INTERNATIONAL DESIGN CONFERENCE – DESIGN 2024
https://doi.org/10.1017/pds.2024.210

The digital thread for system lifecycle management with a native graph database in a polyglot architecture

Nico Kasper [1,✉], Michael Pfenning [2] and Martin Eigner [3]

[1] SAP SE, IBU Industrial Manufacturing and Aerospace & Defense, Germany,
[2] SAP SE, Product Management SAP PLM, Germany, [3] EIGNER Engineering Consult, Germany

✉ nico.kasper@web4k.de

Abstract

The Digital Thread is a system that connects different phases of the product lifecycle and the related data across one or more companies in the supply chain. This work aims to develop a graph data model of the Digital Thread, in the context of the vision of polyglot persistence, that interconnects the different phases of the lifecycle and their corresponding data models, processes, and IT systems. This work proposes a Digital Thread Graph that integrates a Digital Model and a derived Digital Twin, using object and relation attributes for view creation and filtering while minimizing redundancy.

Keywords: digital thread, digital twin, product lifecycle management (PLM), change management, graph

1. Introduction

In the context of digital, data-driven business models, data plays a crucial role in securing competitiveness. This requires new concepts in System Lifecycle Management to gather and manage data and processes, such as the Digital Thread. This work aims to develop a "Digital Thread Graph" that connects the different lifecycle phases and corresponding partial models of the Digital Model and the Digital Twin using a polyglot persistence with multi-view capability.

System Lifecycle Management is the evolution of Product Lifecycle Management. It is an interdisciplinary (mechanical, electrical, software and service) approach to managing all phases of the product lifecycle, including system modeling, production planning and product operations (Eigner, 2021). It is not just about integrating data but linking it lifecycle-wide in a specific context, which is crucial (Kasper et al., 2023; Hedberg et al., 2020). The Digital Thread is a framework of protocols and standards for interoperable and context-based connection and communication of different partial models and processes realized in different IT systems across the lifecycle to streamline design, manufacturing, and operation processes and data (Eigner, 2021; Singh and Willcox, 2018). It creates a cross-system Authoritative Source of Truth, where the latest and valid state of the data is always known (Kasper et al., 2023; Bone et al., 2018; Kraft, 2015). The Digital Thread logically connects Configuration Items throughout the lifecycle, creating a network of semantic links. To accomplish this, graph technologies have emerged as a suitable solution (Kasper et al., 2023; Hedberg et al., 2020; Kwon et al., 2020).

2. Literature review

The Digital Thread connects the phase-specific partial models of the Digital Model and the Digital Twin throughout the system lifecycle (Kasper et al., 2023; Eigner, 2021). Figure 1 illustrates the idealized connection. However, in reality, the Digital Thread is distributed across the supply chain.

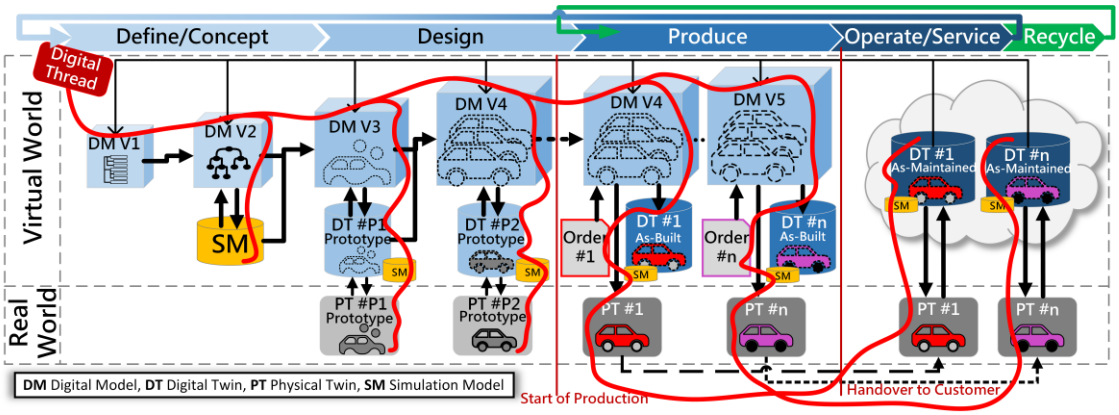

Figure 1. The System Lifecycle & the Digital Thread (Kasper et al., 2023)

