Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2023) 000–000

www.elsevier.com/locate/procedia

34th CIRP Design Conference

Unveiling Causality in Stateful Enterprise Knowledge Graphs: An Exploration of Process-Driven Object Relationships

Marvin Schobert[a,*], Matúš Mala[b], Markus Herhoffer[c], Samuel Farag[d], Jörg Franke[a]

[a]Institute for Factory Automation and Production Systems, Friedrich-Alexander-Universität Erlangen-Nürnberg, Egerlandstr. 7, Erlangen 91058, Germany
[b]Flowable Germany GmbH, Schloßstr. 70, 70176 Stuttgart, Germany
[c]exentra GmbH, Löwenstr. 11, 85276 Pfaffenhofen an der Ilm, Germany
[d]Frends Technology Oy, Tekniikantie 14, Espoo 02150, Finland

* Corresponding author. Tel.: +49 9131 85-28972; fax: +49 9131 302528. *E-mail address:* marvin.schobert@faps.fau.de

Abstract

The exponential growth of enterprise data production demands innovative approaches for efficient information management across highly interconnected domains. While structured data resides comfortably in dedicated systems, handling unstructured data like raw data, emails and documents often involves distributed sources with challenging holistic comprehension of linking, context, and evolution. Graph technology holds promise in establishing adaptable connections among decentralized data sets, transcending deterministic limitations. However, leveraging graph databases comes with challenges. Schema standardization, crucial for framing complex business processes, faces complexities due to evolving data interrelationships. Finding a balance between standardization and changing requirements is a difficult undertaking. This paper proposes a two-fold solution. First, we introduce a reference semantics inspired by the Asset Administration Shell (AAS), that decouples stateful object descriptions from its headers. This allows flexible object associations and historical state versioning while maintaining manageability. Second, we leverage the Business Process Model and Notation (BPMN) for Low Code graph manipulations and improve traceability by involving transactional relationships. Through application to the publicly available Northwind data set using neo4j and Flowable BPM, we illustrate our methodology's efficacy and the seamless integration of type safer data science into dynamic knowledge graphs, contributing to the future of holistic enterprise data management.

© 2024 The Authors. Published by ELSEVIER B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 34th CIRP Design Conference

Keywords: Knowledge Graphs; BPMN; Provenance; neo4j; Design informatics

1. Introduction and motivation

The data-driven culture and trend towards workflow automation are becoming concepts that are increasingly being incorporated into corporate strategies: According to McKinsey & Company, by 2025 most employees will use data to optimize nearly every aspect of their work and smart workflows will be seamlessly integrated between humans and machines [1]. Even though the growth of data is driven by unstructured or semi-structured sources, they claim, that the majority of practical and accessible data continues to be structured through the utilization of relational database tools. Since data is embedded in most decisions, interactions and processes already, a notable challenge faced by engineers today lies in the exploration of datasets, establishing relationships, and joining information, being a time-consuming, non-scalable, and error-prone manual and bespoke processes [1]. Recognizing this industry challenge, our paper aims to contribute to the facilitation of these data engineering processes as well as to better integrability into hyperautomation pipelines [2]. This approach empowers teams to query and comprehend relationships within unstructured and semi-structured data efficiently, accelerating the development

2212-8271 © 2024 The Authors. Published by ELSEVIER B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 34th CIRP Design Conference

of new AI-driven capabilities and fostering the discovery of innovative relationships in the data landscape.

2. Relevant Basics

This chapter will briefly introduce the related background and approaches. First, we will explain the importance of graphs concerning growing data volumes and causal inference, before we then will explain the fundamental influences on our concept.

2.1. Representation of enterprise data using graphs

As described by Hogan et al., data with strong mutual interaction and specifications that change over their lifetime often evolve into a graph-like representation intrinsically. Instead of involving several identifiers and cross-references, the usage of graphs in the common forms of the Resource Description Framework or a Labelled-Property Graph containing nodes, edges and properties are advantageous. [3]

