Knowledge and Information Systems (2021) 63:1271–1304
https://doi.org/10.1007/s10115-021-01558-4

REGULAR PAPER

Comparing ontologies and databases: a critical review of lifecycle engineering models in manufacturing

Borja Ramis Ferrer[1] · Wael M. Mohammed[1] · Mussawar Ahmad[2] · Sergii Iarovyi[3] · Jiayi Zhang[2] · Robert Harrison[2] · Jose Luis Martinez Lastra[1]

Received: 20 December 2017 / Revised: 5 March 2021 / Accepted: 10 March 2021 / Published online: 3 April 2021

Abstract

The literature on the modeling and management of data generated through the lifecycle of a manufacturing system is split into two main paradigms: product lifecycle management (PLM) and product, process, resource (PPR) modeling. These paradigms are complementary, and the latter could be considered a more neutral version of the former. There are two main technologies associated with these paradigms: ontologies and databases. Database technology is widespread in industry and is well established. Ontologies remain largely a plaything of the academic community which, despite numerous projects and publications, have seen limited implementations in industrial manufacturing applications. The main objective of this paper is to provide a comparison between ontologies and databases, offering both qualitative and quantitative analyses in the context of PLM and PPR. To achieve this, the article presents (1) a literature review within the context of manufacturing systems that use databases and ontologies, identifying their respective strengths and weaknesses, and (2) an implementation in a real industrial scenario that demonstrates how different modeling approaches can be used

✉ Wael M. Mohammed
wael.mohammed@tuni.fi

Borja Ramis Ferrer
ramis@ieee.org

Mussawar Ahmad
mussawar.ahmad@warwick.ac.uk

Sergii Iarovyi
sergii.iarovyi@kalmarglobal.com

Jiayi Zhang
jiayi.zhang@warwick.ac.uk

Robert Harrison
robert.harrison@warwick.ac.uk

Jose Luis Martinez Lastra
jose.martinezlastra@tuni.fi

1 FAST-Lab., Faculty of Engineering and Natural Sciences, Tampere University, Tampere, Finland

2 University of Warwick, Coventry, UK

3 Kalmar Global, Tampere, Finland

for the same purpose. This experiment is used to enable discussion and comparative analysis of both modeling strategies.

Keywords Ontology · Database · Comparison · Data modeling · Product lifecycle management

1 Introduction

The advent of computer science and information communication technologies (ICT) in diverse fields such as manufacturing, healthcare and smart cities has improved the manner in which information is created and exchanged between multiple stakeholders [1]. Furthermore, paradigms such as service-oriented architecture (SOA) [2] and cloud computing (CC) [3] can be implemented in order to permit the remote access, storage and manipulation of resources. These can be physical or cyber resources, which are, in turn, mapped to different physical elements such as industrial equipment or measuring devices. This is achieved through the modeling and management of data. More precisely, any domain that can be described within a data model may use models as engineering artifacts for different purposes, e.g., simulation, inference, monitoring and control.

Nowadays, in the case of manufacturing systems, databases seem to be the most common technology used by organizations in order to represent, store and share information [4] with databases currently being used for product lifecycle management (PLM) and product, process and resource (PPR) data modeling [5,6]. Databases are widespread in industry and are more established than ontologies as a means for representing system knowledge [7]. Nevertheless, the design of ontologies for this domain has, in the last decade, gained momentum—particularly in academia. This is evidenced by the increase in published research on this matter in various research portals [8].

Selection of data modeling approach should depend on the needs and desire of end users. However, such decisions are frequently made by software engineers who tend to select a solution according to their knowledge and experience, tending toward those that they are comfortable with. The authors of this paper consider there to be a lack of comparative studies that offer sufficient means for deciding between the employment of ontologies and databases, particularly for those unfamiliar with the former. Some of the few existing studies can be found in [9,10]. However, such works need to be supported by research that provides representative examples to enable a comparison of the capabilities of both technologies within the context of the requirements placed upon them. It is important to state that other knowledge representation (KR)-based solutions, such as production rules or frames, are used in industry; this work considers that, based on contemporary trends, the prominent technologies to be compared are ontologies and databases.

This paper aims to present a qualitative and quantitative comparison between ontologies and databases, demonstrating some of the capabilities of each technology when facing the same issue in the context of manufacturing systems. The main contribution of this article is to provide a study that permits an assessment of the strengths and weaknesses of both technologies for a specific domain—manufacturing systems—which requires robust technologies for storing, accessing and updating data dynamically, at runtime. To achieve this, this research presents a concrete and industrial use case whereby a manufacturing system is described within different semantic models to be accessed and updated. Further, this paper addresses

real-world benefits of both data modeling technologies and discusses a set of research questions that need to be answered.

The rest of the paper is structured as follows: Sect. 2 presents a literature and industrial practices review within the scope of this research. More precisely, the review contains an introduction to data modeling and management as well as the definition of ontologies and databases, including a classification and introduction of their main principles. Finally, there is also a description of state-of-the-art transformation solutions. Section 3 describes the methodology that has been followed in order to achieve the reported results. Section 4 presents the test environment that has been designed for the required implementation of this research. Section 5 presents the use case that has been applied to obtain both qualitative and quantitative results for the comparison between ontologies and databases. Section 6 presents and discusses the results with Sect. 7 concluding the article and identifying future directions.

2 Literature and industrial practices review

2.1 Data modeling and management

A vast amount of data are currently generated throughout the product realization process—from design, through to process planning, and then on to manufacturing system design and engineering. Prior to the emergence of the present-day engineering software, models were limited to those instantiated within the physical world, such as mock-ups and prototypes for testing products and processes, as well as some digital versions such as spreadsheet-based calculations to predict costs and cycle times [11]. However, the amount of complexity in a typical manufacturing system has increased due to the emergence of more sophisticated technologies (both within products and manufacturing systems), requiring the expertise of multiple domains for realization. In addition, there is an ongoing paradigm shift toward mass customization and product personalization [12,13]. These factors are the cause of an exponential rise in the amount of data that are now generated by the manufacturing industry, with sources ranging from customer requirements to production systems and the supply chain [14].

Although much can be gleaned from these data, it is necessary for it to be managed and analyzed, to allow maximum value to be extracted from it. Some data are generated from physical systems during operation; however, a considerable amount of data are also generated through the modeling of products and systems. This is done in order to support design activities and to understand the interactions of components—carrying out simulations to predict performance, and visualization to communicate requirements and the desired outcome.

To help manage these data, the key paradigm, recognized as state of the art by the industry, is PLM. PLM manages the storage and exchange of information supporting design and engineering, and integrates it with business processes [5]. PLM is envisioned to allow stakeholders to make data- or information-driven decisions throughout the lifecycle of the product. The implementation of PLM is intended to create a so-called unbroken "digital thread" that prevents the loss of information and ensures that the data are an up-to-date, truthful representation of reality. PLM software acts as a hub or platform that brings together a suite of engineering tools and business processes. Relational database management systems (RDBMS) are a core part of all major, existing, PLM platforms and are renowned for their scalability and stability [15]. However, with the exponential rise in the volume and type of data that now need to be managed by such systems, the efficacy of RDBMS is called into question, particularly within

the context of adaptability, expressiveness, interoperability and extensibility [10,16,17]. To continue to support the industry, it is vital for some form of PLM to continue to exist; however, with the increasing complexity and demands on such systems, it is necessary to consider whether new data management and modeling techniques are required.

2.2 Ontologies

2.2.1 What is an ontology?

The word "ontology" has different meanings depending on the context. Firstly, there is the philosophical discipline which is an uncountable noun written as "ontology," which deals with nature and the structure of "reality" [18]. Aristotle dealt with this subject and defined ontology as the "science of being." Unlike the scientific ontology, this branch of metaphysics focuses on the nature and structure or reality independent of how this information would be used.

Contrastingly, the use of ontologies in this research stems from the field of computer science, where it refers to a type of information object. An ontology is a form of KR and is defined by Gruber [19] as "an explicit specification of a conceptualization" while Borst [20] extends this definition to "formal specification of shared conceptualization." Ontologies are a form of KR for a given domain that uses formal semantics and can be used to arrange and define a concept hierarchy, taxonomy and topology.

Ontologies can be accessed for querying and/or modification purposes and they can be implemented using several semantic languages [21,22]. Resource description framework (RDF)-based languages remain dominant, using XML as the syntax option for writing expressions. RDF-based models (e.g., RDF graphs) are sets of triples composed of a subject, a predicate and an object. The Ontology Web Language (OWL) [23] is a description language that extends RDF with cardinality constraints, enumeration and axioms, enabling the creation of richer and more accurate models. OWL comprises three sublanguages: OWL Lite, OWL DL and OWL Full, in order of increasing expressivity, respectively. OWL 2 extends OWL with additional features, including extended datatype support and annotation capabilities. However, OWL remains the prevalent ontology language, with a large number of supporting editors.

The information from OWL models can be queried using an RDF-based query language such as SPARQL [24]. In addition, SPARQL Update [25] can be used for retrieving and updating ontological models. Rule-based languages such as the Semantic Web Rule Language (SWRL) [26] can be employed within ontologies. These rules are defined on top of such ontological models, as presented in [27]. Through the use of rules and RDF triples, semantic reasoning engines can infer implicit knowledge and validate the consistency of a model [28,29].

2.2.2 Types of ontologies

There are different types of ontologies, as reported in [30], with two main criteria that are used to categorize them: the level of formalization, and the level of specificity. In the former, there exist "lightweight" and "heavyweight" ontologies, while in the latter, there exist "foundational," "core," and "domain" ontologies.

– Levels of ontological formalization: Lightweight ontologies are based on simple taxonomies with simple parent child relationships between concepts [31]. Examples of this

type of ontology include WordNet [32], and a number of international standards within the context of product data management, such as STEP [33]. This type of ontology has limited concept constraints such that their semantics are insufficient to support interoperability, i.e., to integrate different domain models [34]. To address this, particularly for the STEP format, the ONTOSTEP ontology was developed, which addressed the lack of logical formalism of EXPRESS so that reasoning and semantic operability could be realized [35]. Thus, heavyweight ontologies describe concepts, relationships and logic constraints for automatic prediction and logical inference.

– Levels of ontological specification: Foundational ontologies aim to cover the semantics of "everything" and therefore cover the semantic base for any given domain. Examples of foundational ontologies include DOLCE [36], and the Basic Formal Ontology (BFO) [37]. The concepts in foundational ontologies are generic and as a result are often too broad to be used in a practical engineering context. Core ontologies are limited in the literature and sit at a level of specificity between foundational and domain ontologies. The objective of core ontologies is to cover a set of semantics that are shared across multiple domains [38]. As a result, they lend themselves to reuse and are of particular importance within the context of interoperability. Domain ontologies have the greatest level of specificity and, due to their focus and distinct semantics, interoperability between domain ontologies is challenging. Within the context of supporting manufacturing system lifecycles, it is therefore incumbent on the domain ontology development team to identify domain touchpoints and to ensure that links and mappings exist between the relevant concepts.

2.2.3 Ontology development methodologies

As a result of over two decades of development and learning, a number of methodologies have evolved to support the development of ontological models from the modeling process through to implementation and use. In 1994, the U.S. Air Force defined an ontology method to structure semantic information modeling called IDEF5 [39]. An ontology acquisition process was developed based on five basic steps [40].

1. Organizing and Scoping of Project: The structure and content of the project is described in this part and the main objectives of ontology development are clearly specified.
2. Data Collection: The raw data are defined and classified to enable the development of the ontology and the data collection methods are summarized from different domains.
3. Data Analysis: This part is used to analyze the existing data material to establish an initial ontology for knowledge engineers and the domain developer.
4. Initial Ontology Development: By developing prototype ontologies, ontology classes, properties, attributes and relationships are refined and given detailed specifications.
5. Ontology Refinement and Validation: This phase integrates the known information with the ontology. Through a refinement procedure, ontologies are summarized in specification form for evaluation by domain experts.