The Digital Model is the complete product model at the type level, including all valid product variants. It includes all lifecycle phase-specific partial models of the type level (Kasper et al., 2023; Eigner, 2021; Pfenning, 2017). **The Digital Twin** is the uniquely assigned digital representation of a specific actual manufactured product at the instance level. It documents the product's configuration at the time of delivery and throughout its entire lifecycle, including selected characteristics, states, and behavior. The Digital Twin is a specific instantiation from the Digital Model and can be enriched or reduced with additional instance-related data from System Lifecycle Management and other IT systems (Kasper et al., 2023; Eigner, 2021; Singh and Willcox, 2018). **The Digital Thread** connects the lifecycle phase-specific partial models of a Digital Model as well as the Digital Twin and its Physical Twin across the entire system lifecycle. It links the information and data of different processes and IT systems across the system lifecycle in a specific logical context. The Digital Thread is a framework of protocols and standards for interoperable and context-based connection and communication of different partial models and processes realized in different IT systems. It establishes an Authoritative Source of Truth across multiple systems. (Kasper et al., 2023; Hedberg et al., 2020; Kwon et al., 2020; Bone et al., 2018).
Particularly, in complex interdisciplinary system solutions, the Digital Thread plays an essential role as the backbone of the technical and organizational data and process models. The Digital Thread has two major benefits. Firstly, it improves the efficiency of integrated and interdisciplinary release/change management and configuration management by establishing a framework that can anticipate the impact of any changes made. Secondly, it enables reconfiguration according to the DIN 9001 in the event of any damage by accessing the Digital Model (Eigner, 2021; Singh and Willcox, 2018). It is essential to integrate data and processes in a logical context across the entire system lifecycle to maintain consistency within the partial models realized in different IT systems (Eigner, 2021; Hedberg et al., 2020). As a result, various standardization activities are underway to standardize different parts of the Digital Model, the Digital Twin and the Digital Thread (Gorringe et al., 2023; Hedberg et al., 2020). Although research has primarily focused on the core aspects of the Digital Thread, particularly in the aerospace industry, with the connection between engineering and manufacturing lifecycle phases (Kraft, 2015). However, there is still a need for a more systematic approach to integrate additional phases of the product lifecycle, along with their corresponding data and structures. Current approaches to the technical realization of the Digital Thread rely on a logical linkage of the data in a context-based graph. The vision is to use multiple databases optimized for specific tasks or domains. This approach, known as polyglot persistence, is more efficient than relying on a single SQL database. One way to achieve this is by building a semantic federation layer like the Digital Thread Graph and a link repository. Graph data models and databases are crucial for this, due to their superior performance for networked data (Kasper et al., 2023; Shilovitsky, 2022; Eigner, 2021; Hedberg et al., 2020; Kwon et al., 2020; Bone et al., 2018). According to Neo4j Inc (2024), a graph database is over 1100x faster than a recursive relational database for a 4-level network problem. Kasper et al. (2023) proposed a data model for the Digital Thread Graph based on objects and relations, which are described by attributes. Filters can be used to define and generate an "Engineering View", a "Manufacturing View" and a "Digital Twin View"

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

based on the lifecycle phases (Kasper et al., 2023). This system of filters can also be used to generate further definable views across the lifecycle from early engineering to disposal (Kasper et al., 2023). This raises the research question of how a graph data model of the Digital Thread, in the context of the vision of polyglot persistence, can be designed and realized and how the different phases of the lifecycle and their corresponding data models, processes, and IT systems can be interconnected and integrated.

3. The Digital Thread for System Lifecycle Management

The core of the Digital Thread Graph data model are objects and relations that are described by attributes. The product configuration can be mapped via a variant model, which is linked to the Digital Thread Graph through the variant tag (VAR_TAG). This allows the 150% model, with all possible variants, to be reduced to a 100% model by setting boolean values and applying them to the graph. The variant table is versionable. Filters can be used to define and generate different views, such as a "Define View", a "Design View," a "Produce View," and a "Digital Twin View" based on the lifecycle phases. This systematic approach of using filters can also be used to generate other definable views. The attributes of the objects and relations can be used to specify and narrow down the views. Each relation has attributes for validation and release, so called effectivity ("Valid from: (date)", "Valid until: (date)", "Released: (boolean)"), and some have additional attributes (Table 2). Therefore, all information is stored in the complete graph and accessed via a "View" whenever required. The context-based semantic linking of the data creates an Authoritative Source of Truth. Views present the same information and data in a new context.

The data model was visualized according to the scheme for graph databases based on the work of León et al. (2023). It is essential to note that relationships are bidirectional in a graph data model, even if they point in one direction. Successors always know their predecessors in such models. Additionally, the class in a graph data model is a label for nodes. These labels can be assigned to nodes flexibly without constraints on attribute definition. Multiple labels can be assigned to one node, allowing it to belong to more than one class simultaneously. Figure 2 shows the main modeling objects for graph data models according to León et al. (2023) and the data model for the Digital Thread Graph with each relation color-coded according to its corresponding lifecycle phase.

