ESKAPE: Information Platform for Enabling Semantic Data Processing

André Pomp, Alexander Paulus, Sabina Jeschke and Tobias Meisen
Institute of Information Management in Mechanical Engineering, RWTH Aachen University, Aachen, Germany

Keywords: Semantic Computing, Semantic Model, Knowledge Graph, Internet of Things, Data Processing.

Abstract: Over the last years, many Internet of Things (IoT) platforms have been developed to manage data from public and industrial environmental settings. To handle the upcoming amounts of structured and unstructured data in those fields, a couple of these platforms use ontologies to model the data semantics. However, generating ontologies is a complex task since it requires to collect and model all semantics of the provided data. Since the (Industrial) IoT is fast and continuously evolving, a static ontology will not be able to model each requirement. To overcome this problem, we developed the platform ESKAPE, which uses semantic models in addition to data models to handle batch and streaming data on an information focused level. Our platform enables users to process, query and subscribe to heterogeneous data sources without the need to consider the data model, facilitating the creation of information products from heterogeneous data. Instead of using a pre-defined ontology, ESKAPE uses a knowledge graph which is expanded by semantic models defined by users upon their data sets. Utilizing the semantic annotations enables data source substitution and frees users from analyzing data models to understand their content. A first prototype of our platform was evaluated by a user study in form of a competitive hackathon, during which the participants developed mobile applications based on data published on the platform by local companies. The feedback given by the participants reveals the demand for platforms that are capable of handling data on a semantic level and allow users to easily request data that fits their application.

1 INTRODUCTION

Recent trends in Smart Cities, (Industrial) Internet of Things (IoT), enterprise and mobile applications and various other areas lead to an enormous increase of data sources and available data. Due to the high degree of globalization in today's international companies, developers and data scientists face the problem of utilizing data from multiple technically independent company sites. Each site may use different data formats or data models as well as different languages resulting in a massive amount of heterogeneous data sources, even within a single company structure.

However, the common idea of the Internet of Things and various industrial and enterprise applications is to enable each participating object, which is part of the system, to collect and exchange data that can be used in multiple domains and sites at the same time. One possible solution for solving this challenge is the standardization of data models, formats and exchanged information. Here, it is necessary to create a fixed standard for each available information that may be recorded by a sensor or be produced by an application. An example for standardized sensor readings is SensorML,[1] which provides a model consisting at least of an identifier, a definition and a unit of measurement for each reading, whereas all attributes can be chosen by the user but are fixed afterwards.

However, defining standardized data models leads to different problems. If we want to solve the problem of heterogeneity by using standards and if we do not want to violate the common idea of IoT, we have to define them in such a way that they comprise all existing concepts. For example, for the concept *Temperature*, a standard also has to define all sub concepts, such as *Indoor, Ambient or Room Temperature*. To allow a common understanding of the general data schema, the standard has to define at least those basic concepts. Here, further problems arise since the usage of those concepts depends on their context. For instance, if a person is located in a room, the room temperature corresponds to the indoor temperature. However, the ambient temperature may either be the temperature outside the room or outside the building. Most of today's data modeling approaches cannot cope with such unpredictable cases.

[1] http://www.ogcnetwork.net/SensorML

Pomp, A., Paulus, A., Jeschke, S. and Meisen, T.
ESKAPE: Information Platform for Enabling Semantic Data Processing.
DOI: 10.5220/0006324906440655
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 2*, pages 644-655
ISBN: 978-989-758-248-6

The example shows that the definition of standards for a common data model among various sensors is already difficult and it becomes more complicated when considering further data sources. For the definition of a corresponding standard to be successful, all software developers and hardware manufacturers would have to accord it and apply it to already rolled out devices or software. While parts of these problems can be simplified by limiting the standard to a specific domain, e.g., smart home or a specific company site, it results in the drawback that those devices can then only be used in this domain unless their standard is translated to be compatible with another one. This leads to additional work and is a contradiction to the common idea of connected systems (e.g., Industrial IoT) in which each participant's data should be readable regardless of domain knowledge.

Another possibility for solving the described problems is the use of ontologies, which are widely spread in the semantic web. Instead of standardizing data models, we can use ontologies to define the current view of the world. This allows the definition of semantic models based on the ontology, creating an abstraction layer and thus being able to view the data on an information level rather than a data level. This meta level enables the comparability and analysis of data sources with different data models but identical information.

However, ontologies also suffer from various disadvantages. First, the definition of an ontology is a complex task which in advance requires to collect and model all the knowledge that will be required. Second, ontologies suffer from inflexibility. Due to the complex generation, an ontology is usually generated for a specific use case in a specific domain, e.g., medicine. Every annotation that is missing in the ontology will not be available when creating a semantic model later on.

Since the (Industrial) Internet of Things is fast and continuously evolving and new devices and sensors are proposed every day, a static ontology will not be able to model each requirement. In addition, covering the complete required knowledge from the beginning will also be challenging, even within an enterprise.

To solve the described problems of heterogeneous data sources and the sophisticated definition of static ontologies, we are developing the information-based data platform *ESKAPE* (Evolving Semantic Knowledge and Aggregation Processing Engine) for structured as well as unstructured batch and streaming data that is capable of handling data on a semantic information level. To feed data into ESKAPE, users (e.g., data source owners) can add data sources and describe them with a custom semantic model. Instead of generating the model based on a pre-defined ontology, users create the semantic model either by using concepts available in the *knowledge graph* managed by ESKAPE or by defining their own domain-specific concepts leading to an expansion of ESKAPE's knowledge graph. Hence, compared to ontologies created top-down, this knowledge graph is capable of adapting new and unknown concepts and relations bottom-up based on the added data sources and their semantic models. Other users can use the published data sources for analyzing data or developing applications based on it.

We run ESKAPE in a closed enterprise environment, but due to compliance issues of enterprises, we evaluated our approach in an open real-world scenario. We set up a challenge in which teams had to develop smart city applications based on various data sets (batch and streaming data) that we collected from different local providers, such as the local bus company or bicycle sharing company. The evaluation shows that the provided platform features, like a semantic search, information conversion, data format conversion or semantic filtering simplify the development of real-world applications significantly. Hence, ESKAPE forms the foundation for enabling true semantics in the evolving IoT.

