ExeKG: Executable Knowledge Graph System for User-friendly Data Analytics

Zhuoxun Zheng*
zhuoxun.zheng@de.bosch.com
Bosch Center for AI, Germany
Oslo Metropolitan University, Norway

Baifan Zhou*
baifanz@ifi.uio.no
SIRIUS Centre
University of Oslo, Norway

Dongzhuoran Zhou
dongzhuoran.zhou@de.bosch.com
Bosch Center for AI, Germany
University of Oslo, Norway

Ahmet Soylu
ahmet.soylu@oslomet.no
Department of Computer Science
Oslo Metropolitan University, Norway

Evgeny Kharlamov
evgeny.kharlamov@de.bosch.com
Bosch Center for AI, Germany
University of Oslo, Norway

ABSTRACT

Data analytics including machine learning (ML) is essential to extract insights from production data in modern industries. However, industrial ML is affected by: the low transparency of ML towards non-ML experts; poor and non-unified descriptions of ML practices for reviewing or comprehension; ad-hoc fashion of ML solutions tailored to specific applications, which affects their re-usability. To address these challenges, we propose the concept and a system of executable knowledge graph (KG), which represent KGs that rely on semantic technologies to formally encode ML knowledge and solutions. These KGs can be translated to executable scripts in a reusable and modularised fashion.

CCS CONCEPTS

• **Information systems → Information systems applications**.

KEYWORDS

Knowledge Graph, Artificial Intelligence, Data Analysis

ACM Reference Format:
Zhuoxun Zheng, Baifan Zhou, Dongzhuoran Zhou, Ahmet Soylu, and Evgeny Kharlamov. 2022. ExeKG: Executable Knowledge Graph System for User-friendly Data Analytics. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557195

*Z. Zheng and B. Zhou contributed equally to this research as first authors.

CIKM '22, October 17–21, 2022, Atlanta, GA, USA.

ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557195

1 INTRODUCTION

Data analysis technologies play an important role in a wide range of modern industries and applications, such as recommendation system in internet, production monitoring in automatic manufacturing, pose estimation in robotics etc [2, 8, 22]. Among the technologies, machine learning (ML) attracts substantial yet increasing attention, for its strong modelling capability without the need of explicit programming [7] and the voluminous data that become available due to the introduction of internet of things into manufacturing [6, 24].

Take the quality monitoring of *automated welding* at Bosch as an example, which is an impactful automatic manufacturing process accounting for the production over 50 million cars globally in a year [20]. During the welding process, a high current flows through the car body work pieces to melt the metal materials, which then congeal after cooling down, to form connecting spots to connect the work pieces. Traditional monitoring approaches require to destroy the welded cars to measure the diameters of the connecting spots as prescribed in international and German standards [3, 5], which is extremely costly and produces much waste. In contrast, data-driven methods will reduce the need of destroying welded cars, thus reducing the waste and contributing to more economical and sustainable manufacturing industry [19]. The data analytics projects here involve experts from various domains with asymmetric knowledge background: welding experts, data scientists, measurement experts, managers, etc. They need to discuss extensively, formulate and prioritise the questions according to technical feasibility, company strategies, and invest-return ratio. After that, they design data analytics pipelines to process massive data from many sources like different customers, factories, to solve various questions.

Challenges. Development of such ML anlaytics solutions exist still many challenges of ML practice in the industry. In ML projects where interdisciplinary teams of experts with distinct background are involved (which is often), the transparency of ML (C1) to non-ML experts (e.g., domain experts, managers) is usually challenging [21], since the latter often specialise in their domain knowledge and did not receive excessive training of ML that is often required to understand the sophisticated ML methods and interpret the ML results. The non-ML experts need to understand ML and trust that ML applied in manufacturing robots operating with high electricity