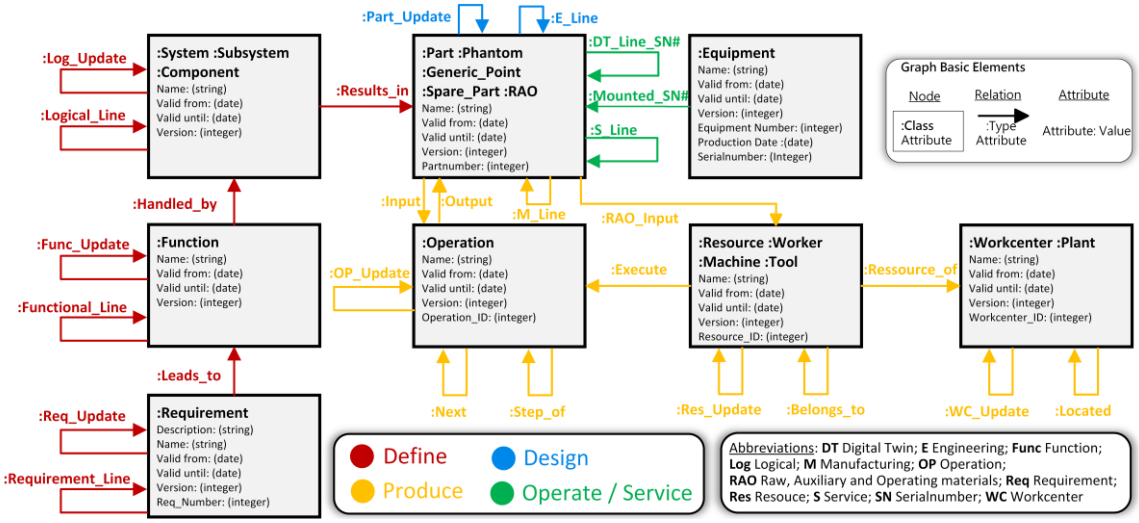

Figure 2. Data Model of the Digital Thread Graph

Three fundamental design principles were identified for systematic implementation:

1. The graph nodes are independent of the lifecycle phase and can be reused by connecting different types of relation to it.
2. The relations are specifically assigned to a lifecycle phase and are unique for every combination of node classes.
3. Lifecycle-specific views can be generated using filters. Nodes, relations, and attributes are designed to facilitate view filtering.

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

The Digital Thread Graph data model (Figure 2) objects and classes are described in Table 1. Relations with additional attributes that go beyond standard release control are listed in Table 2.

Table 1. Objects and object classes

Object Class	Description
:Component	Subsystems are composed of components, which are the lowest level of the logical structure.
:Equipment	Component of a physical product tracked by a unique equipmentnumber.
:Function	Function describes a generic function of the product.
:Generic_Point	Generic point in the product structure. Serves as a generic placeholder for variants.
:Machine	Describes a machine of the production.
:Operation	The operation object represents the individual operations and the steps they contain.
:Part	Part of the product.
:Phantom	**Manufacturing Phantom:** An assembly that results from the Manufacturing Bill of Process and was not defined in engineering. Connected to the Manufacturing_Line. **Engineering Phantom:** A generic CAD geometry. Its characteristics, such as color, are defined separately from its geometry. Connected to the Engineering_Line.
:Plant	Describes a production plant.
:RAO	Raw materials, auxiliary materials and operating materials (RAO).
:Requirement	Describes a requirement for the product.
:Resource	Describes a generic manufacturing resource.
:Spare_Part	Spare part for service.
:Subsystem	Systems are composed of subsystems.
:System	The system represents the highest level in the logical structure hierarchy.
:Tool	Describes a tool of the production.
:Workcenter	Describes a station in the production.
:Worker	Describes a worker of the production.

Table 2. Relations and additional attributes beyond standard release control

Relation	Attributes
:DT_Line_SN#	Direction_X: (float,float,float), Direction_Z: (float,float,float), Logical_Position: (string), Update_Date: (date), Production_Date: (date), Quantity: (integer), VAR_TAG: (string), Vector: (float,float,float)
:E_Line	Direction_X: (float,float,float), Direction_Z: (float,float,float), Logical_Position: (string), Quantity: (integer), VAR_TAG: (string), Vector: (float,float,float)
:Execute	Affected: (boolean), VAR_TAG: (string)
:Handled_by	Affected: (boolean)
:Input	Affected: (boolean), Quantity: (integer)
:Leads_to	Affected: (boolean)
:M_Line	Quantity: (integer), VAR_TAG: (string)
:Mounted_SN#	Direction_X: (float,float,float), Direction_Z: (float,float,float), Logical_Position: (string), Mouting_Date: (date), VAR_TAG: (string), Vector: (float,float,float)
:Output	Affected: (boolean), Quantity: (integer)
:(obj)_Update	Compatible: (boolean)
:RAO_Input	Affected: (boolean), Quantity: (integer)
:Ressource_of	Affected: (boolean)
:Results_in	Affected: (boolean)
:S_Line	Quantity: (integer), VAR_TAG: (string)

As described in the design principles, the objects in the Digital Thread Graph are defined independently of the lifecycle phases to minimize redundancy. They can be labeled with classes, which structure the

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

graph. In addition to various structures such as functional, logical, and product structures, a requirements model and production processes with associated resources can also be mapped. Furthermore, position indicators and equipments can be documented.
To ensure accuracy and completeness, each relation has a set of attributes that can be used for validation and release purposes, known as effectivity ("Valid from: (date)", "Valid until: (date)", "Released: (boolean)"). As per standard, the "Update"-relations also have the "Compatible: (boolean)" attribute. In addition, some relationships have additional attributes that provide more context (Table 2). The attribute "Compatible: (boolean)" indicates whether a change is compatible or not. If the change is incompatible, the "Affected: (boolean)" relation highlights the objects that are impacted. Additionally, the attributes "VAR_TAG: (string)" and "Quantity: (integer)" are used to describe the different variants and configurations of the product. When creating Digital Twins, the Physical Twin of a product is described in an as-built or as-maintained Digital Twin structure and the serial number is part of the relation type. To describe these structures, the attributes "Update_Date: (date)", "Production_Date: (date)", and "Mounting_Date: (date)" are utilized. By including this data, the product's current structure and complete history can be stored and accessed.

