General Model for Tracking Manufacturing Products Using Graph Databases

Jorge Martinez-Gil, Reinhard Stumptner, Christian Lettner, Mario Pichler, Salma Mahmoud, Patrick Praher, Bernhard Freudenthaler

General Model for Tracking Manufacturing Products Using Graph Databases*

Jorge Martinez-Gil[1], Reinhard Stumptner[2], Christian Lettner[1], Mario Pichler[1], Salma Mahmoud[1], Patrick Praher[1], Bernhard Freudenthaler[1]

[1]Software Competence Center Hagenberg GmbH
Softwarepark 21, 4232 Hagenberg, Austria
[2]FAW GmbH
Softwarepark 35, 4232 Hagenberg, Austria
e-mail: jorge.martinez-gil@scch.at

Abstract. One of the major problems in the manufacturing industry consists of the fact that, when manufacturing a product, many parts from different lots are supplied and mixed to a certain degree during an indeterminate number of stages, what makes it very difficult to trace each of these parts from its origin to its presence in a final product. In order to overcome this limitation, we have worked towards the design of a general solution aiming to improve the traceability of the products from several manufacturers. This solution is based on the exploitation of graph databases which allows us to significantly reduce response times compared to traditional relational systems.

Keywords: Data Engineering, Graph Databases, Knowledge Graphs, Manufacturing

1 Introduction

Quality related errors in manufacturing create a lot of problems for the industrial sector mainly because they lead to a great waste of resources in terms of time, money and effort spent to identify and solve them [18]. For this reason, researchers and practitioners aim to find novel solutions capable of tracking and analyzing manufacturing products in an appropriate and easy to use manner [15]. There are already several approaches belonging to different manufacturing domains: additive manufacturing [17], toy manufacturing [4], electrical equipment [13], or fabrication of cylindrical markers [12].

In this context, one of the challenges of modern manufacturing is that a final product usually consists of different components which themselves can also

* This work is an extended version of: Jorge Martinez-Gil, Reinhard Stumptner, Christian Lettner, Mario Pichler, and Werner Fragner. *Design and implementation of a graph-based solution for tracking manufacturing products*. In New Trends in Databases and Information Systems. ADBIS Workshops 2019. Communications in Computer and Information Science, volume 1064, pages 417-423. Springer Cham, 2019

consist of different components, and so forth. At the lowest level, there is raw material (e.g. steel coils, sheet metal, steel rolls, etc.) which nature is also very relevant to the quality of the final product. In this way, tracking and connecting all the data of the different manufacturing stages is crucial to finding the causes of quality-related errors in the final product.

In case the components and raw materials have one-to-one or one-to-many relationships to the final products, the tracking process is quite straightforward and has already been implemented satisfactorily in some of the existing solutions. However, manufacturing products can also be made up of parts that are in lots, and lots can be combined to make other lots. Also, many different lots can be combined into a new one that can be part of many other ones. This means that some manufacturers have to deal with a problem involving many-to-many relationships with blurred relationships among the single parts.

One of the most important problems here is that providing these relationships is very important for both the operators and the quality engineers, so they can drill down from the final product to the assembled components and ultimately to the used raw materials and can identify causes to problems that are not obvious at the first glance.

Unfortunately, the existing solutions based on relational databases are not very useful for the people who are in charge of examining these lots. On the one hand, the existing solutions have bad response times when dealing with this problem, and on the other hand, there are no meaningful probability values available. So the quality engineers must invest a lot of time to analyze all possibly related data and cannot just focus on the relevant data.

To alleviate this problem, we have tried to look for a solution so that it can be possible to track all items from different lots that were used in final products. Our proposed solution is based on the exploitation of graph databases. The major advantage of such approach, concerning the existing ones, is that it allows for informed queries, i.e. queries that can lead to early termination if nodes with no compatible outgoing relations are found. As a result, we have got a software system that presents lower execution time for most of the use cases concerning the tracking of items.

