INTERNATIONAL DESIGN CONFERENCE – DESIGN 2022
https://doi.org/10.1017/pds.2022.73

From a Monolithic PLM Landscape to a Federated Domain and Data Mesh

Y. Hooshmand [1,✉], J. Resch [1], P. Wischnewski [2] and P. Patil [3]

[1] Mercedes-Benz AG, Germany, [2] PROSTEP AG, Germany, [3] HCL Technologies Germany GmbH, Germany
✉ yousef.hooshmand@mercedes-benz.com

Abstract

Product Lifecycle Management (PLM) is one of the most business-critical IT backbones of manufacturing companies. It often consists of numerous, rigidly interwoven monolithic applications and is seen as synonymous with costly maintenance, lack of extensibility, and poor scalability. This paper proposes an approach for transforming a monolithic PLM landscape into a federated Domain and Data Mesh. This enhances semantic interoperability and enables data-driven use cases by treating data as first-class citizens. User-centric PLM domains moreover help to increase productivity in the workplace.

Keywords: product lifecycle management (PLM), semantic modelling, software engineering, data mesh, domain-driven design

1. Introduction

Managing product data along its lifecycle has always been a challenging task for enterprise IT in manufacturing companies. Based on ISO/IEC/IEEE 15288 (2015), the generic life cycle of a product consists of the following stages: Concept, Development, Production, Utilization/ Support, and Retirement. During these stages, various tools are used to manage the generated product data. For example, in early phases of the Concept stage, Requirements Management (RM) tools are often used to define, to document and to manage the requirements. As development continues, various applications such as Product Data Management (PDM), Application Lifecycle Management (ALM), Enterprise-Resource-Planning (ERP), and Internet of Things (IoT) Platforms come into play. Each of these applications is used to manage a subset of product data that is generated during product lifecycle and pertaining to a product context. The product data is exchanged between these applications realizing business use cases across applications and across departments. Figure 1 shows the product lifecycle phases as well as the PLM landscape of a manufacturing company with more than 100 software tools for data handling and numerous interfaces for data exchange.

The business issues facing manufacturing companies, such as the abrupt electrification of powertrains, the demand for connected devices, and the handling of (big) data, are emerging at an accelerating pace. This is leading to challenges in realizing unconventional business requirements using conventional IT Landscape. To overcome this, on one end companies are trying to align their IT Landscape to new internal and external requirements. On the other end, independent software vendors are offering platform solutions that promise a homogeneous IT landscape to be competitive in disruptive situations. However, the platform solutions do not cover all aspects of changing business requirements. Furthermore, the challenges related to semantic interoperability, semantic configuration, and data consistency in the IT landscape, as well as data discoverability and usability, are not properly addressed. These are essential for an organization to be both data-driven and user-centric in order to thrive in disruptive situations.

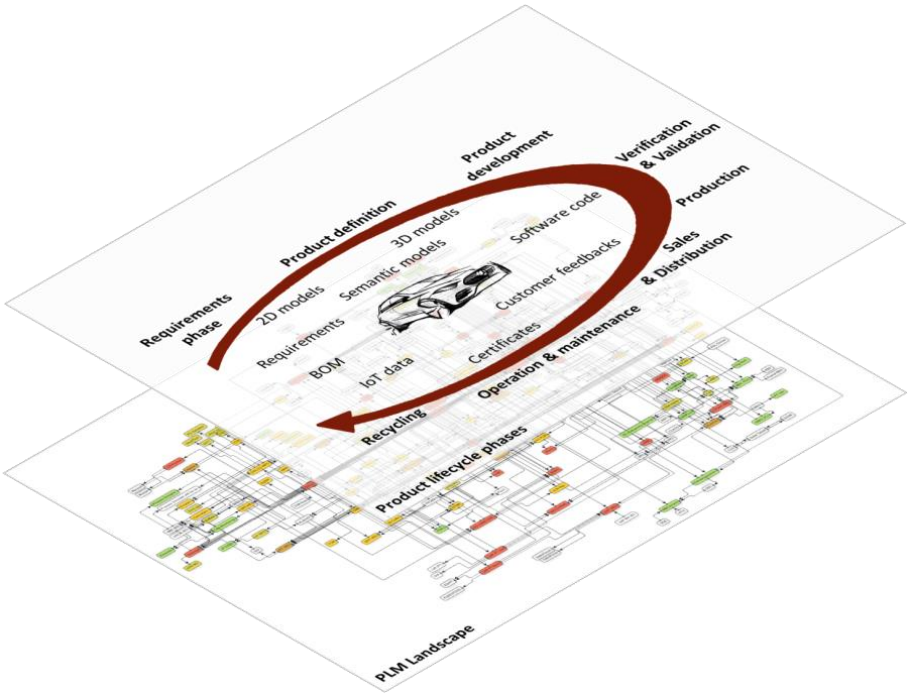

Figure 1. Product lifecycle phases and PLM landscape

The challenges along the product lifecycle can be considered both from a business process perspective and from a technological perspective. The PLM challenges from the process perspective have been the dominant topic in the literature. To overcome these challenges, various authors including Eigner (2021), Dickopf (2020), Pfenning (2017) and Hooshmand et al. (2017, 2018) have proposed sophisticated approaches that cover either the entire product lifecycle or a specific aspect of it.