Based on the IDEF5 methodology, [41] a documentation stage is added to standardize the ontologies and to support the foundation for future ontology development. METHONTOLOGY introduced the iterative development of an ontology, with a focus on maintenance [42]. Reusing knowledge from existing ontologies forms part of a seven-step guide for ontology creation by Noy and McGuinness [43]. Other than the knowledge reuse aspect, the method remains similar to what is proposed by [40]. An important conclusion derived by Noy and McGuinness is that there is no single correct ontology for a given domain, despite following a

common methodology. Determining whether the "right" ontology has been created can only be done by using it in the application for which it was developed [44].

2.3 Databases

2.3.1 What is a database?

The concept of the database appeared following development of direct-access memory, i.e., immediately after stored computer data became feasible. The term database appeared in the early 1960s, and since that moment, multiple database implementations have emerged [45]. The term database in computer science is understood as a structured collection of data [46]. This collection includes several kinds of objects, such as schemas, tables and queries, that permit the representation of data to enable it to be interpreted and reused by computer systems and by humans.

As the deployment, across multiple domains, of IT-based solutions for managing and storing data has grown over the last few decades, specific types of databases have been selected and implemented, according to the requirements of the field of application. The authors suggest the following non-exhaustive expectations for a database in the modern production environment:

1. Databases are expected to be medium sized, i.e., smaller than global social network databases occupying the data centers around the world, but large enough to accommodate all relevant production information.
2. Database models are expected to be of average complexity. Due to standardization of the production processes and components, the allowed abstraction level can be relatively high.
3. Database models are expected to be stable. This is because changes made to adequately designed database models are only made in relation to significant changes in the production.
4. Databases are expected to ensure data consistency, i.e., corrupt data should be spotted early, while the failure is still recoverable.
5. Databases are expected to provide high data throughput, accessibility and robustness. Basically, a database should not be a bottleneck of the production process.

The authors of this research work claim that, in the manufacturing domain, databases should, at least:

1. be medium-sized models in order to easily manage, access and update them;
2. allow the description of static schemas, not affected by a highly dynamic number of requests;
3. ensure data robustness and consistency;
4. permit the processing of multiple requests from/to different data providers and consumers in parallel.

2.3.2 Types of databases

There are many ways to classify different kinds of databases as they can be differentiated according to their structure, contents or application area. These characteristics affect interrelated concepts such as data storage, organization and access. Data storage typically refers to the number of levels of abstraction between the data and its representation in computer

memory. The levels of abstraction add functionality to the database but increase memory usage and, generally, the access time. The data can be stored in its binary form directly in the database (DB) program memory, on a disk as a file in binary format, in DB-specific format, or even in plain or marked-up text format. Furthermore, the stored data can be present in a single place in memory or can be replicated across clusters.

Different organization approaches influence the storage and access options as well as the performance and applicability of certain techniques for representing data. In turn, represented data affects several aspects such as consistency, synchronization, redundancy and robustness. Among the access options, the most common ones include direct access as well as several querying languages, such as SQL [47], NoSQL [48] and some customized and/or proprietary ones, such as Hyper Text SQL (HTSQL).[1] One of the main objectives of this manuscript is to provide a qualitative and quantitative comparison between ontologies and databases. This research considers relational SQL, NoSQL and graph databases [45], which can be mapped to RDF-based models.

From the first databases that emerged in the 1960s, such as the network-based CODASYL [49] or the hierarchy-based IMS [50], to the recent models, many different types of database have been designed and implemented for diverse applications. It is important to note that database engines are capable of handling specific types of databases. For instance, Oracle[2] allows, as a primary data model, a relational database management system (DBMS). However, some database engines permit additional secondary data models or even multi-model functionality, i.e., processing different database types as primary data models. For example, while Oracle permits document store and key-value store as a secondary data model, the OrientDB[3] allows the implementation of document store, graph DBMS and key-value store.

2.3.3 Database development methodologies

There are many methodologies that database designers and developers may follow in order to create coherent and consistent data models [51–53]. As there are many different types of databases that can be developed, it is not feasible to find a methodology that supports the creation of any kind of data model, covering all the required steps.

Nevertheless, there are many common steps regardless of the database type being created. Fundamentally, the development of a database model starts with a decision on terminology for certain concepts such as entities, relationships, attributes and constraints. This convention of terms is similar to the fourth step presented in the basic step list for designing ontologies in the previous section. Following this step, is it necessary to check for any redundancy in the model for simplification purposes [54].

2.4 Previous work on comparison of different approaches for data modeling

A number of works have been published that present some level of comparison between ontologies and databases. In some cases, only a passing comment is made, while others delve deeper. In order to demonstrate the value and contribution of this work, the authors present what has already been discussed in this area alongside the remaining questions.

Ontologies differ from a database approach as their focus is on the preservation of meaning to facilitate interoperability, whereas the main purpose of a database schema is to store and

[1] http://htsql.org/.

[2] https://www.oracle.com/database/index.html.

[3] http://orientdb.com/orientdb/.

query large data sets [9]. One of the most comprehensive reviews on the topic was presented in [10], which aimed to clarify the differences and similarities between ontologies and databases. Similar points were also raised in [55]. A summary of the conclusions made concerning the differences are as follows:

- Design approach: Databases are created from scratch for a specific purpose, whereas ontologies may be created by reusing existing ontologies. Although ontologies can also be created from scratch to be used for a specific purpose, their inherent dependence on semantics facilitates reuse for unforeseen applications—unlike databases.
- Manner of KR: Databases relies on closed world assumption (CWA), which means that the assumption is that the model represents complete information. This has the consequence that what is not known to be true, must be false. Ontologies, however, use an open world assumption (OWA), whereby if a query does not return a result, the interpretation would be that the information is unknown.
- Syntax: Databases utilize entity-relationship diagrams, which represent the logic of the database, whereas ontologies are expressed in languages with which you can describe logics. By extending this notion, semantic features are the underlying foundation of ontologies, but are unimportant for databases.

There are also some similarities: the expressiveness of the respective tools resembles each other to some degree (classes $\ncong$ tables, properties $\ncong$ attributes, and axioms $\ncong$ constraints). Thus, we conclude that the key differences are derived from how the respective tools are used: databases are for storing large data sets, while ontologies are focused on integrating semantic data or exchanging information between heterogeneous systems. An example of exchanging data between heterogeneous software is present within PLM [56,57]. As such, it has been proposed that ontologies can make databases entirely redundant within this context. This is because the conceptual model is stored together with the instances. Additionally, when transforming the conceptual model associated with a database to physical and logical models, there is an associated semantic loss [58].

In [59], a framework for representing functional knowledge within ontologies to retain design intent is presented. The authors explicitly decide not to use databases specifically because they have been known to hinder the reuse of documents due to the lack of semantic constraints for functional knowledge. Within the context of the work, the authors define a semantic constraint as a restriction that allows the description of a model that complies with the conceptualization committed by the author.

Research that presented a comparison of databases and ontologies through implementation within a medical data management system concluded that SPARQL, querying triple stores, retrieved instance data via the Virtuoso Universal Server faster than SQL, querying a relational database [16]. Comments were also raised concerning the usability of SPARQL versus SQL, whereby the former adhered to a clearer standard which was not always the case for SQL-capable systems. A further advantage of the ontological approach was its flexible schema that could be extended without comprehensive system redesign. On the other hand, the OWA offers no constraint validation, requiring implementation of this functionality in the application layer.

Finally, in the scope of this research, Bizer and Schultz presented the Berlin SPARQL Benchmark (BSBM) [60], which is a study of the querying performance on a variety of different RDF and SQL-based stores via SPARQL and SPARQL-to-SQL queries. The authors performed a set of queries where the data size and the number of clients (representing the number of the end users) changed in order to add realism to the conducted tests. As a result, the SPARQL-to-SQL rewriters slightly outperformed the RDF as the data set increased. It is

important to highlight that the authors did not discuss the low-level specification of different technologies used. An analysis of their experiment infrastructure would provide a benchmark for comparing the results presented in this research work.

2.4.1 Transformation tools

Moving on from the comparison above, it is clear that there are some areas where the respective data modeling methods are complementary. A number of transformation tools have been developed to allow the benefits of the respective approaches to be exploited. Such tools enable the sharing and reuse of knowledge structures to support domain experts in addressing the integration and analysis of existing data sets. Relational database-based conversion tools serve as a method to facilitate ontology development by reducing development lead times, examples include DB2OWL, RDB2Onto and OWL2DB [61].

- DB2OWL: DB2OWL is a conversion tool that can automatically generate ontologies from relational databases via mapping database tables and description logic using OWL DL language [62]. Based on their algorithm, database concepts are translated to a related ontology component. For example, tables will be classes in ontology description; columns and rows are represented by properties and instances; the relation in database schema is relationship between domain ontologies. The advantage of this tool is that it can automatically generate records for logging ontology mapping processes including (1) each corresponding description for the ontology component, (2) the conceptual relationship between ontology and database, and (3) the mapping history of instances and attributes [63]. However, this tool depends only on a particular case or database table, and current databases only support Oracle and MySQL due to limited metadata. In addition, data mapping cannot occur across different databases to generate one ontology.
- RDB2Onto: The automatic generation of ontologies usually focuses on mapping a relational database with ontology concepts, such as, such as DB2OWL, D2R and R2O [64]. RDB2Onto is an SQL-query-based RDF/OWL translation tool that can transfer existing data to ontology templates using only SQL queries. In order to analyze XML schema in an ontology template, data are merged to an ontology data format. This tool is developed in the Java environment using the Sesame and Jena library, which support SPARQL to connect an ontology with a MySQL database, but it can also be used for any other relational database. The advantage of this solution is its simplicity and ease of operation via a visual user interface [65]. RDB2Onto also provides the ability to customize instances and create decision-making rules through an ontology library. Unlike DB2OWL, this approach cannot directly generate database instances to ontology. Furthermore, the main components of this tool are the OWL Builder and the OWL Writer, which cannot preserve the ontology structural constraints. Thus, this tool does not support reasoning tasks for extending ontology with predication of rules.
- Others: There are solutions that permit transformation from OWL to relational databases [66]. In fact, the work reported in [66] describes the main principles for mapping OWL elements to relational database schemas within a specific tool, based on the OWL2DB transformation algorithm. Furthermore, a qualitative comparison between similar transformation solutions is provided. The research works compared are, predominantly, the ones reported in [67–72]. The aforementioned articles demonstrate that the mapping between ontology and database models is feasible and must be taken into account in environments that employ both types of modeling approach. However, OWL2DB focuses on a one-to-one class relationship and breadth-first search method. As a result, the per-

Table 1 Research questions about ontologies and databases

#	Research question	Requirements
1	What is the effect on performance (as a set of characteristics) as the volume of data within a given model is incrementally increased?	Investigate the differences on processing when using each technology. This should be tool-agnostic and focus on the performance differences between fundamentals of ontologies and databases
2	How could database models and ontologies interact with each other in a future scenario?	Investigate the employment of ontological models in conjunction with databases to support complex and demanding needs of ICT-based platforms
3	What is the perceived difference in effort for implementing and maintenance of a database versus an ontology for common applications?	Evaluate the required effort to create script, model and resources of each solution. In addition, research about the effort of modification and maintenance of models at different phases (e.g., design and operation)

formance of this tool is limited by the transformation algorithm. Depending on different cases, this tool may not create all the relationships between tables or classes. Further, knowledge can only be transformed in terms of OWL Lite syntax and a part of OWL DL syntax.

3 Open questions and a methodology for comparing digital data modeling technologies

The review of literature and industrial practices in the scope of PLM and PPR data modeling led to the discovery of a set of unanswered questions. As shown in Table 1, this section presents a set of research questions (RQs) and required research actions to address them.

