Enabling Model-Based Engineering in a CAx-Neutral PLM System

A case study for the implementation of semantic networks into Bluestar PLM

Johannes Bach Larsen
Manufacturing Technology, 2025-08

Master Thesis - 30 ECTS

Department of Materials and Production
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Enabling Model-Based Engineering in a CAx-Neutral PLM System

Theme:
Product Configuration of Complex Technical Products

Project Period:
Spring Semester 2025

Project Group:
N/A

Participant(s):
Johannes Bach Larsen
<Jolars20@student.aau.dk>

Supervisor(s):
Ole Madsen

Copies: 1

Page Numbers: 47

Date of Completion:
July 31, 2025

Use of Generative AI:
Generative AI, in the form of ChatGPT, third-party modifications of ChatGPT, and Microsoft Copilot, was utilized to summarize findings, rephrase human-written text, and identify relevant academic sources for use in the thesis.

Abstract:

This study examines the application of semantic networks within Bluestar PLM to support high-variance, low-volume engineering manufacturers, particularly through a case study with ETO LLC. These companies often face challenges automating data validation and transformation due to unique internal logic and the lack of built-in modeling capabilities in Bluestar PLM. The research explores digital modeling technologies, including model-based systems engineering (MBSE), model-based enterprise (MBE), product-process-resource (PPR), and Digital Twins, identifying semantic networks as a foundational approach to encapsulate reusable logic and reduce manual work. A new Bluestar entity, the data model, is introduced to represent these semantic networks, and an iterative development process validates their feasibility. While some complexity in feature adoption may conflict with Bluestar PLM's self-implementation strategy, the approach promises significant reductions in data validation effort and improved integration with Microsoft Dynamics. The study concludes by outlining necessary extensions for broader customer adoption, including support for diverse CAx toolchains and potential alignment with MBSE toolsets.

Executive Summary

Bluestar PLM's customers are high-variance, low-volume engineering manufacturing companies. These companies choose Bluestar PLM because they want to synchronize their engineering data and designs with commercial and production operations. Bluestar PLM's users often struggle with representing the data relationships between objects and files, even though these are easily explainable and formalizable. However, their internal technical and business logic is often unique to them and thus cannot be easily automated by Bluestar PLM, even with its extensive settings, requiring either manual data entry and validation work or a bespoke customization by Bluestar PLM's developers.

This study therefore explores the implementation of digital models within Bluestar PLM to enable ETO LLC's transition. It investigates key digital modeling technologies, such as Model-Based Systems Engineering (MBSE), Model-Based Enterprise (MBE), Product-Process-Resource (PPR), and Digital Twins, and identifies semantic networks as a foundational enabler for representing data relationships in and between digital models and furthering Bluestar PLM's competitive edge in its integration with Microsoft Dynamics by improving data transfer and interpretation.

ETO LLC serves as a use case, revealing that their current operations involve significant manual data validation and transformation, which could be algorithmically automated but currently require costly custom software. The study proposes using semantic networks to encapsulate recurring business and technical logic at the class level within Bluestar PLM, reducing the need for redundant work across similar products.

Due to the novelty of semantic networks for both Bluestar's developers and customers, an iterative development approach is employed to validate six core hypotheses. This results in the creation of a new Bluestar entity, called a data model, which is used to represent semantic networks. While ETO LLC's users show the capacity to adopt this modeling approach, some features are complex and would require training, potentially conflicting with Bluestar's self-implementation strategy.

Although the implementation was not finalized at the time of writing due to limited developer availability, detailed specifications were completed. The anticipated benefits include substantial reductions in data validation effort and improved consistency in product development. To generalize the approach across Bluestar's broader customer base, additional features are identified—especially to support integration with diverse CAx tools without requiring changes in customer workflows. Finally, the feasibility of using existing MBSE tools to define these semantic networks is evaluated, and a potential implementation path is proposed.

Thus, to further the development performed in this project, PDM technology should:

1. Continue the development of the already specified development tickets.
2. Perform a feasibility study in creating the data models using external MBSE tools such as Capella.
3. Create new Bluestar PLM object types to represent production processes and resources.
4. Redesign the file-to-object field synchronization system to be more flexible and show file-to-file dependencies.

Preface

This project is my master's thesis and has been supervised by Ole Madsen. It represents the end of my studies at Aalborg University, Department of Materials and Production. As I had coursework during the semester, representing 15 extra ECTS points, the duration of this project has been extended, and has taken place from February to the end of July 2025. The project has been performed in collaboration with PDM TECHNOLOGY ApS. A number of companies have participated in the development performed in this thesis, each of whom are customers of Bluestar PLM and whom I have visited either during or before the writing of this thesis.

Aalborg University, July 31, 2025

List of Abbrieviations

Abbreviation	Definition
ERP	Enterprise Ressource Planning
PLM	Product Lifecycle Management
SysLM	System Lifecycle Management
BOM	Bill Of Material
MBOM	Manufacturing Bill Of Material
EBOM	Engineering Bill Of Material
BOP	Bill Of Procesees
CAD	Computed Aided Design
MCAD	Mechanical Computed Aided Design
ECAD	Electrical Computed Aided Design
CAx	Electrical Computed Aided Design
FEA	Finite Element Analysisis
CPS	Cyber Physical System
IPS	Impact Propagation Scope
EOS	Extended Object Scope
F&SCM	Finance and Supply Chain Management
BSPLM	Bluestar Product Lifecycle Management
SysML	System Modelling Language
UML	Unified Modeling Language
API	Application Programming Interface
ETO	Engineer-To-Order
CTO	Configure-To-Order
PDMt	PDM Technology ApS

Table of Contents

1 Introduction to Bluestar PLM and ETO LLC **1**
1.1 A Troublesome Implementation - An Introduction to ETO LLC . . . 1

2 Models in Digital Manufacturing and Engineering **3**
2.1 What is a Model? . . . 3
2.2 Digital Twins . . . 3
2.3 Model-Based Systems Engineering . . . 5
2.4 Product-Process-Resource Models . . . 7
2.5 Model-Based Enterprise . . . 8
2.6 Semantic Networks . . . 11

3 The Challenges at ETO LLC **14**
3.1 Operations Data Structure Modelling . . . 14
3.2 Interobject Data Structure at ETO LLC . . . 14
3.3 Intraobject Data Structures at ETO LLC . . . 15
3.4 Business Case for ETO LLC . . . 18

4 Solution Development **21**
4.1 Hypotheses . . . 21
4.2 Solution Architecture . . . 21
4.3 First Development Iteration . . . 24
4.4 Second Development Iteration . . . 26
4.5 Third Development Iteration . . . 28

5 Discussion **31**
5.1 Usefullness Beyond ETO LLC . . . 31
5.2 Further Development for Current Features . . . 36
5.3 Interactions with CAx . . . 38

6 Conclusion **42**

Bibliography **43**

A Development Tickets for First Iteration **46**

B Development Tickets for Second Iteration **47**

1 Introduction to Bluestar PLM and ETO LLC

Bluestar Product Lifecycle Management (PLM) by PDM Technology ApS (PDMt) in Denmark is a cloud-based product lifecycle management system fully embedded in Microsoft Dynamics 365 Finance & Supply Chain Management (F&SCM). Bluestar is "fully embedded" in Dynamics 365, so engineers and production teams operate on the same data without custom interfaces. This enables non-technical users to access the single source of product truth and for fast data sharing between commercial and technical departments. It is a strategic priority that PDMt's involvement in customers' Bluestar PLM implementations is minimized.