The approach proposed by Pfenning et al. (2021), on the other hand, focuses on technological aspects of PLM space and addresses several problems arising from the monolithic nature of PLM tools. The proposed solution demands a modern cloud-native Web architecture based on Microservices and Linked Data that enables companies to introduce Model-Based Systems Engineering (MBSE) on the large scale. Likewise, this article focuses mainly on the technological aspects and proposes an approach towards a data-driven and user-centric PLM landscape, enabling companies to tackle various challenges.

1.1. PLM landscape challenges

One of the most critical underlying problems in manufacturing companies is that most information systems are monolithic legacy systems. Bisbal et al. (1999) define legacy information systems as the backbone of an organization's information flow and the primary tool for consolidating business information. Legacy systems are typically mission-critical, expensive to maintain due to lack of adequate documentation and system understanding, difficult to integrate with other systems, extremely difficult to extend or modify, costly and time-consuming to troubleshoot, and running often on obsolete hardware (Bisbal et al., 1999; Bennett, 1995).

Monolithic applications usually consist of monolithic User Interfaces, large Application Servers, Databases, and Data Models (Newman, 2019). This makes it almost impossible to react flexibly and quickly to emerging requirements that require changes to PLM processes, methods, or tools. The applications would become rigid even if one of the aspects is monolith. This also leads to poor and restricted scalability. Complex applications and a highly interwoven PLM landscape make any change or customization too expensive or even risky for the company.

A further critical problem is that the ability to access, semantically link, and use the data stored in IT applications is limited. This is often only possible through various time-consuming and cost-intensive data processing operations (Alharthi et al., 2017). In addition, interoperability between IT applications is extremely restricted, which, among other things, makes an End-to-End Configuration Management along the product lifecycle nearly impossible or very difficult.

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

Finally, IT tools do not reflect the company's business capabilities and domains, which hinders the natural flow of information. The domain data is often scattered in various tools with overlapping functionalities. As a result, many domain experts work with multiple tools daily to accomplish a single business use case. Therefore, the current philosophy is: User adapts to Technology. This in turn slows down decision-making processes and thus the ability to react quickly to arising change requests and new requirements.
To overcome some of the above challenges, various approaches have been evolved to migrate a monolithic legacy system to solutions with modern architectures such as Microservices Architectures (Taibi et al., 2017). However, these approaches are not really practiced by most application vendors in the PLM space, nor are they seriously considered by most manufacturing companies.
There exist several strategic options for migrating a legacy system, namely Conversion, Reimplementation, Redevelopment, Wrap, and Replace (Sneed and Verhoef, 2019). Due to the strategic importance of the PLM landscape for manufacturing companies, reimplementation and redevelopment are more promising, for example, to ensure the long-term goals like flexibility. However, different strategies can be pursued for different business domains and capabilities. For one PLM domain it may be sufficient to acquire a commercial software application from the market, for another a completely new development may be required. It is only important not to compromise the overall strategic goals such as semantic interoperability or usability.

1.2. Business needs demanding a new PLM landscape

To remain competitive in the current highly dynamic market situation, manufacturing companies must react quickly to new internal and external requirements. Among other things, this requires companies to quickly adapt their processes and IT tools along the entire product lifecycle. To achieve this, they must stay flexible in the face of unforeseen and disruptive requirements that can arise at any time (Highsmith et al., 2020). Furthermore, they need to be data-driven to gain insights faster and to make quicker decisions based on the knowledge gained from the internal and external data. This, in turn, requires fast access to and semantic linking of their data, which is often scattered in various IT tools or data silos throughout the company.
This paper presents a methodology for designing a PLM landscape that meets the following three requirements, among others: First, the solution is designed according to the user-centric principle. This provides users with the right tool with proper usability to perform their tasks in the most efficient way. Second, the landscape consists of inherently adaptable and extendable solutions, capable of providing new functionalities on a daily basis. Consequently, the software can be quickly adapted to meet new requirements that arise from the ever-faster changing markets. Third, data is available in the landscape as a self-service to users. Following the data-driven principle, this enables effective use of the available data to quickly obtain information and gain new insights based on data.
The next section briefly presents selected concepts, methodologies and philosophies used intensively in this work to enable the transition from the current monolithic PLM landscape to the desired federated Domain and Data Mesh.