These RQs are the starting point of this research. The following steps have been performed to compare different data modeling technologies:

1. Model and environment design: This consists of the selection of the main methods and tools for designing similar data models to describe and control the same system using different data modeling technologies (ontologies and databases). The decisions and the final environment that was selected for this research are described in the following section; the decision has three core aspects: data collection, applications and tools, and test environment.

2. Implementation: This concerns the implementation of the test environment and the data models. The completion of this step provides an experimental setup that permits the assessment of different features of the data modeling technologies. In relation to RQ3, this step illustrates some of the aspects of model implementation that require management.
3. Test and compare: This is the final step of the methodology. It involves testing both technologies to obtain results for analysis and discussion. The testing of each model demonstrates directly the effects on performance, which is the concern of RQ1. In addition, the discussion of the experimental tests leads to the identification of potential synergies between databases and ontologies.

4 A test environment for the lifecycle engineering databases and ontology comparison

As a representative case in the scope of PLM and PPR data modeling, this research work implements a means of data collection for retrieving information from a discrete assembly line. Principally, the objective is to collect random events that are triggered by industrial controllers located in such a line. These events describe the status of each machine in the line. More precisely, an orchestrator engine has been designed in order to produce one variation among the available products. While the line is producing the products, the orchestrator records all triggered events. Over a period of 12 hours, the line produced 100,000 random events. The authors of this research believe that the randomness and unconstrained nature of the data are representative of a real manufacturing environment. In fact, the randomness is generated by the nature of the manufacturing systems, where events can be triggered once a change occurs. For example, a change can include pallet position, machine status, operations feedback or safety alarms. The event collection routines are not linked to the current process or status of the production system.

Besides the data collection, the research methodology requires the selection of the applications that are to be used for the comparison. The goal is to exploit the same tools and frameworks for each kind of modeling technology to ensure a valid comparison. The selection study took more effort than was expected, since the available support for both technologies is significantly different. This contrast originated from several factors; for example, the life span of the technology was the major factor, since the two technologies have many years of difference in terms of maturity. Besides that, the level of usage and maturity also plays an important role in terms of technical and programming support. The authors use the Java development environment[4] with similar libraries, due to the availability of frameworks such as SQL[5] provided by Java, which suits each of the technologies being compared. In addition, the Java Microbenchmarking Harness (JMH)[6] framework is used for measuring the operation-to-time ratio for both implemented technologies. Within this setup, each technology to be compared had similar programs for making the required benchmarking test. The MySQL[7] data store has been used to implement the database store, and Jena ARQ[8] has been used to store the data for the ontology model.

[4] https://www.java.com/en/.

[5] https://docs.oracle.com/javase/7/docs/api/java/sql/package-summary.html.

[6] http://tutorials.jenkov.com/java-performance/jmh.html.

[7] https://dev.mysql.com/.

[8] https://jena.apache.org/documentation/query/.

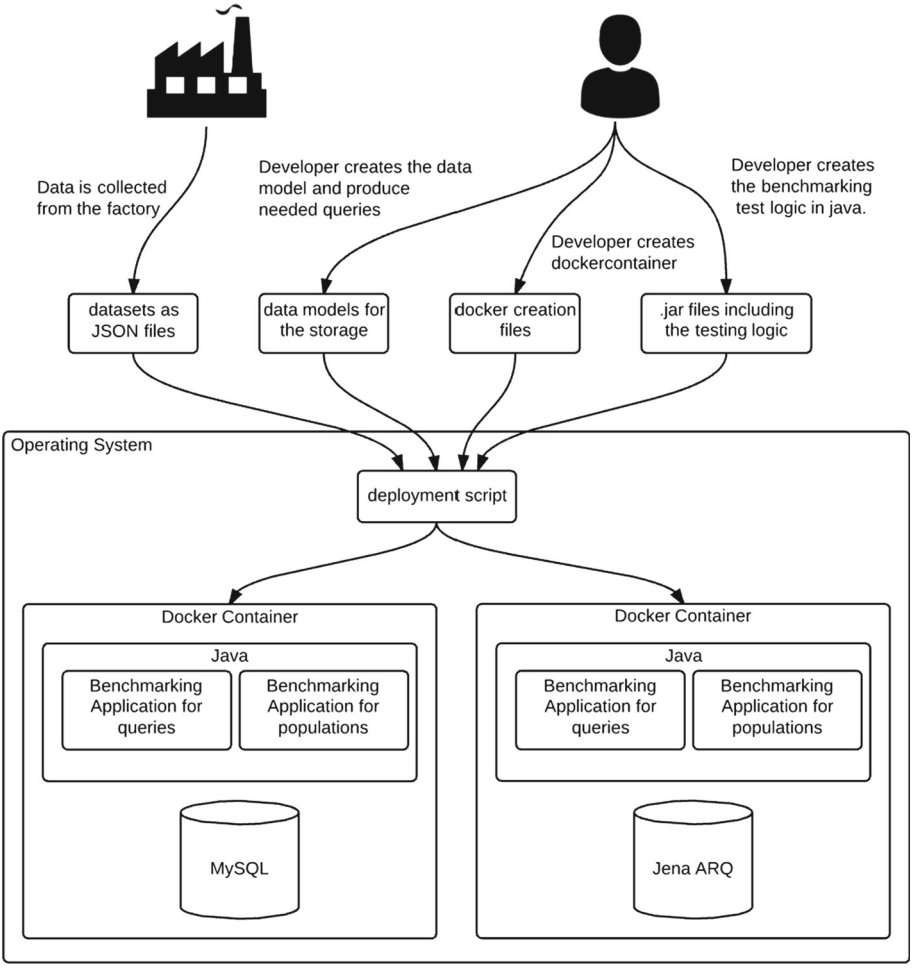

Fig. 1 Deployment environment for database and knowledge-based technology validation

Finally, the decision on the deployment environment has ensured similar impact due to computational resources such as central processing unit (CPU) capabilities, random access memory (RAM) size and background services. The objective is to execute both benchmarking applications on the same machine with the same operating system (OS) conditions and background services. Furthermore, it is important to deploy both applications without them affecting or interfering with each other. As these requirements could be achieved by employing virtual machines or containers, "Docker"[9] containers have been built based on Linux Ubuntu images for the deployment of the benchmarking tests. As shown in Fig. 1, both tests are deployed on similar Docker images—the difference being the data store, since the technology is different.

[9] https://www.docker.com/.

Fig. 2 A completed mobile phone from the FASTory line

5 Lifecycle engineering models in manufacturing: a use case

5.1 The FASTory line

The FAST-Lab (Future Automation Systems and Technologies Laboratory)[10] FASTory line is a production line that demonstrates the assembly process of mobile phones by drawing different variants on sheets of paper that are located on special pallets. Up to three components are drawn: the frame, screen and keyboard. Each of the mobile parts may be drawn in any one of three different colors and three different models. This means that the line can produce 81 different mobile models and 729 different mobile variants, taking into account different color options. Figure 2 shows the FASTory line and an example of a finalized product.

The FASTory assembly line contains a workstation (WS1) for loading/unloading papers to/from the pallets using a SCARA robot. Another workstation (WS7) is used for loading/unloading pallets from the assembly line served by a human operator. Ten workstations are used for drawing purposes. These workstations are identical and are able to draw all mobile models with different colors. All workstations include a segment of the central transport system, which is based on a belt conveyor. All workstations used for drawing operations have a path for pallets to bypass the workstation if it is operating—reducing the possible delays or traffic in the overall production process. This is appreciable from Fig. 3, which shows the interface of a FASTory line web-based simulator. In addition, each conveyor is divided into multiple zones that have one presence sensor to detect the presence of the pallet, one stopper to stop the pallet and an RFID reader for pallet recognition. Each red-filled triangle of four represents a different stopper, each located at a different zone of the conveyor. Each drawing workstation has up to five conveyor zones, while WS7 and WS1 have only four zones.

The FASTory line has evolved during the implementation of several European projects, such as the eSonia[11] and eScop[12] projects. Some of the tasks performed during the eScop project made it possible to create a remotely accessible virtual replica of the production line to support the project developers. This virtual replica is referred to as the FASTory Simulator. In this research work, the FASTory Simulator[13] is used to collect the event logs for the comparison tests of ontologies and databases, with it being the system description container.

[10] https://www.tuni.fi/en/research/fast-lab#expander-trigger-field-group-members.

[11] https://artemis-ia.eu/project/18-esonia.html.

[12] https://cordis.europa.eu/project/id/332946.

[13] http://escop.rd.tut.fi:3000/fmw.

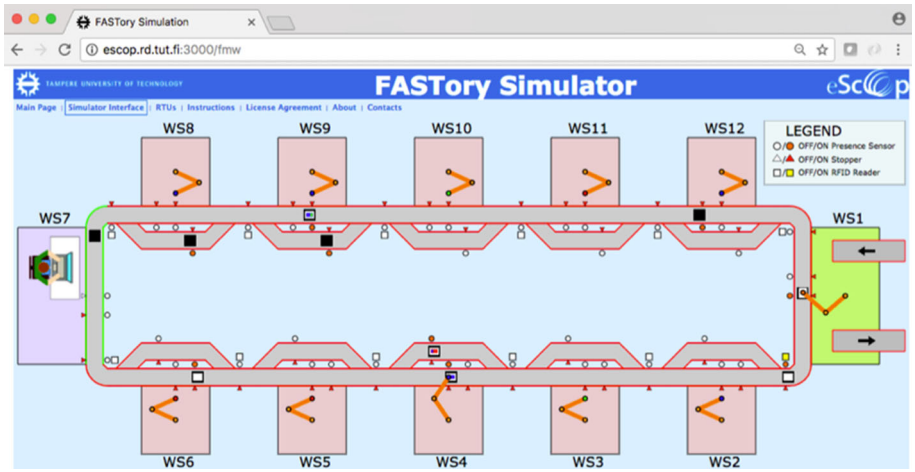

Fig. 3 FASTory layout shown in the FASTory web-based simulator

5.1.1 Collecting data in the FASTory line

To achieve the aim of collecting data from the FASTory Simulator, a web-service-enabled orchestrator has been designed. This engine consists of two main blocks: the JobExecuter and the Logger. As depicted in Fig. 4, the orchestrator is an application that runs on a normal personal computer (PC) and is connected to the FASTory network through an Ethernet socket. Figure 4 shows that a switch has been used to connect to different remote terminal units (RTUs) which, in turn, are connected to different devices (robots and conveyors) of the FASTory line. The RTUs are devices profile for web services (DPWS)-enabled devices that permit the description of service operations that can be executed in order to control the performance of the robots and conveyors. In Fig. 4, the RTUs are labeled according to the type of device and number of web services that they control, i.e., ROB1 RTU denotes an RTU connected to the robot located at WS1 and CNV12 RTU denotes an RTU connected to the conveyor located at WS12.

At the initialization phase, the orchestrator subscribes to each event in the FASTory line. This subscription allows the Logger block to be notified whenever any change occurs in the line. The Logger then creates a JSON object to store all the notifications. Hourly records are stored for each day. In the experiment performed for this research, up to 106,154 events have been collected over a total of 12 operating hours. The JSON-formatted records allow parsing for further analysis.

The JobExecuter (JE) block is capable of managing a simple production process. The production process tested in this research experiment requires the participation of all workstations. This scenario provides several events of different types, sent from multiple senders. Figure 5 shows all possible event types that could be generated, including their main information. As depicted, each event type (ET) includes three principal objects: timestamp, event (which in turn contains id and senderId), and finally the payload object. It is the payload object that categorizes the ET. While ET1 includes a payload object with palletID, ET2 also contains the recipe item. Additionally, ET3 includes color information.

