Quality in Color: Using Knowledge Graphs for Enhanced Quality Control in an Automotive Paintshop

Bram Steenwinckel[1(✉)], Colin Soete[1], Pieter Moens[1], Joris Mussche[2], Sofie Van Hoecke[1], and Femke Ongenae[1]

[1] IDLab, Ghent University – imec, 9000 Gent, Belgium
bram.steenwinckel@ugent.be
[2] Volvo Cars Group, 9000 Gent, Belgium
joris.mussche@volovcars.com

Abstract. Sensors and their derived data have reshaped manufacturing by enabling real-time monitoring, improved quality control and enhanced safety, particularly in automotive processes. Despite these benefits, challenges arise within the realm of data discovery when they have to align all this data, originating from different systems and built by different manufacturers using different technologies. This paper shows our solution to resolve these challenges within the Volvo Cars paintshop. By organizing sensor metadata, link them with the originating devices and the place where the data will be stored in a knowledge graph, our approach facilitates seamless data integration for multiple paintshop applications. We show how the resulting knowledge graph helps with dashboard techniques, machine learning, and semantic reasoning to provide insights for the paintshop operators. The obtained findings clearly show the potential of knowledge graphs in production lines, paving the way for future advancements in automotive manufacturing.

Keywords: Automotive paintshop · Knowledge Graph · Semantic reasoning · Machine learning · Dashboard visualization · Industry 4.0

1 Introduction

Industry 4.0 has fundamentally transformed manufacturing operations due to the creation of autonomous systems for production processes [13]. In automotive production lines, many of these automated processes are nowadays applied to manufacture cars. Applying paint is one such process, playing an essential role in creating the final aesthetics and durability of cars [1]. This process typically involves applying various paint layers, including primer, topcoat, and clear coat, to the vehicle bodies. Each paint layer can be seen as a different process that each exists out of multiple procedures that follow one another.

Within the Volvo Cars paintshop, these different paint procedures are physically segregated across various stations. Stations are equipped with a range

G. Demartini et al. (Eds.): ISWC 2024, LNCS 15233, pp. 236–252, 2025.
https://doi.org/10.1007/978-3-031-77847-6_13

of devices, so-called Internet of Things (IoT) sensors and actuators, e.g. paint robots and ventilation units, which are systems from various manufacturers. This results in a diverse and complex ecosystem of technologies that must seamlessly interact. Figure 1 schematically shows car bodies on a conveyor belt going through four different stations. At the end of all these paint processes, each car body undergoes a final inspection at the Quality Control (QC) station. This station is vital to ensure that the vehicles meet the stringent quality standards of the Volvo Cars group before leaving the production line. The QC inspection is a manual process. An operator conducts visual checks to identify any defects or errors that may have occurred during the painting process, such as uneven paint layers, scratches, or paint defects.

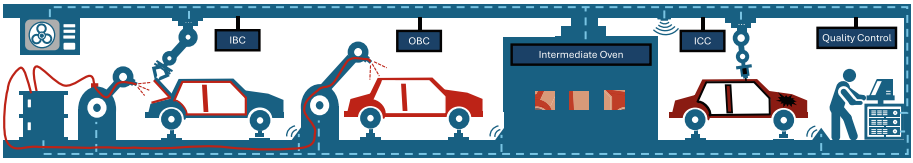

Fig. 1. Schematic overview of a subprocess within an automated paintshop. The Interior Base Coat (IBC) station is responsible for painting the interior parts of the car. Next, the Outer Base Coat (OBC) station applies the exterior base coat. An intermediate oven is used to dry the car body before moving towards the Interior Clear Coat (ICC) station. At the end, a manual inspection of the car is performed to control the quality, e.g. detect scratches.

The robots within these stations are precisely programmed to perform a predetermined sequence of actions. They are programmed using industry standards, and controlled by Programmable Logic Controllers (PLCs). PLC code can be seen as a set of instructions that are sequentially executed. It is the PLC that steers every actuator within the station and starts the different robot programs based on available parameters and sensor inputs. The PLC plays a crucial role in coordinating all these operations. As car body trackers are placed at the input and output regions of each station, different robot programs with adjusted parameters can be loaded by the PLC based on the car type and the desired car color.

While the tracker data directly steers the dynamic behaviour of the paint application, the Volvo Cars group extensively collects all the available station data. This includes car types, color compounds and more, beyond basic production information, including the robot operations, robot faults, PLC operations, PLC faults, ventilation metrics and temperature values at each station. The records defining the paint faults observed during the QC process are also stored alongside each car body. The data is stored in separate databases, laying the foundation for further analytics. It opens the possibility for system operators to optimize the production and enhance the overall efficiency or quality of the product [2].

An important shortcoming of the current QC process is the lack of additional information that the operator can use to identify the causes of any defects and assess whether subsequent cars might have similar issues. While it is not directly indicated in Fig. 1, many stations are duplicated in separate lines to parallelize and speed up the production process. Cars with similar quality issues might have passed the same station and identifying these issues provides information towards necessary maintenance operations for that particular station. On the other side, if the quality operator has station-related information alongside each car body they have to inspect, quality issues could be more easily detected or even predicted upfront.