3.1. Define phase & production processes

In the Define Phase, the foundation for the Digital Model is laid by carrying out early engineering work. This phase includes creating essential information and models such as the requirements model, system architecture model, functional and logical structure, and various simulation models (Eigner, 2021; Schuh and Riesener, 2018). According to Schuh and Riesener (2018), the functional model plays a crucial role in determining the product architecture. The Digital Thread establishes a logical connection between the requirements, functional/logical models, and the product architecture. It is important to note that in the context of this work, tests are not integrated into the Digital Thread data model.
The purpose of the "Operation" object is to map production processes to parts and the Bill of Processes (BOP). An operation represents a work step, which may include sub-operations, such as assembly. In order to execute an operation, a resource, such as a worker or a machine, is assigned. The "Resource" object and corresponding classes describe the assigned resource, which is then assigned to a factory and work center using the "Workcenter" object and corresponding classes. This allows the mapping of various alternative production processes in the graph and the representation of deviations between different factories. The manufacturing structure can be derived almost automatically by assigning parts as input and output of operations. Raw, auxiliary and operating materials are a class of the object "Part".

3.2. Topology & dynamic mock-up

Dynamic mock-ups are an essential tool for product visualization, particularly for structure-independent visualization. The main goal of using dynamic mock-ups is to facilitate the visualization of different parts of a product in any desired structure. One of the significant challenges faced in practice is the restructuring of the product structure, as it involves a transition from, for example, the Engineering Bill of Material (E-BOM) to the Manufacturing Bill of Material (M-BOM) (Eigner, 2021; Wardhani and Xu, 2016; Xu et al., 2007). The product's topology must be integrated into the graph to create dynamic mock-ups based on the structures in the Digital Thread Graph. By using transformation matrixes and linear combinations, the relative vectors of the CAD files from engineering can be transformed into absolute coordinates (Eigner, 2021). The origin of the transformation is the origin of the assembly at the entry point into the graph. Figure 3 illustrates the implementation of the relative/absolute transformation concept in the graph.
In order to integrate the structure topology into the Digital Thread Graph, this work is based on the STEP format, which enables the interoperable exchange of CAD drawings (*ISO 10303-242*, 2022). When creating STEP assemblies, the coordinate systems of individual parts are positioned relative to the coordinate system of the entire assembly. This is achieved by specifying three values in the STEP file that indicate the position of the part coordinate system relative to the assembly coordinate system. These values consist of the origin of the part coordinate system, the direction of the Z-axis of the part coordinate system, and the direction of the X-axis of the part coordinate system (*ISO 10303-21*, 2016; Ungerer and Rosché, 2002). In the STEP file, information about the position of the part coordinate

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

system and its relative position can be found under "AXIS2_PLACEMENT_3D" and "CARTESIAN_POINT" (Ungerer and Rosché, 2002). The visualization software can determine the exact relative positions of the parts in a right-handed coordinate system using this information.

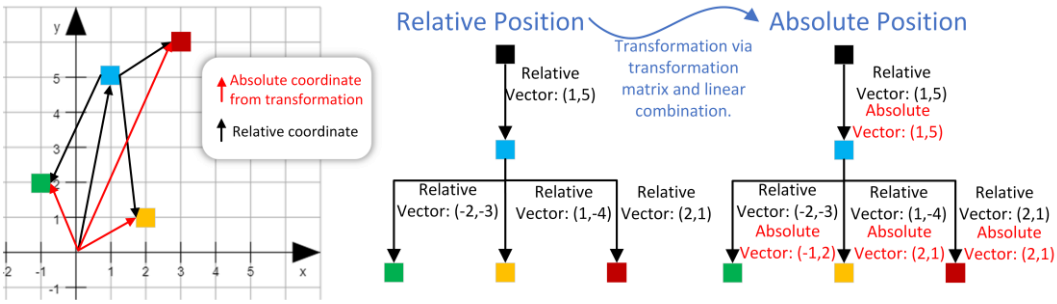

Figure 3. Transformation of relative coordinates to absolute coordinates in a graph

By merging the STEP approach with the Digital Thread Graph, a dynamic mock-up across different structures is possible. To achieve this, information about the relative position of the origin and direction of the X- and Z-axis is attached to the "Engineering_Line" relation (Attributes: Direction_X: (float,float,float), Direction_Z: (float,float,float), Vector: (float,float,float)). This information always reflects the position of the part in the parent assembly. Figure 4 shows the integration into the Digital Thread Graph based on the validation example from Chapter 4.