Bluestar PLM has a policy of Computer Aided Something (CAx) neutrality. CAx-neutrality means that for each category, MCAD, ECAD, etc., there are several possible applications from different vendors, and that compatibility with new applications could be developed. Consequently, any developed solution for Bluestar PLM must not be reliant on interfacing between one or mere particular CAx-applications.

Bluestar PLM is primarily aimed at discrete manufacturing, especially engineer-to-order (ETO) and high-variance producers. Typical customers are in industrial equipment/machinery and similar sectors. Many have complex, customized products with thousands of variants and distributed operations. Due to the low volumes per product, their operations are very sensitive to changes in the design time for products. For them to make use of a developed solution, it must not significantly increase the amount of design time per product.

Much has changed in the requirements of these companies since Bluestar PLM's commercial launch. The complexity of products has escalated, both in the multitude of data associated with products, and the inter-relatedness of products has increased [1], likewise the interest of Bluestar PLMs users in representing these relations have increased along with it. Bluestar currently cannot represent this increasing interrelatedness and interdependency between various files and products. However, PDM International ApS, who develop Bluestar PLM, is interested in developing a solution to these challenges. To do this, a particularly troublesome implementation will be used as a customer case, thus, it will be used throughout this thesis, but the solution must also be generalizable to other customers of Bluestar PLM and must remain CAx-neutral.

1.1 A Troublesome Implementation - An Introduction to ETO LLC

ETO LLC is an engineer-to-order (ETO) company that sells, designs, and produces customized steel lighting poles for use in the urban lighting infrastructure. While the company and its data structures have been anonymized, it reflects many of the ambitions and challenges held by other customers of Bluestar PLM.

When creating a new pole for an order, a design is created from the requirements provided by the customer. Once a design has been finished, the raw materials that will be used to produce the pole must be set. The rules for selecting the raw material and routes are straightforward, but due to the large number of new products being designed, a significant number of mistakes are made, resulting in the incorrect type or amount of material being ordered. Currently, this is all manually entered and validated, which substantially increases lead time.

ETO LLC produces the same kinds of poles from several different production facilities, these however have different equipment and thus have different constraints, such as which geometries they can accommodate. These constraints are not easily traceable, as the constraints are only available inside design documents. Applying a new machine's constraints is thus tedious and time-consuming. ETO LLC has repeatedly emphasized that it aims to create digital twins of its products to monitor the condition of poles and perform Finite Element Analysis (FEA) using the simulation tool Ansys after the point of sale.

Post-approval changes may still occur due to changes in customer requirements and design mistakes. For each change, the impacts must be estimated and sent to the customer for approval. This requires ETO LLC to

create an estimate on the impact on the mechanical properties and the production schedule. Next, the technical changes often have commercial impacts, thus, the people required to estimate or approve the changes can vary depending on many different factors, today, this is tracked manually in spreadsheets.

To summarize ETO LLC's ambitions, they have been enumerated below. The ambitions and the lifecycle stages where they would be achieved is shown in Figure 1.1

1. Automatically creates and updates derived product documentation in Bluestar.
2. Reduce the amount of time needed for manual data entry and validation.
3. Be able to determine a change's impact on cost, production, and lead time before it is committed to.
4. Be able to determine the persons responsible for enacting and approving a given change.
5. Be able to select the raw materials and routes automatically.
6. Create digital twins of their products.

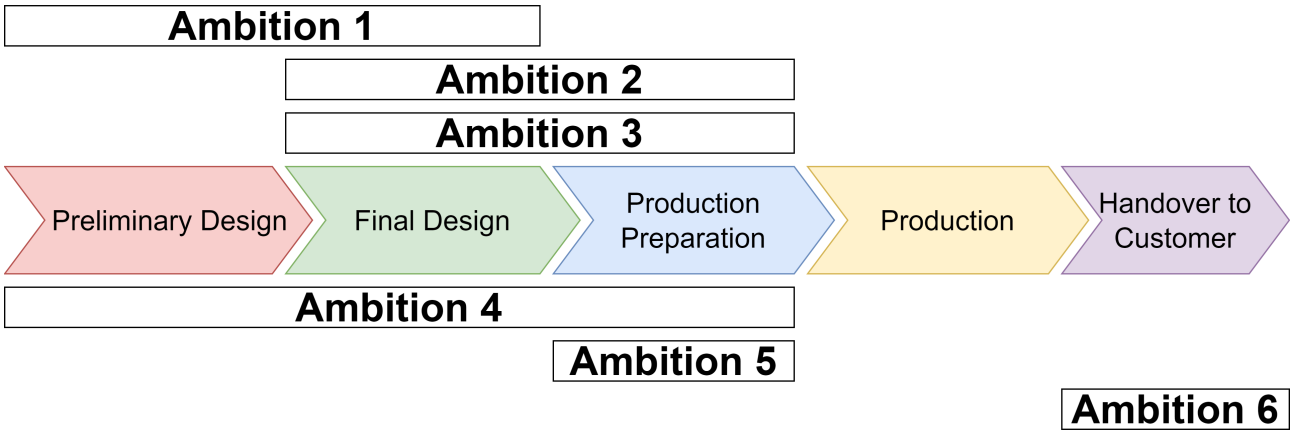


Figure 1.1: The digital ambitions of ETO LLC spread out across the lifecycle of a pole. All except digital twin creation is performed before production and handover to the customer.

In order to investigate the feasibility of and methods to fulfill these wishes, a number of technologies related to the use of digital models will be investigated. Identified technologies were then chosen based on interest from ETO LLC and Bluestar PLM's management. The choice to focus on the use of models is due to the ability of a single authoritative data source to ensure data accuracy, consistency, and effective product lifecycle management. As well as support digital product development and lifecycle phases by enabling digital representations of real processes and products [2]. Due to the high-variance, low-volume nature of Bluestar PLM's customers, an emphasis will be given to the ability to automate the effort related to the digital models.

2 Models in Digital Manufacturing and Engineering

The pressure on companies to shorten development times, deliver complex multi-disciplinary products, reduce costs, etc. has resulted in an ongoing digitalization of manufacturing and engineering operations. These operations have long been bound to the use of documents and design files, though many have called for a transition to models, including senior leadership at Bluestar. This chapter will first identify a number of ways models can be used during digital engineering and manufacturing, then an enabling technology in the form of semantic networks will be discussed. Next, a challenging case from one of Bluestar PLM's customers will be presented as a use case for the use of semantic networks, which will set the objective of the rest of this thesis.