Providing interoperability over different stations and different devices with different technologies is challenging. This challenge is indicative of broader issues faced within Industry 4.0, where data standardization is crucial for achieving efficient and interconnected manufacturing systems. Data stored within the databases is most of the time defined at an overly granular level. This data exists with meaningless identifiers, making it challenging for operators to further use and digitize the data [11]. To be able to build the necessary insights directly from the data, one often neglected aspect is the comprehensive management of metadata and knowledge about these sensors, defining not only their location and relation to specific processes, but also what they observe and what the semantic significance is of this observation. Such a semantic meaning to the data requires human expertise and it is the station operators that understand how the different devices work together to define the paint process. A common understanding of each subprocess is required to further contextualize and semantically annotate the data.

In this paper, we demonstrate the substantial benefits that can arise from the implementation of a Knowledge Graph (KG) within an Industry 4.0 car paintshop process to obtain further insights. We resolve the following 4 challenges:

- We enable the semantic annotation of the process data using a KG which is created and maintainable with minimal human effort.
- We show how we can use this KG to enable interoperability across production lines and stations and how the data sources can be linked to the available process knowledge.
- We showcase how semantic enrichment enables more insightful visualization of the data for quality control.
- Due to enhancing the identification and investigation of quality issues in the paintshop, we can perform semantically enhanced reasoning and machine learning on the data.

The remainder of this paper is defined as follows. Section 2 defines the process of how the Volvo Cars KG was created. Section 3 showcases three applications benefiting from the created KG for enhanced quality control. Section 4 and Sect. 5 define the different lessons learned from this setup and conclude this work respectively.

2 From Knowledge to Knowledge Graph

To address the uniformity challenges mentioned earlier, a solution based on KGs was devised, integrating data, metadata information, and insights from the paintshop processes. This KG was developed in three sequential phases: 1) establishing unique identifiers, 2) incorporating expert knowledge using ontologies, and 3) connecting the knowledge with existing data resources. Important requirements for this design process were (1) to enable minimal human intervention, automating annotation as much as possible, and (2) when manual input is needed, making this input as user-friendly as possible.

2.1 Establishing Unique Identifiers

Device identifiers in a factory are usually vendor-specific. This can cause confusion, errors, and inefficiencies in the production line. Diverse machinery and stations need to work together, but different identifiers complicate inventory, asset management, tracking, risk management, and quality control. They also hinder system interoperability and data integration. Thus, standardizing identification systems across vendors is crucial for streamlining production and maximizing efficiency.

Device identifiers are typically vendor-specific. Using different methods to identify machines within a production line can cause confusion, errors, and inefficiencies. This can complicate inventory and asset management, tracking, risk management, and quality control. They hinder further interoperability between systems and obstruct data integration efforts. Therefore, standardizing the identification of machines and systems across vendors is essential for streamlining production processes and maximizing efficiency.

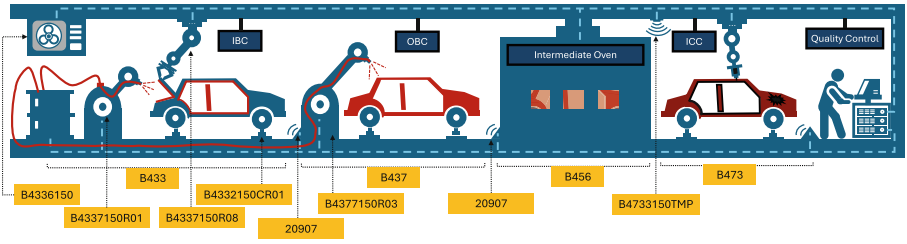

Fig. 2. Overview of the identification of the different components within the paintshop production line. Formatting the identifiers is based on a hierarchical schema where the location of a certain component can be deduced from the identifier.

When the identification of systems and machines adheres to a well-defined standard, it not only ensures the uniqueness of the numbers, but can also provide substantial information through the formatting of the IDs. As shown in Fig. 2, Volvo let their IDs follow a specific standard to define the different equipment within the whole factory. Different shop floors are first identified with a single character (here 'B', indicating the specific building where the paintshop resides).

Next, three numbers are used to define the zone and specific line followed by three digits to specify the station. Finally, more specific devices within each station are defined, such as 'R' with two digits to identify a specific robot. These types of hierarchical identifiers help operators to understand and directly identify each component within the factory. As opposed to the car body tracking devices, which are identified in Fig. 2 by simple integer numbers, they do provide, even across Volvo Cars factories, some of the necessary information about tracking and tracing components. Hierarchical unique identifiers are necessary to start linking machines and systems in process together. However, the semantic description of each component's functioning and how they interact with each other is limited.

2.2 Production Line Ontology Design

While unique identifiers offer precision in tracking and tracing, it is the consolidation of shared knowledge of the domain experts, i.e. operators, on how the paintshop operates and how the data should be interpreted that presents a significant stride towards a holistic understanding and operational optimization [7]. This knowledge can be contextualized in the form of domain-specific ontologies. However, the journey of ontology design in specific domains is full of challenges, time-consuming and labor intensive [21]. They do, however, offer advantages, such as facilitating collaboration and consensus-building among experts within a particular context [15]. In the context of Industry 4.0, domain experts may also face a learning curve, as ontology design paradigms might not be familiar to them. Moreover, the true benefits of a domain ontology manifest only upon its integration into various applications, forming a threshold for domain experts to value its importance during construction. Hence, expediting the construction process, perhaps through automation techniques like extracting common-sense knowledge from documents or using user-friendly tools, is often desired.