2. Tools to tackle PLM landscape challenges

A range of approaches, technologies and methods are required to holistically cope with the challenges mentioned above. For example, the Semantic Web Technology is used to improve the interoperability of data along the product lifecycle and to enable much-needed Semantic Configuration Management. In addition, the Semantic Web is leveraged to enable model-driven software development and to introduce model-driven software components. This enables efficient and effective extensibility and creates inherent flexibility to meet new arising requirements. Furthermore, it empowers companies to respond quickly to new internal and external requirements.
Domain-Driven Design and Data Mesh are used as main architectural design approaches and methodologies to cut the monolith PLM applications in smaller user-centred pieces or domains. Thus, both operational and data-centred use cases can be addressed with priority and based on value-driven metrics. The operational use cases help knowledge workers to work more effectively and efficiently. The data-centred use cases enable the company to be data- and knowledge-driven. Modern architectural design approaches such as microservices and micro-frontend are also used to ensure independence and flexibility of components in the landscape.

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

2.1. Semantic Web Technology

The Semantic Web aims to represent data and information on the Internet in such a way that machines are able to independently analyse data and interact with other machines and even humans (Berners-Lee et al., 2001). This is an attempt to make the Web "intelligent" by using various technologies forming the Semantic Web Technology Stack (W3C, 2015). As illustrated in the Figure 2, established Web technologies form the foundation of the Semantic Web. Linked Data is seen as a subset of this stack. Communication mechanisms such as HTTP, encoding standards such as Unicode, identification mechanisms such as IRI/URI, and authentication mechanisms are implemented in this foundational layer. On top of that, representation syntaxes and formats such as TURTLE, JSON-LD, RDFa and μFormats as well as the RDF (Resource Description Framework) are built. RDF is indeed one of the key building blocks of the semantic Web, standardizing the description and exchange of data across the Web. RDF also makes the data searchable via SPARQL (SPARQL Protocol And RDF Query Language), which is another important building block for both querying and handling data.

The technologies described so far are truly essential for standardized data exchange and data integration on the Web. However, true semantics can only be realized through the introduction of modelling languages (such as OWL and RDFS) and rule languages (such as RIF and SHACL). These languages are crucial for creating sophisticated models of real-world phenomena and systems, breathing semantic soul into the data, and making it machine interpretable.

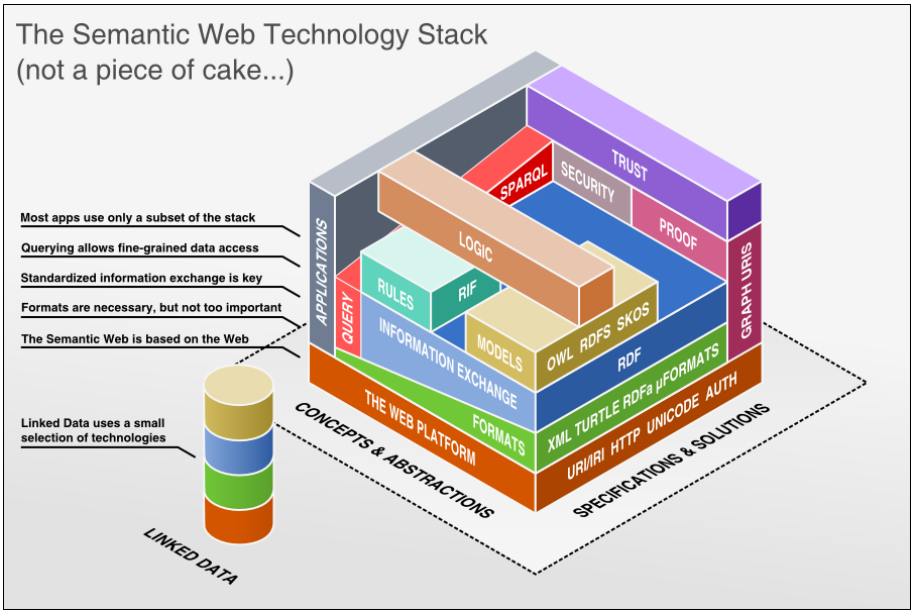

Figure 2. Semantic Web Technology Stack (Nowack, 2009)

The core promise of the Semantic Web is achieved by using the aforementioned highly standardized technologies as we build applications. Top layers of the stack (including Logic and Trust) are not yet standardized and are mainly still in academic stages.

2.2. Domain-Driven Design

Evans (2003) proposed Domain-Driven Design (DDD) as a software development approach, in which development is done based on domain models with rich content about the business processes and rules of the related domain capabilities. This approach is especially helpful in developing software solutions with large domains and complex logic. To prevent ambiguity in DDD, a Ubiquitous Language or a common language have to be used to embed the domain terminology into the software systems. It is important that both software developers and domain experts collaborate and jointly develop the ubiquitous language of the domain model (Vernon, 2016).