2.1 What is a Model?

The concept of models is often used in many different industries, from manufacturing to fashion and art. This thesis will use the following definitions as defined by [3]. This defines that a model can be understood as an approximation, representation, or idealization of selected attributes or characteristics pertaining to the structure, behavior, operation, or conceptual nature of a real-world system, process, or phenomenon. It functions as an abstraction that encapsulates essential properties of its real-world counterpart, enabling users to engage with complex entities in a simplified or structured manner. Models are instrumental in communicating design intent, simulating real-world behaviors, analyzing performance, or defining operational processes. In the context of manufacturing, a model typically exists as a digital artifact that is created and utilized within one or more software environments to support engineering, planning, or production-related activities.

Digital models may assume multiple views or representations, each tailored to fulfill the requirements of distinct functional domains. These views reflect the system from the perspective of specific users or disciplines, such as design, simulation, production, or quality assurance. Each model exists for a given purpose and is constrained to a given view. Meaning that models can exist in different levels of abstraction and in different levels of granularity. Thus, some models may represent entire organizations and supply chains, while others can represent the buckling of one component under a load. With the concept of a model defined, a number of digital model related technologies in engineering or manufacturing will be outlined.

2.2 Digital Twins

The use and development of digital twins have received much interest from both academia and industry. There is often disagreement on the specific definition of digital twin versus digital models. For the scope of this paper, the term digital twin (DT) will be distinguished from the terms digital model (DM) and digital shadow (DS). The differentiating factor is the data interchange between the physical and digital, and vice versa, as seen in Figure 2.1.

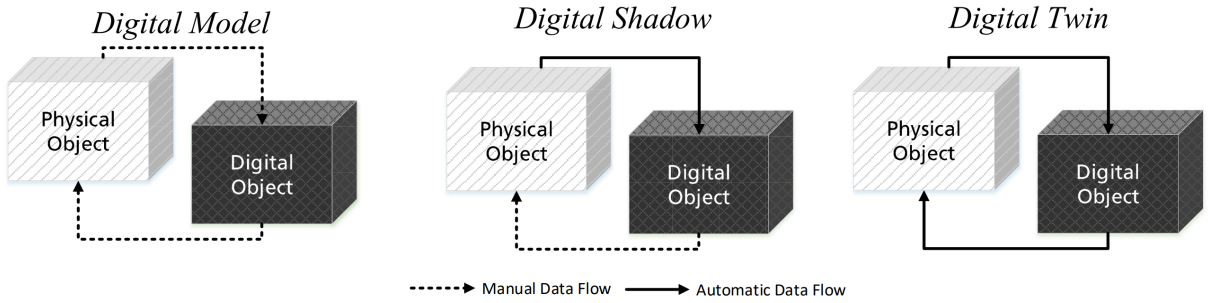


Figure 2.1: The differences between digital model, shadows, and twins lie in the level of automaticity of data transfer as described by [4]

The digital twin and shadow include three sub-systems: the physical product, the virtual product, and the connection between them. The virtual object can again be split into two interacting parts, being the digital model and the gathered physical world data. The digital twin has found applications throughout the lifecycle stages and in three main use cases within manufacturing: production planning and execution, maintenance, and simulation. [4, 5]

1. **Manufacturing planning and execution**
    1. **Manufacturing execution** is performed by *digital twins* by using sensors to apply feedback to existing digital models for the manufacturing system, such as using a digital twin to perform mobile robot navigation [6] or real-time production scheduling and monitoring [7].
    2. **Manufacturing Item Tracking** can be performed by *digital shadows* to track the state of complex physical objects on the production floor, thus allowing the entire organization to access the updated model of the physical object as it crosses through the production.[8]

2. **Maintenance**
    1. **Predictive maintenance** may be performed by using a *digital shadow* to predict whether the physical object will require maintenance. This is performed by using the feedback from the physical object in a predictive model, which is made to perform the prediction.[9–11]
    2. **After-sales Servicing** is sometimes performed on a physical object. Since it is after the point of handover, the manufacturer does not have access to or knowledge about the physical object daily. Here a *digital model* is continually updated, and subsequent servicing will know the last recorded state of the machine physical object [12–14]

3. **Simulation**
    1. **Product, Process, and Plant** is simulated through *digital models*, especially at the beginning of the product lifecycle, to validate the design, such as through simulations like finite element analysis (FEA), multiphysics simulations, and discrete event simulations.[5, 15–17]
    2. **Worker training** is performed by using *digital models* to allow the user to interact with simulations of the real object, such as through the use of virtual reality or augmented reality, which includes the digital models of the product, process, and plant.[5, 18, 19]

All of these concepts are based on the use of a digital model, in the case of a digital shadow and digital twin, this model is then combined with feedback from the real physical systems, such as can be seen in Figure 2.2. As the digital model predates the physical object, the digital twin itself has to cross a threshold during its lifecycle, thus, its lifecycle can be segmented into at least two phases: one before and one after instantiation. Before instantiation, the digital twin does not exist; only the digital model does. Only after instantiation is the model fed with data from the physical world to become a digital twin; the instantiation thus also makes the particular instance of the digital twin dynamic.[20, 21]

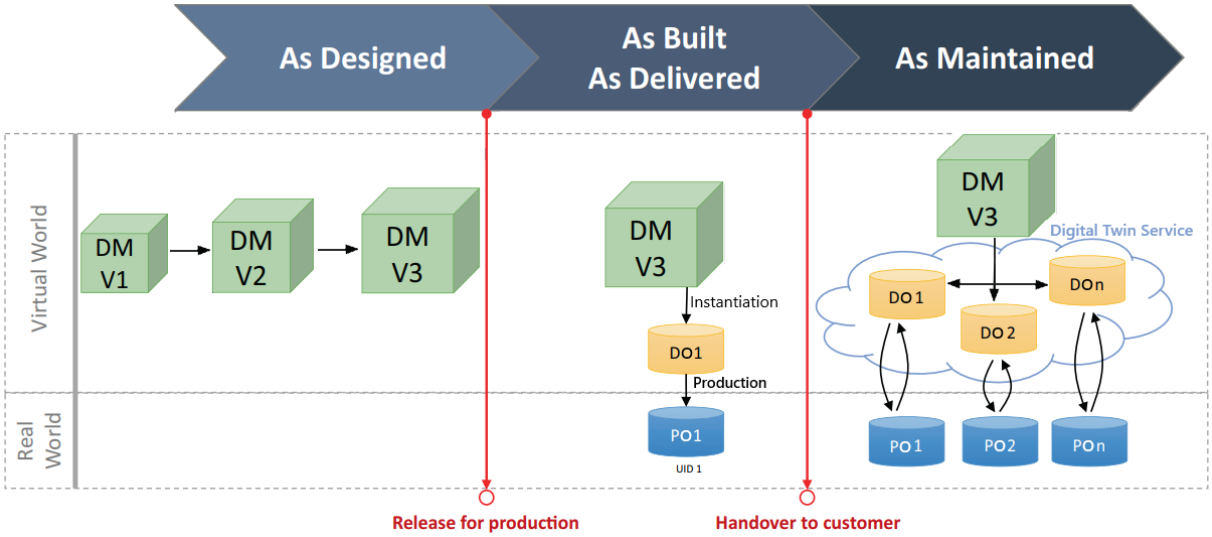


Figure 2.2: An overview of the relationship between the digital model, DM, digital object, DO, and physical object, PO, along the lifecycle of the product. Note that the customer may be both internal and external, and the product may also be manufacturing equipment not sold by the given company. The digital twin service is the backend service responsible for the reception of data from POs, application of this data to models, updating DOs and lastly transmitting to the PO. Figure adapted, original from [21]