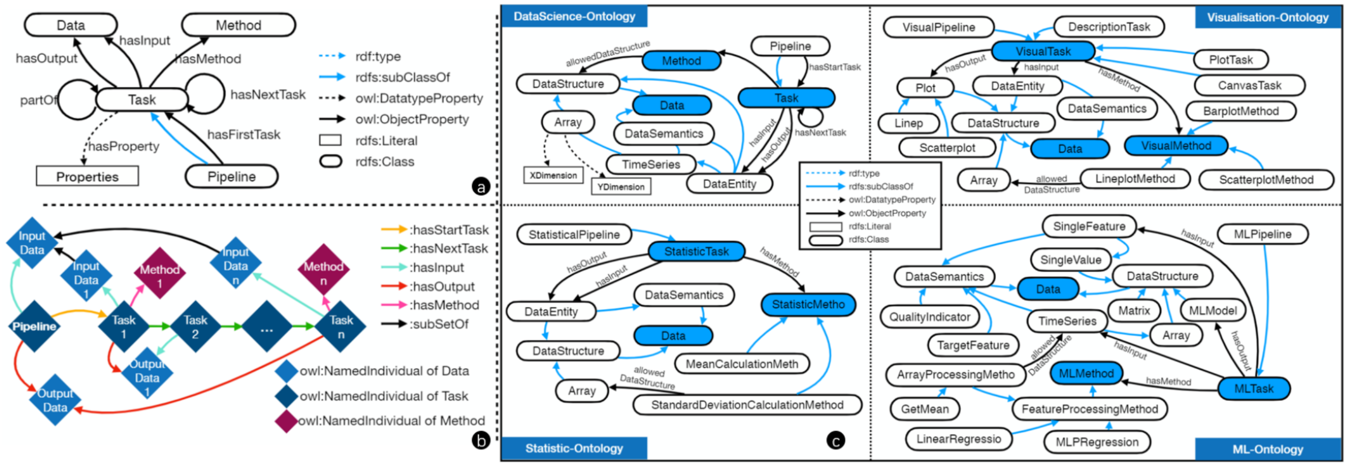

Figure 1: (a) Executable KG framework in the schema level, and (b) in the individual level, (c) KG schemata (ontologies) for the executable KG

can ensure product quality and personnel safety. In addition, in traditional ML projects, the ML procedures, methods, scripts, and decisions are described in the technical language of ML, which is highly dependent on the person who writes the document. ML knowledge and solutions are hardly described or documented in a standardised way (C2), causing difficulties for later review and retrospective comprehension of the projects in big companies like Bosch, which have strict regulations in reporting the details for later audit and analysis. Moreover, ML solutions are often developed in an ad-hoc fashion and tailored to specific applications, which complicates its reusability (C3) for new data or questions [13, 23].

ExeKG System. To address C1-C3 challenges, we present a novel system that allows to combine semantic technologies and ML and enables users with minimal training of data analytics to do data analytics through GUI-based KG construction, without coding. ExeKG system enables this by encoding ML solutions in knowledge graphs (KG), which helps in describing ML knowledge and solutions in a standardised way that follows our KG framework and pre-defined schemata, which contains data science knowledge in formal language. Our system offers KG construction that represent executable data pipelines via GUI-based system for creation, modification, integration and visualisation of KGs. We name our system executable KGs (ExeKG), because our KGs can be translated to modularised and executable ML scripts that can be modified and reused for new data and new questions, besides, these KGs can also be used in other industrial applications such as pipeline verification and selection based query answering [17]. In particular, we focus on three important activities of data analytics practice: (1) visual analytics using various plotting methods to visualise data for intuitive data understanding; (2) statistical analytics with statistical methods to extract insights from data; (3) ML analytics relying on classic ML methods as well as neural networks for classification or regression. The former two are often known as exploratory data analysis and seen as important preceding steps for ML analytics [9].

Demo Overview. The attendees will experience how easy one can analyse data for welding quality monitoring on anonymised industrial data provided by Bosch. With zero knowledge in semantic technologies and a minimal common knowledge of data anlaytics, the users can use our GUI tools in three scenarios for three analytical tasks: to visualise, modify, and create executable KGs for visual analytics, statistic analytics and ML analytics, and see the execution results of the data pipelines translated from these KGs.