Various concepts needed within industry 4.0,e.g. processes, sensors, and faults, are already defined in well-established upper ontologies such as SNN [3] and FOLIO [18]. Both were reused and linked together in the Systematized Procedure for Automating the Retrieval of Knowledge within Smart industries (SPARKS) upper ontology [19]. Instead of focusing on how observations are created following a procedure with one or more devices, SPARKS defines the influence observable properties have on the procedure. These influences are more commonly known as process or procedure parameters. SPARKS additionally allows to order procedures after each other, making it more clear how systems interact with multiple procedures and how the different procedures eventually lead to a process. An overview of this SPARKS ontology is provided in Fig. 3 and this ontology has been open-sourced for future industry 4.0 use cases[1].

To extend this upper ontology to a domain-specific one, e.g. for a paintshop, domain experts will need to define information regarding their specific production processes, systems and faults and link the unique identifiers to the defined concepts. To automate this tedious manual process and to prevent the domain

[1] https://predict-idlab.github.io/SPARKS/.

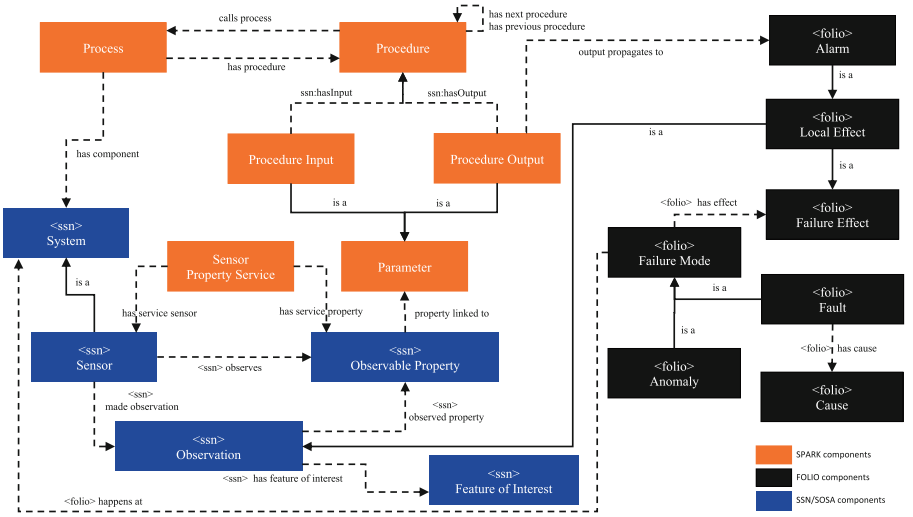

Fig. 3. Overview of the SPARKS upper ontology, which combines different upper ontologies and aligns them to processes and procedures

expert from requiring experience with semantic web technologies, a graphical user interface (GUI) has been developed to provide this information, as shown in Fig. 4. This GUI uses automated scripts to transform the graphical input provided by operators to semantic representations, based on the SPARKS ontology.

To ensure the expert does not have to start from scratch, the GUI enables to load expert documents and automatically map them on the SPARKS ontology. In this research, we made it possible to directly load PLC code. Transformation scripts were created based on the PLC Statement List (STL) standard to connect PLC function blocks and their relationships to the occurring paintshop production processes. These transformation mechanisms are standardized in a well-defined specification, allowing the GUI to be easily extended to also enable the seamless import of other structured files, beyond PLC code. A video defining the different GUI interactions is made available online[2]. Figure 4 shows the semantification process of the *IBC* process and the corresponding station is shown as an example. The GUI interface has a canvas where different widgets can be specified and linked. Each widget corresponds to a key concept within the SPARKS ontology, but can be further characterized by the settings pane (see Fig. 4 on the left). It is also possible to provide additional concepts and provide more information regarding occurring faults at each component, their cause and possible mitigation action. The GUI, the transformation script and the SPARKS methodology can be found in the corresponding repository[3].

[2] https://youtu.be/BzLiNpzNHU8.
[3] https://github.com/predict-idlab/SPARKS.

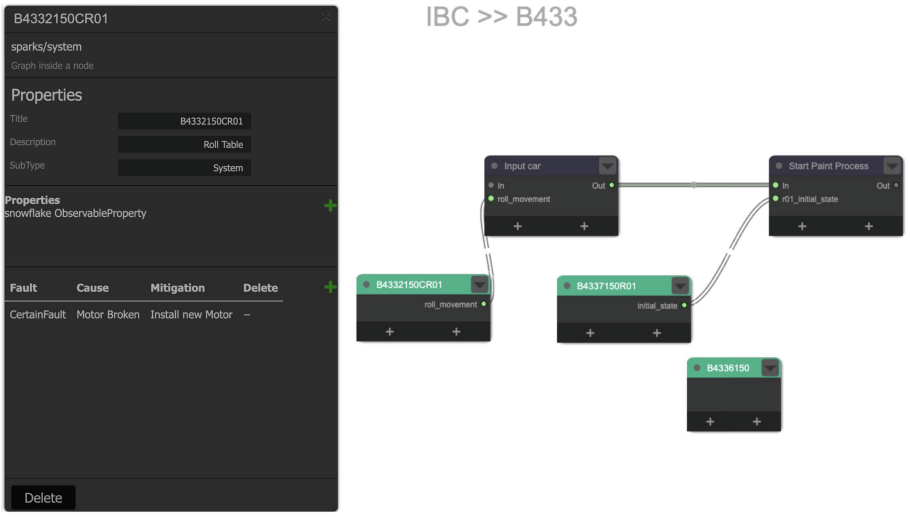

Fig. 4. Example of the information can be specified in the designed GUI. It shows two procedures within the IBC process, which was originally already uniquely defined. Three systems are provided, of which two influence two subsequent procedures following each other. A detail panel on the left is provided for the roll table, responsible for the transport of the car bodies within the station.