As the digital model may change over time, a system must be in place to replace or revise the active models connected to a physical object. Thus, a central repository for the models must be available such that the newest version of the model is available for the instantiation of new digital objects and to overwrite old, faulty models that have already been instantiated [22]. As suggested by [23], the digital model is to be managed by the PLM system, as the definition and revision of a digital model is intrinsically linked to the work of engineering and falls within the scope of PLM.

The architecture and feature set of Bluestar PLM does not easily support the instantiation and maintenance of the digital objects, such projects have previously been attempted, though with little succes. Instead, a separate digital twin application should be responsible for instantiating the digital model into a digital object and maintaining the connection to the physical object, creating the digital twin.

ETO LLC has voiced its interest in creating digital twins of its products. However, there is no data to be transmitted from ETO LLC's models to the poles themselves. Therefore, ETO LLC is looking to create digital models and digital shadows, but not digital twins. Furthermore, the instantiation and management of the digital objects and the communication with the physical objects should be relegated to a separate digital twin application. whereas the management and vaulting of the digital models fall within the scope of PLM. Therefore, the creation and representation of digital models is to be investigated.

2.3 Model-Based Systems Engineering

As products and systems have become more complex, the importance of early design decisions for total lifecycle cost has not decreased [24, 25]. This is also seen at ETO LLC, which has invested heavily in flexible, customized automation. However, the design and requirements management for these production systems have been onerous and error-prone, as ETO LLC has no way to express the interactions between these automation systems and the surrounding production and products beyond the text documents.

A potential solution to this is the multi-disciplinary design paradigm called *model-based systems engineering*, which aims to perform model-driven development instead of a document- or file-centric development process.[26] MBSE is broadly defined by the International Council on Systems Engineering as "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle". [27]. Thus, models are used to drive or support the analysis, specification, design, and verification of the system being developed [28]. The goal is a single source of truth where system requirements, logical architectures, and physical product data coexist. This supports model-based analysis of the product lifecycle and

continuous model-based verification in manufacturing contexts [29, 30].

In practice, MBSE replaces much of the traditional document-based approach with a focus on developing and maintaining explicit system models throughout development and operation. This paradigm shift addresses the data inconsistencies arising in document-centric processes by centralizing knowledge into models rather than disjointed files, reports, and spreadsheets. The emphasis on modeling yields better communication across disciplines and thorough traceability of requirements and design decisions. Thereby, MBSE provides a structured means of capturing a system's architecture, behavior, requirements, and interfaces in an integrated form, which streamlines collaboration and helps detect design faults early in the process.[26] MBSE often follows the Requirements-Functional-Logical-Physical (RLFP) development framework, often combined with the V-model for engineering of complex products. The resulting structure can be seen in Figure 2.3.

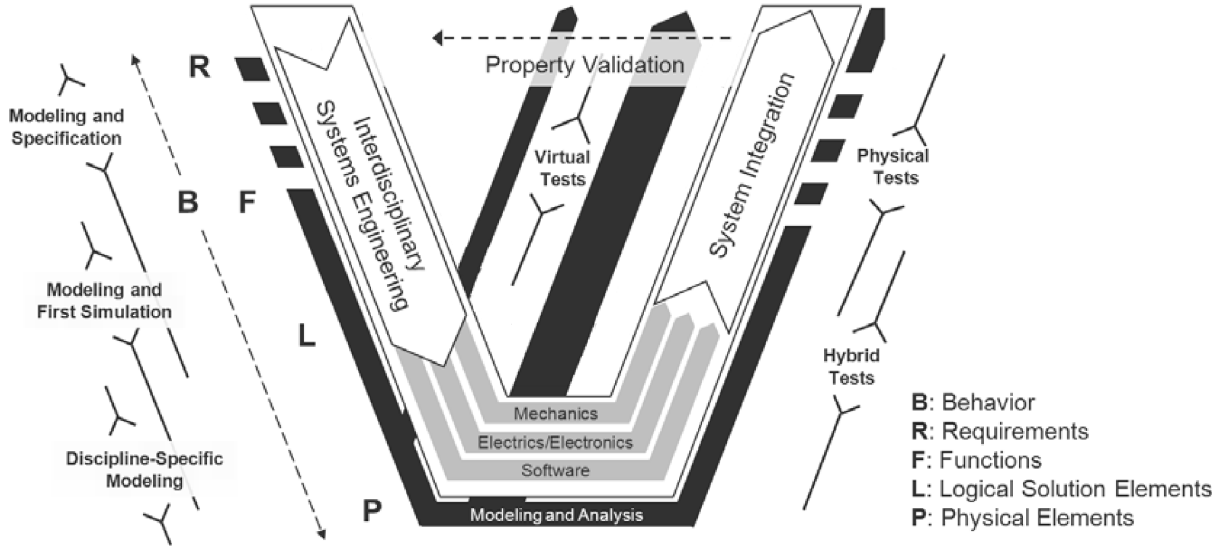


Figure 2.3: The V-model used to guide the development process of complex products and systems using MBSE. Figure adapted, original from [31]

The V-model breaks the development of complex systems into design and integration, the two edges of the V. The design is comprised of initial interdisciplinary systems engineering, which defines the main requirements and functional definitions of the system. This is then used to guide the discipline-specific modelling by defining the logical and physical modelling of the system. Afterwards, the output of the different disciplines is then integrated and tested to determine whether the original architecture and requirements are upheld. The V-model shown here places a large weight on digital tools for modelling, simulation, and testing both for the discipline-specific and interdisciplinary models before the execution of physical tests.