With knowledge being a resource of high value for companies, the establishment of an Enterprise Knowledge Graph is an approach to connect and evaluate structured and unstructured data relevant for the enterprise and distributed in different sources [4]. We understand a Knowledge Graph as a "*graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities, […] potentially enhanced with representations of schema, identity, context, ontologies and/or rules"* [3]. We refer to a schema being a conceptual data model describing the logical structure of data and to an ontology as a semantically enriched structure specifying certain domain knowledge [5]. For further graph science explanations, we recommend the publication by Hogan et al. referenced above [3]. We share the Open World Assumption and assume, that the graph well never be complete in describing reality. Consequently, there is an interest in extracting knowledge that is not explicitly mentioned. Hernán et al. classify data into the fields of description (quantitative summary of certain features), prediction (map input features to outputs) and counterfactual prediction (predict outputs through causal inference, if input features had been different) [6]. We refer to causality "as the influence, by which one event, process, state or object contributes to the production of another event, process, state or object" [7]. Since, according to Hernán, expert knowledge is necessary to formulate and answer causal questions in more complex systems, we would like to briefly mention some techniques for graph analytics to get descriptive knowledge (see Iosup et al. [8] for further reference):

- *Centrality* finds a graph's most important nodes or edges
- *Community detection* aims to identify above-average interconnected groups of entities (or sub-graphs)
- *Connectivity* reveals the reachability and resilience of entities by determining how well-connected it is
- *Node similarity* compares, how similar two entities are based on their structure and connections
- *Path finding* finds the shortest connection between two or more entities taking edge weights into account.

2.2. Data versioning and traceability

For tracking the influences between entities, keep track of change and possibly do what-if-analysis, we could not find any satisfactory out of the box solutions in the literature for knowledge graphs. For the consideration of object and state, we endorse a blog post published on Medium and based on the neo4j NODES2019 conference contribution by Lazarevic [9]. They follow the approach to separate the object from its state and link those by a time-dependent relationship. The Asset Administration Shell (AAS) similarly separates objects into headers and bodies consisting of several sub models that aim to standardize the representation of assets (such as physical objects, software or documents) in the context of the manufacturing industry [10]. As an industry-neutral standard, it ensures cross-manufacturer interoperability by defining a uniform data model, characteristics, properties, behaviours, functions, relationships and other relevant asset information. To build representations of entities, people and processes involved in the creation of data objects, the PROV data model provides a generic model for provenance [11]. Entities (nodes) are derived from other entities, generated and/ or used by activities and are attributed to agents (e.g., people, software or organizations). VDI/VDE 3682 on the other hand structures the interaction of assets involving processes as follows: Products, energies or information (inputs) are converted to new products, energies or information (outputs) by a process operator, which is executed by a technical resource [12]. In this way, the influences on the creation of each asset can be traced.

2.3. Transactions based on the BPMN standard

We claim that every database interaction and thus change in data object's states is motivated through a (possibly less transparent) process context. Thereby we consider atomic units of work as "transactions" constructed of several operations, while a process describes a series of events, transactions and activities to produce certain results. Since software and the orchestration of distributed systems is becoming increasingly process-oriented, the design standardized of standardized to-be processes is favourable [13]. As processes are driven by domain experts and their activities serve as the connecting elements, a process definition language must possess qualities such as intuitiveness, standardization, completeness, precision, and the ability to easily transform into executable software. The Business Process Model and Notation (BPMN, [14]) allows for the modularization of complex behavioral models. It facilitates a hierarchical breakdown into sub processes featuring fundamental activities, conditions, transitions, and events. Furthermore, not only business processes, but also technical and IT-heavy processes including system or database transactions can be expressed [15]. The Process-Driven Approach (PDA) suggests an BPMN-based architecture, where the business logic is executed within a process engine and external services can be accessed using a predefined service contract describing its interface structure via an integration layer [13]. In contrast to alternative process notations like the event-driven process chain, BPMN stands out due to its standardization and the support for direct workflow execution

within a process engine [16]. Considering its advantages, we strive to use BPMN as the lingua franca for the traceability of transactions and their causal effects. We would also like to mention the supplementary Case Management Model and Notation (CMMN, [17]) standard, which supports processes that are adapted at runtime and therefore unpredictable, non-repeatable, weakly structured and knowledge-intensive.