The complexity of large models and teams is handled by dividing them into different Bounded Contexts (domains) and being explicit about their interrelationships. Each domain can also be divided

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

in subdomains. Contexts are defined based on the actual customer use cases of a domain. A context thus bundles the functionality that is coherent for a particular domain (Vernon, 2016). A microservice architecture can then be designed based on the contexts and sub-contexts (Newman, 2019). The interfaces between the services of different contexts should be minimal and simple (Evans, 2003). This allows each domain to implement and publish its functionalities and services independently. This is a prerequisite for the daily provision of new functions in the IT landscape. Furthermore, a Context Map can be used to illustrate the relationships between bounded contexts (domains) and to make the interfaces and the interconnections among them more transparent.
Various patterns introduced by Evans (2003) and later Vernon (2016) offers useful tools for an effective strategic (re-)design of complex enterprise applications or even the entire IT landscape. Leveraging the DDD approach, Newman (2019) also defines a set of patterns for migrating a monolithic system to a modern microservice architecture.
By concentrating on domains with clear boundaries, the complexity of software coupling will be significantly reduced. This accelerates the development and increases the release cycle and the stability of the software components in each domain.

2.3. Data Mesh

The key idea of Data Mesh is to apply DDD and Product Thinking to the challenges in the data and analytics space. Product Thinking is a product-oriented way of funding and organizing software development as opposed to software projects (Narayan 2018). Dehghani (2022) proposed Data Mesh as a new approach, aiming at effective and efficient sourcing, managing, and accessing data for analytical use cases at scale. To address aforementioned challenges, Data Mesh relies on four principles, namely domain-oriented decentralized data ownership and architecture, data as a product, self-serve data infrastructure as a platform, and federated computational governance (Dehghani, 2022).
Based on the first principle, data decomposition takes place according to domains as explained in section 2.2. Consequently, the domain owns the data and the dataset. Thus, operational capabilities and data access as well as analytical data provided and owned by the domain. This is in contrast to a Data Lake approach, where one data team is responsible for providing analytical data to all business domains. The Data Mesh approach enables the companies to be data-driven with end-to-end responsibility for data owners. Based on this principle the duplication of data is allowed and a domain can have multiple data products. The second principle aims to serve data as product and ensures that the data products are Discoverable, Self-describing, Addressable, Trustworthy, Interoperable and Secure. Input Data Ports (IDP) and Output Data Ports (ODP) are used to receive data from upstream systems and to provide data to downstream systems.
The third principle leverages platform thinking by creating a shared domain-agnostic self-service data infrastructure to lower the cost and specialization needed to build data products. It helps to create secure data products quickly and execute their pipelines. Furthermore, it focuses on supporting and enabling the collaboration between data producers and data consumers. The last principle ensures domain self-sovereignty as well as interoperability through global standards. It can cope with a dynamic topology by automatic execution of decisions and by computational validation. Standardization of semantics and syntax as well as global rules are major enablers for this principle. The global interoperability standards ensure that the mesh behaves as a consistent interoperable ecosystem.

3. Domain-driven PLM landscape

A domain-oriented redesign and development of the PLM landscape is essential to overcome the challenges discussed in the previous sections. This facilitates especially the consolidation of data and functionalities scattered across various systems. Therefore, the new philosophy will be: Technology adapts to User. It also addresses the goals of fast deployment and making data available for self-service analytics. In this section, the proposed domain-driven PLM landscape in which the DDD and Data Mesh principles are embedded is first schematically presented at a high level. A sample domain is then used to describe various aspects and the domain structure in more detail. The approaches and technologies presented in section 2 are combined and slightly adapted to meet the requirements of the desired PLM landscape. The data products are always considered as part of a bounded context forming a (sub)domain and RDF, OWL and SHACL are used extensively to describe every aspect of domains.

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

3.1. Domain's context map

The context of this article is limited to the business domains in design, manufacturing and after sales phases, which comprise the classic PLM core functionalities of manufacturing companies. However, the presented approach is extensible to the entire lifecycle of the product. Depending on how we define the bounded contexts and delineate the domains, about 12 to 16 domains will be required to replace the systems currently used in the above phases. Figure 3 shows schematically selected PLM domains and their interactions.