2 EXEKG SYSTEM

2.1 Executable Knowledge Graph Framework

We define data, methods and tasks as follows: *Data* $\mathcal{D}$ is a set of facts, statistics, or items of information in forms such as numerals, diagrams or strings organised in different structures such as tables, etc. A *Method* $\mathcal{F}$ is a function in the form of language-dependent script (such as in C++ or Python). A method can take some data which fulfils certain constraints $C_{\mathcal{F}}$ as input and can output specific data. Formally, $\mathcal{D}^{out} = \mathcal{F}(\mathcal{D}^{in})$, if $C_{\mathcal{F}}(\mathcal{D}^{in}) = True$. A *Task* $\mathcal{T}$ is the process of invoking a method by feeding it with some data that meets certain constraints, and by doing so to obtain some other data. Formally, $\mathcal{T}\langle\mathcal{D}^{in}, \mathcal{F}\rangle = \mathcal{F}(\mathcal{D}^{in}) = \mathcal{D}^{out}$, if $C_{\mathcal{F}}(\mathcal{D}^{in}) = True$.

Some tasks have a single method, while other more complex tasks can not solved by invoking a single method but can be unfolded into a sequence of tasks where each task is a part of the complex one. We refer the complex tasks as pipelines $\mathcal{T}_p$. Formally, a pipeline $\mathcal{T}_p$ with input data $\mathcal{D}^{in}$ to get $\mathcal{D}^{out}$, expressed as $\mathcal{T}_p\langle\mathcal{D}^{in}, \mathcal{F}\rangle = \mathcal{D}^{out}$ can be unfolded in the sequence $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$, where:

$$\mathcal{T}_1\langle\mathcal{D}_1^{in}, \mathcal{F}_1\rangle = \mathcal{D}_1^{out}, \mathcal{D}_1^{in} \subseteq \mathcal{D}^{in}, C_{\mathcal{F}_1}(\mathcal{D}_1^{in}) = True; \ldots \tag{1}$$

$$\mathcal{T}_n\langle\mathcal{D}_n^{in}, \mathcal{F}_n\rangle = \mathcal{D}_n^{out}, \mathcal{D}_n^{in} \subseteq \dot{\bigcup}_{i\in\{1,...,n-1\}} \mathcal{D}_i^{out} \cup \mathcal{D}^{in}, C_{\mathcal{F}_n}(\mathcal{D}_n^{in}) = True$$
$$\longrightarrow \mathcal{D}^{out} \subseteq \dot{\bigcup}_{i\in\{1,...,n\}} \mathcal{D}_i^{out}, \ C_{\mathcal{F}} = \dot{\bigcap}_{i\in\{1,...,n\}} C_{\mathcal{F}_1}(\mathcal{D}_i^{in}). \tag{2}$$

Based on the above definitions, we determine the framework for the executable KGs as Fig. 1 a, such executable KG should take the form as Fig. 1 b. Here we split the properties from the data $\mathcal{D}$, which strictly speaking also belong to $\mathcal{D}$, but correspond to the properties rather than objects of a $Task$. Except those $Tasks$ with their *Methods* already been integrated in script, all other $Tasks$ can be modularised in a *Pipeline* and be unfolded into a sequence of $Tasks$. The objectProperty *:hasFirstTask* connects the *Pipeline* with the first task in its unfolded sequence, while *:hasNextTask* connects the task in the sequence with its following task. In this framework, as long as the *Data* and *Properties* of every *Task* fulfil the constraints of the *Method* in the *Task*, the *Task* is executable. If every *Task* in a *Pipeline* is executable, the *Pipeline* is executable. In addition, as a *Task*, the *Pipeline* can also be a part of another *Task*, which represents the modularity of the executable KG.

2.2 Architectural Overview

We now give an architectural overview of our system, which consists of five layers (Fig. 2 a), namely: (non-KG) data layer, application layer, KG database layer, semantic modules layer, and semantic artefacts layer. From the bottom left, we start with the welding raw data collected from production lines. These data are transformed by the Data Integration module (with the help of domain ontologies) to Domain-ML KG, it is a type of welding data KG with its datatype properties are annotated by some classes in data science ontology and thus carries ML annotation. These KGs are used by four types of analytics applications in the application layer.