3. Need for action, structure and methodology

To systematically collect and merge data in a purposeful manner for smarter applications, we claim that it is necessary to relate processes, business objects and transactions. In addition, it must be ensured that all activities contained in processes are automatically documented and related to the affected objects. Furthermore, we claim that it is advantageous to keep multiple versions of objects and their connections and to make their causal influences traceable. These properties would lead to increased data quality resulting in increased transparency and informative value regarding causality and thus influence on decisions. Although there are partial approaches, to our knowledge a comprehensive solution has not yet been satisfactorily proposed. To conduct our investigation, we followed the Design Science Research in Information Systems methodology as proposed by Hevner et al. [18]: We propose a reference model and method (*Design as an Artifact*: Chapter 4) in order to enhance traceability and meaningfulness (*Problem Relevance*: Chapters 1 to 3), strictly following the separations of concerns and divide- and-conquer principles (*Research Rigor*: Chapter 4). We conduct an expressive and openly reproducible experiment for its discussion (*Design evaluation*, chapters 5 and 6) and provide clear and verifiable contributions in the area of the design artifact (*Research Contributions*: Chapter 6 and 7). By writing this paper, we share results (*Research Communication*), illustrate limitations and will finally adapt future steps based on feedback and discussions (*Design as a Search Process*).

4. Reference structure and process-driven manipulation

Our approach is mainly influenced by adapting the ideas communicated in four different sources: We take the concept of dividing objects in (semi) standardized sub models as the AAS does [10]; we enable state versioning as suggested by Lazarevic [9] and keep track of activities as proposed by VDI/ VDE 3682 [12] as well as the PROV data model [11].

4.1. Reference structure

Our proposition aims to enable to trace the correlation between business objects and activities at a specific point in time. We therefore endorse, similar to the AAS, to distribute systems into sub-models having their own object node, each representing a set of certain properties, features, parameters or characteristics. The reference structure in form of a labelled property graph is shown in Fig. 1 and described below. Chapter 5 later will show the application to an order fulfilment use case.

Each object is represented by a unique identifier or locator (“Object head”) and its descriptive content having a certain validity (“Object state”). Thus, a head can contain multiple states, whereby only one state is valid at a time. Heads can also have relations to other object heads with certain validity rules (e.g., time- or revision-based). Separating systems in individual parts leads to reduced maintenance effort and better readability.

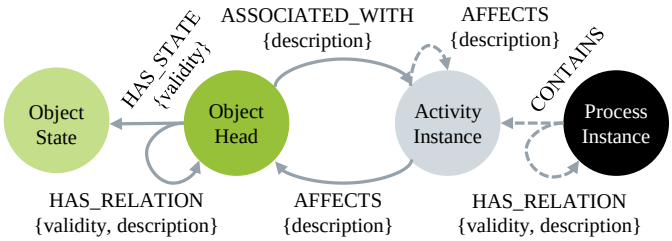

Fig. 1: Reference structure as labelled property graph with induced processes

To incorporate the influence of activities and processes, we propose a combination of the data models mentioned before: An activity instance can affect several objects or other activities, be assigned to process-hierarchies (eventually making use of the previously mentioned graph analytics methods) and multiple objects can be involved in the execution of the activity. In accordance to VDI/ VDE 3682, we demand that objects are associated with an activity for execution (e.g. resources, auxiliary objects) and that the effects on objects are explicitly described in order to improve causality. With regards to naming, we endorse the use of natural language combined with the paradigms for expressive end to end process modelling according to Silver [19]. We propose the following rules:

- The identifier or header of an object is unique after its creation and remains unchanged over its lifetime.
- An object has exactly one valid state at a point in time
- An object can have any number of relations to other objects at one point in time
- A change in the object state is always caused by an activity. The activity takes a duration, while the state creation occurs at a discrete point in time. We refer to a transaction as activities that affect objects at a point in time

4.2. Process-driven data manipulations

Even though the processes can be executed by different resources, we endorse the orchestration of technologies using BPMN executed within a process engine according to the specifications of the process-driven approach [13]. This enables business and IT experts to work closely on the same artifacts and thus close the well-known gap. Since process execution ultimately takes place in distributed environments (referring here towards “hyperautomation” [2]), we require an open architecture for various interfaces (see Fig. 2).

If process execution takes place within a business process management suite (BPMS), Low-Code functionalities should be provided by the software built-in or explicitly created developers. Regardless of the system from which the processes are executed, we require the creation of activity logs in tandem with any updates. Those can be carried out either synchronously (e.g., HTTP-REST) or through event-driven architecture with pre-defined formats (e.g., Apache Kafka).