The remainder of this paper is organized as follows. Section 2 provides a motivating example and Section 3 discusses related work. Based on the example and current state of the art, we discuss the concepts of ESKAPE in Section 4 as well as its functionality (Section 5) and architecture (Section 6). Finally, we present the evaluation results in Section 7 before we conclude in Section 8.

2 MOTIVATING EXAMPLE

In this section, we provide a motivating example illustrating the necessity for future IoT platforms to consider knowledge about the semantics of data and to offer a semantic-based processing. In addition, we discuss why usual ontologies are not sufficient for modeling the semantic knowledge of all data sources that are added to the platform.

The scenario consists of an app developer D_1, who wants to create an application A_1 suggesting to the end users if they should take a bus or rent a bicycle to reach a certain location. For giving the suggestion, the application uses the current weather conditions, the current position of a bus and the number of available bicycles at the nearest station.

To create the application, the developer searches for data available in her city. On the local Open Data platform, multiple publishers P_i offer data. P_1 offers weather data DS_1 via its HTTP API. The weather data is available in XML format and includes a timestamp, the current temperature in $^{\circ}C$ as well as the current amount of rain. P_2, a public transportation company, provides DS_2 as an AMQP stream in JSON format containing the current timestamp and the GPS position of each available bus represented by bus number and a target stop. P_3 offers DS_3 as CSV table available via HTTP containing the current number of available bicycles at a renting station, the position of this station as address (city, street, house number) as well as the date and time of the last update.

When examining this scenario, we identify different drawbacks of common solutions for the current process of data publishing as well as data consuming. When using the data, D_1 currently has to deal with three different data models in three different formats (JSON, CSV, XML) available via two different approaches (HTTP, AMQP) leading to unnecessary implementation overhead. In addition, D_1 has to deal with information in different representation forms. There exist two different location formats (address and GPS location), a temperature in a predefined unit as well as date and time information formatted either as date or as timestamp. These representations again lead to a higher unnecessary implementation overhead.

When publishing data, the parties may also face different challenges. A common platform without semantics will not be able to provide any knowledge about the data. For example, the platform would not be aware of the unit of the temperature and could therefore not convert it into an appropriate unit. If we assume that the platform is already using an ontology for representing information, the ontology would need to cover any concept and relation that will be available in data sets that are added in the future. Otherwise, the platform could not link those unknown data attributes to other data sets containing the same information. While defining a comprehensive ontology may be possible in a closed and controllable environment (e.g., inside a single department of a company), it is very unrealistic for a global company with multiple sites in different countries or in an open scenario where data sources are added over a long time period by multiple independent actors, such as local companies, city administration or private end users. This shows that we are in need of a more generic and evolving approach when dealing with semantics in the (Industrial) Internet of Things.

3 RELATED WORK

In a broad field like the IoT, many other research topics are related to our work. Such topics are data integration and semantic model generation, ontology learning, community-driven ontology engineering and IoT platforms.

In the area of data integration, multiple approaches focus on solving the problem of heterogeneous data sources, such as query reformulation techniques (Global or Local as View) or ontology-based information integration (Ahamed and Ramkumar, 2016).

The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) (Garcia-Molina et al., 1995) focuses on integrating and accessing heterogeneous data sources by following a Global as View approach. TTIMMIS acts as a translator between data sources and a common data model (CDM) allowing to translate queries to the CDM into data source specific queries and returning the result back to the user in the CDM. By additionally providing mediators, TSIMMIS identifies data sources that contain queried information. Compared to TSIMMIS, ESKAPE follows an ontology-based information approach where the data is integrated based on semantic models enabling the use of queries on an information-focused level.

Knoblock et al. propose in multiple papers (Taheriyan et al., 2014) (Knoblock and Szekely, 2015) (Taheriyan et al., 2016) (Gupta et al., 2015) a platform, called KARMA, which follows the ontology-based information integration approach. This platform allows integrating data of multiple formats (JSON, XML, etc.) based on a pre-defined ontology. Data sources that are attached to Karma use semantic models. For that, Knoblock et al. focus their work on automatically creating semantic models based on a fixed ontology per use case where the ontology does not evolve over time. Another project proposed by Meisen et al. describes a framework for handling heterogeneous simulation tools for production processes by using semantics based on a static ontology (Meisen et al., 2012). Compared to these approaches, ESKAPE focuses on an evolving knowledge graph that learns from semantic models that were created by users for their provided data sources.

In the area of ontology generation and learning, OntoWiki (Hepp et al., 2006) addresses the problem of the complex ontology generation by crowdsourcing ontologies with the help of community members. In OntoWiki, users create Wiki articles for concepts and describe them in natural language. If a concept is missing, users can create a new article for it. However, the evolving ontology OntoWiki is created by community members where each one can modify the work of the other. Without well-defined processes, the ontology will not achieve stability. Hence, ESKAPE mitigates this problem by supervising the knowledge graph generation using external knowledge databases.

The approaches (Xiao et al., 2016) and (He et al., 2014) deal with learning ontologies from unstructured text in an automatic fashion for coping with the high effort that arises from creating ontologies by hand. Therefore, the authors use DBPedia as a supervisory tool for guiding a user during the learning of an ontology. Their work is a first attempt to automatically generate ontologies. Xiao et al. focus on learning ontologies from unstructured text where the learned ontology is created based on a fixed corpus of input sources that are not provided by different users whereas ESKAPE's knowledge graph learns from user provided semantic models.

Furthermore, the project Semantic Data Platform (SDP) proposed by (Palavalli et al., 2016) uses ontologies and semantic models to enable IoT-based services. They also have the goal to deal with the number of upcoming heterogeneous data sources in the Internet of Things. Palavalli et al. provide a concept to solve the problem by allowing vendors to specify semantic models for their new devices. These models are then used for evolving the underlying ontology, which can be used as a common data model. Based on SDP, devices and applications can subscribe to events (e.g., temperature higher than 25°C) that are obtained from the collected IoT data. This collection is enabled by the use of semantic queries. Palavalli et al. just focus on adding devices with semantic models that were provided by vendors. Their work does not consider other data sources, such as applications, that may deliver data, as well as devices where the semantic model may change over time (e.g., modular smart phones). Moreover, the future work presented includes extending the semantic approach to collaboratively evolve IoT data model standard among device vendors whereas our approach does not require to establish a standard.