In our previous work [19], we presented a specific approach for the design and solution of a tailored solution for a manufacturing company located in Upper Austria. In the present work, our major contribution here is an extension of our previous work to build a general solution for appropriately tracking the manufacturing products that have different kinds of dependencies (evolution, distribution, and so on). Our solution is intended to outperform traditional systems based on relational databases in the specific context of tracking defective items in lots of manufacturing products. Besides, to illustrate our proposal, we include a complete use case that shows some of the functionality that can be derived from a solution of this kind.

The rest of this work is structured in the following way: Section 2 presents the state-of-the-art concerning the current graph-based solutions for the manufacturing industry. Section 3 describes the design and implementation of our

solution and a use case whereby our solution outperforms the traditional tracking systems. Section 4 discusses the lessons learned from this research. Finally, in Section 5, we remark the conclusions that can be extracted from this work as well as the possible future lines of research.

2 State-of-the-art

In spite of the great need for tracking solutions in manufacturing environments, it seems that most of the quality assurance processes that require controlling and supervising the whole production chain to timely detect human errors and defective materials need to be further automatized. The major reason is that little attention has been paid to this problem due to technical limitations, and as result, there are not too many solutions in the manufacturing domain, but just a few works have been proposed to date [9, 20, 24, 25]

We can go even a step further beyond to see that the problem can be aggravated, even in the case of passing all quality assurance controls. The reason is that products can be rejected by end-users or other manufacturers if unknown problems appear. Therefore, the capability to track each part of a lot from its origin is of vital importance for the manufacturing industry.

In this context, it is necessary to remark that existing manufacturing systems are far from being trivial since the data models they work with often consist of many data types and data sources that are in no relation to each other. Due to this, modeling and optimization, as well as process analysis, represent often a hard task [26].

One of the major limiting factors for the solution to this problem is that traditional relational database systems (i.e. the systems that have so far been used mostly in the industry) are unable to model this specific problem effectively. Therefore, we have focused our research on graph databases [23]. The idea behind graph databases is their capability to store data in nodes and edges versus tables, as found in relational databases [3]. Each node represents an entity, and each edge represents a relationship between two nodes [5]. This way of modeling the problem is much more natural and is in line with a problem arising from dependencies such as this.

It is generally assumed that graph databases have some key advantages over relational databases in this context. The reason is that unlike relational databases, graph databases are designed to store interconnected data what makes it easier to work with these data by not forcing intermediate indexing at every time, and also making it easier to facilitate the evolution of the data that we are working with.

In the literature, recurrent mention is made of some of the advantages of graph databases concerning traditional relational models. It has been possible to identify the three major advantages of graph databases in comparison with traditional relational databases to tackle this problem:

(1) The first advantage when dealing with a graph is that as opposed to the relational world, foreign key relationships are not relations in the sense of edges of a graph.
(2) The second important advantage of graph databases in relation to relational databases is that, when referring to the latter ones, it is not possible to assign properties or labels to relationships. It is possible to give them a name in the database, but it is not possible to visualize them (e.g. derivations, transformations, etc.). When, in fact, in graph databases is the natural way to model data.
(3) Last, but not least, traditional systems based on the exploitation of relational databases are not able to scale as well as graph databases when dispatching relationship-like queries [2].

Some of these advantages are making graph databases gaining popularity among big industrial players. Moreover, their application domain is very broad [1]. In fact, many organizations are already using databases of this kind for detecting fraud in monetary transactions, providing product and service recommendations, documenting use cases and lessons learned in a wide range of domains, managing access control in restricted places, network monitoring to identify potential risks and hazards, and so on. Moreover, if we focus strictly on the manufacturing industry, graph-based solutions have already been proposed in forecasting and recommendation [22].

3 General Model for Tracking Manufacturing Products Using Graph Databases

To illustrate the problem that we are facing with an example, let us think of a situation where a finished part of a product is rejected by the customer because of a number of quality errors. In that case, it is quite common that the manufacturer of the finished part must make a statement within 24 hours if this error can be restricted to the single rejected product or if a greater amount of parts is affected. In case a greater amount of parts is affected, then the exactly affected lots must be reported to the customer.