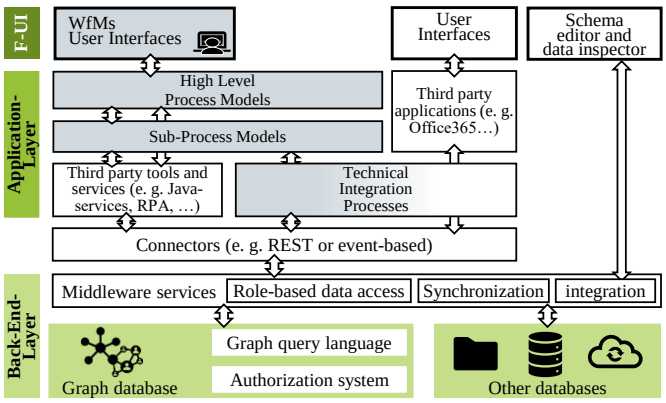

Fig. 2: Architecture of process-driven data manipulations

For the purpose of type safety and simplified interaction, we recommend the use of middleware software or integration platforms. Thus, data schemata and validations can be defined (e.g., Apollo GraphQL; we want to prevent users from being able to perform direct manipulations within the graph database) and migration and synchronization processes between various data sources handled (e.g., Hasura for relational databases). In this way, cross-system processes can also be tracked.

5. Example case study/ experiment implementation

The case study we would like to present makes use of the Northwind sample database, that is shipped along with the Microsoft Access application under the Microsoft Public License and has served as the basis for different tutorials. The database stores the sales information of a fictional company named "Northwind Traders", engaged in the global import and export of specialty foods. Our case study will examine the following, sample problems of descriptive insights, serving as blueprints by being transferable to similar structural queries:

- System state at a certain historical point in time
- Trace of state development over time
- Recommendations based on certain activities
- Statistical analysis on activity occurrence
- Trace of workflow progression and different versions
- Analysis of influences where activities have led to success

5.1. Project set up and stack description

The concept explained in chapter 4 is set up making use of Flowable BPM for process execution, which features a process engine as well as a design tool for low-code creation of processes, services and forms and the so called “GRANDstack” for database management. GRAND thereby is an acronym for the frameworks GraphQL as query language, react for front-end visualization, Apollo.js as middleware based on node and finally a neo4j-instance as database. For the database we used the Aura-DB-Version, which is a (limited) free, online-hosted instance provided by neo4j. The Apollo middleware is running in a local docker environment. With regard to the BPMS, we have collaborated with Flowable and used its similar named product “Flowable Work” in its Spring Boot version and back end in a docker container.

5.2. Data set preparation

For the import to neo4j-Aura-DB, we have used the official guide on how to import data from relational databases to neo4j [20]. Slightly simplified, the data represents 830 orders, 91 customers, 77 products within 8 categories, 29 suppliers, 3 shippers and 9 employees among others. The migration of the master data to the proposed format was realised by a simple migration process within the process engine, which temporarily stored each object and then created it via a service task in the new format using GraphQL. Orders are not yet considered, as we regard them as dynamic objects generated by transactions within a dedicated order fulfilment process. In order to enable type safe GraphQL-processing, the data formats were defined in schemata within the middleware Apollo. Therefore, explicit data types were created for each object head and state, being each connected via edges with time-dependent properties. Fixed relations were added between object heads according to Northwind’s ontology. All object heads inherit from an abstract type "business object" enabling flexible relations in between.

5.3. Experiment set up and execution

Having the master data set up as proposed, the following will describe the conducted experiment. Based on the data structure, we have interpreted an underlying business process that spans from the order’s creation to shipment (see Fig. 3a).