Similar interest in interoperability of IoT systems has been stated in a new project named 'BIG IoT' (Dorsch, 2016). It focuses on the task of offering a single API for multiple IoT applications, bridging the gap between multiple independent data storages. Another project, called Anzo Semantic Data Lake (Cambridge Semantics, 2016), focuses on developing a data lake by adding context to all kinds of data sources. In both projects, the participants do not state how this task can be achieved but describe the need for adding semantics to raw data sources.

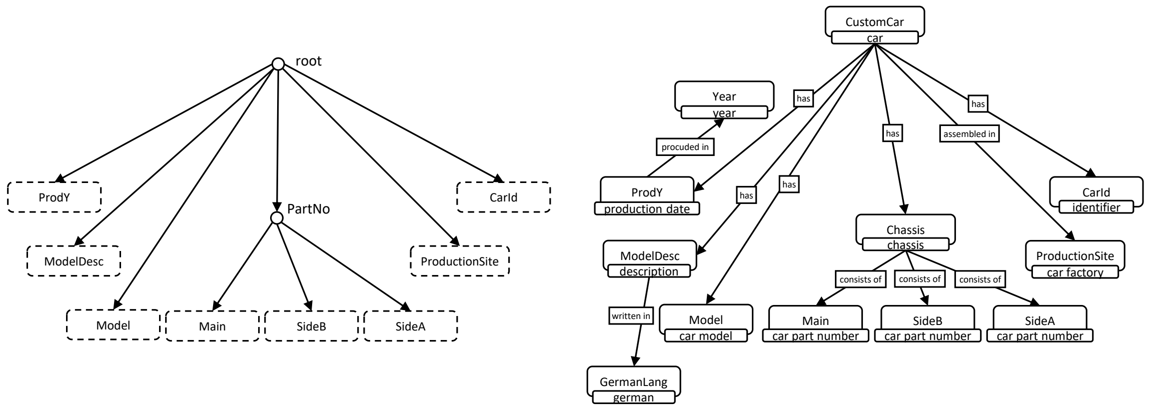
4 CONCEPT

In this section, we will describe the concept behind the platform, specifically the semantic concepts used by ESKAPE. First, since there exist contrary definitions of terms used in literature, we define the following terms to accomplish a common understanding in the upcoming sections:

- **Data Attribute:** Property of a single point of data; e.g., column in a table described by a header or the key in a map identifying all values available under this key.
- **Data Value:** Describes the value of one specific *Data Attribute* (e.g., the cell in a table or the value available under a certain key in a map).
- **Data Point:** Describes an actual object including *Data Values* for *Data Attributes* (e.g., a row in a table or a map with data values for available *Data Attributes*).
- **Data Set:** Represents all available *Data Points* (e.g., all rows in a table) that are present at a certain time.
- **Batch Data:** One *Data Set* that is completely added to ESKAPE.
- **Streaming Data:** Consist of multiple *Data Sets* where each *Data Set* arrives at an unknown time t. Consequently, at a certain time t_i multiple *Data Points* can arrive simultaneously.
- **Data Source:** A source from which ESKAPE receives data, such as a stream, a file system or a URL.

To address the challenges that we identified in the motivating example and in the related work, we are developing ESKAPE, an information-based platform that is capable of integrating data from heterogeneous data sources on a semantic level by using a maintained knowledge graph. Common approaches tend to use ontologies for semantic data modeling (cf. Section 3) or other static approaches to annotate data sets with semantic information. A drawback of these approaches is the missing ability to adapt to changes in a rapidly changing environment where new devices come into play almost every day. With ESKAPE, we want to provide a new approach of creating semantics for data sources.

Using ESKAPE, a user typically starts with a *data model* deducted from a sample data set. The data model contains all data attributes which have been found in the analyzed data points thus representing the structure of the observed input data (Figure 1(a)). The identified data model has to be mapped to a semantic representation (cf. Section 5.1).



(a) Data model (b) Semantic model

Figure 1: Identified data model and respective semantic model for an example data set of cars.

4.1 Semantic Models

ESKAPE extends the technique of matching data source models to (static) ontologies and their vocabulary. The platform uses semantic models to describe single data sources. A *semantic model* is a semantic annotation of a data model adding semantic vocabulary and relations to it. This model gives a semantic representation of the information contained in the raw data points. The model itself is user maintained and fixed at the time the data integration starts.

A semantic model (see Figure 1(b)) is created by instantiating an *Entity Type* for each data attribute identified during the data analysis which is to be integrated later. Each Entity Type contains a user-defined name for this attribute in this specific semantic model (e.g., 'ProductionSite') and has a semantic concept assigned to it, defining the semantic properties of this attribute (e.g., 'car factory'). The Entity Type is therefore an instantiation of a generic concept for a specific semantic model and is defined by it. Furthermore, the user is not limited to attribute-assigned Entity Types inferred by the source data and can define advanced Entity Types to create more complex models (e.g., 'Chassis'). Those advanced types are not related to a specific data attribute as they resemble a higher order construct, either by combining multiple Entity Types or by specifying an Entity Type in more detail. We require that each Entity Type has a concept assigned to it as well as each concept to be attached to at least one Entity Type. To relate two inferred or created Entity Types, the user can also define *Entity Type Relations* between those two (e.g., CustomCar *has* CarId). An exemplary result of a semantic model for the data model shown in Figure 1(a) can be seen in Figure 1(b).

4.2 Knowledge Graph

Upon submission of a final semantic model in ESKAPE, the information contained in the model are added to the *knowledge graph*, which is maintained and supervised by the platform. The knowledge graph cannot be modified directly by users and resembles an auto-generated collection of semantic knowledge from all available semantic models. It consists of three types of core components:

- *Entity Concepts*: Entity Concepts are labeled semantic descriptors which cover and uniquely identify a generic or specific mental concept (e.g., *car*).
- *Entity Concept Relations*: Entity Concepts are connected to other Entity Concepts via Entity Concept Relations which describe invariant connections (e.g., car *has* production date).
- *Relation Concepts*: Relation Concepts are descriptors for kinds of relations, such as *isA*, *producedIn* or *consistsOf* which might be used by Entity Concept Relations and Entity Type Relations. Each Relation Concept contains a name and a set of properties defining the mathematical classes of relations it belongs to like transitive, reflexive, symmetric, etc.