The difference between the ETs is originated by each event sender. Each type of event is linked to specific operations that are executed in the line. ET1 originates from CNV RTUs

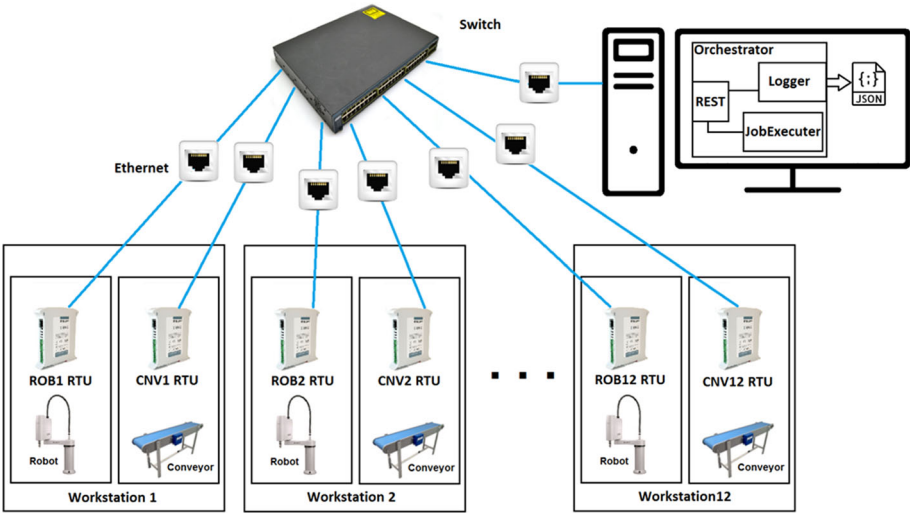

Fig. 4 Data collection orchestrator integration

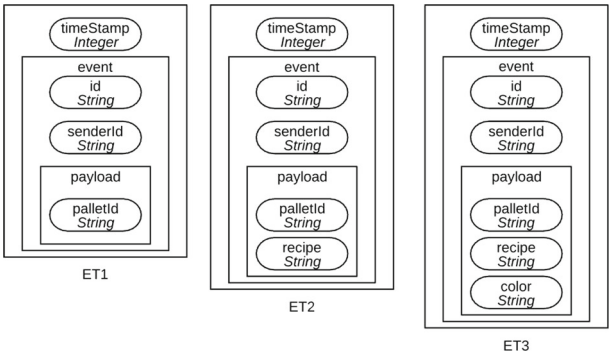

Fig. 5 Information included in each ET

whenever a pallet moves to a new conveyor zone (i.e., a new position). The first operation linked to ET1 is Z_CHANGED. In addition, ET1 is issued when executing PAPER_LOADED and PAPER_UNLOADED operations, which notify the load or unload of papers in WS1. Secondly, ET2 is linked to the DRAW_START_EXECUTION operation, which is executed for starting a drawing with any robot from WS2 to WS6 and from WS8 to WS12. Finally, ET3 is linked to the DRAW_END_EXECUTION, which is triggered once a robot finishes the drawing process. Figure 6 shows an example of the JSON format of an ET3.

5.2 An ontology-based modeling approach for the FASTory line

This subsection presents the ontology model designed to describe the FASTory line. In the UML class diagram presented in Fig. 7, the model that this research employs for storing, retrieving and reasoning information generated from FASTory events via ontologies is described.

```
{
        "timeStamp":1501750807464,
        "event":{
        "id":"DRAW_END_EXECUTION",
        "senderId":"ROB2",
        "payload":{
                "palletId":"1501749696163",
                "recipe":"4",
                "color":"RED"
        }
        }
}
```


Fig. 6 ET3 JSON format example

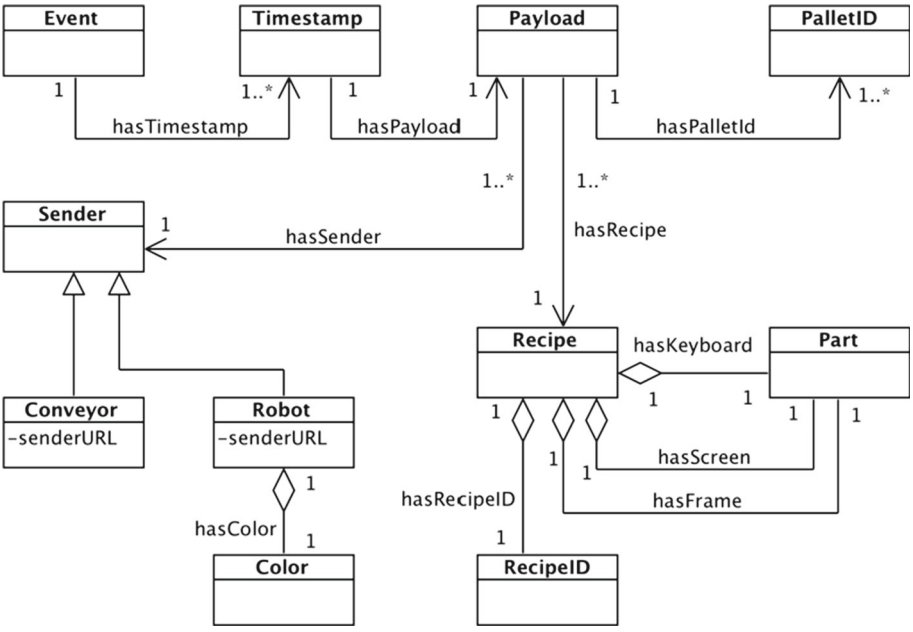

Fig. 7 FASTory ontology model represented within the UML class diagram

Figure 7 illustrates that the ontology is composed of 11 classes. Besides the depicted range and domain of the object properties, the model includes the datatype property senderURL, which is used for describing the URL of event senders, i.e., robot and conveyor RTUs. In order to demonstrate the implementation of the presented model, Fig. 8 shows the model in Protégé.[14] Protégé has been used as the ontology editor at the design phase to create the model and to perform both consistency tests and execution of queries in order to validate the model.

As presented in Fig. 8, the model includes certain instances by default. There is one instance for each robot and conveyor as well as the senderURL datatype property value, which is a string indicating the sender URL. Furthermore, all robots are linked to the color RED, as the default color that any robot of the FASTory line will use for drawing operations. This research requires the population of different models in order to evaluate each modeling

[14] https://protege.stanford.edu/.

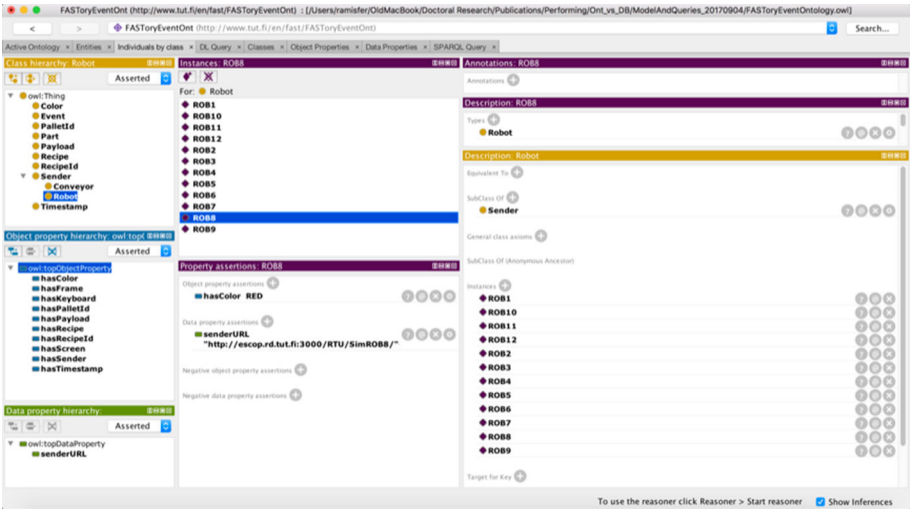

Fig. 8 FASTory model seen via the Protégé user interface

```
PREFIX ont: <http://www.tut.fi/en/fast/FASToryEventOnt#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
INSERT DATA
{
ont:%EVENT_ID% rdf:type ont:Event.
ont:%TIMESTAMP% rdf:type ont:Timestamp.
ont:%EVENT_ID% ont:hasTimestamp ont:%TIMESTAMP%.
ont:%PAYLOAD% rdf:type ont:Payload.
ont:%TIMESTAMP% ont:hasPayload ont:%PAYLOAD%.
ont:%PALLET_ID% rdf:type ont:PalletId.
ont:%PAYLOAD% ont:hasPalletId ont:%PALLET_ID%.
ont:%PAYLOAD% ont:hasSender ont:%SENDER_ID%.
ont:%PAYLOAD% ont:hasRecipe ont:%RECIPE%.
ont:%PAYLOAD% ont:hasColor ont:%COLOR%.
}
```

Fig. 9 SPARQL query for ET3 population

approach. The population of each model has been done in the environment described in Sect. 4. Figure 9 shows, as an example, the query that permits populating events of type 3 (ET3), as presented in Fig. 6. This illustrates the structure of the implemented update queries.

Since the ontology population is about updating the model, the executed queries are SPARQL Update queries, which are usable within RDF-based models, as described in Sect. 2. It is important to mention that the words in between "%" characters are variables, to be replaced by the Java code in order for a query to be fully executable. These variables are taken from each incoming event for the model population.

Besides similar queries for populating the ontology with ET1 and ET2 events, this research required the design and implementation of SPARQL SELECT queries for retrieving information that is useful for supporting RQ1 and RQ3. This is then presented to aid discussion and to demonstrate the results and required efforts during the research.

Table 2 Data type events

Path	Data type	Mandatory
timeStamp	TIMESTAMP	Yes
event.id	VARCHAR[20]	Yes
event.senderId	VARCHAR[10]	Yes
event.payload.palletId	CHAR[13]	Yes
event.payload.recipe	VARCHAR[5]	No
event.payload.color	VARCHAR[5]	No

5.3 A DB-based modeling approach for the FASTory line

An open-source database system named PostgreSQL[15] is used for DB-based modeling. The data modeling for this research employed generic RDBMS and does not exploit the specific features provided by PostgreSQL, being instead representative of SQL technology in general. The RDBMS's features include the ability to store and to query the data in JSON format—similar to other document-oriented DBs. Storing the event data in JSON format may decrease the querying performance of a DB, but does significantly simplify the design process, as the message itself can persist in the DB. There are several steps in the design of a relational DB. Firstly, the data to be stored in the system should be classified to data primitives. Secondly, the data should be organized across DB tables and connected via relations.

As described in Sect. 5.1.1, the events generated by FASTory line are similar in structure and share multiple similar data parts. These events must therefore have an associated timestamp, type, sender, pallet, and may have recipe and color in the model. The timestamp can be directly mapped to the TIMESTAMP data primitive to allow advanced operations with the column. The event id, sender id, recipe and color are varying length strings, while the pallet id is a constant length string. All the fields shown in Table 2 are present in all the events with the exceptions of color and recipe.

The next step involves defining the tables and the relations between them. The structure of the tables depends mainly on the data structure, considering aspects such as the relation between fields in the data and which parts of the data are continuously updated. The design of the queries that are needed could affect the table's structure where some SQL commands depend on the technology employed. As a result, the developer needs to find the balance between query performance and the nature of data for constructing tables in the database. In some cases, the exact set of expected queries to be run on the database is known in the design phase; this can also be true of industrial DB deployments. This issue becomes more apparent with the contemporary trend for a more iterative approach. As a result, the process of DB design becomes more complicated, as not all the requirements are available and future changes cannot always be anticipated.

Hence, the most simple and straightforward approach for the case described in the paper would be to place all the data in the same table. Such an approach provides a reasonable structure for available data, since only a few fields could be skipped in the events in the system. In addition, such an approach should deliver a good performance for the expected queries. The structure for such a table is shown in Fig. 10.