As it is not difficult to imagine, a situation of this kind happens very often, and the worst thing is not that, but that it is really expensive in terms of effort, time, money and brand image. So it should be tried by all means to minimize its impact as far as possible.

To do that, the first task of the manufacturer is to try to find the cause of the quality error. Therefore it must be possible to analyze the captured data of the finished part but also the captured data of all assembled components and raw materials. In case the cause of the quality error lies in a specific component or raw material lot, the manufacturer must find all finished parts that contain this lot. This means that finding the right affected lots in a short period of time can decide whether the manufacturer must recall millions of finished parts or just a few. This means that if we can identify the affected lots quickly, it is not necessary to recall all delivered lots.

It is possible to think a situation whereby to perform this analytic task, the manufacturer needs to search back and forth across all the data of the finished parts. If a relational database is used, this means that many queries and their corresponding responses would need to be combined. In our approach, the solution is much more intuitive since it is in general possible to easily write queries capable of running over the data in any direction as we will see later in this paper. The capability to discover and see the connections between different parts of a product allows a human operator to effectively perform this tracking.

3.1 Notation

Let L_m^t be a lot of parts produced on machine m at time step t. Every machine m has a buffer b_m where lots to be processed at this machine are poured into, i.e. they are getting blurred. Then, the relation $usage : (L_m^{t2}, L_n^{t1})$ defines that at time slice $t2$ the lot L_m^{t1} has been poured into the buffer of machine m for processing. So beginning at time slice $t2$ parts of lot L_n^{t1} are installed with a certain probability into parts of lot L_m^{t2}.

The relation $pred : \{(L_m^{t2}, L_m^{t1})\}$ defines that lot L_m^{t1} is produced before lot L_m^{t2} on machine m and the buffer b_m of machine m was not empty when the production of L_m^{t2} started. Lots that are delivered by other suppliers, i.e. raw materials, are treated the same way. They will be assigned a virtual production machine number and time slice which uniquely identifies the batch number from the supplier.

Using the same mathematical notation, simple examples for one-to-one and the one-to-many relationship between lots without blurring are depicted in Fig. 1 and Fig. 2. An example with a many-to-many relationship that includes blurring of lots is given in Fig. 3. By following the edges of the graph, the lots that are built in other lots can be determined easily, i.e. for lot L_3^4 parts of the lots $\{L_1^1, L_1^2, L_2^2, L_2^3\}$ may be included, or for lot L_3^5 parts of the lots $\{L_1^2, L_1^3, L_2^3, L_2^4\}$ may be included.

The missing relation *pred* between L_3^4 and L_3^5 indicates that the buffer of machine 3 was empty before the production of lot L_3^5 started so no blurring of lots could have occurred. The distance between lots can be used as a basis to determine the probability a certain part of a lot is used in another lot. A more precise determination of probabilities would also require to consider buffer levels during manufacturing but that is beyond the scope of this work.

3.2 Implementation

We have implemented a prototypical software solution to we can see how a given tracking system could use the graph database OrientDB[1] in order to implement common operations. We have chosen OrientDB, since it is a multimodal NoSQL (which stands for not only SQL) database that combines properties of document-oriented and graph databases [7]. It allows users to define graph structures using

[1] https://orientdb.com/

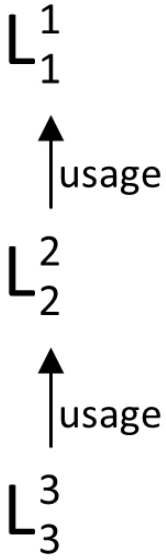

Fig. 1: One-to-one relationship between lots. This is the simplest relationship that we can find in the manufacturing industry. In principle, a solution for dealing with this type of relationship is quite simple and can be implemented efficiently in a wide range of database systems, including those databases making use of the traditional model.