2.3 Linking Data to Knowledge

The interaction between data and semantic components in manufacturing environments unlocks the full potential of unique identifiers, especially to monitor and evaluate the different processes [13]. The challenge arises when attempting to align the data generated by different manufacturers' software with the predefined semantic structure. Often, data arrives in various formats and may be stored across multiple databases, complicating the efforts to integrate it seamlessly into a unified semantic representation. In an ideal scenario, a comprehensive overhaul and alignment of data silos would be the optimal solution. However, practical constraints, such as costs and time constraints often limit the feasibility of such endeavors. The solution used here is to semantically describe the data source alongside the device and the properties that the device can observe.

The SPARKS ontology provides a concept to define the link between sensors and observable properties in the form of a `Sensor Property Service`. Subclasses of this concept can work twofold: 1) They provide a service concept that links the sensor and the property that it observes, and 2) They can add additional data properties to provide all the necessary information to acquire the corresponding data, e.g. add data properties to define the database schema, login information and specific location within the database to reach the correct sensor data.

The GUI editor is able to provide this link for each component by specifying the `Sensor Property Services` for each device. This can be seen in the left panel in Fig. 4, where different sensor property services can be added using the

'+' sign. Different types of property services were defined in our KG for this project, based on the different data silo technologies. In the example of Fig. 4, a Snowflake Sensor Property Service was defined for the roll table and an arbitrary Observable property (e.g. Tracking). Snowflake is a SQL-typed database [4], and we defined this Sensor Property Service concept with additional data properties to access this Snowflake database. These data properties can also be specified within the GUI editor directly when the specific Sensor Property Service is instantiated. Under the hood, the GUI uses automated mapping scripts to transform all these operator-provided graphical inputs to semantic representations [19].

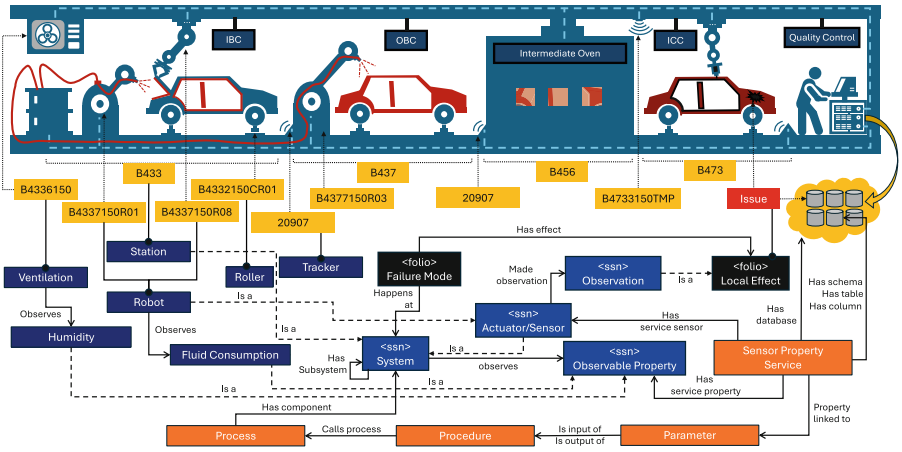

Fig. 5. Example of information about a paintshop production line being semantically annotated and interlinked to data available in data silos using the SPARKS ontology

The data linked to occurring faults of devices or quality issues is also available within different databases and can be linked in a similar way as described above. The overall process of identification, semantic annotation and linking of data resulted in a system where both the knowledge of the system procedures and faults occurring at those systems are uniformly defined and become easily queryable for further analyses. Figure 5 shows this linked knowledge schematically.

In addition to the equipment in each station, car bodies themselves can also be uniquely defined. They are tracked using RFID scanners to follow up on the production configuration. It is based on these body numbers that configuration parameters are defined, such as the paint color, which are in turn directly used as input for the PLC program to start the correct paint processes. While what happens in the production line might affect the quality of the product, all the data and sensor/actuator information available in the data silos about each station does not have a direct link to the car body that was in a specific station at

the particular moment these observations were made. The KG can be used to enable this interlinking. The car body, or in more general terms the product, is a so-called feature of interest. The station trackers in our example are responsible for scanning the car body identifiers and providing timestamps when a car body enters the assigned station. These scanning devices are defined in our KG, making it possible to request the timestamps for a car body identifier when it is in a particular station. The Sensor Property Services can be used to query the data of all devices within this station, taking into account the ranges of the obtained timestamps. This results in data that can be linked directly to an instantiated car body concept.