With the availability of an evolving knowledge graph, the creation of a semantic model for a data source is simplified as elements from the knowledge graph can be directly re-used in the annotation process. Re-using elements from the knowledge graph's vocabulary saves the user from defining common concepts and relations multiple times. In addition, it creates links between elements of the newly created semantic model and the knowledge graph. This allows ESKAPE to make suggestions during the modeling process as well as using reasoning based on the previously available knowledge in the knowledge graph, e.g., noticing if a relation is probably used in the wrong direction. However, the user's semantic model can be valid even without elements of the knowledge graph and will not be modified by the system automatically.

The only exception to that is the auto-detection of probable concepts during the analysis phase. Based on learned patterns for certain concepts, ESKAPE can provide an initial semantic model using only previously known concepts and relations. The system learns about those patterns when data is integrated for a specific semantic model. If concepts of the knowledge graph have been reused, example values and/or patterns for certain concepts can be identified and analyzed to build patterns.

4.3 Evolving the Knowledge Graph

Upon submission of a new semantic model to ESKAPE, the knowledge graph adapts the newly gathered information by merging a model's concepts and relations into the graph. Re-used elements act as anchor points for matching elements. Previously unknown concepts are matched using heuristics and external resources to detect, e.g., synonyms. If no suitable match is possible, a new Entity Concept is created from the custom concepts and added to the graph using the provided relations to previously known elements of the graph. The same procedure is applied for newly encountered relations, which are converted to Relation Concepts. Entity Concept Relations are deducted from Entity Type Relations (which implicitly relate Entity Concepts) which occur multiple times, further extending the knowledge graph. Figure 2 illustrates the described information model schemes.

5 FUNCTIONALITY AND IMPLEMENTATION

To enable users sharing their data with others, we differentiate between providing data sources to ESKAPE and retrieving and processing the provided data sources from it. The former process includes an initial analysis of the data source (cf. Section 5.1), the semantic modeling of the available information (cf. Section 5.2) as well as the data integration (cf. Section 5.3). After ESKAPE integrated the data, users can start different kinds of processes on the data in-

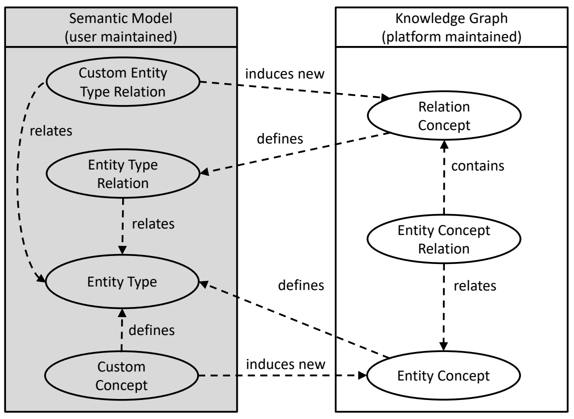


Figure 2: Overview about the different components that are used for constructing semantic models for data sources and their related elements in the knowledge graph.

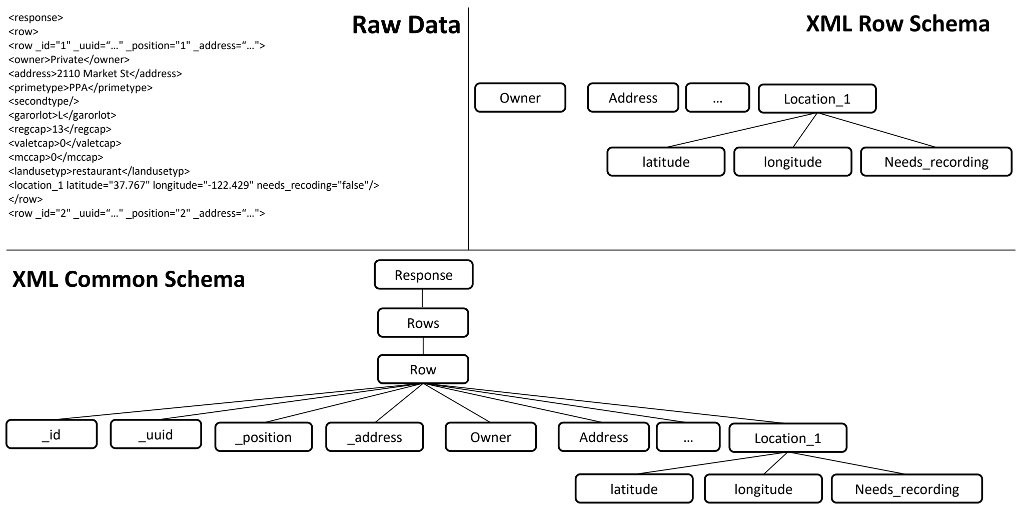
cluding data enrichment, transformation or analysis (cf. Section 5.4) as well as querying and obtaining data (cf. Section 5.5).

5.1 Schema Analysis

When a user connects a data source to ESKAPE for the first time, it collects an example data set and analyzes it to identify a potential schema. Afterwards, ESKAPE presents and visualizes the result to the user to simplify the semantic model creation in the next step. However, gathering and analyzing the raw data schema leads to different challenges.

For analyzing the schema, a sufficient amount of data points is required since a single data point may not contain all data attributes. For batch data, no problems occur as it is a self-contained data set that does not change over time and ESKAPE can just consider each data point to obtain a comprehensive schema. For streaming data, it is unknown if the number of collected data sets and their included data points cover the complete schema. Thus, our current solution observes streaming data for a certain amount of time (configurable by the user). Based on all collected data sets and data points, ESKAPE proposes a comprehensive schema to the user. If the users recognize any flaws, they can refine the schema, e.g., by manually adding missing data attributes that were not covered.