[15] https://www.postgresql.org/.

Fig. 10 Structure of the table

```
CREATE TABLE events (
  id SERIAL PRIMARY KEY,
  ts TIMESTAMP NOT NULL,
  action_id VARCHAR(20) NOT NULL,
  sender_id VARCHAR(10) NOT NULL,
  pallet_id CHAR(13) NOT NULL,
  recipe VARCHAR(5),
  color VARCHAR(5)
);
```

Fig. 11 Organization for query population

```
INSERT INTO events (ts,
    action_id,
    sender_id,
    pallet_id,
    recipe,
    color)
VALUES (?, ?, ?, ?, ?, ?);
```

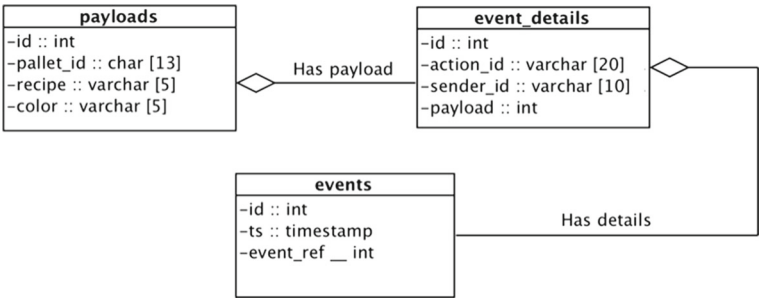

Fig. 12 DB model tables

For such a structure, the population queries are to be organized as shown in Fig. 11. During the population phase, all question marks are to be replaced with the proper values following the same order in the insert command.

Another approach to data modeling used in this research comprises splitting the data according to the content into three separate tables: one defining event payloads, one defining other event description, and one connecting the events to the timestamps. The events table should include a reference to an entry of event details table, which in turn should include a reference to the payload entry. The separation of the data into three tables, shown in Fig. 12, makes the queries more complicated, which leads to higher execution overhead in some cases.

The creation of the three tables is presented in Fig. 13. Once created, the tables are populated by the query as shown in Fig. 14. In a similar manner to the single-table structure, the question marks are replaced by proper values to form the raw data during the population phase.

6 Results and discussion

This section presents the querying benchmark tests performed on different data models (DB and ontology), which have been populated as described in Sects. 5.2 and 5.3. More precisely, three query types have been generated in order to compare the performance of the two data

```
CREATE TABLE payloads (
  id SERIAL PRIMARY KEY,
  pallet_id CHAR(13) NOT NULL,
  recipe VARCHAR(5),
  color VARCHAR(5)
);

CREATE TABLE event_details (
  id SERIAL PRIMARY KEY,
  action_id VARCHAR(20) NOT NULL,
  sender_id VARCHAR(10) NOT NULL,
  payload INT NOT NULL REFERENCES payloads(id)
);

CREATE TABLE events (
  id SERIAL PRIMARY KEY,
  ts TIMESTAMP NOT NULL,
  event_ref INT NOT NULL REFERENCES event_details(id)
);
```

Fig. 13 Creation of DB tables

```
WITH payload AS (
     INSERT INTO payloads (pallet_id, recipe, color)
     VALUES (?, ?, ?)
     RETURNING id
), event_dtls AS (
     INSERT INTO event_details (payload, action_id, sender_id)
     SELECT id, ?, ?
     FROM payload
     RETURNING id
)
INSERT INTO events (ts, event_ref)
     SELECT ?, id FROM event_dtls;
```

Fig. 14 Query to be executed for population

modeling technologies. Since the tests depend on the technology implementation, this may affect the overall results. Nevertheless, the tools employed for these experiments are among the most frequently used by data model designers, and so the results are reflective of common implementation performance.

The design of the three queries provides similar functionality for both DB and knowledge base (KB, ontology) implementations. These queries are presented in Tables 3, 4 and 5.

- The first query counts the number of triggered events for a given event ID.
- The second query counts the number of products that were produced using a specific pallet, which is filtered using the desired pallet ID.
- The third query returns all events that have been triggered in a specific period by giving the start and end timestamps.

A performance analysis of both data modeling technologies is achieved through the execution of the aforementioned queries. In the case of DBs, as discussed in the previous section, two different data models have been designed in order to study the effect of data structure on the performance. The first model considers the data to be stored as one single table, whereas the second approach stores the data in three tables (payloads, event_details and events). This difference in the database structure requires different queries for each functionality. Table 3 presents the three types of query for a single-table database. Although it is simpler to build a

Table 3 DB single-table queries

Query type	Query statement

Count events types	`SELECT events.action_id,`
	`COUNT(events.action_id)`
	`FROM events;`
	`GROUP BY events.action_id;`

Count products made on pallet	`SELECT events.pallet_id,`
	`COUNT(events.pallet_id)`
	`FROM events`
	`WHERE events.action_id ='DRAW_END_EXECUTION'`
	`GROUP BY events.pallet_id;`

Count events in time scope	`SELECT events.sender_id,`
	`COUNT(events.sender_id)`
	`FROM events`
	`WHERE events.ts`
	`BETWEEN '2017-08-03 11:00:00.000'::timestamp`
	`AND '2017-08-03 11:15:00.000'::timestamp`
	`GROUP BY events.sender_id;`

query for a single table, this could decrease the flexibility should the data structure change. The query "Count events types" allows the counting of the number of events for a certain event ID. It returns the list of event_id and the count of appearance for each event. The second row shows the count of the products that have been in a certain pallet. Finally, the third query counts the number of events within a given time window.

Similar to the single-table data structure, the three-table DB structure uses the same queries but with some changes, as shown in Table 4.

The queries for the KB are presented in Table 5; it is apparent that the structure of the queries is somewhat different from the DB queries. This is due to the different syntax of the languages that are used for querying the distinct data models. Nonetheless, the queries maintain, at some level, the same structure since they are used to extract the same information.

As mentioned in Sect. 4, the JMH framework has been used to create benchmarking tests. This framework allows programmers to examine the performance of Java routines. These tests measure the performance of a Java routine by executing the routines successively within a preconfigured period, which is known as iteration. Next, the programmer configures the framework to run the iteration several times. Depending on the configuration, the programmer can allocate some of the iterations to the warmup iteration. During the measurement, the JMH returns the number of times that the routine is executed during the specified period for each iteration. Then, once the framework completes all benchmarking tests, the average value of each benchmark across all iterations is obtained. A higher number indicates a better performance of the tested routine.

To achieve a satisfactory and relevant result for the objectives of this research, the JMH framework has been configured to run ten warmup iterations and ten benchmarking iterations taking one second for each iteration—this means that the unit for the results are operations per second (Ops/s). For the test routines, four benchmarking tests have been created for each model. Three routines execute the queries listed in Tables 3, 4 and 5. In addition, another

Table 4 DB three-table queries

Query type	Query statement

ount events types	`SELECT DISTINCT action_id, COUNT(action_id)`
	`FROM event_details;`
	`GROUP BY action_id;`

Count products made on pallet	`SELECT payloads.pallet_id, COUNT(payloads.pallet_id)`
	`FROM event_details, payloads`
	`WHERE event_details.action_id = 'DRAW_END_EXECUTION'`
	`AND payloads.id = event_details.payload`
	`GROUP BY payloads.pallet_id;`

Count events in time scope	`SELECT event_details.sender_id, COUNT(event_details.sender_id)`
	`FROM events, event_details`
	`WHERE events.ts`
	`BETWEEN '2017-08-03 11:00:00.000'::timestamp`
	`AND '2017-08-03 11:15:00.000'::timestamp`
	`AND event_details.id = events.event_ref`
	`GROUP BY event_details.sender_id;`

routine executes the populating routine for the DB and KB models in Figs. 9, 11, and 14, correspondingly. These benchmarking tests have been deployed in Docker containers to avoid disturbances from different OS processes. For this purpose, three Docker containers have been created and deployed, one for each model. For the single-table and three-table DB models, the measurements were similar but a slightly better performance is shown by the three tables. It is noticeable that there are, in both cases, drops in performance that could be related to the DB query cache feature.[16] In this context, MySQL caches the results of SELECTED queries by default in order to improve the performance. Due to conditions such as the complexity of the query, size of data, size of the results and hardware capability, this cache can be flushed or cleared automatically by the optimizer to prevent any possible errors. On the other hand, the Population benchmarking test showed greater variation from normal performance, as illustrated in both Figs. 15 and 16.

In the ontology model case, the performance is different from that of the DB, as presented in Fig. 17. During testing of the three queries routines, the warmup iteration showed incremental performance improvement until it reached a saturation value. Afterwards, the warmup continued at a steady performance. This steady performance was present in the benchmarking measurements as well. The dramatic difference between the performance results of ontologies compared with DBs is due to several technological and tool-based factors. The main reason for such a difference could be the buffering of the queries in the Jena ARQ,[17] since KB do not repeat the existing instance. This allows for caching of the paths between nodes. Unlike the query tests, the population was similar to the DB population benchmarking results.

After finishing all the benchmarking tests, the JMH framework presented the overall results, which are presented in Fig. 18 and Table 6. As shown, the results show a very distinct

[16] https://dev.mysql.com/doc/refman/5.5/en/query-cache.html.

[17] https://docs.oracle.com/database/122/RDFRM/rdf-suipport-for-apache-jena.htm#RDFRM246.

Table 5 KB queries

Query type	Query statement

Count events types	PREFIX ont: <http://www.tut.fi/en/fast/FASToryEventOnt#>
	SELECT DISTINCT ?Event (count(?Event) as ?Count
	WHERE {
	?Event ont:hasTimestamp ?Timestamp.
	}
	GROUP BY ?Event

Count products made on pallet	PREFIX ont: <http://www.tut.fi/en/fast/FASToryEventOnt#>
	SELECT ?Pallet (count(?Pallet) as ?FinishedProducts)
	WHERE {
	ont:DRAW_END_EXECUTION ont:hasTimestamp ?Timestamp.
	?Timestamp ont:hasPayload ?Payload.
	?Payload ont:hasPalletId ?Pallet.
	}
	GROUP BY ?Pallet

Count events in time scope	PREFIX ont: <http://www.tut.fi/en/fast/FASToryEventOnt#>
	PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
	SELECT ?Sender (count(?Event) as ?Count)
	WHERE {
	?Event ont:hasTimestamp ?Timestamp.
	?Timestamp ont:hasPayload ?Payload.
	?Payload ont:hasSender ?Sender.\n"
	Filter (STR(?Timestamp) >=
	"http://www.tut.fi/en/fast/FASToryEventOnt#1501747200000").
	Filter (STR(?Timestamp) <=
	"http://www.tut.fi/en/fast/FASToryEventOnt#1501748100000")).
	}
	GROUP BY ?Sender