3 Enabling Quality Control Through KG-Enhanced Decision Support

The outcome of Sect. 2 is a KG that uniquely connects and identifies instances within the paintshop production line. These links between data and identifiers present numerous opportunities to assist paintshop operators with their daily tasks. As indicated previously, the operator was not able to directly access the necessary insights or station errors that occurred during manual inspection of the car's paint quality. This also hindered solutions to detect quality issues automatically. To show how the KG can assist in resolving this, we elaborate on three predictive maintenance use cases that were realized.

3.1 Knowledge-Driven AI on Semantic Data

Beyond concepts and links between these concepts in a KG, rules can further encapsulate the domain-specific insights derived from the expertise of operators and quality control engineers. They can play a role in decision-making processes by providing actionable guidelines based on observations and experiences. An illustrative example of such a rule could identify the correlation between temperature fluctuations in a specific station and an increase in paint drip quality issues observed on the car bodies that traversed the station when the temperature fluctuations occurred. Such an expert-driven rule is formally provided in Eq. 1.

$$\begin{aligned} &\text{car}(?x) \wedge \text{hasLog}(?x, ?y) \wedge \text{metricTempValue}(?y, ?v) \wedge \\ &\text{greaterThan}(?v, 18) \rightarrow \text{sparks:Fault}(?x, ?v) \end{aligned} \tag{1}$$

The availability of our KG, the fact that we can combine all data and link it to a car body, and the ability to perform rule-based reasoning upon all this data, make it possible for these cases to come up with a knowledge-driven AI system to automatically inform quality operators of possible faults occurred at the car body they are inspecting. We created a solution based on rules expressed in the Semantic Web Rule Language (SWRL) [8] in combination with the OWLReady2 [10] Python package to gain the capability to effortlessly load knowledge

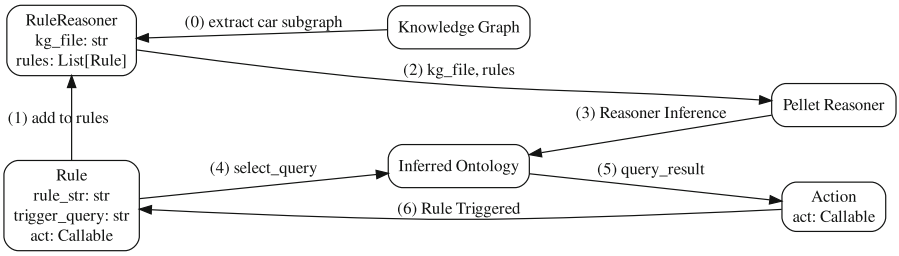

Fig. 6. Flow of the Rule-based Python reasoner created to act upon triggered rules.

graphs in OWLReady, interact with ontology classes and individuals, and execute reasoning tasks, all through a Python syntax that prioritizes convenience.

Our solution consists of the following steps. First, the overall KG is queried to generate a new, smaller KG that only contains all data from a particular car body. This smaller KG connects and transforms the data provisioned by the Sensor Property Service to SSN observations. Next, the newly generated KG and rules provided by the expert are given to our created Python rule-based reasoning framework. The flow of this framework is provided in Fig. 6. A rule class within this framework consists of 3 arguments: the SWRL rule, a SPARQL selection query and a Python callable function. The SWRL rule (or multiple SWRL rules when more than one rule is defined) is added together with the car-specific knowledge graph to the RuleReasoner as seen in step one of Fig. 6. Step two in our framework will trigger the Pellet reasoner to infer the materialized knowledge graph based on the available concepts and the provided rules. After this step, the SPARQL query can be used to retrieve the inferred knowledge that we are interested in, e.g. all the derived faults. If knowledge was inferred that adheres to the query, the Python function linked to this query is called and executed.

A simplified Python example is provided in Listing 1.1 to show the rule-based Python reasoning framework functionality. Car data is queried from the general knowledge graph using the query function (line 4), and the retrieved data is stored in the graph variable. Subsequently, a rule named rule1 is defined (lines 6–7) with conditions specifying Eq. 1. Additionally, a corresponding SPARQL query and a callable function are encapsulated to handle the results of the rule (lines 8–9). The fault function is defined (lines 11–12) to process the results of the rule by printing a message indicating a fault in the car. Finally, a RuleReasoner object named reasoner is initialized (line 14), passing the car data (graph) and the defined rule (rule1). The reason method is called on the reasoner object (line 16), initiating the reasoning process where the rule is evaluated against the provided data, and if new knowledge is inferred, the fault function is called.

Our rule-based framework shows the additional benefits of having a KG to steer this whole process. Without having to define additional rule-specific context, our framework and rules can be applied to multiple stations. New stations providing data linked to the `metricTempValue` can possibly trigger the callback

function. This makes dynamically extending the KG and using this rule-based framework even more interesting.