As described by the definition given above, MBSE is *formalized*, meaning that the modelling is performed using a formal or semi-formal modelling language such as Unified Modeling Language (UML), Systems Modeling Language (SysML), or Cappella [32]. These formalized tools allow for standardized system modelling across systems and applications since the meaning and rules of various symbols and relations are formally given by the laguage used. The net result is enhanced coordination across design teams and disciplines, requirements, interfaces, and constraints in centralized models that can be shared throughout the organization. Such model-centric practices support concurrent engineering and reduce costly iterative loops in design.[26]

Despite their complementary goals, MBSE and PLM have historically evolved separately. The full-scale integration of MBSE into PLM would allow linking system models to downstream product data, whereby continuity and traceability across the lifecycle can be created. Integrated MBSE–PLM approaches can yield enhanced communications, reduced development risks, improved quality, increased productivity, and enhanced knowledge transfer when MBSE artifacts, namely requirements, functions, and models, are connected to PLM-managed data [33]. Coupling MBSE and PLM can mitigate excessive complexity and uncertainty by ensuring that system-level changes propagate downstream into domain-specific engineering such as shown in a previous case study, where designers were able to propagate changes and validate consistency across

disciplines [34].

The representation of MBSE in PLM involves both increased data linkage, meaning connecting existing PLM items to MBSE elements, and data type enlargement, by extending PLM to include systems models. Representing MBSE artifacts in a PLM system involves extending the PLM data model to accommodate system-level entities, including components, functions, requirements, and their interrelationships. In practice, this integration often involves linking system-level entities in MBSE models with corresponding parts, assemblies, or documents in PLM.[29]

MBSE is often performed as a combination of language, tool, and method. The artifacts required to be represented in PLM will depend on the chosen language. As suggested by [35], while it has limitations due to its semi-formality, the most popular language is SysML. For instance, Siemens Teamcenter enables integration of product design optimization workflows with system models by treating SysML model artifacts as part of the PLM-managed data[36].

MBSE has emerged as a critical methodology for managing the complexity of modern engineering systems, offering a model-centric alternative to traditional document-based processes. To realize these benefits across the full product lifecycle, MBSE must be integrated into PLM systems. Recent literature and commercial applications describe an approach for representing system models within PLM, from aligning ontologies to enabling SysML data exchange to ensure consistent information flow from requirements to the final product. When MBSE and PLM are harmonized, organizations can achieve a robust link between system architecture with detailed product data, ultimately improving quality and reducing risk when making complex products or systems.

However, Bluestar currently has no way to represent system architecture, functions, interfaces between parts, and requirements. Today, these can only be performed file-centrically using Microsoft Office files to encode them. The system architecture, and so on, can not be extracted and used within the wider Bluestar PLM Environment. One of the challenges for many of Bluestar's customers is the coevolution of product and production systems, thus, a modeling framework for this particular task will be further explored.

2.4 Product-Process-Resource Models

As the use development of computing power, sensors, and the internet-of-things has progressed, the use of cyber-physical systems has become more widespread. These cyber-physical systems have both a physical dimension and a digital dimension, which together allow the system to interact with both the real world and the digital. These cyber-physical systems have seen great use both in products designed and sold. Additionally, they have found use within production systems, which themselves have become cyber-physical systems.[37]
Product lifecycle management has historically focused on the product and the management of product-related data. However, the product is only one part of a wider production context due to the integration and co-evolution of products, the processes for their production, and the resources to produce them. As some of Bluestar PLM's customers transition towards flexible automation and mass customization, the complexities of modern engineering and manufacturing increase, and with it, the need to unify and manage product, process, and resource information becomes critical.[38] A promising solution lies in Product-Process-Resource (PPR), modelling. PPR is a conceptual and data architecture framework that links the three core elements of manufacturing, the products, the processes, and the resources, into a single model. It enables a holistic representation of a system's manufacturing logic by linking what is being made, the product, how it is made, the process, and what is used to make it, the resource. A Simple PPR-structure and its relations can be seen in Figures 2.4 2.4a and 2.4b.[39]

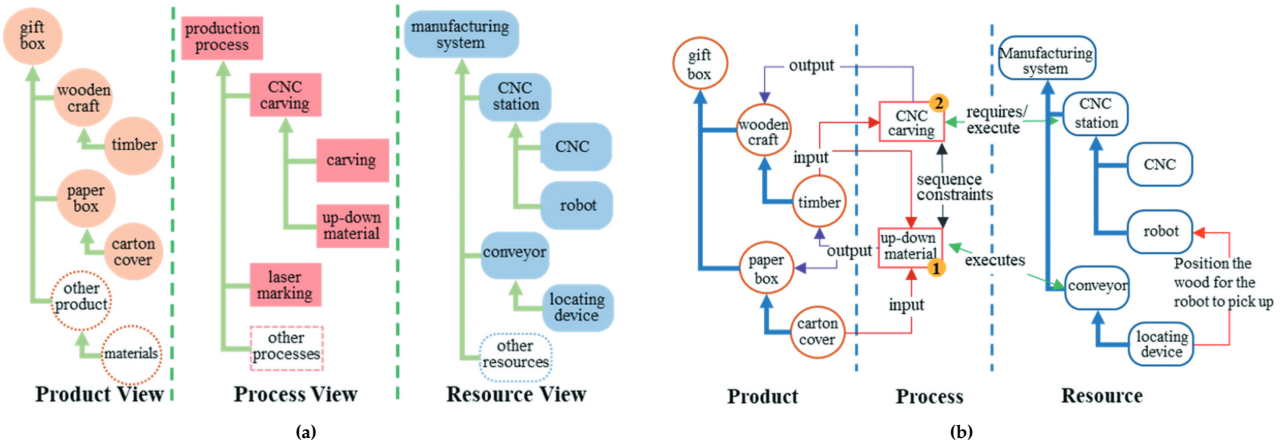


Figure 2.4: An example of a PPR-structure can be seen in Figure 2.4, showing the different associated process and resource views of a simple product. On the right, the relationships between the different views can be seen. Figure from [39]

Traditional engineering tools often segregate product design, manufacturing process planning, and resource allocation, such as creating models and drawings for products in CAD, creating processes in ERP systems, and programming workstations in CAM software. The use of such disparate systems creates media breaks in which the traceability and transfer of data are hindered by the discontinuity between the digital applications, often resulting in manual data transfer to and from each CAx [40]. These applications are rarely integrated due to the use of proprietary file types. Subsequently, the PPR-related knowledge and data are spread throughout these several applications and files. Similarly, the applications used for simulation and validation of a given design may not be integrated with the software used for design. This conflicts with the need to create traceability between them and to perform concurrent engineering [38, 41]. Moreover, ETO, CTO, and other high-variance low-volume enterprises, which make up Bluestar PLM's customer base, will often only use a specific CAD model, process plan, or CNC-machining program a few times or even once, as the given design is often order-specific and may never be used again. Therefore, Bluestar PLM requires a more integrated approach to manage complex assembly lines, custom products, and flexible automation.