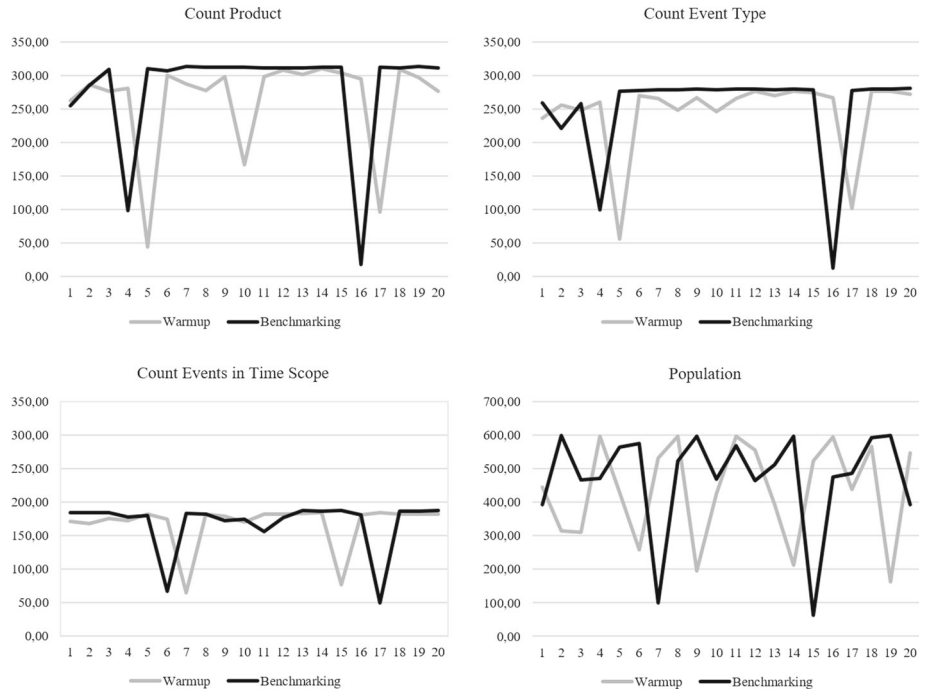

Fig. 15 Single-table DB benchmarking measurements

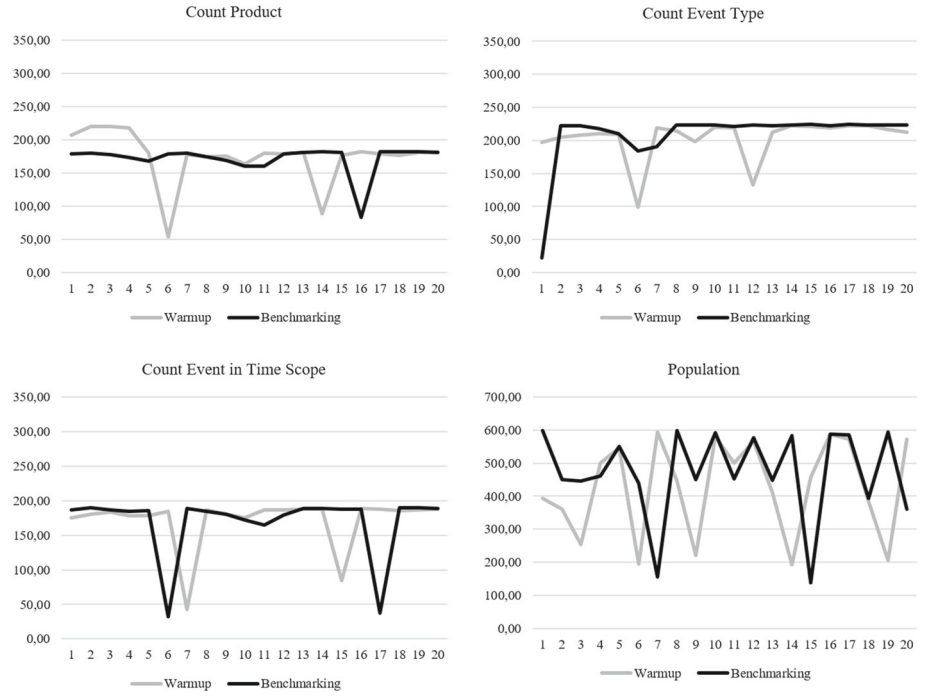

Fig. 16 Three-table DB benchmarking measurements

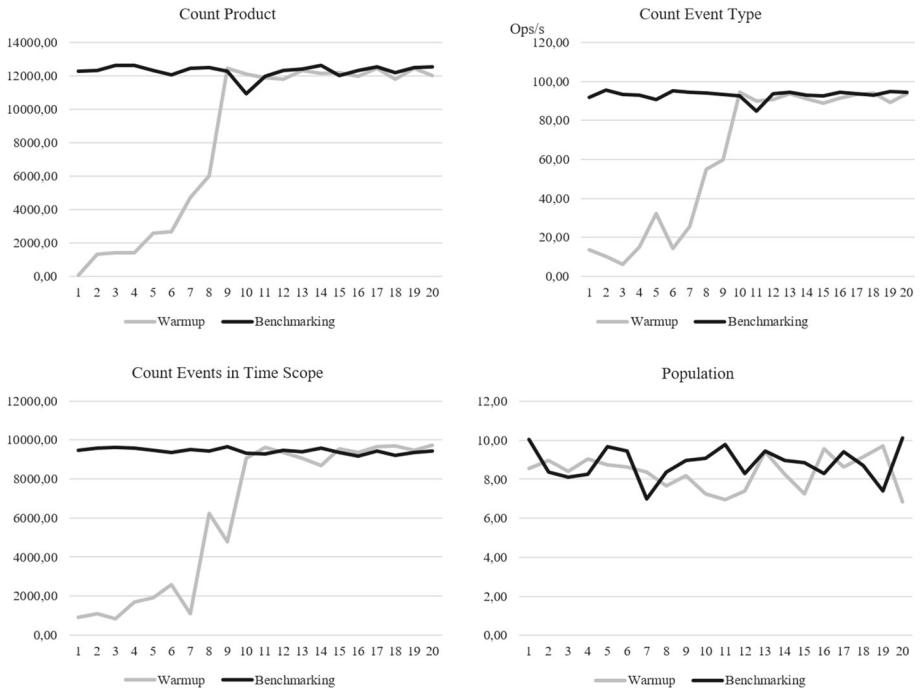

Fig. 17 KB benchmarking measurements

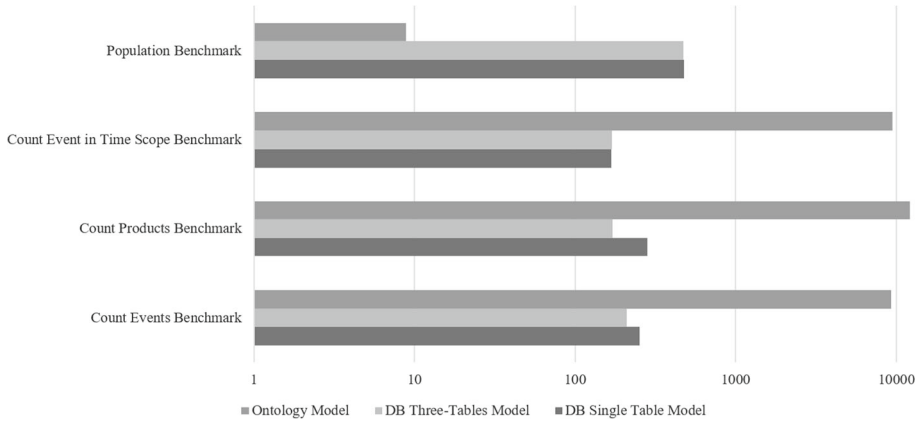

Fig. 18 Overall results for all benchmarking tests

performance difference between updating and querying the models. In the case of population benchmarking, the DB shows much better performance, with an average difference of 200 Ops/s. This difference could be generated by the nature of the technology, since the DB inserts data directly without extra mapping operations as the DB table is constructed. In contrast, the ontology update operation requires a mapping process in order to not duplicate the instances of the classes.

Table 6 Overall average benchmarking results

Test name	DB single-table model [Ops/s]	DB three-table model [Ops/s]	KB [Ops/s]
Count events benchmark	251.727	208.406	931.421
Count products benchmark	282.039	171.624	12291.206
Count event in time scope benchmark	168.307	169.658	9437.583
Population benchmark	475.498	473.017	8.831

Furthermore, the ontology model had the upper hand in the data retrieval process with a difference of approximately 9000 Ops/s. This difference could be caused by the caching feature, provided by the JENA ARQ, which caches the path between nodes for a querying operation—providing better performance. In this research, these parameters were kept at the default configuration settings to represent a common client trying to use the tool directly from the box. It is important to highlight that these tests depend on the search engine for each technology; therefore, the specific indexing algorithm should be further analyzed. These results can be compared with other research that addressed the same problem; however, the comparison can be unfair for some technologies since the tests and experimental techniques might vary, with different configurations and parameters. As an example, while the BSBM in [60] used an HTTP interface to connect with the JENA TDB; this research interfaces with the JENA TDB using the Java API (application programming interface) directly. This could affect the test results, where HTTP services can add latency to the system.

This section has presented the experimental results that have enabled a performance comparison of DB and ontology technologies. On the performance level, the tests intend to eliminate the technology effects, as close as possible, by following the same test conditions and using the same OS. However, as the vision was to try to deploy these tests with the most commonly used tools available in order to reflect the real-world scenario, this has been a challenging experiment. Instead of comparing both technologies, it is important to highlight that both technologies may work side by side—each providing unique features for the user. As an example, from the tests, ontologies should perform better than DBs at querying processes, which makes it a more reasonable choice to be used as a knowledge provider. On the other hand, DBs show more consistent performance for both querying and updating, which makes them suited for use as a data store. In addition, an ontology-based model permits more rich representation of the data. This fact suggests that ontologies would be a better choice for applications that require reasoning and for inferencing implicit knowledge of the data model.

With regard to RQ2, the boundaries between databases and knowledge-base technologies must be investigated. Due to the involvement of the authors of this research in several EU projects, they have experience of the evolution of both data modeling technologies (DBs and ontologies) with new concepts. As an example, the Cloud Collaborative Manufacturing Networks (C2NET)[18] project includes both technologies within the same solution, in order to exploit different features of each one. In this context, the C2NET project provides key functionalities for the smart and medium-sized enterprises (SMEs) on the enterprise resource planning (ERP) level in the well-known automation pyramid described in the ISA-95 standard [73]. These features, which are provided as web services in the C2NET project, include i) optimization, ii) monitoring and iii) assessment of production, delivery and logistics plans. Besides this, the C2NET platform also allows the companies to interact with other companies

[18] https://cordis.europa.eu/project/id/636909.

in the same supply chain, acting as a network for the exchange of information and facilitating communicating through the web.

In regard to its architecture, and related to the synergy of databases and ontologies, the C2NET platform employs both technologies. The data collected by the C2NET platform from SMEs—potentially ERP or factory shop-floor data—pass through a transformation process where it is homogenized with the schemas or standards of the C2NET platform data. The transformation is applied within the ontology technology since each company can provide the data in a different schema or format. The C2NET platform then uses the database technology to manage the transformed data before it is utilized by the aforementioned features.

Similar to C2NET, and as described in Sects. 5.2 and 5.3, both DB- and KB (ontology)-based approaches may work in conjunction and support each other since each one might have it is own specialized role to play. Although it is discussed in the following section, at first glance it can be argued that the knowledge base provides more flexibility and adjustability for the data format, whereas the database provides better performance and robustness to the system.

7 Conclusions and further work

The core objective of this research was to compare two data modeling approaches that are used in the context of PLM and PPR: ontologies and databases. In order to achieve this, the authors explored the literature to identify what work existed in this area. The knowledge gap that was identified gives rise to a limited comparison of the two technologies for common applications with limited quantitative and qualitative analyses. To address this gap, three RQs were synthesized:

- RQ1 focused on understanding how the data modeling approaches performed as data volume increased. This is important because databases are currently used extensively in industrial environments, handling large volumes of data, and it is necessary to understand how the ontological approach compares. The authors worked to create data models in a way which enabled a fair comparison with benchmarks that presented data on the following: event counting, product counting, event-in-time counting, and data model population. The results found that the ontology performed more than three times better in the event counting benchmarks, and orders of magnitude better than databases in all other test—apart from the population test. It is proposed that this is due to the fact that when an instance is created in an ontology, the respective mappings must also be created based on rules, which is not necessarily the case for databases (accounting for the poor performance of ontologies in executing population tasks). By comparison, once the instance exists, it is much easier to access and manipulate it in an ontological model than in a database due to the benefits that these mappings provide. In addition, it is important to keep in mind the effect of the tools used for such a study. The tool itself might play an important role since each tool is supported by optimization algorithms to enhance the performance. This could be the subject of further study to gain insight into the potential optimization process.
- RQ2 considered the idea that, ultimately, databases and ontologies have been developed for two quite distinct purposes and it is therefore necessary to understand how these respective technologies may complement each other. Drawing on experience from previous projects (e.g., [6,74,75]), the authors describe a scenario where myriad standards are homogenized using an ontological model and the data then instantiated within a database.