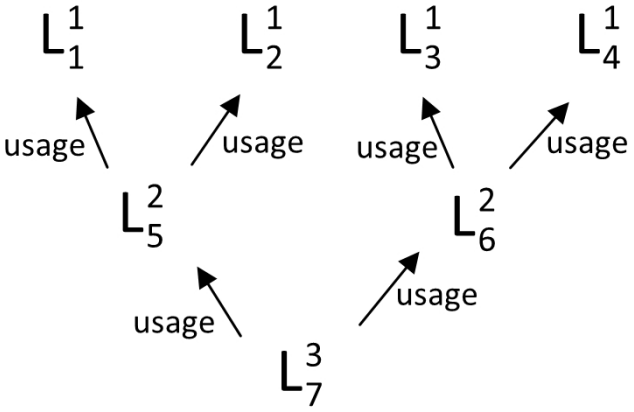

Fig. 2: One-to-many relationship between lots. It is a fairly common type of relationship in the manufacturing industry. The content of one batch is distributed or transformed in turn into other batches. This type of relationship can also be implemented efficiently in most current database systems, including relational systems.

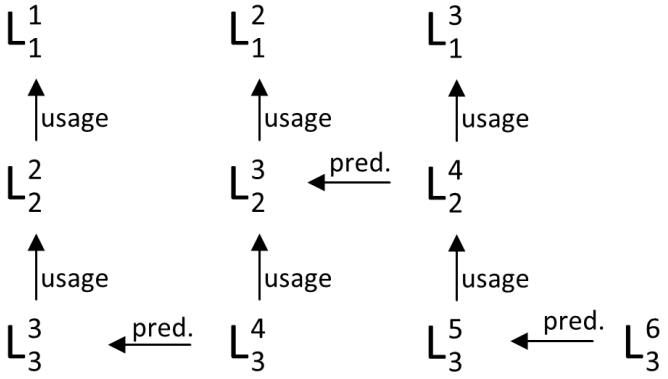

Fig. 3: Many-to-many relationship between lots i.e. blurred lots. It is a complex, yet a fairly common, type of relationship in the manufacturing industry. It takes place when many lots evolve or are distributed among other lots making their trace very difficult to follow. We hypothesize that only graph database systems can model and implement a solution efficiently. This type of relationship represents the central problem around which our research work revolves.

concepts for nodes and edges but also allows us to append complex data to nodes in the form of documents. Nodes and edges can have attributes (e.g. edge weight or similar).

Moreover, inheritable classes can be defined for the nodes and edges which can be extended flexibly. The query language is an adapted form of SQL and it is very intuitive and easy to use. In fact, the queries can be easy expressed through a user interface as represented in Figure 4.

It is important to note that the smallest unit that can be loaded and saved from the database is a record. OrientDB distinguishes between four types of records: A record can be a document, a RecordBytes (BLOB), a node (Vertex) or an edge (Edge). In our specific case, working with Vertex and Edge data types is enough. But as future work, we can envision that it would be useful to use the data type document to offer explicit support when dealing with situations that have been already before. Therefore, choosing OrientDB also represents an efficient and scalable alternative, which is why an OrientDB-based implementation has been set up.

The rationale behind the election of this solution is that, compared to implementations based on relational database systems, using a graph database leads to an efficient and scalable solution in which the problem at hand can be modeled easily [14]. Maybe the most clear example is the action of traversing graphs. The fact is that traversing graphs modeled in relational database systems would require to write nested and recursive queries that are difficult to maintain and provide bad comparable performance.

The following SQL source code shows an brief example of how to model the problem using a traditional relational database approach to illustrate our viewpoint.