By encoding dependencies among these three domains, PPR models ensure semantic consistency and traceability across the product lifecycle. Therefore, the use of semantic modeling in PPR systems can increase system modularity and data reuse, which is essential in highly customized manufacturing [42]. For enterprises performing small-batch productions, PPR enables knowledge representation, reconfiguration of process sequences, and resource allocations based on standardized modules. Thus, PPR supports both product variant reuse and manufacturing adaptability, which are crucial in high-variance scenarios characterized by custom solutions and low volumes.

PLM systems increasingly implement PPR structures into their architectures to support cross-disciplinary collaboration and product data modeling and traceability. The integration of PPR into PLM systems expands them from product-centric databases into comprehensive production lifecycle and information systems. Several modern PLM platforms embed PPR to accommodate not only CAD data but also process planning, robot programming/simulation, and resource allocation [43]. As of the writing of this thesis, Bluestar PLM has a very significant ability to manage product BOMs, relations, data, and files. The foundational artifacts for process and resource representation have been implemented. However, the data granularity and ability to represent semantic links are largely absent for processes and resources. Therefore, the implementation of features to represent the PPR models and linkages into Bluestar PLM would be a significant enabler of further digitalization of engineering and manufacturing for its users.

2.5 Model-Based Enterprise

A model-based enterprise (MBE) is "an organization that adopts modeling technologies, such as Model-Based Definition (MBD) solutions, to integrate and manage both technical and business processes related to product design, production, support, and retirement" [2]. MBE fundamentally relies on annotated models as the

product information backbone. Instead of paper drawings, an MBE uses a digital model to define product specifications. Thus, MBE is a digital engineering paradigm that utilizes a single, authoritative source of product and process data throughout the product lifecycle. This fits with ETO LLC's ambition to use knowledge from engineering and design phases to generate derived product documentation throughout the entire lifecycle. This has often been complicated by ETO LLC having multiple versions of the same product available in different systems. Moreover, MBE fits with the general ambition of Bluestar PLM's customers to create data only once and then reuse it.

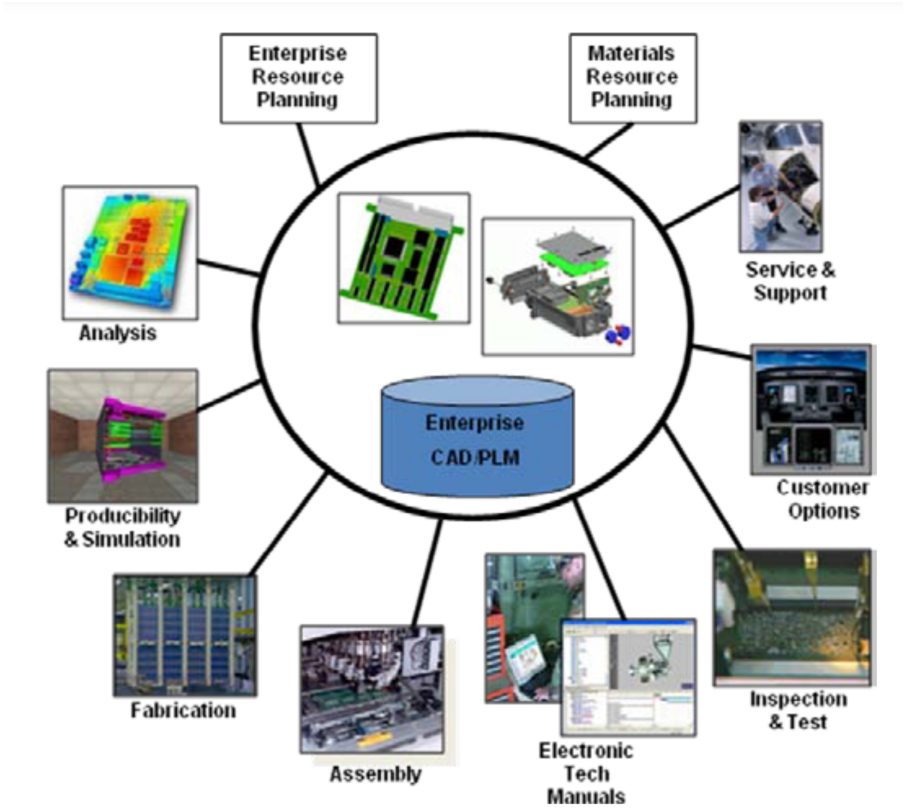


Figure 2.5: An overview of the various data consumers, Enterprise resource planning, analysis, producibility, and simulation, all referencing the same model, contained in the enterprise PLM instance. From [3].

This model is often the 3D model of a given product, but can also represent a process or resource. Through MBD, manufacturing organizations aim to achieve seamless connectivity across design, production, quality, and support processes. The MBD approach means that each product's requirements and manufacturing information are embedded directly in the CAD model, which then drives all engineering, manufacturing, and inspection activities. In practice, MBD involves 3D representations with associated product and manufacturing information (PMI), such as dimensions, tolerances, and annotations. The core principle is that model data is created once and reused throughout the enterprise. This single-source-of-truth principle means that data is created once and directly reused by all data consumers.[3]

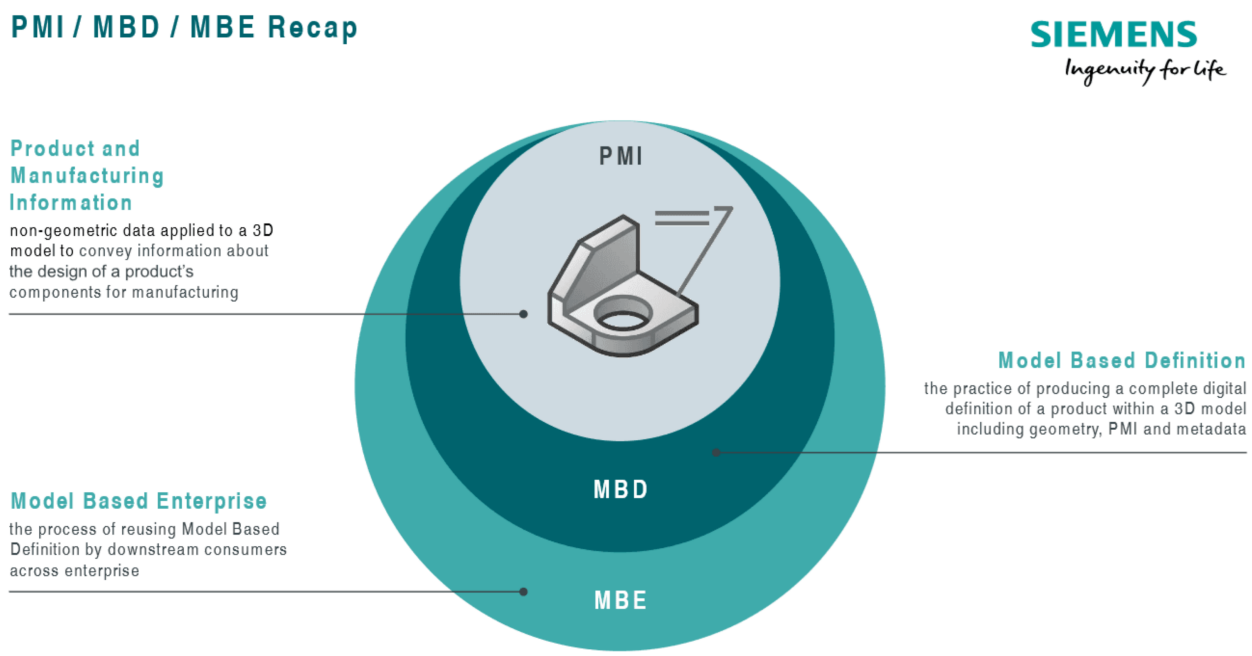


Figure 2.6: The definitions and relationships for product and manufacturing information(PMI) model-based definition(MBD) and model-based enterprise(MBE). From [44].