This type of complementary working addresses an environment where there may be a need to realize interoperability between heterogeneous software. Such an environment is typical of a manufacturing system. The results do demonstrate, however, that the population rate of an ontological model gives cause for concern in a high-data-volume environment. Given the significant differences in performance between ontologies and databases, it would be of value to investigate if the high-speed data instantiation of databases could be brought into a system where the high-speed querying of ontologies could be exploited.
- RQ3 aimed to examine the maintainability of databases in comparison with ontologies, but was not directly addressed in this work. This question was included as the authors appreciate the need to examine the respective technologies holistically and therefore require a lifecycle assessment from system design, through to implementation and then reconfiguration—this is true for both physical and digital systems. This article lacks a study that evaluates the maintenance efforts, although the authors do shed some light on the creation of the respective models. To address this, the authors are working on a further piece of research to introduce a change to system requirements and to assess the efforts required to realize them. This proposed study will consider the hypothesis that the perceived difference in effort for implementing and maintaining the data models will depend significantly on the engineer's familiarity with the respective technology, even though design tools can abstract users from fully understanding the model syntax.

This research concludes that ontologies and databases should not replace, but rather complement each other. The experiments show that both technologies present differences in their performance and that the decision for using one instead of the other will depend on the implementation and application. Nevertheless, it may be seen that the experiments presented do not allow the full exploitation of ontologies, due to the low expressivity of event information. To address this, the authors will further increase the complexity of event content—enabling the demonstration of other features such as implicit knowledge inference.

In summary, the work presented in this research contributes significantly to the body of knowledge by:

- Developing a methodology for comparing data modeling approaches.
- Quantitatively comparing ontological models and databases with a view to understanding how data volume affects performance.
- Considering how databases and ontologies may complement each other in the future and the scenarios in which they exist in a system whereby their whole is greater than the sum of their parts.

Acknowledgements The research work by Wael M. Mohammed is supported by the Doctoral School of the President call 2018 of the Tampere University of Technology. Also, the authors gratefully acknowledge the support of the graduate school funding of Tampere University and the UK EPSRC under the iCASE Ph.D. studentship (1377735) in carrying out this work.

References

1. IEC (2021) IEC White Paper Future Factory:2015 | IEC Webstore | water automation, water management, smart city. https://webstore.iec.ch/publication/23389
2. Delamer IM, Lastra JLM (2006) Service-oriented architecture for distributed publish/subscribe middleware in electronics production. IEEE Trans Ind Inform 2(4):281–294. https://doi.org/10.1109/TII.2006.885188
3. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M (2010) A view of cloud computing. Commun ACM 53(4):50–58. https://doi.org/10.1145/1721654.1721672
4. Liu C, Ding J, Toprac AJ, Chai T (2014) Data-based adaptive online prediction model for plant-wide production indices. Knowl Inf Syst 41(2):401–421. https://doi.org/10.1007/s10115-014-0757-8
5. Stark J (2011) Product lifecycle management: 21st century paradigm for product realisation, 2nd edn. Springer, London. https://doi.org/10.1007/978-0-85729-546-0
6. Ahmad M, Ferrer BR, Ahmad B, Vera D, Martinez Lastra JL, Harrison R (2018) Knowledge-based PPR modelling for assembly automation. CIRP J Manuf Sci Technol 21:33–46. https://doi.org/10.1016/j.cirpj.2018.01.001
7. Verborgh Ruben, Sande Miel Vander (2020) The Semantic Web identity crisis: in search of the trivialities that never were. Seman Web J 11(1):19–27
8. Ramis Ferrer B, Ahmad B, Vera D, Lobov A, Harrison R, Martínez Lastra JL (2016) Product, process and resource model coupling for knowledge-driven assembly automation. Automatisierungstechnik. https://doi.org/10.1515/auto-2015-0073
9. Martinez-Cruz C, Blanco IJ, Vila MA (2012) Ontologies versus relational databases: are they so different? A comparison. Artif Intell Rev 38(4):271–290. https://doi.org/10.1007/s10462-011-9251-9
10. Sir M, Bradac Z, Fiedler P (2015) Ontology versus database. IFAC-PapersOnLine 48(4):220–225. https://doi.org/10.1016/j.ifacol.2015.07.036
11. Schmidt D (2006) Guest editor's introduction: model-driven engineering. Computer 39(2):25–31. https://doi.org/10.1109/MC.2006.58
12. Michalos G, Makris S, Papakostas N, Mourtzis D, Chryssolouris G (2010) Automotive assembly technologies review: challenges and outlook for a flexible and adaptive approach. CIRP J Manuf Sci Technol 2(2):81–91. https://doi.org/10.1016/j.cirpj.2009.12.001
13. Fogliatto FS, da Silveira GJC, Borenstein D (2012) The mass customization decade: an updated review of the literature. Int J Prod Econ 138(1):14–25. https://doi.org/10.1016/j.ijpe.2012.03.002
14. Lee J, Lapira E, Bagheri B, Kao Ha (2013) Recent advances and trends in predictive manufacturing systems in big data environment. Manuf Lett 1(1):38–41. https://doi.org/10.1016/j.mfglet.2013.09.005
15. Crescenzi V, Fernandes AAA, Merialdo P, Paton NW (2017) Crowdsourcing for data management. Knowl Inf Syst 53(1):1–41. https://doi.org/10.1007/s10115-017-1057-x
16. Kilintzis V, Beredimas N, Chouvarda I (2014) Evaluation of the performance of open-source RDBMS and triplestores for storing medical data over a web service. In: 2014 36th annual international conference of the IEEE Engineering in Medicine and Biology Society, IEEE, Chicago, IL, pp 4499–4502. https://doi.org/10.1109/EMBC.2014.6944623
17. Lin HK, Harding JA, Shahbaz M (2004) Manufacturing system engineering ontology for semantic interoperability across extended project teams. Int J Prod Res 42(24):5099–5118. https://doi.org/10.1080/00207540412331281999
18. Staab S, Studer R (eds) (2009) Handbook on ontologies. International handbooks on information systems, 2nd edn. Springer, Berlin. https://doi.org/10.1007/978-3-540-92673-3
19. Gruber TR (1993) A translation approach to portable ontology specifications. Knowl Acquis 5(2):199–220. https://doi.org/10.1006/knac.1993.1008
20. Borst P, Akkermans H, Top J (1997) Engineering ontologies. Int J Hum Comput Stud 46(2–3):365–406. https://doi.org/10.1006/ijhc.1996.0096
21. Zdravković M, Panetto H, Trajanović M, Aubry A (2014) Explication and semantic querying of enterprise information systems. Knowl Inf Syst 40(3):697–724. https://doi.org/10.1007/s10115-013-0650-x
22. Kalibatiene D, Vasilecas O (2011) Survey on ontology languages. In: Grabis J, Kirikova M (eds) Perspectives in business informatics research. Springer, Berlin, pp 124–141
23. Bechhofer S, van Harmelen JHF (2021) OWL web ontology language reference. https://www.w3.org/TR/owl-ref/
24. Eric Prud hommeaux AS (2021) SPARQL Query Language for RDF. https://www.w3.org/TR/rdf-sparql-query/
25. Paula Gearon AP Alexandre Passant (2021) SPARQL 1.1 Update. https://www.w3.org/TR/sparql11-update/

26. Horrocks I, Patel-Schneider P, Boley H, Tabet S, Grosof B, Dean M (2007) SWRL: a semantic web rule language combining oWL and ruleML. W3C Member submission 21
27. Puttonen J, Lobov A, Lastra JLM (2013) Maintaining a dynamic view of semantic web services representing factory automation systems. In: 2013 IEEE 20th international conference on web services, pp 419–426. https://doi.org/10.1109/ICWS.2013.63
28. Efthymiou K, Sipsas K, Mourtzis D, Chryssolouris G (2015) On knowledge reuse for manufacturing systems design and planning: a semantic technology approach. CIRP J Manuf Sci Technol 8:1–11. https://doi.org/10.1016/j.cirpj.2014.10.006
29. Delamer IM, Lastra JLM (2006) Ontology modeling of assembly processes and systems using semantic web services. In: 2006 4th IEEE international conference on industrial informatics, pp 611–617. https://doi.org/10.1109/INDIN.2006.275631
30. Usman Z (2012) A manufacturing core concepts ontology to support knowledge sharing. thesis, Loughborough University, http://articles/thesis/A_manufacturing_core_concepts_ontology_to_support_knowledge_sharing/9524204/1
31. Borgo S, Leitão P (2007) Foundations for a core ontology of manufacturing. In: Sharman R, Kishore R, Ramesh R (eds) Ontologies: a handbook of principles, concepts and applications in information systems, integrated series in information systems. Springer, Boston, pp 751–775. https://doi.org/10.1007/978-0-387-37022-4_27
32. Miller GA (1995) WordNet: a lexical database for English. Commun ACM 38(11):39–41. https://doi.org/10.1145/219717.219748
33. Pratt MJ (2001) Introduction to ISO 10303: the STEP standard for product data exchange. J Comput Inf Sci Eng 1(1):102–103. https://doi.org/10.1115/1.1354995
34. Dartigues C, Ghodous P, Gruninger M, Pallez D, Sriram R (2007) CAD/CAPP integration using feature ontology. Concurr Eng 15(2):237–249. https://doi.org/10.1177/1063293X07079312
35. Krima SI, Barbau R, Fiorentini X, Rachuri S, Sriram RD (2009) OntoSTEP: OWL-DL Ontology for STEP. NIST Interagency https://www.nist.gov/publications/ontostep-owl-dl-ontology-step, last Modified: 2017-02-19T20:02-05:00
36. Gangemi A, Guarino N, Masolo C, Oltramari A, Schneider L (2002) Sweetening ontologies with DOLCE. In: Gómez-Pérez A, Benjamins VR (eds) Knowledge engineering and knowledge management: ontologies and the semantic web. Lecture notes in computer science. Springer, Berlin, pp 166–181. https://doi.org/10.1007/3-540-45810-7_18
37. Ontology BF (2021) Basic Formal Ontology (BFO) | Home. http://basic-formal-ontology.org/
38. Deshayes L, Foufou S, Gruninger M (2007) An ontology architecture for standards integration and conformance in manufacturing. In: Tichkiewitch S, Tollenaere M, Ray P (eds) Advances in integrated design and manufacturing in mechanical engineering II. Springer, Dordrecht, pp 261–276. https://doi.org/10.1007/978-1-4020-6761-7_18
39. Li Q, Chen YL (2009) Ontology capture methods. In: Li Q, Chen YL (eds) Modeling and analysis of enterprise and information systems: from requirements to realization. Springer, Berlin, pp 227–243. https://doi.org/10.1007/978-3-540-89556-5_12
40. Peraketh B, Menzel C, Mayer R, Fillion F, Futrell M (1994) Ontology capture method (IDEF5). Armstrong Laboratory, Arlington
41. Uschold M, Gruninger M (1996) Ontologies: principles, methods and applications. Knowl Eng Rev 11(2):93–136. https://doi.org/10.1017/S0269888900007797
42. Fernández-López M, Gómez-Pérez A, Juristo N (1997) METHONTOLOGY: from ontological art towards ontological engineering. In: Proceedings of the ontological engineering AAAI-97 Spring symposium series, Facultad de Informática (UPM), Stanford University, EEUU. http://oa.upm.es/5484/
43. Noy N, Mcguinness D (2001) Ontology development 101: a guide to creating your first ontology. Knowl Syst Lab 32
44. Fokou G, Jean S, Hadjali A, Baron M (2017) Handling failing RDF queries: from diagnosis to relaxation. Knowl Inf Syst 50(1):167–195. https://doi.org/10.1007/s10115-016-0941-0
45. Angles R, Gutierrez C (2008) Survey of graph database models. ACM Comput Surv 40(1):1–39. https://doi.org/10.1145/1322432.1322433
46. Elmasri R, Navathe S (2016) Fundamentals of database systems, 7th edn. Pearson, Hoboken
47. Gulutzan P, Pelzer T (1999) SQL-99 complete, really: an example-based reference manual of the new standard. R&D Books, Lawrence
48. Sadalage PJ, Fowler M (2012) NoSQL distilled a brief guide to the emerging world of polyglot persistence. Addison-Wesley, Upper Saddle River
49. Taylor RW, Frank RL (1976) CODASYL data-base management systems. ACM Comput Surv 8(1):67–103. https://doi.org/10.1145/356662.356666