Listing 1.1. Example code to detect possible faults on a car body using our Python rule-based reasoner.

```
from reasoner import *
from cars.query_car import query

# Query car data for analysis
graph = query(3076989)

# Define a rule for detecting faults in cars
rule1 = Rule(
    'car(?x) ^ hasLog(?x, ?y) ^ metricTempValue(?y, ?v) ^
       greaterThan(?v, 18) -> sparks:Fault(?x, ?v)',
    'SELECT ?x ?y WHERE {?x sparks:Fault ?y}',
    lambda res: fault(res))

# Define an action function for the fault rule
def fault(res):
    print(f'FAULT: car {str(res[0]).split(".")[-1]} is faulty
         due to a too high VP413P04BF1 value of {res[1]}')

# Initialize the rule reasoner with the graph and rules
reasoner = RuleReasoner(graph, [rule1])

# Trigger the reasoning process
reasoner.reason()
```

3.2 Semantically Enriched MLOps Pipeline

Besides knowledge-driven AI, data-driven solutions, e.g. machine learning (ML), emerge frequently in the industry 4.0 domain, as they can use the available sensor data to search for unknown patterns or unwanted system behaviour automatically [2]. MLOps, the integration of DevOps principles within ML workflows, revolutionizes this digital transformation [6]. It streamlines the deployment, monitoring, and optimization of ML models, ensuring seamless integration into production environments. A typical ML pipeline involves data ingestion, preprocessing, model training, evaluation, and deployment [5]. MLOps separates these stages, facilitating the reuse of components and iterative improvements, thereby making it possible for industries to leverage ML for the dynamic purposes of Industry 4.0.

Despite the advancements in ML and data science techniques, it remains evident that considerable efforts are still required to ensure the acquisition of correct and high-quality data from within databases. This initial step is fundamental, as the effectiveness of subsequent feature extraction and ML methods heavily relies on the quality and relevance of the available data. Within our paintshop,

this first step was extremely challenging due to the missing link between sensor metadata and the provided data. The generated KG resolved this mismatch and data from specific observable properties can now be easily queried using multiple SPARQL queries. The SPARQL Query in e.g. Listing 1.2 can be used to acquire all table, signal and column information from a specific Sensor Property Service concept for all systems within a given station. With this information, a generic component within the ML pipeline can be provided to acquire all the necessary data over different stations for that particular property-sensor combination. Decoupling the ML pipeline from the underlying data structure has the additional benefit of introducing fewer faults when this data structure changes. When e.g. moving to a new database, only the Sensor Property Services and connectors will have to be rewritten, but the links towards the stations remain the same.

Listing 1.2. Example SPARQL query to acquire the Sensor Property Service data properties for a particular station

```
SELECT ?schema ?table ?signal WHERE
{
    ?service a sparks:SnowflakeSensorPropertyService .
    ?service  sparks:hasSensor ?sensor .
    <B433> ssn:hasSubSystem* ?sensor .
    ?service sparks:hasSchema ?schema .
    ?service sparks:hasTable ?table .
    ?service sparks:hasColumn ?signal .
}
```

While data acquisition is one such process within the ML pipeline that clearly benefits from a KG, the proceeding feature engineering and ML training steps can also be linked to the Sensor Property Services and process information within the KG [16]. Such a semantic pipeline could enable ML engineers to easily create, adapt or even directly apply different ML models to different stations in different factories by only specifying and reasoning upon which station identifiers to use within the KG.

3.3 Dynamic Dashboard

In the modern era of data analytics, dashboards have emerged as necessary tools for gaining insights, monitoring systems, and facilitating informed actions [9]. Dashboards provide so-called widgets, visual interfaces that consolidate complex data into interpretable visualizations, enabling operators to derive actionable insights. However, traditional dashboard creation methods often require significant manual effort and expertise in how the data is outputted, posing challenges in adapting to evolving data landscapes and dynamic operational needs. Therefore, in previous research, we created the Dynamic Dashboard, a dashboard solution designed to automate the process of dashboard creation by leveraging Semantic Web technologies [22].

The constructed KG provides hierarchical information, grouping sensors and subsystems as indicated in Fig. 5. A link to the data is also provided using the Sensor Property Service. The Dynamic Dashboard uses these concepts and instances of both the ontology and KG and further extends this semantic knowledge in two parts.

First, for each observable property within the KG, additional metric information is added. This metric information includes details about how the data from that observable property is provided, e.g. the data type and unit of measurement defined by the Ontology of units of Measure (OM) [14]. An example of such a metric is provided in Listing 1.3.

Listing 1.3. Example Code of metric information

```
metrics:quantity a dashb:Metric; a om:Quantity;
        dashb:datatype xsd:double .
```

The different available widgets to visualize the data are also semantically described. Each widget has a component list that defines how many sources it accepts, but also which metrics it can visualize. An example of such a widget description is provided in Listing 1.4.

Listing 1.4. Example Code of widget description

```
<time-series-line-chart> a dashb:LineChart;
    rdfs:label "Line chart"@en;
    dashb:component [
        dashb:accepts [ dashb:datatype xsd:double ];
        dashb:min 1;
        dashb:max 1 ].
```

The above combination of metric information and widget description enables the Dynamic Dashboard to reason upon which visualizations are valid and interesting for a provided observable property. More information about this reasoning procedure and the overall architecture of the Dynamic Dashboard can be found in the corresponding research papers [12,22]. Within the paintshop, the Dynamic Dashboard offered the operators two distinct functionalities for visualizing insights: user-driven and event-driven dashboards [12].

In the user-driven dashboards, operators can create visualizations tailored to their specific needs and preferences. Through an intuitive interface, operators can explore the available sensors and systems available within the KG and can select the properties for visualization. As discussed, semantic reasoning suggests suitable visualizations based on the selected observable properties and registered metrics, streamlining the process of dashboard creation.