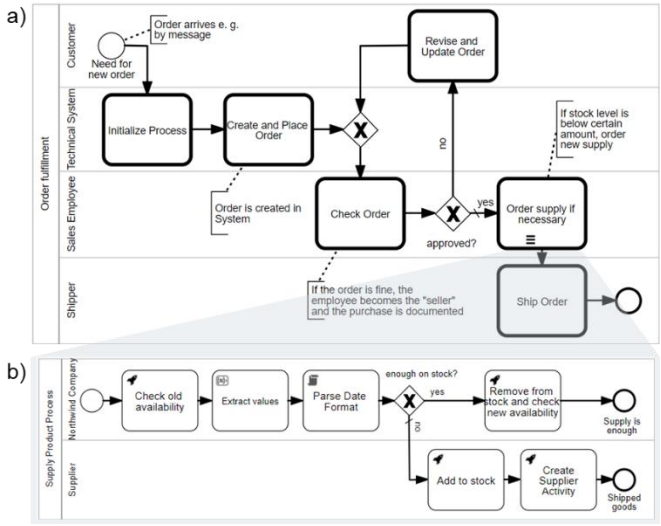

Fig. 3: a) Order-fulfillment-process, b) integration process for product supply

For simplicity, we assume that each of the 830 orders would have gone through this (or a similar) process in the past. The process shown depicts our simplified order-to-shipment-process on its highest hierarchy level and features six call activities with separated sub processes for technical execution. In our scenario, the receipt of the order represents our system’s perceptual horizon (scope of truth), as no information is available as to how the customer came to place the order. As input, we take the fictional orders existing in the Northwind graph, which run through the process one by one. Since the data doesn’t provide information about any iterative order clarifications (e.g., changes in order quantities or spelling

errors), we assume, that one out of ten orders won't be approved immediately and will need revision by the customer. For simplicity, if the stock level of any of the products is less than 100, 300 units are reordered from the supplier. Each sub process will take care about the technical database interactions (see Fig. 3b). Simulating 30 random orders this way results in the graph shown in Fig. 4, while the processing of all 830 orders results in a total of 9470 nodes and 18200 relations.

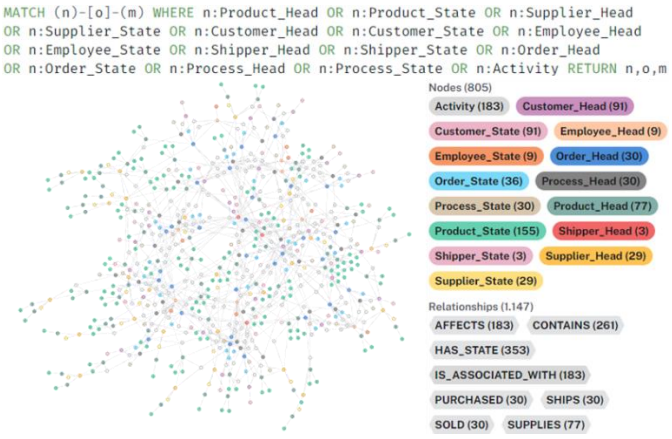

Fig. 4: Data set after thirty orders

6. Evaluation and discussion

In the following, we want to demonstrate sample queries based on the blueprint problems mentioned in chapter 5. We will explain the solutions and approaches by dividing the results into state-related and state-activity-combined insights.

6.1. State-related insights

Firstly, we want to show the status of a single order at a point in time by passing a validity range to the query (see Fig. 5). This can be queried here via Cypher, graph visualization tools or the GraphQL interface given by Apollo providing a JSON format result that can be incorporated by other services.

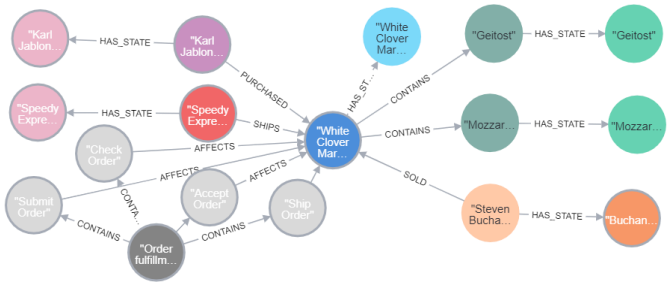

Fig. 5: Order "White Clover Market" at a point in time and affecting activities

Even more interesting is tracing the development of selected object states over the course of time: Stock levels for the exemplary product types Chai and Pavlova were queried, exported and charts visualized in Excel (see Fig. 6), which enables statistical evaluation. We conclude the insight, that the range of Pavlova's 300 reordered products has shortened from 200 days at the beginning to 130 days in the winter of 1997/98.