When analyzing the schema, we are not interested in a superficial schema for the data format, but in the relationship between multiple data points and especially the structure of a data point (more precisely, in the relationship between the different data attributes of a single data point). While a data set is available in a fixed format (e.g., CSV or JSON) and follows a



```
<response>
<row>
<row _id="1" _uuid="…" _position="1" _address="…">
<owner>Private</owner>
<address>2110 Market St</address>
<primetype>PPA</primetype>
<secondtype/>
<garorlot>L</garorlot>
<regcap>13</regcap>
<valetcap>0</valetcap>
<mccap>0</mccap>
<landusetyp>restaurant</landusetyp>
<location_1 latitude="37.767" longitude="-122.429" needs_recoding="false"/>
</row>
<row _id="2" _uuid="…" _position="2" _address="…">
```

Figure 3: The example illustrates a raw XML data snippet and two possible recognized schemas depending on the used format. For the XML Common Schema, the result schema includes all available XML properties. In contrast, the result for the XML Row Schema Analysis considers also the semantic structure of the data (table) and solely focus on the important concepts.

pre-defined standard, (e.g., RFC 4180[2] for CSV), the data may be represented in different ways. Depending on the way these formats are used, the semantic of the data attributes and data values inside these formats changes. This might imply whether the data attribute or value is significant or not.

Thus, the schema analysis does not include all the format specific details, but rather the semantics implied when using the format. Figure 3 depicts an example containing raw data from the San Francisco Open Data platform[3] in XML format. When performing a common schema analysis for this data snippet, we receive the result shown in the lower part. This schema contains data attributes that do not generate any actual semantic value (e.g., response, rows, row). Thus, the user should not define a semantic model for those parts of the data.

To increase usability for the semantic model creation (cf. Section 4), we examined multiple data sets from different Open Data platforms and identified patterns allowing to strip all the unnecessary data attributes. One example (XML Row) is illustrated in Figure 3. This schema implicitly considers that the contained data represents a table consisting of multiple rows, just like a CSV file. Thus, this schema analysis automatically strips all the data attributes that should not be covered by the semantic model. Further identified subtypes of formats are:

- **CSV:** covers all data sets where the values for data attributes are separated by a delimiter.
- **JSON Lines:** covers all data sets where the single data points are in line separated JSON format.
- **JSON Tabular:** used for tabular data represented as JSON. This format is recommended by the W3C[4] for tabular data that contain additional meta data.
- **JSON Common:** used for all JSON data sets that are not covered by any other subtype (e.g., JSON Tabular).
- **XML Row:** tabular data represented in XML format.
- **XML Common:** used for all XML data sets that are not covered by any other subtype (e.g., XML Row).

Since maintaining a schema analysis per subtype leads to high implementation overhead, especially when changing constraints for the analysis, we decide to translate each identified subtype into a platform internal format (JSON) and perform the schema analysis based on this format. We define for each subtype an appropriate conversion process which translates the subtype into an equivalent representation in JSON. This allows us to define all schema analysis constraints based on a single format. If constraints for a single subtype change, we just have to adopt the corresponding translation process.

[2]https://www.ietf.org/rfc/rfc4180.txt

[3]https://data.sfgov.org/Transportation/Parking/9qrz-nwix

[4]https://www.w3.org/TR/tabular-metadata/

5.2 Modeling Information of Data Sources

After recognizing a schema for a data source, we present the result to the user. Based on the determined schema, the user creates the semantic model using the buildings blocks described in Section 4. Following the example illustrated in Figure 1, the user assigns each data attribute (*ProdY, ModelDesc, Model, Main, SideB, SideA, Plant, CarId*) an Entity Type. For instance, the Entity Type *ModelDesc* referencing the Entity Concept *description* is assigned to the attribute *ModelDesc*. In addition, the user defines more complex Entity Types which are not mapped to data attributes directly (e.g., *CustomCar* referencing the Entity Concept *car*). By defining Relations between the Entity Types, the user completes the view of a car entity and precisely defines the characteristic of a custom car entity named 'CustomCar' consisting of attributes 'ProdY'(production date), 'Model'(car model), 'Chassis'(chassis), 'ProductionSite'(car factory) and 'CarId'(identifier). Figure 4 illustrates the described scenario using the modeling GUI during the semantic model creation phase in the prototype web client of ESKAPE. The user creates new Entity Types by dragging concepts onto attributes or into free space. An Entity Type is then created automatically, initially assuming the name of the attribute or the concept, respectively.

Our approach comes with three novelties allowing to continuously evolve the knowledge graph over time. This is important for a growing system, such as the Internet of Things, as it is constantly expanded by newly added sensors and data emitters with new semantic models. Thus, we first allow the user to use concepts that are not available in the knowledge graph yet. If the user wants to define an Entity Type but no appropriate Entity Concept is available, then the user will add this concept to the knowledge graph. To ensure that the concept is added at an appropriate position in the graph, we use external data sources, such as semantic networks (cf. Section 4).

The second novelty is that the knowledge graph learns from the semantic model how entities are related to each other. Let us assume that the Entity Concept *car* was only connected to the Entity Concepts *description*, *identifier* and *production date* before the new semantic model was added to ESKAPE. After defining the semantic model and adding it to ESKAPE, the knowledge graph learns that the entity *car* can also have a *car factory*, a *chassis*, and a *car model* which can then be proposed to users which intend to add other data sets containing information about cars.

Another novelty of our approach is that the data attributes recognized by the schema analysis are not mapped in a one-to-one fashion onto entities of the semantic model. Instead, we allow the user to refine the schema and map the semantic model on the refined schema. In the example illustrated in Figure 1, we created a semantic model where the *SideA* attribute was modeled by a *car part number* Entity Concept and a relation to the *Chassis* Entity Type. If the user wants to define a more fine granular model, she can split the *SideA* attribute based on a regular expression into two attributes c_1 and c_2 where c_1 contains the serial number and c_2 the manufacturer number. By mapping c_1 to a corresponding Entity Concept, such as *serial number*, and c_2 to the Entity Concept *manufacturer number* a more detailed semantic model will be created. Besides the splitting of a data attribute into multiple ones, we also offer to split lists into multiple lists or single data attributes by using regular expressions and defining repetition cycles. We call fields that are split into multiple ones *Composite Fields*. Furthermore, we also allow users to remove unwanted data attributes from the schema for publishing.

This example shows that the definition of a semantic model is not unique. The user just provides her view of the data even if there might exist more precise semantic models.