Each domain has its own data model, its own data storage and infrastructure. As it contains all the data that is needed to be fully functional, the domain is independently operational. For example, the part domain has about 50 classes with a few hundred attributes. The BOM domain needs few classes and attributes from the part domain to be independently operational. This data is provided asynchronously and near-real-time via event streaming platforms like Kafka. Thus, the BOM domain remains functional even if the part domain is temporarily unavailable. Each business object can be created or modified only in one domain, but its data can reside in various consecutive domains. The business objects can also be extended with further data in subsequent domains (e.g., by adding new attributes to the object model) but their changes will not be synchronized back to the source domain. The definition of Single Source of Truth is thus altered to Nearest Source of Truth based on Single Source of Change.

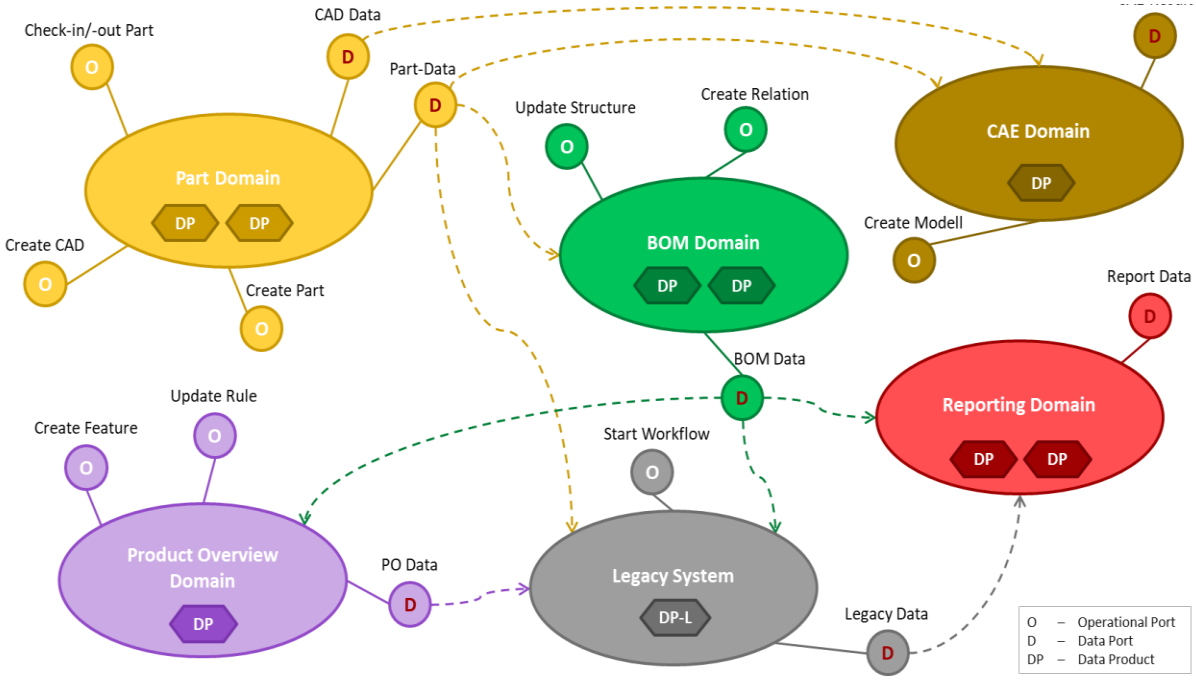

Figure 3. Schematic context map of selected PLM domains

Although the domain models can be changed locally and independently in each domain or bounded context, global governance ensures consistency and interoperability across domains. This is discussed in more detail in following sections.

The legacy systems remain part of the PLM landscape as long as the functionalities have not been fully migrated or the data supply has not been switched over. The migration takes place in smaller steps so that the provision of the process chain remains guaranteed. It is also worth noting that domains can be built on proprietary components, open-source components, in-house developments, or a combination of these. The only important thing is that they are not built as monolithic, rigid solutions and that interoperability is ensured.

Using part domain as an example, the next section will dive deeper and show how a domain is constructed and what technologies are used.

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

3.2. Domain structure and setup

In accordance with the domain-driven design and data mesh approaches, each domain has its own domain model and one or more data products with their own data storages. In addition, communication with the domains takes place via ports. The operational ports are marked with O and are required for data processing based on a business logic. As can be seen in Figure 4, services such as Create Part or Create CAD can be accessed only through corresponding operational ports. The data ports, on the other hand, are marked with D and are responsible for data import through IDP and data export through ODP. Both operational ports and data ports are subject to version control and global governance to ensure interoperability between the domains.

Although most domains initially have only one data product with one database, new data products can be added as the domain grows or business, technical, or legal requirements necessitate. As shown schematically in Figure 4, a second data product with a higher SLA (Service Level Agreements) is used for parts that are released for manufacturing. This data product can also be enriched with additional manufacturing-specific information through further operational ports. The same technologies are used for intra-domain data exchange as for cross-domain data exchange. Furthermore, local intra-domain governance ensures the interoperability between data products.