The domain ontologies include various domain specific knowledge models, e.g., resistance spot welding ontology [12], hot-staking ontology. These ontologies are created based on the upper domain ontology [11], the manufacturing ontology. We briefly introduce ontology here and refer the readers to [1, 4]. In essence an ontology is a formal specification of a domain of interest written in a set of first-order logic formulae of a special form over atomic classes and properties, where each formula essentially says that one atomic class (resp. property) is a subclass (resp. subproperty) of another, and complex classes (resp. properties) are composed from the atomic classes and properties (resp. properties) using logical *and*, *or*, *not* as well as universal and existential quantifiers. Reasoning over ontologies allows to compute logical entailments. The manufacturing ontology is semantically connected with an upper task ontology, the data science ontology (O^{ds}), in a way that the datatype properties in the former one are annotated by some classes in the latter one. A series of task ontologies, including the visualisation ontology (O^{visu}), the statistical ontology (O^{stats}), and ML ontology (O^{ml}), are created based on the O^{ds}. These task ontologies serve as the schemata for the Executable KG Construction module, which encodes the executable data pipelines in the executable KGs, including the visualisation KG, statistical KG, and ML pipeline KG. These executable KGs then can be translated by the Executable KG Translator module to executable scripts for three analytics applications: Visual Analytics, Statistic Analytics, and ML Analytics, which generate the analytics results.

2.3 Semantic Artefacts

Manufacture Ontology and Domain Ontologies are OWL 2 ontologies and can be expressed in the Description Logics $\mathcal{S}(\mathcal{D})$. With its 1170 axioms, which define 95 classes, 70 object properties and 122 datatype properties, the manufacture ontology as the upper domain ontology models the general knowledge of discrete manufacturing process, which refers to a broad range of manufacturing processes. The domain ontologies describe several manufacturing domains at Bosch. These ontologies are created by domain experts in such a way that all classes/properties in the domain ontologies are sub-classes/sub-properties of that in upper domain ontology.

Data Science and Visual/Statistic/ML Ontologies are also OWL 2 ontologies and are created by Bosch data scientists. This ontologies (Fig. 1 c) are expressed using $\mathcal{ALH}(\mathcal{D})$ Description Logic. The data science ontology (O^{ds}) as the upper task ontology formalises the general knowledge of data science activities. It contains three most important classes (Fig 1 b): *Data* that is the class of all data concepts (the existential being in data science), *Method* is the class of all algorithms and functions (the way that data move), whose allowed input, output and parameters are defined, and *Task* is the class of the scripts that invoke the functions, which has an important sub-class, *Pipeline* that consists of a series of ordered tasks (the way that the data movement is organised). Based on O^{ds}, the O^{visu}, O^{stats}, and O^{ml} are created in such as way that all classes/properties in the task ontologies are sub-classes/sub-properties of that in O^{ds}.

Besides, these ontologies also explicitly identify the rules that constrain the input data for these methods. For example, the rule $\textit{stats:Concatnate}(v1, v2) \wedge \textit{ds:TimeSeries}(v1) \wedge \textit{ds:hasDimension}(v1, x) \wedge \textit{ds:TimeSeries}(v2) \wedge \textit{ds:hasDimension}(v2, y) \Rightarrow \textit{ds:equal}(x, y)$ indicates the input *ds:TimeSeties* of the statistical task *stats:Concatnate* should have the same dimension in the concatenation dimension.

2.4 Executable Knowledge Graph Construction

The executable KGs construction follows the task ontologies O^{visu}, O^{stats}, and O^{ml} as schemata [15, 16, 18], and rely on KG templates [27, 28], which are parameterised ontologies with pre-defined structures and a set of variables of entities and properties. We adopt a solution similar to Reasonable Ontology Templates framework [10]. By providing values (arguments) for each parameter, users create an instance of a template, which is then serialised as OWL axioms. When the KG templates are designed legal and consistent, they possibly ensure legality and consistency of the generated knowledge graphs as well as the relative simplicity of the KG construction process [25]. In our system, we created a template library that relies on the classes, properties and constraints defined within O^{visu}, O^{stats}, and O^{ml}.

Based on the GUI, users are able to construct executable KGs in three ways [14]: *creation*, *modification* and *integration*. *Creation* refers to represent specific data analytics pipelines by instantiating templates from the scratch, choosing the appropriate template which determines the domain and structure of the KG, and filling variables of entities and properties guided by the GUI step by step (Fig. 2 b). *Modification* refers to changing variables of entities or properties of an existing KG to represent a different but similar data pipeline, e.g., modifying the input data node of the visual pipeline KG (Fig. 2 b) makes the visual pipeline applicable for other input datasets. *Integration* refers to merging existing executable KGs to form bigger KGs. This is possible because each executable KG represents a data *pipeline*, which is a *task* according to Section 2.1, and thus an existing executable KG can be treated as a single *task* for forming bigger KGs that represent integrated pipelines.