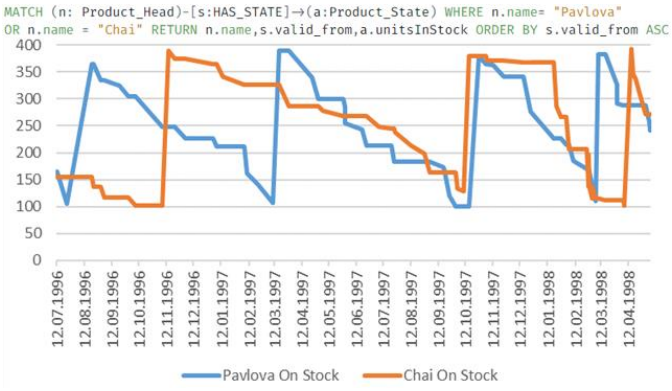

Fig. 6: Analysis of stock development of selected products over time

Additional queries with a similar structure could explore the consistency of product demand or the frequency of purchases made by different customers (XYZ-Analysis). Of course, these conclusions can also be reached in conventional ways, but our approach has the advantage that no further data transformations or preparations need to be carried out after the one-off query.

6.2. State and activity combined insights

Based on the activities, there are large potentials for recommendation-based optimization or root cause analysis. To stay with the scenario from above, the frequency of reorders can directly be determined based on the "add to stock" activity. Raclette has a relatively high frequency of reorders, indicating that other product supply strategies could be developed. In a similar fashion, we can examine the products contained in orders with the highest reject/ update to accept order ratio (see Fig. 7), the most experienced regarding a certain skill based on activity success rate or busiest employee considering durations of tracked activities, useful for resource and project planning.

Fig. 7: Quantity of matching products in the revised orders

We can further trace a process flow via the activity nodes only and calculate lead times for different variants. The development of the state can be tracked either by varying the time range or by graphically revealing related nodes. Since activities are also time-dependent, past activities that caused, affected or even led to the inspected system state can be inspected and interpreted. This way, loops and successful "break out"-conditions within processes can be analyzed and ideally result in digitizable decision rules. Since the graph is fundamentally independent of the workflow execution environment, any considerations can be done across contexts. In our case, the seller can only commission delivery if products

are on stock. Although we already hard-coded this in our BPMN process model (thus being a form of decision rule), similar insights that can be interpreted and concluded from the graph would be highly valuable in less transparent cases. This would be particularly interesting for iterative, agile contexts (e.g., project management) in order to analyze which rework has frequently led to approvals or other outcomes. Additionally, dynamic knowledge can be deduced: In our example, we assume that customer satisfaction is 50% dependent on the ratio of "Ship Order" to "Revise and Update Order" activities and 50% on the ratio of delayed and timely deliveries, resulting in "Pascale Cartrain" being the happiest and "Laurence Lebihan" the most unsatisfied customer.

6.3. Concept discussion

Finally, we want to conclude our experiment evaluation with a short discussion mentioning the successes, challenges and conclusions of our concept. One of the key achievements is based on the traceability over time and context and lies in the ability to comfortably gain insights that are conventionally challenging and usually require associating information from various sources. Specifically, our approach allows us to answer questions pertaining to when, who, how (much) and the factors contributing to any changes. We also have achieved significant successes in leveraging automatic, process-driven data creation and activity logging within a graph database. Challenges remain with the automatic deduction of meaningful cause/ effects. Why changes occurred is still reliant mainly on the reader's interpretation, since the analysis queries we came up with facilitate the identification of interesting areas within the dataset, which aids in directing the reader's attention to relevant points. Besides that, we found the current manual input of every activity in the process to be labor-intensive and future work should focus on developing automations, such as subscribing to the start and end of activities, to streamline data entry into the database and use dedicated integration software. Another challenge poses the integration of information beyond the known scope of truth: Establishing connections between validated and invalidated information needs automated solutions, similarly as handling compatibility with legacy and third-party systems should be investigated. In conclusion, we are convinced that linking processes with business objects facilitates context-dependent knowledge creation, providing a nuanced understanding of the data. Vice versa we go even further and claim, that the clearer a process is discernible within data, the more meaningful statements can be concluded.

