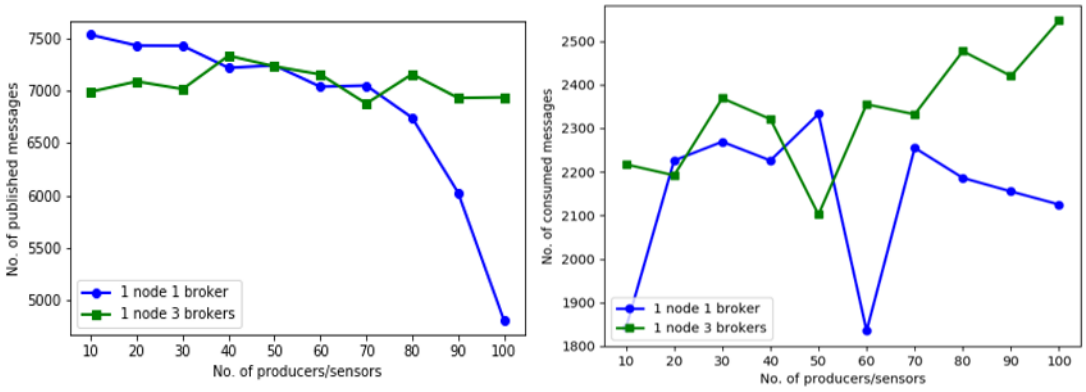

Figure 4. The number of producers versus the number of published or consumed messages in both Kafka-based scenarios

6.2. Smart Factory

The smart factory ecosystem is focused on worker's safety by monitoring health status in real-time using heartbeat, location and proximity sensors. The motivating design problem context is to support a smart factory with a safer workspace allowing emergency interventions and incident management during natural disasters, where ambient and wearable sensors become relevant for preventing workplace injuries or death of employees. Heartbeat and localization sensors are attached to a worker's body to monitor its heart rate, and its spatial coordinates. Proximity sensors are attached to factory machinery to detect the presence of a human body over a distance, such as a worker walking too close to a piece of heavy machinery due to limited visibility

To simulate the data streaming from the aforementioned sensors deployed in a smart factory ecosystem, we have crated Kafka producers and consumers that write and read streams of data in a Kafka cluster. The generated messages are written by producers in different topics. The semantic annotation scheme of sensor streams, originally written in JSON format, is similar to the one used in the smart airport application case. We use the QUDT vocabulary to provide machine-readable access to unit and unit type information (beat per minute for a heartbeat sensor, and the distance in centimeters for proximity sensor), as well as the WGS84 Geo Positioning[5] geospatial vocabulary to model the dynamic location of factory worker.

Once the semantic descriptions are persisted into the triplestore, the resulted knowledge base can be further explored through SPARQL-based reasoning to gain useful insights. We define

[5]https://www.w3.org/2003/01/geo/wgs84_pos

complex SPARQL queries to retrieve the spatial coordinates of a worker whose heartbeat sensor has detected a lower critical observation. Also, we execute SPARQL queries for identifying the closest worker as sensed by a heavy machine's proximity sensor.

In the smart factory application case, we focus on the core processing pipeline to identify where is preferable to process the raw sensor data. To assess the viability of the SSP pipeline, two different architectures were implemented: semantic modeling on consumer side (S1); and semantic modeling on producer side (S2). In S2 architecture, raw sensor streams are semantically processed at the edge of the sensor network, near the sensing devices.

Table 1. Annotated messages and corresponding RDF triples in both implemented architectures

Producers	S1		S2		Throughput multiplication
	Messages	RDF triples	Messages	RDF triples	
10	529	8993	5703	96951	10,78
20	558	9486	6253	106301	11,21
30	696	11832	6207	105519	8,92
40	579	9843	6023	102391	10,40
50	357	6069	6043	102731	16,93
60	652	11084	5714	97138	8,76
70	707	12019	5902	100334	8,35
80	851	14467	5904	100368	6,94
90	751	12767	5911	100487	7,87
100	705	11985	6091	103547	8,64

Table 1 summarizes the number of annotated messages and their corresponding RDF triples, in the alternative architectures (S1 and S2). It can be seen that when the semantic modeling is accomplished by edge sensors, even ten times more annotated messages are generated (and implicitly, the number of corresponding RDF triples it is about ten times larger. In the semantic edge architecture, the data does not waste time to travel through the pipeline because the semantic annotation is performed locally, on edge sensors.