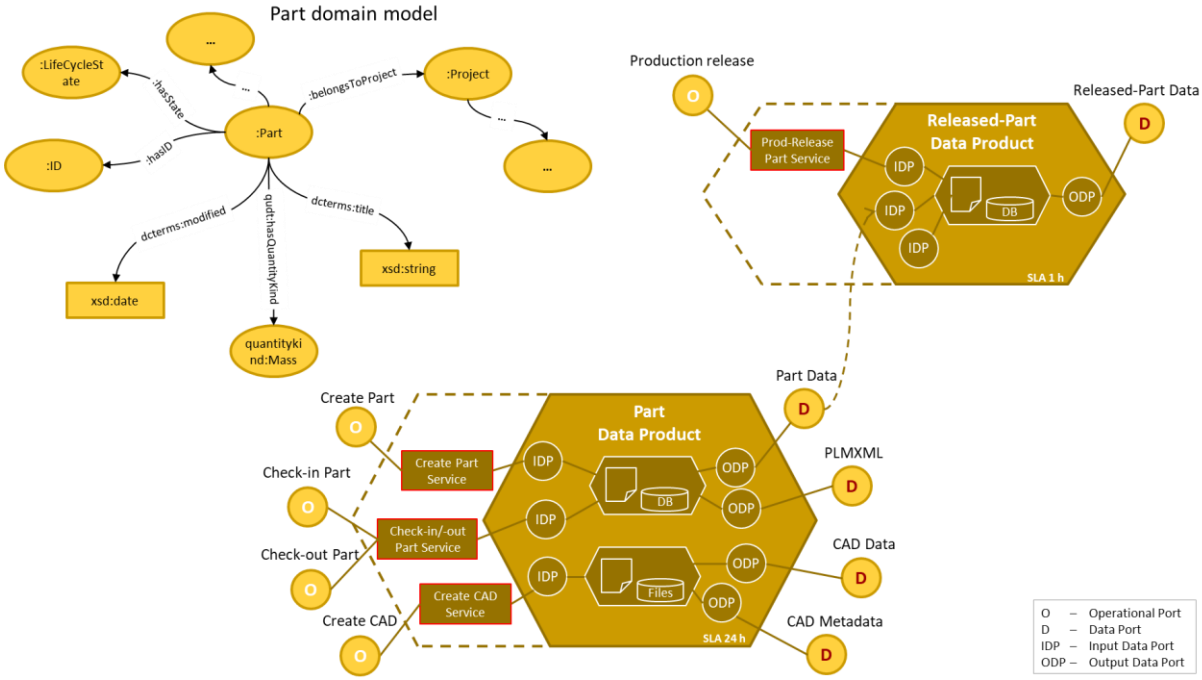

Figure 4. Internal structure of a domain

The modelling languages OWL and SHACL (W3C, 2015) are used to create a semantically precise domain model. Although the domain models are small, with an average of 40 classes, the domain models are moreover built in a modular format to cover different aspects in independent small modules. For example, the information needed for data validation is stored in a separate module to reduce memory requirements in corresponding services and make the collaborative maintenance of the domain model more efficient. The modules comprise the data model, various aspects of the business logic, and the domain interfaces. The domain model is thus the sum of all modules that are semantically linked and form a unit.

3.3. Model-driven software components and services

The software components are mainly designed according to a software development approach in which the behaviour of each software component is generically controlled by the domain model. For example, data validation is performed in different domains by using the same software library but different domain-specific models to produce different behaviours. This drastically increases efficiency

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

in the development, maintenance, and reuse of software components needed in multiple domains. Furthermore, new requirements can often be met only by extending or modifying the domain model without having to change the software code at all.
The domain model acts as an interface between domain experts and software developers to define the desired domain behaviour. Thus, new requirements and ideas can first be discussed based on semantic models to define the intended behaviour. The first software prototypes can often be developed in hours and days instead of weeks and months.
A consolidated user interface approach across all domains and services helps to provide a seamless experience for users. Although numerous independent services and software solutions are offered in different domains, a unified role-based user interface is provided to support users in their daily tasks. In most cases, users don't even realize which domain they are interacting with. At the same time, by using different approaches such as Micro Frontends or Backends For Frontends, each segment of the user interface can be developed and deployed almost 100% independently. Thus, domains retain sovereignty over the release cycle of their software components from backend to user interface.