2.5 Executable Knowledge Graph Execution

The KG execution includes three steps: *verification*, *translation*, and *execution*. In *verification*, all constraints in a constructed KG are verified against the properties of the data entities, methods, and tasks that comprise the KG. KG *translation* refers to the process of translating KG into executable scripts, which is language-dependent [26]. In our system we use Python as the language for discussion. Each executable KG representing a data *pipeline*, which consists of a series of *Tasks* of sequential or parallel structures connected with

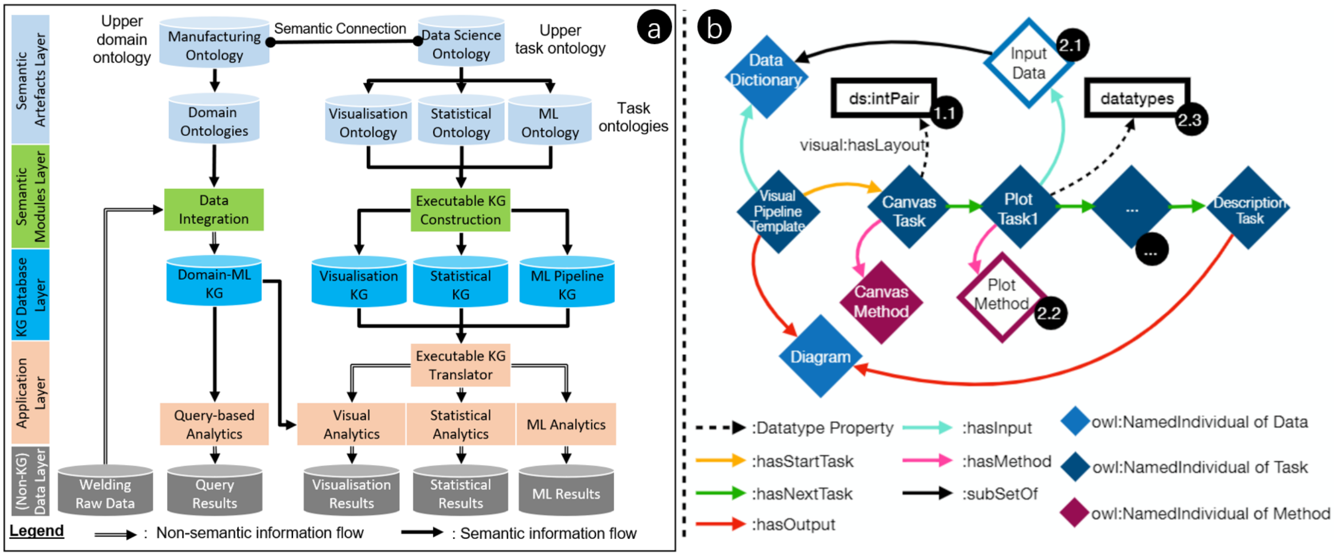

Figure 2: (a) Architectural overview of ExeKG system; (b) KG template instantiation Example: For specific visual analytics, instantiation of visual pipeline template includes the configuration of the first *visual:CanvasCreationTask* by identifying (1.1) the canvas layout with a pair of integers; then *visual:PlotTask*, the identification of (2.1) input data from the *ds:DataDictionary*; (2.2) plot method such as *visual:Lineplot*; (2.3) other properties corresponding to selected visual method. The configurations of the following *visual:PlotTask* (if required) are similar.

hasNextTask (Fig. 1 a). Each *Task* is connected with *hasMethod* to an individual of *Method*, which is a Python function script, whose inputs/outputs and parameters are clearly defined. Thus, the translation of an executable KG invokes the Python function scripts with the inputs/outputs and parameters given by *DataEntity* and datatype properties of KGs (which are arguments of the Python function script), according to the order defined by *hasNextTask*. In the special case of merging two parallel structures, the translator will search the preceding dependency with *hasNextTask*, until no preceding *Task* is found. In *Execution*, the translated scripts are executed. In this step, the dataset in the system back-end will be updated, the results can be seen in the data-panel, and the result-panel will present the numerical or visual results.