50. IBM (2005) IMS - Introduction - History of IMS: Beginnings at NASA. https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.imsintro.doc.intro/ip0ind0011003710.htm
51. Yannakoudakis EJ (1988) Database design methodology. In: Yannakoudakis EJ (ed) The architectural logic of database systems. Springer, London, pp 135–161. https://doi.org/10.1007/978-1-4471-1616-5_6
52. Teorey TJ, Yang D, Fry JP (1986) A logical design methodology for relational databases using the extended entity-relationship model. ACM Comput Surv 18(2):197–222. https://doi.org/10.1145/7474.7475
53. Teorey T, Lightstone S, Nadeau T, Jagadish HV, Safari OMC (2011) Database modeling and design, 5th edn. Morgan Kaufmann, Burlington
54. Gani A, Siddiqa A, Shamshirband S, Hanum F (2016) A survey on indexing techniques for big data: taxonomy and performance evaluation. Knowl Inf Syst 46(2):241–284. https://doi.org/10.1007/s10115-015-0830-y
55. Konstantinou N, Spanos DE, Mitrou N (2008) Ontology and database mapping: a survey of current implementations and future directions. J Web Eng (JWE) 7:1–24
56. Choi SS, Yoon TH, Noh SD (2010) XML-based neutral file and PLM integrator for PPR information exchange between heterogeneous PLM systems. Int J Comput Integr Manuf 23(3):216–228. https://doi.org/10.1080/09511920903443234
57. Gunpinar E, Han S (2008) Interfacing heterogeneous PDM systems using the PLM Services. Adv Eng Inform 22(3):307–316. https://doi.org/10.1016/j.aei.2007.08.009
58. El Kadiri S, Kiritsis D (2015) Ontologies in the context of product lifecycle management: state of the art literature review. Int J Prod Res 53(18):5657–5668. https://doi.org/10.1080/00207543.2015.1052155
59. Kitamura Y, Koji Y, Mizoguchi R (2006) An ontological model of device function: industrial deployment and lessons learned. Appl Ontol 1(3–4):237–262
60. Bizer C, Schultz A (2009) The Berlin SPARQL benchmark. Int J Semant Web Inf Syst 5(2):1–24. https://doi.org/10.4018/jswis.2009040101
61. M Li, X-Y Du, S Wang (2005) Learning ontology from relational database. In: 2005 International conference on machine learning and cybernetics, vol 6, pp 3410–3415. https://doi.org/10.1109/ICMLC.2005.1527531
62. Cullot N, Ghawi R, Yetongnon K (2007) DB2OWL: a tool for automatic database-to-ontology mapping. In: Proceedings of the fifteenth Italian symposium on advanced database systems, p 494
63. Jayakumar P, Shobana P (2014) Creating ontology based user profile for searching web information. In: International conference on information communication and embedded systems (ICICES2014), pp 1–6. https://doi.org/10.1109/ICICES.2014.7033893
64. Barrasa J, Corcho O, Gomez perez A (2004) R2O, an extensible and semantically based database-to-ontology mapping language. In: Proceedings of the 2nd workshop on semantic web and databases(SWDB2004). Springer, pp 1069–1070
65. Laclavík M (2006) RDB2Onto: relational database data to ontology individuals mapping. Information and Knowledge Oriented Technologies Group
66. Ho LTT, Tran CPT, Hoang Q (2015) An approach of transforming ontologies into relational databases. In: Nguyen NT, Trawiński B, Kosala R (eds) Intelligent information and database systems. Lecture notes in computer science. Springer, Cham, pp 149–158. https://doi.org/10.1007/978-3-319-15702-3_15
67. Gali A, Chen CX, Claypool KT, Uceda-Sosa R (2004) From ontology to relational databases. In: Wang S, Tanaka K, Zhou S, Ling TW, Guan J, Yang D, Grandi F, Mangina EE, Song IY, Mayr HC (eds) Conceptual modeling for advanced application domains. Lecture notes in computer science. Springer, Berlin, pp 278–289. https://doi.org/10.1007/978-3-540-30466-1_26
68. Eder J, Haav HM, Kalja A, Penjam J (eds) (2005) Advances in databases and information systems: 9th East European conference, ADBIS 2005, Tallinn, Estonia, 12–15 September 2005. In: Proceedings. Information systems and applications, incl. Internet/Web, and HCI. Springer, Berlin. https://doi.org/10.1007/11547686
69. Trinkunas J, Vasilecas O (2015) A graph oriented model for ontology transformation into conceptual data model. Inf Technol Control. https://doi.org/10.5755/J01.ITC.36.1.11832
70. Lv Y, Xie C (2012) An ontology-based approach to build conceptual data model. In: 2012 9th international conference on fuzzy systems and knowledge discovery, pp 807–810. https://doi.org/10.1109/FSKD.2012.6234141
71. Lee J, Goodwin R (2006) Ontology management for large-scale enterprise systems. Electron Commer Res Appl 5(1):2–15. https://doi.org/10.1016/j.elerap.2005.08.003
72. Saccol DB, Andrade TC, Piveta EK (2011) Mapping OWL ontologies to relational schemas. In: 2011 IEEE international conference on information reuse integration, pp 71–76. https://doi.org/10.1109/IRI.2011.6009523

73. Commission IE (2021) ANSI/ISA-95.00.03-2013 enterprise-control system integration—part 3: activity models of manufacturing operations management. https://www.isa.org/products/ansi-isa-95-00-03-2013-enterprise-control-system-i
74. Ramis Ferrer B, Martinez Lastra JL (2017) Private local automation clouds built by CPS: potential and challenges for distributed reasoning. Adv Eng Inf 32:113–125. https://doi.org/10.1016/j.aei.2017.01.007
75. Iarovyi S, Ramis B, Xiangbin X, Sampath A, Lobov A, Lastra JLM (2015) Representation of manufacturing equipment and services for OKD-MES: from service descriptions to ontology. In: 2015 IEEE 13th international conference on industrial informatics (INDIN), pp 1069–1074. https://doi.org/10.1109/INDIN.2015.7281883

Dr. Borja Ramis Ferrer received the B.Sc. degree in electrical engineering from the Universidad de las Islas Baleares, Islas Baleares, Spain, in 2011 and the M.Sc. degree (with distinction) in Factory Automation from Tampere University of Technology (TUT), Tampere, Finland, in 2013. In addition, he obtained the Dr. Sc. (Tech.) degree in Automation from TUT, in 2018. With more than 8 years of experience in R&D&i in Industrial Informatics, Knowledge Representation (Ontology) and Embedded Systems for Factory Automation, Dr. Ramis contributed to many EU projects. He has (co)authored over 50 scientific papers and his research interests include the deployment of knowledge-based and cyber-physical systems in the fields of Automation, Robotics and Industrial Informatics. Since 2019, he is working with cybersecurity solutions for factories of the future and smart environments.

Wael M. Mohammed is a Doctoral Researcher at Tampere University. He received a M.Sc. in Automation Engineering from Tampere University of Technology in 2017 and a B.Sc. in Mechatronics Engineering from university of Jordan in 2010. As he is pursuing a Doctoral Degree in Engineering Sciences; Mr. Mohammed's research interests include Robotics, Digital Twins, Knowledge-Based Reasoning Engines and Factory Automation. In addition, Mr. Mohammed has been involved in writing proposals for research and innovation project funded by the EU commission. In 2010, he worked as a research assistant in the Production Engineering Department at Tampere University of Technology. Then, in 2011, Mr. Mohammed worked as a head of the technical department in the Traffic Management System project at Etihad Alafandi L.L.C. in Saudi Arabia.

Mussawar Ahmad has a background in Mechanical Engineering and a Ph.D. in Industrial Automation from the University of Warwick. He is currently a Senior Research Engineer at the Manufacturing Technology Centre in the Automation and Robotics team. His current research interests are around the use of knowledge-based to realize the adoption and deployment of intelligent automation systems.

Sergii Iarovyi is a Software Engineer at Kalmar. He received a M.Sc. degree (with Distinction) in Electro-mechanics from the National Technical University "KhPI," Kharkiv, Ukraine, in 2011 and the M.Sc. degree in Factory Automation from Tampere University of Technology, Tampere, Finland, in 2014. He has worked at FAST laboratory, Tampere University of Technology as a Project Researcher. His research interests are in the application of semantic web services, cyber-physical systems, and enterprise integration for factory automation.

Jiayi Zhang is a Software Developer at WMG. He received his Ph.D. in Automation Systems from the Department of WMG, University of Warwick. He received his B.E. degree in Software Engineering and M.Sc. degree in Computing, respectively, from Tianjin Normal University (China) and University of Teesside (UK). His main research interests are semantic ontology predictive modeling, intelligent manufacturing, and knowledge-based system integration. He has worked as an IT engineer, participating in different manufacturing projects related to software development technologies, ontology modeling design, and semantic data analysis. Currently, he is researcher and software developer of the Automation Systems Group at WMG. He has been working closely with the automotive industry on several collaborative research projects focusing on data integration system, automated ontology generation, and audio emotion analysis. His current projects include Smart Seat Manufacturing (Realtime process data analysis), Lear Smart Manufacturing (ProductProcessResource data integration), vueOne auto code generation.

Robert Harrison is Professor of Automation Systems and Head of the Digital Technologies Directorate at WMG. Having joined University of Warwick in March 2013, he was previously Director of Automation Systems at Loughborough University. Current research includes ISCF Manufacturing Made Smarter SIMPLE, DIALOG and DAMPO projects, EU DIGIMAN, and numerous contract research and HVM Catapult projects, with BAE Systems, Lear Corporation, Atlas Copco, and others. Between 2013 and 2018, he led the EPSRC Knowledge Driven Configurable Manufacturing project. From 2005 onwards, his Automation Systems Group were partners on several large EU projects related to serviceoriented distributed systems, including SOCRADES, RIMACS, Arrowhead, and IMCAESOP. Robert received a Royal Academy of Engineering Global Research Award in 2004 with visiting research roles at Schneider Electric and Ford. He was a Research Fellow at NUS Singapore from 1993 to 1995 and Ford Research Fellow from 1996 to 2002. He holds a PhD in automation from Loughborough University.

Jose Luis Martinez Lastra joined Tampere University of Technology in 1997 and became University Full Professor in 2006. His research interest is on applying Information and Communication Technologies to the field of Factory Automation. Prof. Lastra leads the FAST-Lab with the ultimate goal of seamlessly integrating the knowledge of humans and machines. Prof. Lastra has co/authored over 300 scientific papers and holds a number of patents in the field of Industrial Informatics and Automation. He serves as an Associate Editor of the IEEE Transactions on Industrial Informatics, and he served a Technical Editor of the IEEE/ASME Transactions on Mechatronics.