5.3 Data Integration

After the user has created the semantic model for a data source, ESKAPE is capable of integrating the data. The main goals of the data integration are:

- Syntactic homogenization
- Splitting composites into single attributes
- Linking specified entity types to the data attributes of the refined schema
- Marking data values that could not be integrated
- Data point homogenization
- Discarding invalid data points

The syntactic homogenization ensures that all data types (text, number, Boolean and binary values) are integrated into a unified representation, which simplifies the later handling of the data. For instance, Boolean values that occur as *0*, *1* or *F*, *T* are mapped to *False* and *True* and numbers are represented by the American notation. We do not perform a semantic homogenization (e.g., convert all temperatures to a uniform unit such as $^\circ F$) since this information is encoded in the semantic model anyway and can thus be used during processing.

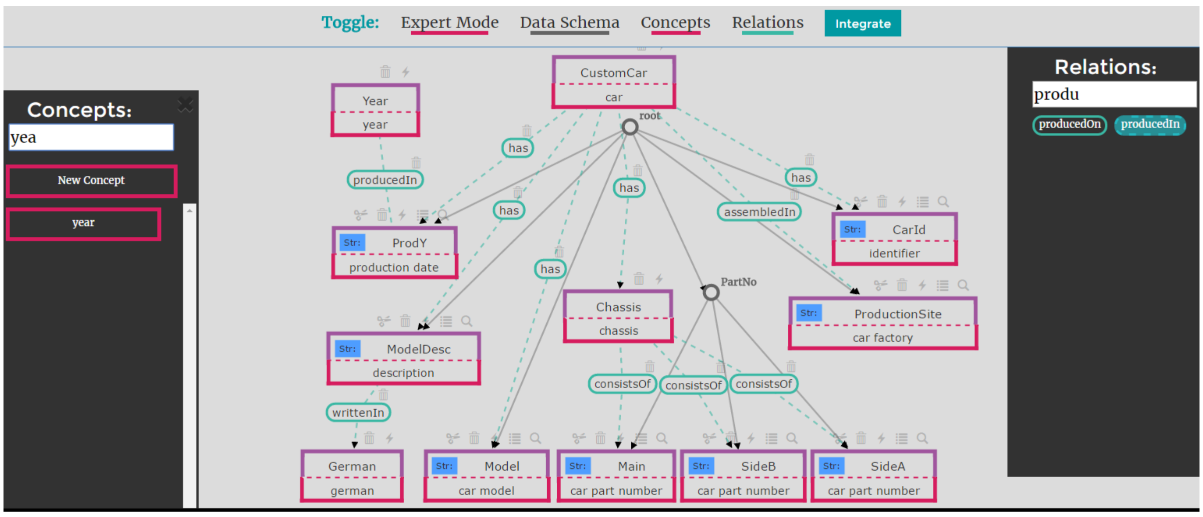


Figure 4: Prototype view of the modeling GUI of ESKAPE. For overview reasons, Entity Types and Entity Concepts are combined. Node top: Entity Type (e.g., 'CustomCar'), node bottom: Entity Concept (e.g., 'car'). The view can be modified by toggling displayed elements such as the data model.

In addition, the integration splits composite fields into multiple data attributes and links all the data attributes to their semantic entity type. If, for instance, the splitting or the syntactic homogenization fails (e.g., data type is marked as number but the data value was a string), then the whole data value is marked as invalid. We explicitly store those values, since we do not want to lose information and we do not know the scenario where the data might be used. We only discard complete data points if errors occur that do not allow us to link the data to its semantic type (e.g., parsing errors).

After integrating data sets (batch or streaming), ESKAPE stores the integrated data in our platform specific Semantic Linked Tree format, called *SLTFormat*. This format offers the user information about the semantic type, the data value, the syntactic type and if the value was successfully integrated, enabling true semantic interoperability on the data sets. To additionally enable real-time analysis, streaming data are directly forwarded to the next processing step.

5.4 Data Enrichment, Transformation and Analysis

After the successful integration of data sources, the data can be used for data enrichment, transformation and analysis. This step enables the user to perform *heavyweight processing* based on one or more selected data sources, their semantic models and additionally parameters defined by the user. The result is a new data source with its semantic model that again is persisted and can be published on ESKAPE.

In processing, we differentiate between *Data Enrichment*, *Data Transformation* and *Data Analysis*. We define Data Enrichment as the process of extracting and persisting new information by using data attributes that are already available in the data. For instance, assumed we have a data source containing a data attribute that holds a description text. A typical data enrichment step would be to determine the language of the description, so we explicitly generate information that is implicitly available and persist it. Moreover, the information that was added will not change over time and is use case independent. Opposed to this, *Data Transformation* is the process of transforming or converting data attributes based on knowledge that is offered by the semantic model. An example is the conversion of a data attribute containing temperature data in $^\circ C$ into $^\circ F$. Compared to *Data Enrichment*, *Data Transformation* changes the data depending on the user's use case. Finally, *Data Analysis* involves all tasks that extract new information by, e.g., aggregating multiple data points or joining data sources based on defined criteria.

After the processing of the data was successful, ESKAPE persists the newly created data source on the storage and adds the corresponding semantic model to the knowledge graph. In case of streaming data, the data is directly forwarded to satisfy possible real-time constraints.

5.5 Querying Data and Information

Aside from data integration and processing, users can also extract data from ESKAPE. Since users have dif-

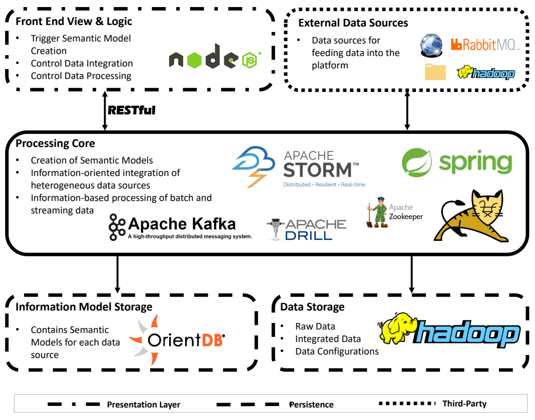

Front End View & Logic
Trigger Semantic Model Creation
Control Data Integration
Control Data Processing
External Data Sources
Data sources for feeding data into the platform
RabbitMQ
hadoop
RESTful
Processing Core
Creation of Semantic Models
Information-oriented integration of heterogeneous data sources
Information-based processing of batch and streaming data
APACHE STORM™
Distributed • Resilient • Real-time
spring
Apache Kafka
A high-throughput distributed messaging system.
APACHE DRILL
Apache Zookeeper
Information Model Storage
Contains Semantic Models for each data source
OrientDB
Data Storage
Raw Data
Integrated Data
Data Configurations
hadoop
Presentation Layer
Persistence
Third-Party

Figure 5: Overview about the architecture and technologies of ESKAPE.

ferent requirements based on the use case, we offer multiple approaches for the data extraction. The user can either trigger the extraction for an available data source on a semantic level by selecting a data source and the semantic parts of it (e.g., selecting the concrete requested entity types) or by actively querying for specific data sets using SQL, which, however, requires the user to consider the semantic model.