7. Conclusions and future work

In this thesis, we proposed a novel semantic stream processing pipeline for sensor data and offered two interesting application cases to evaluate its performance. The architecture of the proposed pipeline contains multiple components that cover all aspects of an integral data processing chain: data collection, semantic annotation, RDF data storage and query processing of sensor streams. Throughout the entire research we have been directing our efforts towards achieving all the proposed objectives.

Based on a literature review with respect of semantic stream processing, we have analyzed existing authors' approaches for processing in real-time large volumes of sensor data continuously generated by heterogeneous data sources de date and extracting useful information from underlying data. Moreover, we have identified the limitations of existing semantic stream processing systems. The existing systems do not consider the edge computing paradigm for speeding up and enhancing the efficiency of data analysis. Also, the SSN ontology identified in the literature does not model sensor types, data measurements units, location, mobility and other dynamic behaviors. We overcome these limitations by extending the SSN ontology with new classes and properties to model sensor types, specific domains, critical observations and relationships between them.

Our middleware solution for semantic processing of sensor data uses another distributed messaging system called Apache Kafka, because it has better throughput, built-in partitioning for parallel data consumption than most messaging system have, which makes it suitable to build low-latency processing pipelines. Another aspect that differentiates our solution from other existing middleware solutions is the combined approach for semantic annotation mixing the SSN ontology with other vocabularies.

To obtain semantic interoperability with SI that rely on the annotated sensor streams, we integrate the sensor data graph with legacy databases to support the development of a Hybrid Semantic System for Incident Management. In this process, we hybridize the existing SSN ontology with additional terms from other vocabularies to model information about individual persons belonging to emergency intervention teams that act in a smart environment.

The next goal is to setup a highly-distributed Kafka solution that consists of multiple nodes with multiple brokers. This configuration is desirable to advance the technological readiness level of our proposal and is on-going work that will employ a high-performance computing infrastructure.

References

Bellini, P., & Nesi, P. (2008). Performance assessment of rdf graph databases for smart city services. (Elsevier, Ed.) *Journal of Visual Languages and Computing, 45*, 24-38.

Cisco. (2018, February). *Cisco Edge-to-Enterprise IoT Analytics for Electric Utilities Solution Overview*. Preluat pe March 25, 2020, de pe Cisco: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/big-data/solution-overview-c22-740248.html

Domingue, J., Fensel, D., & Hendler, J. (2011). *Handbook of Semantic Web Technologies.* Springer.

Firouzi, F., Chakrabarty, K., & Nassif, S. (2020). *Intelligent Internet of Things: from Device to Fog and Cloud.* Springer.

Govindan, K., & Azad, A. P. (2015). End-to-end service assurance in IoT MQTT-SN. *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)* (pg. 290-296). IEEE Press.

Haller, A., Janowicz, K., Cox, S., Le Phuoc, D., Taylor, K., & Lefrançois, M. (2017, October 19). *Semantic Sensor Network Ontology*. Preluat de pe W3C Recommendation: https://www.w3.org/TR/vocab-ssn/

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly, 28*(1), 75-105.

Janowicz, K., Haller, A., Cox, S. J., Le Phuoc, D., & Lefrançois, M. (2019). SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics, 56*, 1-10.