3 DEMONSTRATION SCENARIOS

During the demonstration we will present our ExeKG system for industrial data analysis tasks with the help of three scenarios *Statistic Analytics*, *ML Analytics* and *Visual Analytics*. The data for the scenarios are anonymised snippets collected from one machine in the production of resistance spot welding, a world-widely applied process at many plants of Bosch and Bosch's renowned customers. In these scenarios, the attendees will construct and execute the KGs to solve the specific data analysis tasks, and experience how easy one can do data anlaytics without coding and a minimal common sense knowledge in data analytics. We have prepared 6 tasks for the 3 scenarios for the attendees to choose from.

Statistic Analytics. Our system *ExeKG* incorporates two basic categories of statistical analysis tasks: the calculation of statistical properties such as average and standard deviation over a certain data set, and the selection of target data based on methods such as filter and sliding window. Combing these basic tasks in a pipeline in *ExeKG* helps the users to gain insights from the data, for example, averaging over a sliding window of time-series data will provide a trend of these data with less influence of noise.

ML Analytics. For ML analysis scenarios, *ExeKG* supports users with basic ML knowledge to select existing KGs for ML pipelines, to modify them by changing variables of the KG nodes, and advanced users to create KGs from the scratch. The attendees will first understand an ML pipeline that uses linear regression (LR) for predicting welding quality (spot diameter). They then need to specify the input data for the ML pipeline and run the pipelines. After that, they can modify the ML pipeline to use multilayer perceptron (MLP), for predicting another welding quality indicator (Q-Value) by changing the method node and the output feature node. Then they will see the results of the ML analysis. We will also present the process creating a ML pipeline by selecting a KG template and filling the variables for nodes and properties.

Visual Analytics. *ExeKG* covers the generation of plots like line plot, scatter, pie chart, bar char etc., that are intended to represent the properties such as the distribution, change, statistical information etc of numerical data. The attendees will visualise the raw data and also as the ML analysis results to intuitively understand the analysis results. The attendees will first identify the layout of the canvas. Then in each grid of the canvas, then select the method node for the line-plot and scatter plot, and their properties like line width, colour, etc., and at last add some descriptions on the canvas such as legend, x and y labels, titles, etc.

Acknowledgements. The work was partially supported by the H2020 projects Dome 4.0 (Grant Agreement No. 953163), OntoCommons (Grant Agreement No. 958371), and DataCloud (Grant Agreement No. 101016835) and the SIRIUS Centre, Norwegian Research Council project number 237898.

REFERENCES

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Peter Patel-Schneider, Daniele Nardi, et al. 2003. *The description logic handbook: Theory, implementation and applications.* Cambridge university press.

[2] Onur Celik, Dongzhuoran Zhou, Ge Li, Philipp Becker, and Gerhard Neumann. 2022. Specializing Versatile Skill Libraries using Local Mixture of Experts. In *CRL*. PMLR, 1423–1433.

[3] DVS. 2016. *Widerstandspunktschweißen von Stählen bis 3 mm Einzeldicke - Konstruktion und Berechnung.* Standard. Deutscher Verband für Schweißen und verwandte Verfahren e. V., Düsseldorf, DE.

[4] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F Patel-Schneider, and Sebastian Rudolph. 2009. OWL 2 web ontology language primer. *W3C recommendation* 27, 1 (2009), 123.

[5] ISO. 2004. *Resistance Welding - Procedures for Determining the Weldability Lobe for Resistance Spot, Projection and Seam Welding.* Standard. International Organization for Standardization, Geneva, CH.

[6] Henning Kagermann. 2015. Change Through Digitization – Value Creation in the Age of Industry 4.0. In *Management of Permanent Change.*

[7] Batta Mahesh. 2020. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), 381–386.

[8] Christoph Naab and Zhuoxun Zheng. 2022. Application of the unscented Kalman filter in position estimation a case study on a robot for precise positioning. *Robotics and Autonomous Systems* 147 (2022), 103904.

[9] Adam Perer and Ben Shneiderman. 2008. Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In *Proceedings of the SIGCHI conference on Human Factors in computing systems.* 265–274.

[10] Martin G Skjæveland, Daniel P Lupp, Leif Harald Karlsen, and Henrik Forssell. 2018. Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In *International Semantic Web Conference.* Springer, 477–494.