3.4. Cross-domain Semantic Layer

As explained in section 3.2 the domain models are built modular to cover different aspects in different modules. The modules describing the data models and the domain interfaces (IDPs and ODPs) are particularly important both to ensure interoperability across domains and to enable self-service data access.
A cross-domain Integration model semantically connects the models originating from different domains. The Integration model also connects the domain models with an Enterprise upper ontology as well as a company-specific Terminology model and forms a Semantic Layer and thus enables, among other things, cross-domain semantic interoperability and analytical use cases. The Integration model is modular and covers various aspects of integration in different modules.
The semantic relationships among domains and their data are explicitly part of the Integration model and not hidden in any piece of business logic code. Since both the domain models and the Integration model are developed based on the same modelling language described in section 2.1, their integration as well as their subsequent exploitation is seamless. This makes the domain data much easier discoverable both by data analysts and by machines. Capabilities for accessing and exploring the models as well as the domain data are provided by the PLM infrastructure, described in section 4, through lookup services and distributed query engines. The infrastructure provides the necessary tools for creating actual data-driven applications.
To enable holistic data-driven solutions, a Semantic Layer consisting of the following semantically integrated aspects is thus required.

- The Enterprise upper ontology is a domain-agnostic ontology model covering the main concepts and business objects of the company. All domain-specific business objects can be derived directly or indirectly from the objects in this model. An Enterprise upper ontology emerges gradually over time as domains evolve. It generally is composed of a few hundred unambiguous classes and properties.
- The Domain models semantically describe the domains in the PLM landscape. Each domain model comprises different domain facets, including the data model, different aspects of the business logic and the domain interfaces in several modules.
- The Terminology model combines generic and company-specific terms in a semantically unambiguous model. This model is particularly important for data-driven use cases that combine structured and unstructured data.
- The Integration model semantically integrates and connects the previous models, forming a coherent Semantic layer.

The Semantic Layer is a prerequisite for transforming the fragmented PLM data silos into a live enterprise knowledge graph.

4. Infrastructure for Domain-driven PLM landscape

Compared to a monolithic landscape, the number of software components in a domain-driven landscape multiplies. Therefore, a unified, cross-domain PLM platform is a prerequisite for the

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

efficient implementation and operation of the desired PLM landscape. This is also a prerequisite for a unified, consistent, and interoperable ecosystem. The generic platform follows the Data Mesh (Dehghani, 2022) approach, which treats data as a first-class citizen in all affairs. The PLM platform encompasses both a domain-agnostic infrastructure and a federated governance of the PLM landscape.

4.1. Domain-agnostic infrastructure

The infrastructure provides generic, domain-agnostic capabilities to efficiently build and maintain the PLM landscape. Using a generic infrastructure to build domains also helps to reduce costs and requires less specialization. Among other things, infrastructure capabilities include distributed file storage provisioning, access control, identity management, monitoring, orchestration mechanisms, and API versioning. Although most cloud providers such as Microsoft Azure or Amazon AWS offer similar capabilities, these still need to be customized and adapted to the needs of the PLM landscape. An Infrastructure-as-a-Service (IaaS) strategy therefore better meets the long-term requirements of the PLM landscape.

In addition to the core capabilities required to operate and maintain each PLM domain and the overall landscape, the PLM infrastructure offers cross-domain functionalities such as distributed query engines or data catalogues. These are especially needed for data scientists and data analysts to facilitate the transition to a data- and knowledge-driven enterprise. In addition, the infrastructure supports and facilitates collaboration between different domains.

4.2. Federated governance

A consistent interoperable landscape requires a governance model that embraces decentralization and domain self-sovereignty as well as interoperability through global standardization. Global standards include schema syntax, data formats, modelling languages, metadata formats and syntax, generic objects, dataset schema, and requirements regarding security and compliance. In addition, global API versioning policies as well as data port release rules are important parts of the governance model for ensuring global interoperability of the landscape. The cross-domain Integration Model described in section 3.4 is also maintained as part of global governance. The local rules and standards, on the other hand, guarantee the decentralization and autonomy of the domains.

A federated computational governance is a prerequisite for efficient maintenance of the dynamic topology of the landscape through automatic execution of decisions. It includes both local intra-domain decisions and global decisions.

5. Summary

In this article, a new PLM landscape in the form of a federated Domain and Data Mesh is proposed. By leveraging modern approaches such as Domain-Driven Design and Data Mesh, as well as technologies such as Semantic Web, the proposed Domain-driven PLM landscape tackles various technical, operational and business challenges faced by manufacturing companies. It also meets new requirements essential to make an enterprise data- and knowledge-driven. In addition, the new landscape is inherently user-centric in design, helping to make enterprise IT more focused on users and their real needs for their daily tasks. The model-driven software components and services help to significantly increase the synergy between the domains and improve efficiency in the development, maintenance and reuse of the software components.

By shifting from traditional Single Source of Truth philosophy to Nearest Source of Truth based on Single Source of Change philosophy, the monolithic legacy PLM solutions are efficiently replaced with modern user-centric domains in a federated landscape. This enables enterprises to respond quickly and flexibly to new requirements. A cross-domain Semantic Layer ensures cross-domain semantic interoperability and enables holistic data-driven solutions and analytical use cases transforming the fragmented PLM landscape into a living knowledge graph.