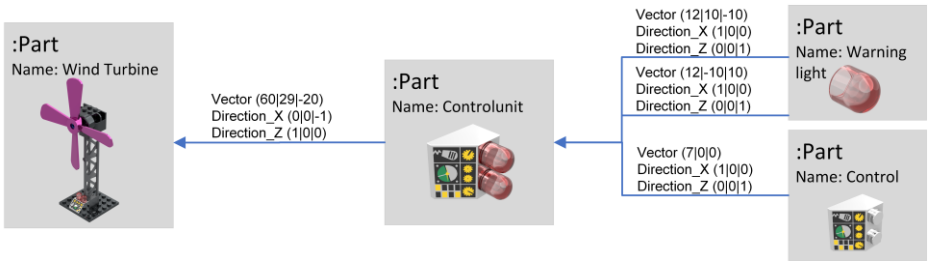

Figure 4. 3D Topology in Digital Thread Graph with a wind turbine control example

By calculating the absolute positions of the parts, any restructuring in the graph can take place without affecting the visualization. In case a structure needs to be visualized that does not exist in engineering, the position of the relevant parts is computed based on the "Engineering_Line" and the attached vectors. It is possible to improve performance when dealing with large assemblies by visualizing the assembly's geometry instead of all individual parts, for example, as an attached STEP file. This computation enables the visualization of the new structure, and dynamic changes can be made as needed.

3.3. Logical position & equipments

In complex mechanical structures, logical positions play a crucial role in maintaining data quality and ensuring data comprehensibility for users. For instance, sensors must be accurately placed in order to assign the collected data to specific locations for use in Digital Twins (Eigner, 2021). One such example is the placement of vibration sensors in piston engines. These sensors must be assigned to the respective cylinders, numbered 1, 2, ..., n. The logical position is primarily an attribute of the "Engineering_Line". If necessary, the "Digital_Twin_Line" can also have this attribute in case different logical positions are needed. For example, it can be helpful for service technicians to logically name areas of particular service relevance, such as the left front wheel. In addition, non-serialized components can be specified via the logical position. If a part exists more than once, it can be assigned to a position. For example, if four identical tires are installed on a car, the logical positions of the front left, front right, rear left, and rear right specify the respective position. The advantage is that despite the lack of serialization, it is possible to trace damage patterns.

It is possible to assign equipments, such as parts with equipmentnumbers and/or serialnumbers, to a specific installation location using logical positions and vectors. This information is linked to the

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

"Mounted" relation. It is important to note that the logical position and vectors in the "DT_Line" and the "Mounted" relation are optional and not always required.

3.4. Engineering Change Management & versioning

Product development requires Engineering Release and Change Management (ERM/ECM) to achieve traceability and transparency. These processes also serve as the basis for configuration management (Eigner, 2021). Both creating a new revision and creating a new part may require changes to the entire structure. There are two types of changes: compatible and incompatible (Eigner, 2021). Compatible changes do not require any further dependencies beyond the actual change to the part, while incompatible changes lead to further changes through dependent parts and processes. Manual decision-making by humans usually determines whether a change is compatible or not (Eigner, 2021). Companies often have a committee that classifies changes, known as a changing circle. A guideline for determining compatibility is the rule of thumb: Form, Fit, Function (Eigner, 2021). Additionally, the operation is often used as a criterion. A change is incompatible if it alters the form, installation, function, or manufacturing process. However, in reality, other factors may come into play when determining compatibility or incompatibility (Eigner, 2021).

The Digital Thread is set to support change management by linking objects (Configuration Items) throughout the lifecycle, and it creates a framework that enables the anticipation of change effects. The relationships reveal the objects that could be impacted (Eigner, 2021). In the Digital Thread Graph, every object (excluding equipments) can have multiple versions. Each object has a relation "Update", which refers to the object itself. This relation includes a "Compatible: (boolean)" attribute that specifies if the change is compatible or not. This ensures that the compatibility of changes is maintained throughout the graph. Figure 5 illustrates the fundamental concept. If any change occurs, the effects are recorded using the attribute "Affected: (boolean)" at the relevant relationships.

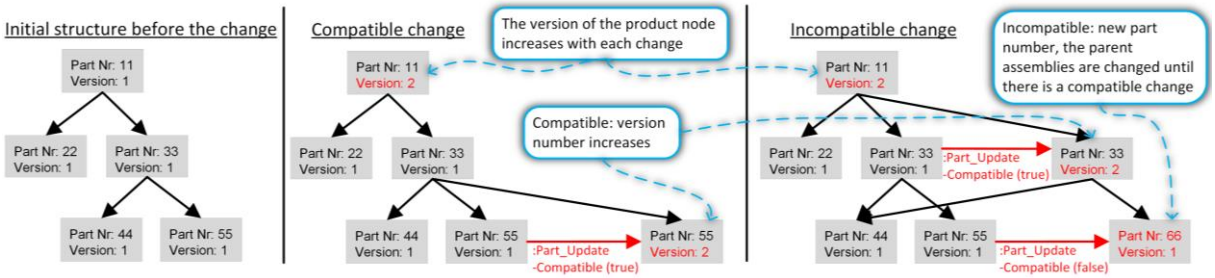

Figure 5. Versioning in the Digital Thread Graph using parts as an example

3.5. Contextual release

In the Digital Thread Graph, the release of parts is carried out through a "contextual release" logic. Objects are only released for specific uses, meaning they are released in the context of something. Consequently, the release is not implemented as an object attribute but as an attribute of the relationship. When a relationship is released, the context in which the object is used is also released. This means that the object itself does not need to be released separately since it is always consistently released in context. The relation attributes include "Valid from: (date)", "Valid until: (date)", and "Released: (boolean)". The "Released" attribute is used to indicate whether the object can be released in its current context, while the validity attributes are used to specify the period of validity of the relation.