PLM systems enhanced with MBE capabilities support not only digital mock-ups but also simulation data, tolerance analysis, and manufacturing instructions, effectively forming the backbone of a digital model. The application of model-based impact analysis improves change predictability in product development cycles by embedding semantic knowledge into PLM-managed models.[45, 46]

MBE integration with PLM also requires new data structures. Modern PLM strategies must accommodate annotated 3D CAD models, importantly some of the data for the model may originate outside the CAD model it self, an example could be annotating the expected welding speed required, which is derived from a model of the welding process. Furthermore, the product structure, change processes, and quality data in the PLM must reference features on the 3D model instead of separate drawings. Therefore, the transition to MBE is significantly more complicated than showing 3D CAD models on the shop floor.[2, 47, 48]

A model-centric approach makes data management more time-efficient and accurate by preserving a connected product structure. In effect, every downstream artifact, CNC code, inspection plan, and assembly work order is linked back to features of the master model, ensuring that changes propagate automatically. This breaks down traditional information silos between design, process planning, and production. For example, if a design tolerance is updated in the model, downstream manufacturing process models can automatically update parameters. In sum, embedding models enables a contiguous digital thread [2, 49, 50].

Today, Bluestar has made some developments towards enabling MBE. A CAD-neutral 3D viewer has been created, which is accessible directly from the shop-floor management system inside F&SCM. Moreover, the dimensions of a model can be extracted and used within Bluestar PLM. However, the extracted dimensions cannot be used in a system model or PPR relationship, the extracted model dimensions and features are only available within the products own data structure and is entirely without context. Thus, there is no way to access these dimensions from related production processes and resources, or to propagate changes anywhere outside the product itself. Moreover, there is no way to automatically generate derived documentation or model views, such as creating the purchasing BOM of required raw materials dynamically, or other product-related documentation, inside Bluestar PLM. Which is one of ETO LLC's main ambitions.

Since the data in the domain-specific models can impact the wider interdisciplinary model or another domain, the propagation of data throughout the model becomes imperative. Among the primary challenges is the representation of a wide variety of data types and sources [51]. As greater amounts of interdisciplinary and domain-specific data are included in PLM systems, changes in a given field must be dispersed throughout the wider system model or enterprise data architecture. Thus, a system architecture is to be investigated that allows for the representation of the relationships between heterogeneous data types.

2.6 Semantic Networks

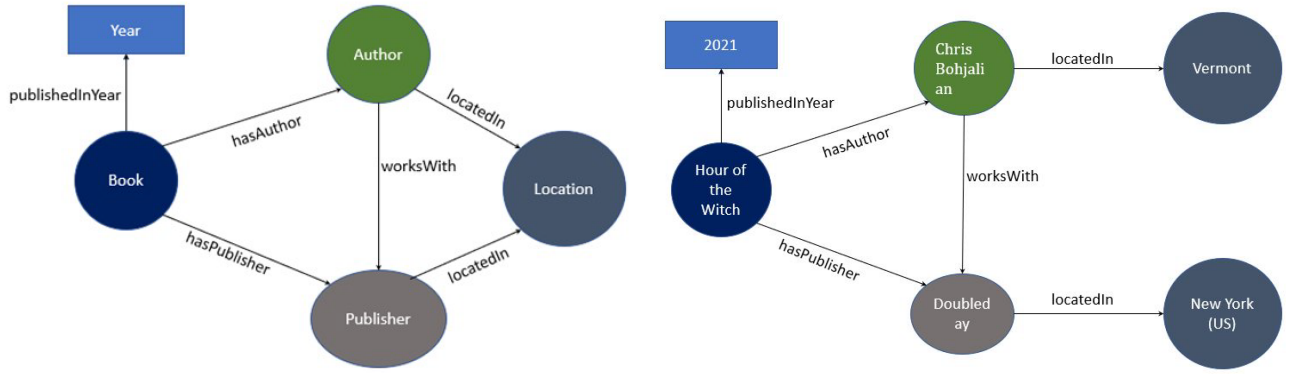
In digital engineering, the different engineering disciplines often use different tools to create their data artifacts, the integration of these is often challenging, as the relations and interactions of these different models are use-case-specific. Therefore, the interdisciplinary knowledge sharing and creation of a shared data model for concurrent engineering often require custom middleware or integrations, which can increase costs, process rigidity, and errors [52]. The use of digital models as the authoritative source of information throughout the product lifecycle results in a need to enable humans to interpret model content and relationships. Furthermore, if Bluestar PLM's customers are to use digital models, the efforts related to digital models must be automated, and thus the models' contents and relationships must be algorithmically interpretable. An example of the required semantic capabilities for interdisciplinary engineering collaboration can be seen in Figure 2.7.

Production System Engineering Needs	
N1	Explicit engineering knowledge representation
N2	Engineering data integration
N3	Engineering knowledge access and analytics
N4	Efficient access to semi-structured data in the organization and on the Web
N5	Flexible and intelligent engineering applications
N6	Support for multidisciplinary engineering process knowledge
N7	Provisioning of integrated engineering knowledge at production system runtime

Figure 2.7: The semantic needs of multidisciplinary machine and production engineering for digitalization and traceability. Adapted from [52].

Within this context, semantic networks can be used as representations of knowledge where data or artifacts are connected through defined relationships, becoming an essential enabling technology. Semantic networks use nodes and edges to represent data and relationships between nodes, thereby enabling knowledge representation by describing the ontology of heterogeneous data sources and concepts.[52, 53]

Semantic networks require an ontology, which is a formalized description of how various elements relate to each other. The formalization is important as this allows the given relationships to be algorithmically processed by machines. Thus, the formalization of engineering knowledge into an ontology can allow various engineering disciplines to interact, but also allow the use of the data artifacts created by those disciplines to be used in a wider algorithmically executed context [52]. The difference between the ontologies, semantic networks, and the semantic webs is often confused, the semantic web often refers to *the semantic web* being the encoding of semantics into the internet, while this is related to the use of semantic networks, the semantic web is beyond this thesis. Semantic networks are a form of knowledge representation that uses an ontology. For this thesis, a semantic network is an instantiation of a given ontology as seen in Figure 2.8b.