The event-driven dashboard, on the other hand, automatically generates dashboards in response to detected anomalies or events present in the data. Analogously to observable properties and widgets, semantic descriptions can be provided for events. These events can originate from a wide range of applications. Within this paintshop setting, these events include the outcome of the

Listing 1.5. Example Code of detected anomaly event

```
<events/1234> a dashb:Event;
dcterms:description "4271656 is predicted to be faulty";
sosa:resultTime "2023-09-05T20:28:11"^^xsd:dateTime ;
ssn:wasOriginatedBy [
    fromObservation [
    sosa:resultTime "2023-09-05T20:28:11"^^xsd:dateTime];
    toObservation [
    sosa:resultTime "2023-09-05T20:28:11"^^xsd:dateTime];
    observedProperty </body_visualization/195/83>].
```

rule-based module or the outcome of a ML model as described in Sects. 3.1 and 3.2. By providing a semantic description of the detected faults or detected quality issues, the Dynamic Dashboard is able to recommend the necessary visualization for further inspection by the operators. An example semantic description for a detected anomalous event is provided in Listing 1.5. In addition to the observable properties, the ranges of when the fault occurred are also defined in these events.

Based on this description, the correct widget is automatically created to visualize the anomaly. As shown in Fig. 7, the dashboard for this example event concerns a ML detected quality issue. It displays the general information about the event (right) and provides a visualization to display the faults on a car diagram (left). The red dots shown in the visualization are indications of where the detected faults occurred on the car.

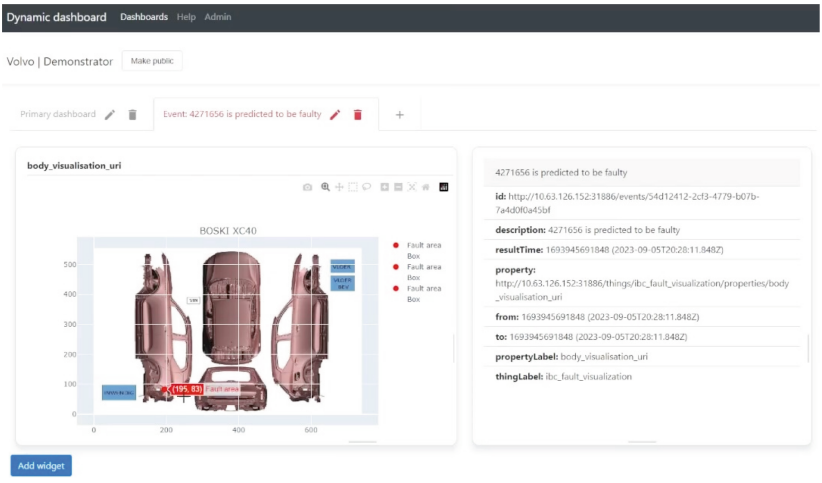

Fig. 7. Expert-driven Dynamic Dashboard example. This example shows a potential predicted quality issue detected by a ML algorithm.

The fact that the Dynamic Dashboard provides this dual functionality enhances situational awareness, enabling both proactive monitoring through user-driven customization and reactive response to system anomalies via event-driven analysis.

4 Impact and Lessons Learned

The impact of this work on the transition of manufacturing operations from traditional knowledge management approaches to a knowledge graph-based system is multifaceted. The primary influences lie in the realm of efficiency, easy transferability to other sites or processes, and traceability. By devising a solution that integrates data, metadata information, and process insights into a KG, this work effectively tackled the uniformity challenges prevalent in manufacturing processes. A key aspect of this is the establishment of unique identifiers and the incorporation of expert knowledge using ontologies.

This work also provided ways to foster collaboration and consensus-building among domain experts. By developing upper ontologies, such as the SPARKS ontology, and tools to easily extend, adapt and maintain available expert knowledge, this work shows the benefits of designing such frameworks for domain experts to collectively understand and optimize manufacturing processes. Those responsible for paint application quality were often unaware of the production line issues. The proposed collaborative approach eventually led to more effective QC measures.

Through the standardization of identification systems across different systems, the data from diverse sources became operable for multiple applications. Within the paintshop pipeline, this provided operators with Dynamic Dashboards and rule-based fault indicator methods. Our rule-based system was developed as a proof of concept to demonstrate the advantages of using reasoning for root cause analyses. The Volvo Cars Group is conducting a cost-benefit analysis to determine the need for more advanced reasoning systems like RDFox[4] or Stardog[5] for more advanced functionalities.

Additionally, the creation of anomaly detection pipelines based on ML could be more automated. This resulted in the applicability of these models for a wide range of stations and driving the continuous improvement initiatives within evolving production environments. Once a unique identifier for new equipment is defined, it can be easily added and linked to the existing KG using the SPARKS GUI. This demonstrates that our approach can grow and adapt to new requirements without significant redevelopment. The integration of the KG within the MLOps pipeline further enhances the system's flexibility, potentially reducing long-term maintenance costs.

In essence, this work can be seen as a prelude to the use of manufacturing intelligence, where the seamless integration of knowledge and data results in innovation, enhances competitiveness and drives the industry to new heights of

[4] https://www.oxfordsemantic.tech/.
[5] https://www.stardog.com/.

productivity. This is also indicated by how the feedback through the Dynamic Dashboard operations could further influence e.g. the ML behaviour. The MLOps use case demonstrated how the Volvo Cars ML group could use the KG to access relevant station data. This enabled them to develop ML pipelines that are easily transferable to other, similar stations, as the data locations for sensors and actuators in these stations were also queryable through the KG. Future work could use the KG and the available data to work towards KG embeddings and enhance the ML models themselves based on the provided domain-specific knowledge for tasks like anomaly detection [20]. This offers a future where manufacturing processes are not just explainable for the operator, but also adaptive to new unknown situations [17].