3.6. Engineering Phantom & Manufacturing Phantom

"Phantom Parts" are a class of the object "Part". The "Phantom Part" is a part that occurs only in certain structures and/or has a generic structure that requires further specification, such as color (Xu et al., 2007). The manufacturing phantom is an assembly created by restructuring the engineering structure to the manufacturing structure. It is created when new assemblies or parts are added to the manufacturing structure that are not present in the engineering structure. Figure 6 illustrates how the manufacturing phantom is integrated into the Digital Thread Graph based on the Manufacturing Bill of Process.

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

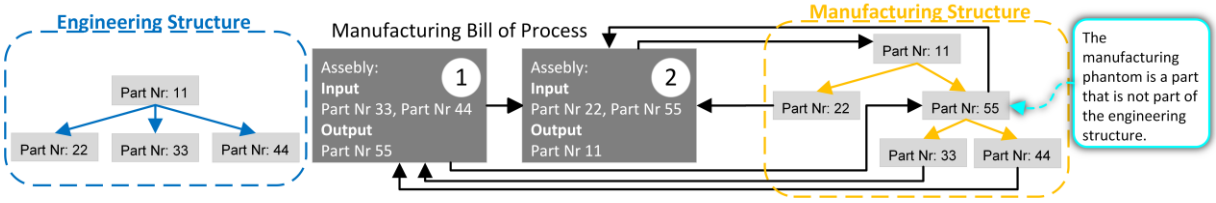

Figure 6. The Manufacturing Phantom in the Digital Thread Graph

The engineering phantom is a generic geometry that can be put to multiple uses. It is created in CAD. Features like color or surface are defined later. Different alternatives of the phantom, with varying characteristics like colors and materials, are created as separate parts, each with a unique partnumber and filled characteristics. The engineering phantom is integrated into the product structure as a regular part and is linked to its different variants. Similar to the feature model of Kasper et al. (2023), the variants of the engineering part are identified using VAR_TAGs at the relation. This allows the product structure to be easily broken down, such as by a sales order configuration.

The phantom is attached to the operation as input and output. If other operations are needed for the phantom part variants because other manufacturing processes are needed, direct linking to the operation is possible. The as-built and as-maintained configuration is captured via the serialnumber as an attribute of the relation. Figure 8 shows the integration of the engineering phantom into the Digital Thread Graph.

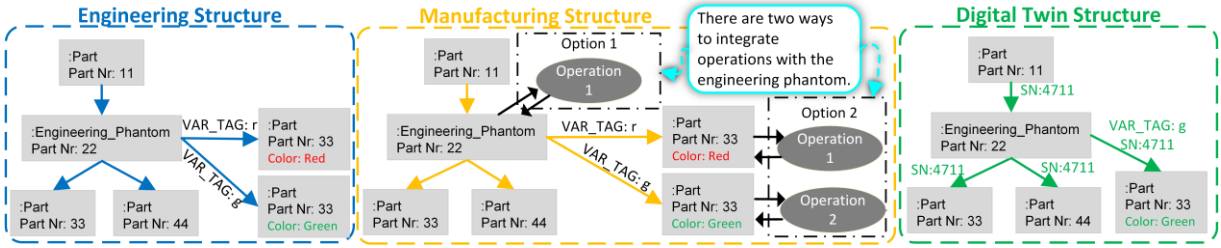

Figure 7. The Engineering Phantom in the Digital Thread Graph

3.7. Engineering and manufacturing BOM transformation and reconciliation

In contrast to the traditional conversion of the E-BOM to the M-BOM and then to the BOP, modern approaches generate the BOP from the E-BOM at the level of parts and possibly already assembly-ready assemblies with system support (Eigner, 2021). The M-BOM, which is still required for logistical purposes, is automatically derived from the BOP. With the Digital Thread Graph, the M-BOM can be derived from the manufacturing structure and the E-BOM from the engineering structure. The E-BOM and M-BOM automatically synchronize with each other and keep track of which parts are included or changed, a process known as reconciliation (Eigner, 2021). By referencing each other, changes made to one structure can be identified and synchronized with the other. Thanks to the multi-view strategy and phase-neutral objects, reconciliation is an integral part of the Digital Thread Graph. It is possible to compare the number of parts between the engineering and manufacturing structures by multiplying and summing up the quantities using the "Quantity: (integer)" attribute on the "Engineering_Line" and the "Manufacturing_Line" relation. The implementation of the concept in the Digital Thread Graph is shown in Figure 9.