7. Summary and outlook

This paper addresses the increasing challenges associated with the ever-expanding landscape of enterprise data. Focusing on the efficient management of information and gaining better causal insights across interconnected domains, the study highlights the inherent difficulties in handling semi structured data combined with its development over time and context. The main contributions of the paper include introducing a novel reference schema, which decouples stateful object descriptions from its identifiers enabling flexible object associations and different state validity while maintaining manageability. Additionally, the paper leverages the BPMN standard in the context of process automation for performing changes within the data, enhancing traceability through trace link relationships. The concept was applied to the public Northwind dataset using neo4j and Flowable, demonstrating its efficacy in integrating type safer data science into dynamic knowledge graphs. These contributions advance the adaptability and traceability of enterprise data, as well as providing new opportunities for gaining descriptive insights that can be used for automatically deducing recommendations, business decision rules and causal inferences in future. Besides improved automatism for data creation and contextualisation, further research will conduct similar experiments on more agile case environments with an even higher need for more transparency and maintainability.

References

[1] McKinsey & Company. The data-driven enterprise of 2025; Available from: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-data-driven-enterprise-of-2025.

[2] Haleem A, Javaid M, Singh RP, Rab S, Suman R. Hyperautomation for the enhancement of automation in industries. Sensors International 2021;2:100124.

[3] Hogan A, Blomqvist E, Cochez M, D'amato C, Melo G de, Gutierrez C et al. Knowledge Graphs. ACM Comput. Surv. 2022;54(4):1–37.

[4] Gomez-Perez JM, Pan JZ, Vetere G, Wu H. Enterprise Knowledge Graph: An Introduction. In: Pan JZ, Vetere G, Gomez-Perez JM, Wu H, editors. Exploiting Linked Data and Knowledge Graphs in Large Organisations. Springer International Publishing; 2017, p. 1–14.

[5] Munir K, Sheraz Anjum M. The use of ontologies for effective knowledge modelling and information retrieval. Applied Computing and Informatics 2018;14(2):116–26.

[6] Hernán MA, Hsu J, Healy B. A Second Chance to Get Causal Inference Right: A Classification of Data Science Tasks. CHANCE 2019;32(1):42–9.

[7] Wikipedia. Causality (disambiguation); Available from: https://en.wikipedia.org/wiki/Causality_(disambiguation).

[8] Iosup A, Musaafir A, Uta A, Pérez AP, Szárnyas G, Chafi H et al. The LDBC Graphalytics Benchmark; 2020.

[9] Ljubica Lazarevic. Keeping track of graph changes using temporal versioning; Available from: https://medium.com/neo4j/keeping-track-of-graph-changes-using-temporal-versioning-3b0f854536fa.

[10] Reich J, Zentarra L, Langer J. Industrie 4.0 und das Konzept der Verwaltungsschale – Eine kritische Auseinandersetzung. HMD 2021;58(3):661–75.

[11] Yolanda Gil, Simon Miles, Khalid Belhajjame et al. PROV Model Primer, W3C Working Group Note 30 April 2013; Available from: https://www.w3.org/TR/2013/NOTE-prov-primer-20130430/.

[12] Verein Deutscher Ingenieure. VDI/VDE 3682; 2015.

[13] Stiehl V. Process-Driven Applications with BPMN. Cham: Springer International Publishing; 2014.

[14] Object Management Group. ISO/IEC 19510: Information technology - Business Process Model and Notation (BPMN): OMG; 2013.

[15] Freund J, Rücker B. Real-Life BPMN: Using BPMN and DMN to analyze, improve, and automate processes in your company. 4th ed. Berlin: Camunda; 2019.

[16] Schäffer E, Stiehl V, Schwab PK, Mayr A, Lierhammer J, Franke J. Process-Driven Approach within the Engineering Domain by Combining Business Process Model and Notation (BPMN) with Process Engines. Procedia CIRP 2021;96(57):207–12.

[17] Object Management Group. Case Management Model and Notation: CMMN; 2016; Available from: https://www.omg.org/spec/CMMN/1.1/.

[18] Hevner A, R A, March S, T S, Park, Park J et al. Design Science in Information Systems Research. Management Information Systems Quarterly 2004(28):75-105.

[19] Silver B, Fischli S. BPMN Methode und Stil. 2nd ed. Cody-Cassidy Press; 2012.

[20] neo4j. Tutorial: Import data from a relational database into Neo4. [November 29, 2023]; Available from: https://neo4j.com/docs/getting-started/appendix/tutorials/guide-import-relational-and-etl/