Jerry, M. (2019, March). *SOA vs. EDA: Is Not Life Simply a Series of Events?* Preluat pe March 5, 2020, de pe Confluent: https://www.confluent.io/blog/soa-vs-eda-is-not-life-simply-a-series-of-events

Klyne, G., Carroll, J. J., & McBride, B. (2009). *Resource description framework (rdf): concepts and abstract syntax, 2004*. Preluat de pe http://www. w3. org/TR/2004/REC-rdf-concepts-20040210

Kreps, J. (2014, July 2). *Questioning the Lambda Architecture*. Preluat pe August 6, 2019, de pe O'Reilly: https://www.oreilly.com/ideas/questioning-the-lambda-architecture

Le Phuoc, D., & Hauswirth, M. (2019). Semantic Stream Processing. În S. Sakr, & A. Y. Zomaya, *Encyclopedia of Big Data Technologies.* Springer. doi:10.1007/978-3-319-63962-8_287-1

Le-Phuoc, D., Parreira, J. X., & Hauswirth, M. (2012). Linked Stream Data Processing. *Reasoning Web. Semantic Technologies for Advanced Query Answering*, (pg. 245-289). Vienna. doi:10.1007/978-3-642-33158-9_7

Li, Y., & Zhong, N. (2004). Web mining model and its applications for information gathering. *Knowledge-Based Systems, 17*(5), 207-217.

Liu, X., Dastjerdi, A., & Buyya, R. (2016). Stream processing in IoT: Foundations, state-of-the-art, and future directions. În R. Buyya, & A. Dastjerdi, *Internet of Things: Principles and paradigms* (pg. 145-161). Elsevier.

March, S. T., & Smith, G. F. (1995). Design and natural science research on information technology. *Decision support systems, 15*(4), 251-266.

Marz, N., & Warren, J. (2015). *Big Data: Principles and best practices of scalable real-time data systems.* New York: Manning Publications Co.

Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., . . . Sattler, U. (2009). OWL 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation, 27*(65), 159.

Narkhede, N., Shapira, G., & Palino, T. (2017). *Kafka: the definitive guide: real-time data and stream processing at scale.* O'Reilly Media, Inc.

Noura, M., Atiquzzaman, M., & Gaedke, M. (2019). Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mobile Networks and Applications, 24*(3), 796-809. doi:10.1007/s11036-018-1089-9

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems, 24*(3), 45-77. doi:10.2753/MIS0742-1222240302

Stack, T. (2018, February). *Internet of Things (IoT) Data Continues to Explode Exponentially. Who Is Using That Data and How?* Preluat pe April 25, 2020, de pe Cisco Blogs: https://blogs.cisco.com/datacenter/internet-of-things-iot-data-continues-to-explode-exponentially-who-is-using-that-data-and-how#comments

Stanford-Clark, A., & Truong, H. L. (2013). MQTT for sensor networks (MQTT-SN) protocol specification. *International business machines (IBM) Corporation version, 1*, 2.

Stonebraker, M., Cetintemel, U., & Zdonik, S. (2005). The 8 requirements of real-time stream processing. *ACM Sigmod Record, 34*(4), 42-47.

Vadivel, V., & Subramanian, S. (2017). Semantic Technologies for IoT. În B. Tripathy, & J. Anuradha, *Internet of Things (IoT): Technologies, Applications, Challenges, and Solutions* (pg. 265-294). CRC Press.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering.* Springer.

Zălhan, P.-G. (2019). Transforming Big Data into knowledge using semantic stream processing technology: challenges and early progress. *Proceedings of the 18th International Conference on INFORMATICS in ECONOMY (IE 2019) Education Research & Business Technologies*, (pg. 365-370). București.

Zălhan, P.-G., Silaghi, G. C., & Buchmann, R. A. (2019). Marrying Big Data with Smart Data in Sensor Stream Processing. *A. Siarheyeva, C. Barry, M. Lang, H. Linger, & C. Schneider (Eds.), Information Systems Development: Information Systems Beyond 2020 (ISD2019 Proceedings).* Toulon, France. Preluat de pe https://aisel.aisnet.org/isd2014/proceedings2019/ManagingISD/8/

Zălhan, P.-G., Stan, A., Teodorescu, L.-R., Saupe, A.-B., & Duma, M. (2016). A Kaldi-based ASR Solution for the Romanian Judicial System. *International Conferece on INFORMATICS in ECONOMY (IE 2016): Education, Research & Business Technologies*, (pg. 191-197). Cluj-Napoca.