Finally, a domain-agnostic PLM infrastructure leverages platform thinking across all domains and ensures global interoperability through global standards as well as domain self-sovereignty through local standards.

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press

Disclaimer: The content of this article originates from a still ongoing project at Mercedes-Benz AG, therefore the final result may differ slightly. In addition, any comment towards vendors or third parties is authors' own opinions.

References

Alharthi, A.; Krotov, V.; Bowman, M. (2017), *Addressing barriers to big data*, Business Horizons, Volume 60, Issue 3, Pages 285-292. https://doi.org/10.1016/j.bushor.2017.01.002

Bennett, K. (1995), *Legacy systems: coping with success*, IEEE Software, vol. 12, no. 1, pp. 19-23. doi: 10.1109/52.363157

Berners-Lee, T.; Hendler, J.; Lassila, O. (2001), *The Semantic Web*, Scientific American, vol. 284, no. 5, pp.34-43. https://www.jstor.org/stable/26059207

Bisbal, J.; Lawless, D.; Wu, B.; Grimson, J. (1999), *Legacy information systems: issues and directions*, IEEE Software, vol. 16, no. 5, pp. 103-111. doi: 10.1109/52.795108

Dehghani, Zh. (2022), *Data Mesh: Delivering Data-Driven Value at Scale* (1.ed - preview version), O'Reilly Media, Inc.

Dickopf, Th. (2020), *A holistic methodology for the development of cybertronic systems in the context of the Internet of Things, [PhD Thesis]*, Kaiserslautern. https://doi.org/10.2370/9783844073690

Eigner, M. (2021), *System Lifecycle Management - Digitalisierung des Engineering*, Springer Vieweg, Berlin. https://doi.org/10.1007/978-3-662-62183-7

Evans, E. (2003), *Domain-Driven Design: Tackling Complexity in the Heart of Software*, Boston, Addison-Wesley Professional.

Highsmith, J.; Luu, L.; Robinson, D. (2020), *EDGE: Value-Driven Digital Transformation*, Boston, Addison-Wesley.

Hooshmand, Y.; Adamenko, D.; Kunnen, S.; Köhler, P. (2017), *An approach for holistic model-based engineering of industrial plants*, 21st International Conference on Engineering Design (ICED17), Vancouver.

Hooshmand, Y.; Adamenko, D.; Kunnen, S.; Köhler, P. (2018), *Semi-Automatic Creation of System Models based on SysML Model Libraries*, INCOSE EMEA Sector Systems Engineering Conference, Berlin.

ISO/IEC/IEEE (2015), *ISO/IEC/IEEE 15288:2015: Systems and Software Engineering - System Life Cycle Processes*, International Organisation for Standardization, Geneva.

Narayan, S. (2018), *Products Over Projects*, [online]. Available at: https://martinfowler.com/articles/products-over-projects.html (accessed 05.11.2021).

Newman, S. (2019), *Monolith to Microservices: Evolutionary Patterns to Transform Your Monolith*, O'Reilly Media, Inc.

Nowack, B. (2009), *The Semantic Web – Not a Piece of cake…*, [online]. Available at: http://bnode.org/blog/2009/07/08/the-semantic-Web-not-a-piece-of-cake (accessed 18.07.2021).

Pfenning, M.; Reichwein, A.; Hooshmand, Y.; Dickopf, T.; Bleisinger, O.; Psota, T.; Masior, J.; Roth, A.; Muggeo, C.; Hutsch, M. (2021), *Killing the PLM Monolith – the Emergence of cloud-native System Lifecycle Management (SysLM)*, Fraunhofer-Publica Verlag.

Pfenning, M. (2017), *Durchgängiges Engineering durch die Integration von PLM und MBSE, [PhD Thesis]*, Technical University Kaiserslautern.

Sneed, H.; Verhoef, Ch. (2019), *Re-implementing a legacy system*, Journal of Systems and Software, Volume 155, Pages 162-184. https://doi.org/10.1016/j.jss.2019.05.012

Taibi, D.; Lenarduzzi, V.; Pahl, C. (2017), *Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation*, IEEE Cloud Computing, vol. 4, no. 5, pp. 22-32. doi: 10.1109/MCC.2017.4250931

Vernon, V. (2016), *Domain-Driven Design Distilled*, Boston, Addison-Wesley Professional.

W3C (2015), *Semantic Web - Leading the web to its full potential*, [online]. Available at: https://www.w3.org/standards/semanticweb (accessed 18.07.2021).

https://doi.org/10.1017/pds.2022.73 Published online by Cambridge University Press