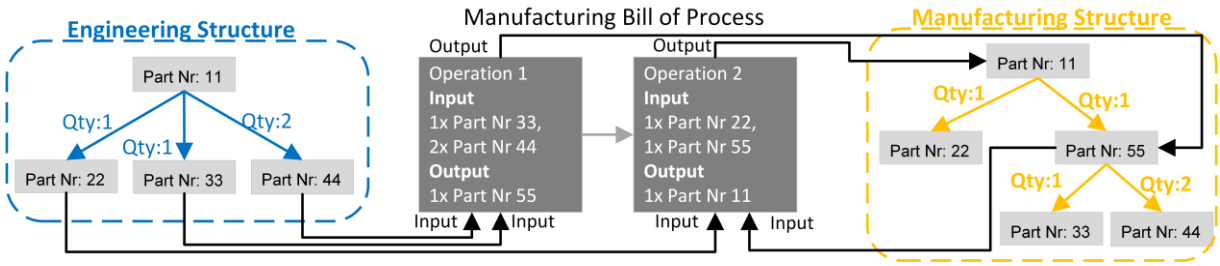

Figure 8. Engineering and manufacturing structure conversion

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

4. Validation

The validation process was conducted using Neo4j (Neo4j Inc., 2023). To create a sample model of a wind turbine made from building bricks like LEGO, a CAD program called Bricklink Studio was utilized (LEGO BrickLink Inc, 2023). Figure 9 illustrates the wind turbine, which has two propeller options: a small and a large one. Aside from the propellers, the wind turbine remains the same. Figure 9 shows the Digital Thread Graph for the wind turbine. The details in the graphs are not crucial, but what is important is the ability to generate different views of the overall graph via filters. For instance, filters can be used to create a "Define & Design View", a "Manufacturing View" or a "Digital Twin & Service View". These views can be modified or created as needed using the filters.

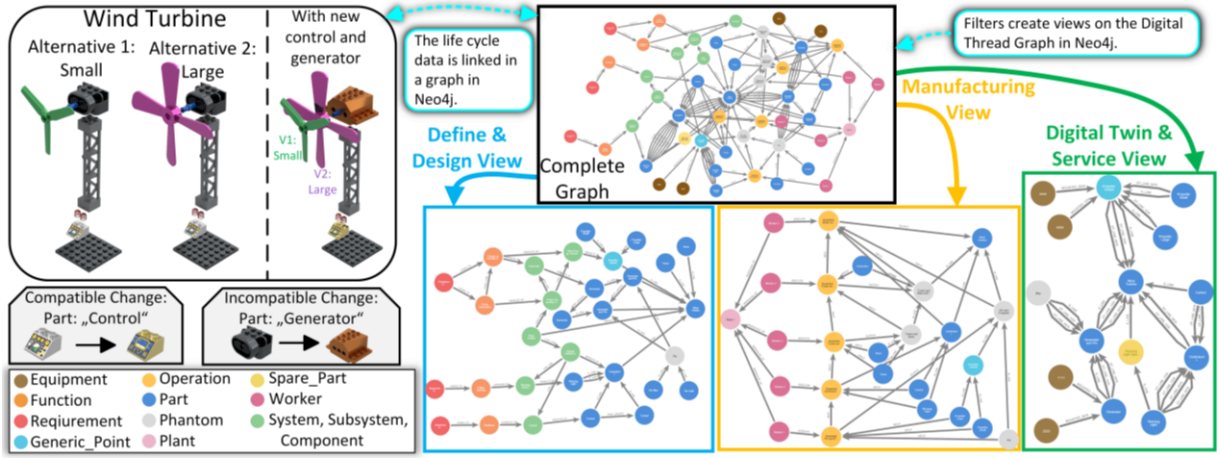

Figure 9. Wind turbine and associated Digital Thread Graph with example views

Define Phase: The wind turbine consists of the Parts *Base*, *Control*, *Warning Light*, *Tower*, *Generator*, *Pin* and *Propeller Small* or *Propeller Large*. The wind turbine was successfully implemented with three requirements, four functions, and a logical structure, all in accordance with the developed data model.
Production Process, Phantom & Reconciliation: The assembly is performed in five steps and by five workers, all assigned to one factory. Based on the assembly sequence, the manufacturing phantoms *Pin and Propeller*, *Tower and Base* and *Tower and Base and Generator* were created.
Contextual Release: The Contextual Release concept is an integral part of the relation data model and is implemented in Neo4j for the wind turbine. It is used in the wind turbine example for the updates.
Engineering Change Management & Versioning: A compatible change from version 1 to version 2 was made for the part *Control*. An incompatible change with a new partnumber was implemented for the part *Generator*, which led to a compatible change to the higher-level module due to the logic of the versioning. The effects of the change are shown via the "Affected: (boolean)" attributes in the Digital Thread Graph. The version-number of the top-level part *Wind Turbine* increases with each change. Due to the high degree of interconnection among the changes, they were excluded from the "Manufacturing View" and the "Operation & Service View" in Figure 9 to avoid overloading the figure and compromising its readability.
Topology: The wind turbine topology is integrated into the "Engineering_Line" via position and location, as described in Chapter 3.3. Figure 4 in Chapter 3.3 shows the implementation of the control unit topology exemplarily. The vectors were stored as data type "3D POINT" in the cartesian coordinate system, with X-, Y-, and Z-coordinates obtained from Bricklink Studio.
Digital Twin, Equipment & Logical Position: Two Digital Twins with serialnumbers 1234 and 5678 were created and a service structure with spare parts was implemented. In addition, equipments were installed at logical positions.
The research question was successfully addressed by developing a semantic federation layer with the Digital Thread Graph data model. This federation layer interconnects objects and models of different life cycle phases and can serves as the backbone of a polyglot persistence. However, further research is needed to explore internal and external extensibility (e.g., a link repository).

https://doi.org/10.1017/pds.2024.210 Published online by Cambridge University Press