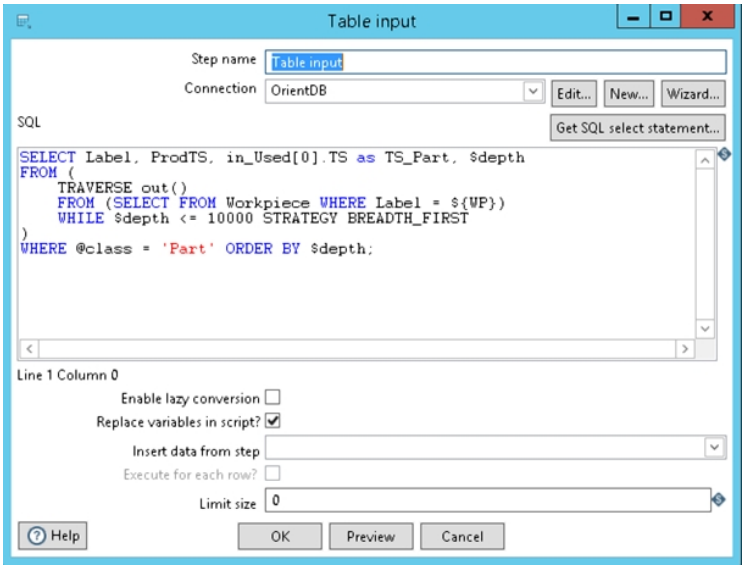

Fig. 4: OrientDB graphical interface that allows users to design and launch queries related to the distribution and/or evolution of the different lots through the manufacturing process.

```
create table workpiece (
  ts int,
  wp_name varchar(50)
)

create table partlot (
  ts int,
  lot_name varchar(50)
  )

insert into workpiece values (1,'WP1')
insert into workpiece values (2,'WP2')
insert into workpiece values (3,'WP3')
insert into workpiece values (4,'WP4')
insert into workpiece values (5,'WP5')
insert into workpiece values (6,'WP6')

insert into partlot values (1,'L1')
insert into partlot values (2,'L2')
insert into partlot values (5,'L3')

select * from workpiece
select * from partlot
```

```
24
25 select
26   l.lot_name,
27   ( select COUNT(*)
28     from workpiece p
29     where p.ts  >= l.ts
30       and p.ts < 5 -- query for workpiece WP6
31   ) distance
32 from partlot l
```

This source code shows how workpieces and their relations (in the form of belonging to a lot) can be modeled. We need a table for workpieces and a different one for lots, Then by means of insertions we can store the corresponding data. Finally, it is necessary to design a query to get the results. However, when using our approach we can appreciate several advantages. In this way we can represent the problem in a very natural way and proceed to the implementation of a solution that is both fast and efficient.

It is important to remark that traversing a graph is the act of visiting the nodes in the graph. For graph databases, traversing to nodes via their relationships can be compared to the join operations on the relational database tables.

The great advantage of this solution is that the operation for traversing a graph is much faster than the traditional joins from the relational databases world. The reason is that when querying the database with a traversal, the model only considers the data that is needed without taking into account any kind of grouping operations on the entire data, as it happens in traditional relational databases.

3.3 Use Cases

Based on the implementation that has been described above, such as the graph-based approach is considered to have a positive impact on the daily operations of the manufacturing industry. In particular, the depth calculation, i.e. the distance between lots, could be used as a basis for calculating the dependency path of a part in a product.

In order to illustrate our approach, we show here some examples of queries that our system can support. The following code shows how to calculate the shortest path between nodes #33:27050 and #29:2667 regardless of edge direction. Note that the unwind directive is useful while performing the aggregation of the nodes in the path.

```
1 SELECT expand(path) FROM (
2   SELECT shortestPath(#33:27050, #29:2667)
3            AS path UNWIND path
4 );
```

Figure 5 shows the resulting graph from the calculation of the shortest path between nodes #33:27050 and #29:2667 regardless of edge direction.

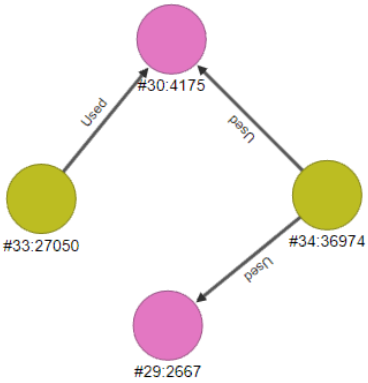

Fig. 5: Result of calculating the shortest path between nodes #33:27050 and #29:2667 regardless of edge direction.

OrientDB solution already implements the so-called search with the so-called BREADTH FIRST, which returns the real depth in the graph. It is trivial to see that the greater the depth, the less likely it is that each lot has been incorporated into the end product. The following code results in Figure 6, and it shows us how to traverse the graph in the direction of the edge direction from node #33:27050 to a depth of 10.

```
1 TRAVERSE out()
2 FROM #33:27050 WHILE depth < 10 STRATEGY BREADTH_FIRST;
```

Fig. 6: Result of traversing the graph from node #33:27050 to a depth of 10.