5 Conclusion

In conclusion, this work exemplifies the transformative potential of KGs and semantic technologies in automating manufacturing operations within an automotive paintshop. By standardizing identification systems, facilitating collaboration among domain experts, and enabling seamless data integration, the showed applications not only address current challenges but also lay the groundwork for future advancements in Industry 4.0. The lessons and impact of this research serve as valuable insights for the continuous adaptation and improvement of various manufacturing processes.

Supplemental Material Statement: The full Volvo Cars KG, station-specific PLC code and QC-related issues cannot be made available as they incorporate company-specific data. However, small examples (including an example KG) are made available, for the sole purpose of replicating the applications in this paper on a GitHub at https://github.com/predict-idlab/SPARKS. The SPARKS ontology and the source code for both the GUI editor and Python rule-based framework are available on this same GitHub repository.

Acknowledgement. This work was executed as part of the O&O HBC.2020.3072 project, funded by Volvo & VLAIO.

References

1. Akafuah, N.K., Poozesh, S., Salaimeh, A., Patrick, G., Lawler, K., Saito, K.: Evolution of the automotive body coating process-a review. Coatings **6**(2), 24 (2016)
2. Bousdekis, A., Lepenioti, K., Apostolou, D., Mentzas, G.: A review of data-driven decision-making methods for industry 4.0 maintenance applications. Electronics **10**(7), 828 (2021)
3. Compton, M., et al.: The SSN ontology of the W3C semantic sensor network incubator group. J. Web Semant. **17**, 25–32 (2012)
4. Dageville, B., et al.: The snowflake elastic data warehouse. In: Proceedings of the 2016 International Conference on Management of Data, pp. 215–226 (2016)

5. Elshawi, R., Maher, M., Sakr, S.: Automated machine learning: state-of-the-art and open challenges. arXiv preprint arXiv:1906.02287 (2019)
6. Francois Regis, D.: MLOps paradigm-a game changer in machine learning engineering? (2023)
7. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum Comput Stud. **43**(5–6), 907–928 (1995)
8. Horrocks, I., et al.: SWRL: a semantic web rule language combining OWL and RuleML. W3C Member submission **21**(79), 1–31 (2004)
9. Kumar, N., Prajapati, S.: Challenges for interface designers in designing sensor dashboards in the context of industry 4.0. Int. J. Ind. Manuf. Eng. **13**(8), 539–542 (2019)
10. Lamy, J.B.: Owlready: ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. Artif. Intell. Med. **80**, 11–28 (2017)
11. Margherita, E.G., Bua, I.: The role of human resource practices for the development of operator 4.0 in industry 4.0 organisations: a literature review and a research agenda. Businesses **1**(1), 18–33 (2021)
12. Moens, P., Vanden Hautte, S., De Paepe, D., Steenwinckel, B.: Event-driven dashboarding and feedback for improved event detection in predictive maintenance applications. Appl. Sci. **11**(21), 10371 (2021)
13. Raja Santhi, A., Muthuswamy, P.: Industry 5.0 or industry 4.0? Introduction to industry 4.0 and a peek into the prospective industry 5.0 technologies. IJIDeM **17**(2), 947–979 (2023)
14. Rijgersberg, H., Van Assem, M., Top, J.: Ontology of units of measure and related concepts. Semant. Web **4**(1), 3–13 (2013)
15. Simperl, E., Luczak-Rösch, M.: Collaborative ontology engineering: a survey. Knowl. Eng. Rev. **29**(1), 101–131 (2014)
16. Soete, C., Steenwinckel, B., Mussche, J., Ongenae, F., Van Hoecke, S.: Enabling ontologies for semi-automated feature extraction and traceability of quality issues in production line monitoring. Information Fusion **under review** (2024)
17. Steenwinckel, B., De Paepe, D., Hautte, S.V., Heyvaert, et al.: Flags: a methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning. Future Gener. Comput. Syst. **116**, 30–48 (2021)
18. Steenwinckel, B., Heyvaert, P., De Paepe, D., et al.: Towards adaptive anomaly detection and root cause analysis by automated extraction of knowledge from risk analyses. In: SSN@ ISWC, pp. 17–31 (2018)
19. Steenwinckel, B., Moens, P., Mussche, J., Van Hoecke, S., Ongenae, F.: Systematized procedure for automating the retrieval of knowledge within smart industries. IEEE Trans. Ind. Inform. **under review** (2024)
20. Steenwinckel, B., Vandewiele, G., Weyns, M., Agozzino, T., Turck, F.D., Ongenae, F.: Ink: knowledge graph embeddings for node classification. Data Min. Knowl. Disc. **36**(2), 620–667 (2022). https://doi.org/10.1007/s10618-021-00806-z
21. Tudorache, T.: Ontology engineering: current state, challenges, and future directions. Semant. Web **11**(1), 125–138 (2020)
22. Vanden Hautte, S., Moens, P., Van Herwegen, J., et al.: A dynamic dashboarding application for fleet monitoring using semantic web of things technologies. Sensors **20**(4), 1152 (2020)