For the semantic extraction, we offer the user to perform additional filters and limits on the data and to select a specific data format, such as XML or JSON, in which the data should be extracted. The user can also decide if she wants to receive the data as a stream or as a file. When requesting batch data, a file can be downloaded or emitted as stream for a certain amount of time enabling the simulation of a data stream. For streaming data, the data is sent directly to the user. We do not offer to stream the data, which is persisted by ESKAPE, into a file since it is identical to extracting historic streaming data for a defined time span.

The SQL extraction, on the other hand, addresses expert users who want to perform more sophisticated data extraction by using familiar SQL syntax on a semantic level.

6 ARCHITECTURE AND TECHNOLOGIES

To ensure scalability, reliability and performance of ESKAPE, we are using technologies that allow us to distribute tasks among a cluster. Figure 5 provides an overview of the developed architecture and used technologies. The architecture of ESKAPE consists of a data processing back end (Processing Core) and a visualization front end used to add data sources to ESKAPE, maintain their state and define their semantic models. It also allows for defining processing, e.g., enrichment steps on selected data sources and data querying including visualization of the returned results. Furthermore, there exist several interfaces using popular data exchange formats to retrieve queried and streaming data from ESKAPE.

To store the aforementioned knowledge graph and the semantic models of each data source, we are using the graph database OrientDB[5]. Data storage is handled in a Hadoop[6] cluster. This cluster contains the raw acquired data from gathered data sources and streams as well as the integrated data sets, including enriched data sets and additional data from third-party enhancements, such as text language analysis. The cluster storage concept enables ESKAPE to redundantly store all acquired data sets in the SLTFormat (cf. Section 5.3).

ESKAPE performs data processing (cf. Section 5), in the Processing Core, which makes up for the largest part of the platform. The Processing Core uses Apache Storm[7] to process data sets in user-defined chains expressed as directed acyclic graphs (DAGs) called *topologies*. A topology consists of spouts, which represent various data sources, and bolts, which contain processing logic inside the process chain. Combining different spouts and bolts yield a DAG running in the Storm Cluster. Figure 6 shows an example of a simple stream processing chain in a public transport domain. Each topology is encapsulated in its own instance but may be distributed over several machines, called nodes. Those topologies are used to form the processing part of any heavy-weight operation as defined in Section 5.4. For each specific processing task, a new Storm topology is created by ESKAPE, besides the always present ones handling data integration and export. By design, topologies run infinitely, waiting for new data to be emitted from the spouts, until ESKAPE shuts them down. This is kept to process streaming data, however, on the batch integration part, a topology will be shut down when the source does not yield any more data. Continuous domain-specific data enrichment tasks on integrated data sets defined by users also tend to run indefinitely.

We implemented the communication between the core platform and topologies running on the Storm cluster using Apache Kafka[8]. It enables the core platform to receive messages, such as errors or status updates from all the topologies. Due to the Kafka de-

[5]http://orientdb.com/orientdb/
[6]http://hadoop.apache.org/
[7]http://storm.apache.org/
[8]http://kafka.apache.org/

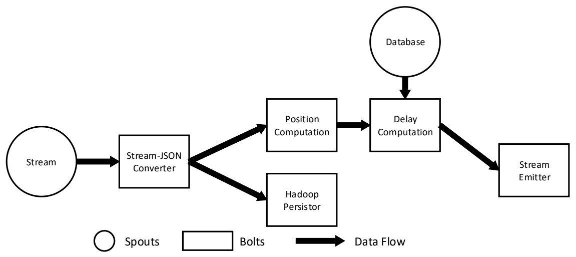


Figure 6: Example Storm processing chain for calculating public transportation delay from current vehicle positions, transmitted as a proprietary stream (not JSON), and the scheduled timetable saved in the database.

sign, it also facilitates to directly persist those messages allowing for a detailed downstream analysis.

Data extraction from the Processing Core is done in three ways. To distinguish between those ways, one has to differentiate data extraction and data enrichment. Topologies which are used to enrich data contain a bolt to write the generated data back into the Hadoop storage. In the case of stream processing, the processed data is stored by default and also forwarded to the aforementioned interfaces providing data exchange technologies. ESKAPE currently supports stream-, HTTP- and file export, where in the case of stream processing RabbitMQ[9], an AMQP implementation, is used.

To allow direct data extraction from the Hadoop cluster (cf. Section 5.5), Apache Drill[10] provides an interface to execute a limited set of SQL queries on the integrated data. ESKAPE can then quickly export the queried data from the Hadoop Cluster. If any kind of processing is needed, which extends simple operations such as joining and filtering data sets, a new Storm topology is needed to handle those requests.

7 EVALUATION

As we could not evaluate ESKAPE in an enterprise setting, we defined an open-world scenario in the first place. For evaluating ESKAPE, we set up a hackathon in which teams had to develop mobile applications based on the data that were published on the platform. The goal of this challenge was to get feedback from users to identify potential design flaws as well as missing features.