An example of SQL-like query to get the shortest path between lot 33 : 27050 and 29 : 2667 can be easily written as:

```
SELECT expand(path) FROM (
SELECT shortestPath(#33:27050, #29:2667, 'OUT')
              AS path UNWIND path);
```

The result of this query is depicted in Fig. 7. The number of edges between the lots gives a basic notion of probability that parts of lot 29 : 2667 are built-in parts of lot 33 : 27050. As we have seen before, the higher the number of nodes visited by the query, the lower the probability that each of the specific nodes is affected by the error or failure that the operator or quality engineer are looking for.

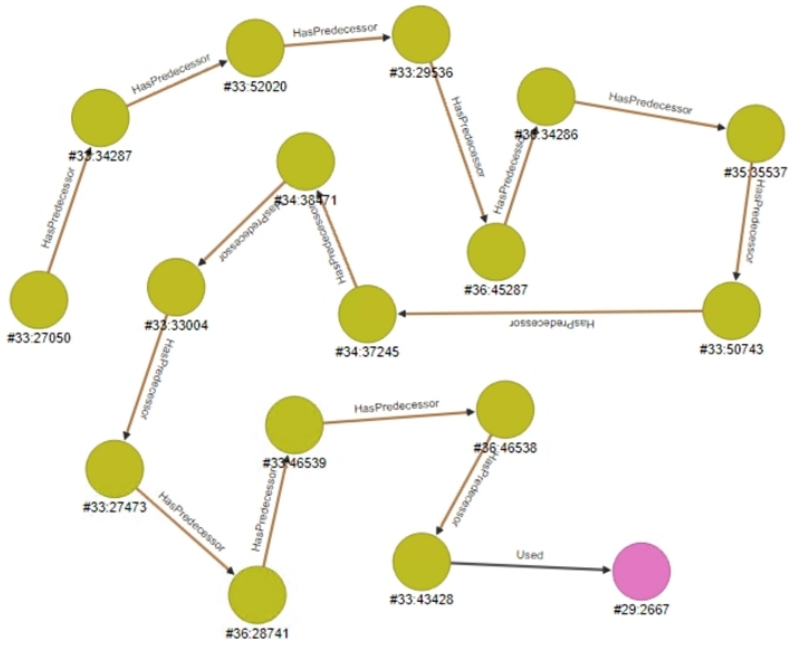

Fig. 7: Result of calculating the shortest path between the nodes #33:27050 and #29:2667.

4 Discussion

During the last years, the ever-increasing technical literature concerning graph-based research clearly shows us that one of the current main challenges of computer science consists of finding proper ways to model the knowledge generated in a specific domain [10].

With this regard, Knowledge Graphs [21] have gained some popularity recently. In this respect, we believe that the manufacturing domain fits very well. This opinion is shared with other researchers who have already been working in this direction to be able to explore the knowledge graph by the industry [16].

In this context, some software solutions being able to facilitate the handling of product-related production for manufacturing enterprises in the industry sector are becoming more popular [11]. Some of the most popular models to represent domain knowledge are based on some kind of the so-called knowledge graphs [8]. Knowledge graphs are intended to represent entities and relationships between entities within a particular universe of discourse. The advantage of using this kind of knowledge representation is that it is easy to understand for both humans and computers at the same time.

The origin of knowledge graphs is the combination of the knowledge bases with the inference engines, what is also referred to as Knowledge-Based Systems in the literature. One of the first approaches that one could think about would be a system that represents knowledge with uncertainty using a set of rules to the that they are given a certainty factor. However, these types of systems based on rules are not very robust, so they have been progressively replaced by another type of more efficient system, being at present the Bayesian networks the most used way of representing and inferring interesting knowledge with uncertainty currently.