[11] Yulia Svetashova, Baifan Zhou, Tim Pychynski, Stefan Schmidt, York Sure-Vetter, Ralf Mikut, and Evgeny Kharlamov. 2020. Ontology-Enhanced Machine Learning: a Bosch Use Case of Welding Quality Monitoring. In *ISWC.*

[12] Muhammad Yahya, Baifan Zhou, Zhuoxun Zheng, Dongzhuoran Zhou, John G Breslin, Muhammad Intizar Ali, and Evgeny Kharlamov. 2022. Towards generalized welding ontology in line with ISO and knowledge graph construction. *ESWC, Springer* (2022).

[13] Zhuoxun Zheng et al. 2022. Executable Knowledge Graph for Machine Learning: A Bosch Case for Welding Monitoring. In *ISWC.*

[14] Zhuoxun Zheng et al. 2022. Executable Knowledge Graph for Transparent Machine Learning in Welding Monitoring at Bosch. In *CIKM.*

[15] Zhuoxun Zheng et al. 2022. Towards A Statistic Ontology for Data Analysis in Smart Manufacturing. In *ISWC.*

[16] Zhuoxun Zheng et al. 2022. Towards a Visualisation Ontology for Data Analysis in Industrial Applications. In *SemIIM@ESWC.*

[17] Zhuoxun Zheng, Baifan Zhou, Dongzhuoran Zhou, Gong Cheng, Ernesto Jiménez-Ruiz, Ahmet Soylu, and Evgeny Kharlamov. 2022. Query-based industrial analytics over knowledge graphs with ontology reshaping. *ESWC, Springer* (2022).

[18] Baifan Zhou et al. 2022. Knowledge Graph-Based Semantic System for Visual Analytics in Automatic Manufacturing. In *ISWC.*

[19] B. Zhou, T. Pychynski, D. Steimer, A. Shakirov, and R. Reischl, M. Mikut. 2019. Predictive quality monitoring in resistance spot welding using machine learning methods with domain knowledge supported feature engineering. *International Journal for Advanced Manufacturing Technology* (2019). submitted.

[20] Baifan Zhou, Yulia Svetashova, Seongsu Byeon, Tim Pychynski, Ralf Mikut, and Evgeny Kharlamov. 2020. Predicting Quality of Automated Welding with Machine Learning and Semantics: a Bosch Case Study. In *CIKM.*

[21] Baifan Zhou, Yulia Svetashova, Andre Gusmao, Ahmet Soylu, Gong Cheng, Ralf Mikut, Arild Waaler, and Evgeny Kharlamov. 2021. SemML: Facilitating Development of ML Models for Condition Monitoring With Semantics. *Journal of Web Semantics* 71 (2021), 100664.

[22] Baifan Zhou, Zhuoxun Zheng, Dongzhuoran Zhou, E Jimenez-Ruiz, G Cheng, T Tran, Daria Stepanova, Mohamed H Gad-Elrab, Nikolay Nikolov, Ahmet Soylu, et al. 2022. The data value quest: A holistic semantic approach at Bosch. *ESWC, Springer* (2022).

[23] Baifan Zhou, Dongzhuoran Zhou, Jieying Chen, Yulia Svetashova, Gong Cheng, and Evgeny Kharlamov. 2021. Scaling usability of ML analytics with knowledge graphs: Exemplified with a Bosch welding case. In *IJCKG.* 54–63.

[24] Dongzhuoran Zhou et al. 2022. Ontology Reshaping for Knowledge Graph Construction: Applied on Bosch Welding Case. In *ISWC.* Springer.

[25] Dongzhuoran Zhou et al. 2022. ScheRe: Schema Reshaping for Enhancing Knowledge Graph Construction. In *CIKM.*

[26] Dongzhuoran Zhou et al. 2022. Towards Executable Knowledge Graph Translation. In *ISWC.*

[27] Dongzhuoran Zhou, Baifan Zhou, et al. 2021. Towards ontology reshaping for KG generation with user-in-the-loop: Applied to Bosch Welding. In *IJCKG.*

[28] Dongzhuoran Zhou, Baifan Zhou, Zhuoxun Zheng, Egor V Kostylev, Gong Cheng, Ernesto Jimenez-Ruiz, Ahmet Soylu, and Evgeny Kharlamov. 2022. Enhancing knowledge graph generation with ontology reshaping–Bosch case. *ESWC, Springer* (2022).