5. Discussion and conclusion

A graph-based approach for modeling the Digital Thread is promising. By using attributes on objects and relations, it is possible to filter and create views while minimizing redundancy in the entire system lifecycle. The focus of this paper is on the product data of the define, design, produce, and operate lifecycle phases. An initial approach to process integration has been developed; however, further research is needed to directly integrate dynamic processes such as quality or change management. In order to implement a polyglot persistence, it will be necessary to integrate additional aspects such as simulation models, software (as an item), logistics and core Enterprise Resource Planning functionality both internally and externally. The Digital Twin must be enriched with additional data to extend beyond a pure as-built or as-maintained structure. Graph-based approaches have been used in Product Lifecycle Management systems, mainly implemented in relational databases. However, recursive operations with networked data in these databases are known to have performance issues. The vision is to have a polyglot persistence with an overarching semantic federation layer and a link repository connecting decentralized databases and applications, which is achievable through the superior performance of graph databases for networked data.

References

Bone, M., Blackburn, M., Kruse, B., Dzielski, J., Hagedorn, T. and Grosse, I. (2018), "Toward an Interoperability and Integration Framework to Enable Digital Thread", *Systems*, Vol. 6 No. 4. https://dx.doi.org/10.3390/systems6040046.

Eigner, M. (2021), *System Lifecycle Management: Engineering Digitalization (Engineering 4.0)*, Springer Vieweg. ISBN: 978-3-658-33874-9.

Gorringe, C., Gould, E. and Neag, I. (2023), "Standards-Based Digital Thread as Authoritative Source of Truth", *IEEE AUTOTESTCON 2023.* https://dx.doi.org/10.1109/AUTOTESTCON47464.2023.10296125.

Hedberg, T.D., Bajaj, M. and Camelio, J.A. (2020), "Using Graphs to Link Data Across the Product Lifecycle for Enabling Smart Manufacturing Digital Threads", *J Comput Inf Sci Eng.*, 20(1). https://dx.doi.org/10.1115/1.4044921.

ISO (2016), *ISO 10303-21*.

ISO (2022), *ISO 10303-242*.

Kasper, N., Pscheid, I., Pfenning, M. and Eigner, M. (2023), "Der Digital Thread Graph als Backbone des Digital Twin im System Lifecycle Management", *Tag des Systems Engineering 2023*. ISBN: 978-3-910649-00-2.

Kraft, E. (2015), "HPCMP CREATE;-AV and the Air Force Digital Thread", *53rd AIAA Aerospace Sciences Meeting*, Kissimmee, Florida. https://dx.doi.org/10.2514/6.2015-0042.

Kwon, S., Monnier, L.V., Barbau, R. and Bernstein, W.Z. (2020), "Enriching standards-based digital thread by fusing as-designed and as-inspected data using knowledge graphs", *Advanced Engineering Informatics*, Vol. 46 No. 101102. https://dx.doi.org/10.1016/j.aei.2020.101102.

LEGO BrickLink Inc (2023), "Bricklink Studio", available at: https://www.bricklink.com/v3/studio/download.page?ltclid= (accessed 8 November 2023).

León, A., Santos, M.Y., García, A., Casamayor, J.C. and Pastor, O. (2023), "Model-to-Model Transformation", *Business & Information Systems Engineering*. https://dx.doi.org/10.1007/s12599-023-00824-9.

Neo4j Inc (2024), "How much faster is a graph database, really?", available at: https://neo4j.com/news/how-much-faster-is-a-graph-database-really/ (accessed 15 February 2024).

Neo4j Inc. (2023), "Neo4j", available at: https://neo4j.com/ (accessed 8 November 2023).

Pfenning, M. (2017), "Durchgängiges Engineering durch die Integration von PLM und MBSE", Dissertation, VPE, Technische Universität Kaiserslautern, 2017.

Schuh, G. and Riesener, M. (2018), *Produktkomplexität managen: Strategien - Methoden - Tools*, 3., vollständig überarbeitete Auflage, Hanser, München. ISBN: 978-3-446-45334-0.

Shilovitsky, O. (2022), "PLM Web Services – Polyglot Persistence, Microservices, and Data Modeling", available at: https://beyondplm.com/2022/09/23/plm-web-services-polyglot-persistence-microservices-and-data-modeling/ (accessed 10 February 2024).

Singh, V. and Willcox, K.E. (2018), "Engineering Design with Digital Thread", *AIAA Journal*, Vol. 56 No. 11, pp. 4515–4528. https://dx.doi.org/10.2514/1.J057255.

Ungerer, M. and Rosché, P. (2002), *Usage Guide for the STEP PDM Schema V1.2: Release 4.3*, available at: https://www.mbx-if.org/documents/pdmug_release4_3.pdf (accessed 29 October 2023).

Wardhani, R. and Xu, X. (2016), "Model-based manufacturing based on STEP AP242", *2016 12th IEEE/ASME (MESA)*. https://dx.doi.org/10.1109/MESA.2016.7587187.

Xu, H.C., Xu, X.F. and He, T. (2007), "Research on Transformation Engineering BOM into Manufacturing BOM Based on BOP", *AMM*, 10-12. https://dx.doi.org/10.4028/www.scientific.net/AMM.10-12.99.