To provide a sufficient amount of data originating from real-world sources, we teamed up with the local city administration as well as local companies coming from different domains. For example, the local bus company provided real-time data about the delay of approaching buses for each stop via an HTTP API and the regular schedule of each stop as CSV files. Other parties published real-time weather data, the locations of available WiFi spots or the number of free slots at bicycle rental stations. Altogether, nine different parties provided 64 different data sets from which 15 were streaming data and 49 were batch data. All streaming data was published via HTTP APIs, which were polled in different time intervals (min=1min, max=1day) whereas the batch data sets were published as files (42) or via an HTTP API (7) without any update frequency (one-time polling). From the 64 data sets, 20 were published as CSV, 11 as XLSX (Excel), 2 as JSON Common, 2 as JSON Tabular, 10 as JSON Lines, 3 as PDF, 2 as SHP, 4 as WMS and 10 as XML Common. Since our prototype did not support all formats at the moment, we converted XLSX files manually to CSV and we did not import PDF (not a M2M format) as well as SHP and WMS resulting in 55 available data sets for the hackathon.

For developing applications, 92 persons grouped into 25 teams participated in the hackathon. Each submitted application used at least three different data sets for developing their application. We discovered that mobile applications developed for the Android operating system requested the data as JSON whereas the submitted iOS applications requested the data as XML. Each team that requested data sets with the semantic concept *Temperature* requested a conversion before extracting the data from ESKAPE. Based on interviews with the teams that used such features, we confirmed that the conversion for units as well as formats simplifies the development process for several OS/platforms. However, we also received feedback about missing features, such as an HTTP API (no team liked AMQP extraction) and merging of data directly on ESKAPE (not enabled during the hackathon). Two teams also requested to integrate own processing logic into the processing pipelines, which will be addressed by the analytic layer in the future.

8 CONCLUSION AND FUTURE WORK

In this paper, we presented our approach to enable semantics in the continuously evolving (Industrial) Internet of Things. The main idea to mitigate problems of current solutions, such as dealing with heterogeneous data sources and defining sophisticated static ontologies, is a platform, called ESKAPE, that uses semantic models based on a knowledge graph for de-

[9]https://www.rabbitmq.com/
[10]http://drill.apache.org/

scribing data sources. Instead of pre-defining all concepts and relations in the graph, we presented an approach where the graph is able to cope with new concepts and relations from data sources that are added to ESKAPE. To create the semantic models, we proposed a detailed schema analysis for which we identified more fine granular subtypes of data formats that directly consider additional semantics, such as table rows described in XML. Based on the schema analysis and the user-defined semantic models, the data is integrated into a unified format that directly links semantic concepts and data attributes. Afterwards, integrated data can be used to perform data enrichment, transformation and analysis and to extract the result based on various approaches, such as SQL queries or an extraction on a semantic level. The latter especially enables the subscription of processed, transformed and enriched real-time data sources on a semantic level, enabling true semantics for the Internet of Things.

In the near future, we plan to improve the current data processing by allowing the user to create own topologies using the web interface. Currently, all topologies are created and made available by ESKAPE's developers. Enabling the user to create and modify own topologies during runtime will allow for complete autonomous usage of ESKAPE. Furthermore, modifying a pipeline requires a restart of the running node resulting in potential data loss. A sophisticated buffering technique will prevent the data loss during modification.

In addition to these improvements, the user will get advanced support when creating semantic models. Our goal is to analyze the given input data and automatically propose a full semantic model to the user. This requires more detailed analysis of the given data attributes and machine learning approaches to estimate the best assignment of Entity Concepts and Types. In addition, adding the capability to change the now fixed semantic models during runtime will help to adapt to changing data sources. By extending ESKAPE's semantic search to support natural language queries, we will additionally improve its usability.

Additional future work will focus on improving the supervising of the knowledge graph creation. By using appropriate machine learning approaches on the provided data of the new concepts, we want to improve the automatic supervising resulting in a more resilient knowledge graph.

REFERENCES

Ahamed, B. and Ramkumar, T. (2016). Data integration-challenges, techniques and future directions: A comprehensive study. *Indian Journal of Science and Technology*, 9(44).

Cambridge Semantics (2016). Anzo Smart Data Discovery. http://www.cambridgesemantics.com/.

Dorsch, L. (2016). How to bridge the interoperability gap in a smart city. http://blog.bosch-si.com/categories/projects/2016/12/bridge-interoperability-gap-smart-city-big-iot/.

Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. (1995). Integrating and accessing heterogeneous information sources in tsimmis. In *Proceedings of the AAAI Symposium on Information Gathering*, volume 3, pages 61–64.

Gupta, S., Szekely, P., Knoblock, C. A., Goel, A., Taheriyan, M., and Muslea, M. (2015). Karma: A System for Mapping Structured Sources into the Semantic Web: The Semantic Web: ESWC 2012 Satellite Events: ESWC 2012 Satellite Events, Heraklion, Crete, Greece, May 27-31, 2012.

He, S., Zou, X., Xiao, L., and Hu, J. (2014). Construction of diachronic ontologies from people's daily of fifty years. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavik, Iceland. ELRA.

Hepp, M., Bachlechner, D., and Siorpaes, K. (2006). Ontowiki: Community-driven ontology engineering and ontology usage based on wikis. In *Proceedings of the 2006 International Symposium on Wikis*, WikiSym '06, pages 143–144, New York, NY, USA. ACM.

Knoblock, C. A. and Szekely, P. (2015). Exploiting semantics for big data integration. *AI Magazine*.

Meisen, T., Meisen, P., Schilberg, D., and Jeschke, S. (2012). *Adaptive Information Integration: Bridging the Semantic Gap between Numerical Simulations*, pages 51–65. Springer Berlin Heidelberg, Berlin, Heidelberg.

Palavalli, A., Karri, D., and Pasupuleti, S. (2016). Semantic internet of things. In *2016 IEEE Tenth International Conference on Semantic Computing (ICSC)*, pages 91–95.

Taheriyan, M., Knoblock, C. A., Szekely, P., and Ambite, J. L. (2014). A scalable approach to learn semantic models of structured sources. In *Proceedings of the 8th IEEE International Conference on Semantic Computing (ICSC 2014)*.

Taheriyan, M., Knoblock, C. A., Szekely, P., and Ambite, J. L. (2016). Learning the semantics of structured data sources. *Web Semantics: Science, Services and Agents on the World Wide Web*.

Xiao, L., Ruan, C., Yang, Zhang, J., and Hu, J. (2016). Domain ontology learning enhanced by optimized relation instance in dbpedia. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. ELRA.