Using this idea of knowledge-based systems, Google launched the so-called Google Knowledge Graph [6] several years ago. This graph seeks to have a universal domain, representing all existing entities and relationships between them, without being subject to a single context. Having such a domain broad has great complexity, and is incomplete. It is complex to introduce a new entity and determine what other entities it relates to and what type of relationship unites them.

This task of generating new entities is not currently automated but is the users themselves who introduce new entities to the network and determine what other entities relate to each other and how. It is expected that within the next years, this technology will be developed and it is possible that some of its foundations can be applied in the manufacturing domain.

5 Conclusions and Future Work

In this work, we have presented the design of a general solution for tracking each component that comprises manufacturing products through diverse stages of the production chain. Our solution has been modeled using graph databases as opposed to most existing solutions that use relational databases. In this way, our approach provides an improved level of both transparency and traceability, since we think that a graph is the natural way to model a problem involving dependencies of this kind.

Transparency is given by the fact that it is for a human operator to see what actions have been performed during the manufacturing process. Traceability is

given by the fact that it is possible to monitor the whole development process followed by the manufacturer. In addition, these two properties are assumed to facilitate the analysis of all manufacturing products as well as the capability to look for final products that could be affected by some specific problems. In this way, our approach presents more efficient modeling and querying mechanisms than traditional approaches based on relational databases.

As future research work, we envision that graph databases hold a lot of unrealized potential in the next years as companies will be moving towards approaches being able to better data analysis and exploration. More specifically, we think that there is a number of pending challenges. For example, it is important to further investigate whether it is possible to document past errors, to facilitate the task of discovering future errors. To do this, it is necessary to investigate whether it is possible to associate documentation with certain error patterns, which have to happen recurrently during the manufacturing process in the past. We also believe that the use of Knowledge Graphs can play a determining role, since these graphs allow not only to model the problem in a natural way, but also to reason with the data we work with.

Acknowledgements

This work has been supported by the project **AutoDetect** (Project No. 862019; Innovative Upper Austria 2020 (call Digitalization)) as well as the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

References

1. M. El Abri. *Probabilistic relational models learning from graph databases. (Apprentissage des modèles probabilistes relationnels à partir des bases de données graphe)*. PhD thesis, University of Nantes, France, 2018.
2. R. Angles, M. Arenas, P. Barceló, A. Hogan, J. L. Reutter, and D. Vrgoc. Foundations of modern query languages for graph databases. *ACM Comput. Surv.*, 50(5):68:1–68:40, 2017.
3. Phininder Balaghan. *An exploration of graph algorithms and graph databases*. PhD thesis, University of Hull, Kingston upon Hull, UK, 2019.
4. Yulian Cao, Wenfeng Li, Wei Song, and W. Art Chaovalitwongse. Collaborative material and production tracking in toy manufacturing. In *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Whistler, BC, Canada, June 27-29, 2013*, pages 645–650, 2013.
5. James Cheng, Yiping Ke, and Wilfred Ng. Efficient query processing on graph databases. *ACM Trans. Database Syst.*, 34(1):2:1–2:48, 2009.
6. Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and YAGO. *Semantic Web*, 9(1):77–129, 2018.

7. D. Fernandes and J. Bernardino. Graph databases comparison: Allegrograph, arangodb, infinitegraph, neo4j, and orientdb. In *Proceedings of the 7th International Conference on Data Science, Technology and Applications, DATA 2018, Porto, Portugal, July 26-28, 2018.*, pages 373–380, 2018.
8. Mikhail Galkin, Sören Auer, and Simon Scerri. Enterprise knowledge graphs: A backbone of linked enterprise data. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016, Omaha, NE, USA, October 13-16, 2016*, pages 497–502, 2016.
9. Mohamed Ghazel, Armand Toguyéni, and Michel Bigand. A semi-formal approach to build the functional graph of an automated production system for supervision purposes. *Int. J. Computer Integrated Manufacturing*, 19(3):234–247, 2006.
10. Jorge Martínez Gil. Automated knowledge base management: A survey. *Computer Science Review*, 18:1–9, 2015.
11. Longlong He and Pingyu Jiang. Manufacturing knowledge graph: A connectivism to answer production problems query with knowledge reuse. *IEEE Access*, 7:101231–101244, 2019.
12. Jan-Patrick Hülß, Bastian Müller, Daniel Pustka, Jochen Willneff, and Konrad Zürl. Tracking of manufacturing tools with cylindrical markers. In *The 18th ACM Symposium on Virtual Reality Software and Technology, VRST 2012, Toronto, ON, Canada - December 10-12, 2012*, pages 161–168, 2012.
13. Virginia Ivanov, Maria D. Brojboiu, and Sergiu Ivanov. Applications of the graph theory for optimization in manufacturing environment of the electrical equipments. In *28th European Conference on Modelling and Simulation, ECMS 2014, Brescia, Italy, May 27-30, 2014*, pages 153–158, 2014.
14. Salim Jouili and Valentin Vansteenberghe. An empirical comparison of graph databases. In *International Conference on Social Computing, SocialCom 2013, SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, Washington, DC, USA, 8-14 September, 2013*, pages 708–715, 2013.
15. Friedemann Kammler, Simon Hagen, Jonas Brinker, and Oliver Thomas. Leveraging the value of data-driven service systems in manufacturing: a graph-based approach. In *27th European Conference on Information Systems - Information Systems for a Sharing Society, ECIS 2019, Stockholm and Uppsala, Sweden, June 8-14, 2019*, 2019.
16. Dmitry Kudryavtsev, Tatiana Gavrilova, Irina A. Leshcheva, Alena Begler, Miroslav Kubelskiy, and Olga Tushkanova. Mind mapping and spreadsheets in collaborative design of manufacturing assembly units' knowledge graphs. In *Joint Proceedings of the BIR 2018 Short Papers, Workshops and Doctoral Consortium co-located with 17th International Conference Perspectives in Business Informatics Research (BIR 2018), Stockholm, Sweden, September 24-26, 2018.*, pages 82–93, 2018.
17. Marco Livesu, Daniela Cabiddu, and Marco Attene. slice2mesh: A meshing tool for the simulation of additive manufacturing processes. *Computers & Graphics*, 80:73–84, 2019.
18. Kuo-Ren Lou and Lu Wang. Optimal lot-sizing policy for a manufacturer with defective items in a supply chain with up-stream and down-stream trade credits. *Computers & Industrial Engineering*, 66(4):1125–1130, 2013.
19. Jorge Martinez-Gil, Reinhard Stumpner, Christian Lettner, Mario Pichler, and Werner Fragner. Design and implementation of a graph-based solution for tracking manufacturing products. In *New Trends in Databases and Information Systems. ADBIS 2019. Communications in Computer and Information Science*, volume 1064, pages 417–423. Springer Cham, 2019.

20. Meysam Minoufekr, Anass Driate, and Peter W. Plapper. An iot framework for assembly tracking and scheduling in manufacturing SME. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2019 - Volume 2, Prague, Czech Republic, July 29-31, 2019.*, pages 585–594, 2019.
21. Natalya Fridman Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019.
22. Martin Ringsquandl, Steffen Lamparter, and Raffaello Lepratti. Graph-based predictions and recommendations in flexible manufacturing systems. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, October 23-26, 2016*, pages 6937–6942, 2016.
23. Ian Robinson, Jim Webber, and Emil Eifrem. *Graph databases*. O'Reilly Media, Inc., 2013.
24. Mohit Singh, I. A. Khan, and Sandeep Grover. Selection of manufacturing process using graph theoretic approach. *Int. J. Systems Assurance Engineering and Management*, 2(4):301–311, 2011.
25. H. Wang, J. Yang, and D. J. Ceglarek. A graph-based data structure for assembly dimensional variation control at a preliminary phase of product design. *Int. J. Computer Integrated Manufacturing*, 22(10):948–961, 2009.
26. Jens Weise, Steven Benkhardt, and Sanaz Mostaghim. A survey on graph-based systems in manufacturing processes. In *IEEE Symposium Series on Computational Intelligence, SSCI 2018, Bangalore, India, November 18-21, 2018*, pages 112–119, 2018.