(a) An ontology showing the relationships between the different types of artifacts. From [54].

(b) A semantic network showing the relationships between instantiations of the types of artifacts. From [54].

Figure 2.8

A semantic integration model in PLM for service and product lifecycle convergence has been presented in [55]. Their work demonstrates how ontologies and linked data principles allow for lifecycle traceability and knowledge reuse. These ideas can be extended with formal semantic annotations that bridge engineering models and process models in PLM, thus enhancing both syntactic and semantic interoperability [56].In conclusion, semantic networks are an enabler of the use and fusion of heterogenous data, providing the structure and meaning necessary for interoperable, automated, and intelligent PLM systems.

Formalizing semantic annotations in engineering models can enhance consistency and machine readability. Thus, through formalization, semantic networks reduce ambiguity and can automate downstream manufacturing tasks, which is essential in realizing the vision of MBE [47]. The increased data consistency can reduce the manual data validation workload at ETO LLC. Furthermore, the formalization allows the model to simulate the impact of a given change. Both of which were an ambition for ETO LLC.

The previously mentioned concepts of digital twins, MBSE, PPR, and MBE all rely on creating or using a model of a given system. Digital twins require this because they require a digital model to instantiate and through which real world feedback can be processed. MBSE creates and analyzes multiple interacting layers of interdisciplinary and domain-specific models. PPR requires that the relationships between the various products, processes, and resources are known and that they are represented in a model. MBE requires that the various files or models used to represent the product and the derived manufacturing information are interoperable with each other. Thus, each of these relies on the modelling of heterogeneous artifacts and data from various sources, to use the known and defined model to derive some digital output, be it updating the state of a digital twin, creating or enforcing rules between product, process, and resource, and so on. Therefore, each of these requires the existence of a semantic model to unify the heterogeneous data and derive the output. Thus, to allow a customer of Bluestar PLM to transition from being file-based to being model-based, Bluestar must implement the use of semantic networks, which will be the focus of the remainder of this thesis.

Summary

This chapter has analysed a number of innovations related to the use of digital models. Firstly the creation and management of digital twins was considered but found to be outside the scope of Bluestar PLM, though the digital model that underpins the digital twin should be managed by Bluestar PLM. Next, MBSE is analyzed as a potential solution to represent the complex interactions between ETO LLC's products and production resources. This led to the introduction of PPR for modelling the interactions between product, process, and resources. Which cannot currently be performed in Bluestar PLM due to an inability to represent the relations. Because Bluestar PLMs users are high-variance producers, they are not willing to create new models for every new product, the creation must be easy, and the modelling must be easily reuasble. Because of this, MBE was introduced to enable the use of models as the single source of truth throughout the entire product lifecycle. This also showed the need to be able to represent the dependence on model data in Bluestar PLM, which is not currently possible. Finally, semantic networks were explored as a way to represent the relationships between different artifacts and models and to ensure consistency between them.

Project Objective The objective of this project is to be able to achieve the stated digital ambitions of ETO LLC by implementing semantic networks into Bluestar PLM by designing a prototype of the interface and functionality of semantic networks in Bluestar PLM. Next, it is to determine the viability of using semantic networks at ETO LLC and other customers. Thus, the solution and further analysis will not include the use of ETO LLC's CAx toolchain, as none of the available developers at PDMt is qualified to perform this development. Therefore, this thesis will proceed with a Bluestar PLM-only solution. The use of CAx-applications along with the developed solution is discussed in Chapter 5. Given the focus made in this chapter on creating semantic networks in Bluestar PLM, the development of the solution will focus on enabling this for ETO LLC. Thus, this company will be used as a case study, and the solution will be developed first for them. This process will be shown in Chapter 4. To gauge the generalizability to other Bluestar PLM Customers, two further customers are discussed in Chapter 5.

How can a CAx-neutral semantic network be implemented into Bluestar PLM to fulfill ETO LLC digital ambitions?

3 The Challenges at ETO LLC

In this chapter, the features that must be developed within Bluestar PLM to satisfy the ambitions of ETO LLC will be identified. To do this, the chapter will present the current data structures, show pseudocode for the manually defined algorithms, and lastly, a theoretical business case will be given to determine the effect of achieving ETO LLC's digital ambitions. First, however, a short note on the structure of data in Bluestar PLM.

Bluestar PLM consists of objects. Objects inherit from a generic data table template, which is then instantiated into a specific object. The object thus inherits the existence of the data fields, but may have its unique value in the given fields. A single object may contain several files of various kinds. There are several kinds of objects, the primary among them is the **engineering object**, which will be used heavily in this and later chapters. An engineering object is used to represent one part, assembly, or document, thus, an engineering object will be defined by and contain one source file and its derived files, such as a SolidWorks model, the drawing, and a STEP model.

Bluestar PLM is embedded into Microsoft Dynamics 365 Finance & Supply Chain Management (F&SCM), the objects in Bluestar PLM and F&SCM all exist within the same database. Therefore, the objects can be linked directly between the two applications, which is a key differentiator between Bluestar PLM and the competing PLM providers. Therefore, the interactions between the objects in Bluestar PLM and F&SCM are a high priority for ETO LLC, and will be considered even though the Bluestar PLM application does not directly manage them. Before the use cases can be presented, an introduction to the used methodology of data structure modeling is provided.

In this and later chapters, the term data structure will refer to a set of several objects that in some manner depend on one another. Meaning that they share data between them. This data structure can comprise objects both in Bluestar PLM and in F&SCM.

3.1 Operations Data Structure Modelling

A central challenge in modeling the use cases is the complexity of the relationships. Many modeling languages utilize HAS-A and IS-A links to describe composition and inheritance data relations, respectively. However, the relationships between many technical data elements are not so simplistically defined. The relationship between an assembly and its constituent parts may be described as a HAS-A relationship. However, some parts are unique to the assembly while others are used in many assemblies. Therefore, the assembly may have a level of ownership over the first but not the latter, thus, the assembly's relationships with the two parts are different. Therefore, this thesis will make use of two different link types, referencing and joining. Both indicate the sharing of information between two objects, however, a reference relationship is directed. One is a data supplier, the other a data consumer, a joining relationship is undirected, allowing both objects to retrieve data from the other.

Using these relationships, three examples of data structures at ETO LLC will be represented. These data models signify the AS-IS data relationships between objects, fields, or sub-tables within those objects. Through these, the features required to fulfill the ambitions laid out in Chapter 1 will be identified.

3.2 Interobject Data Structure at ETO LLC

Because ETO LLC designs new products for every customer order, much of the products is linked through joining relationships. The resulting structure is a hybrid of customized and standard objects. Moreover, as can be seen in Figure 3.1, the poles, in this case denoted as "Pole Assembly", are also connected to other objects, the data structure therefore features objects both in Bluestar and in F&SCM.

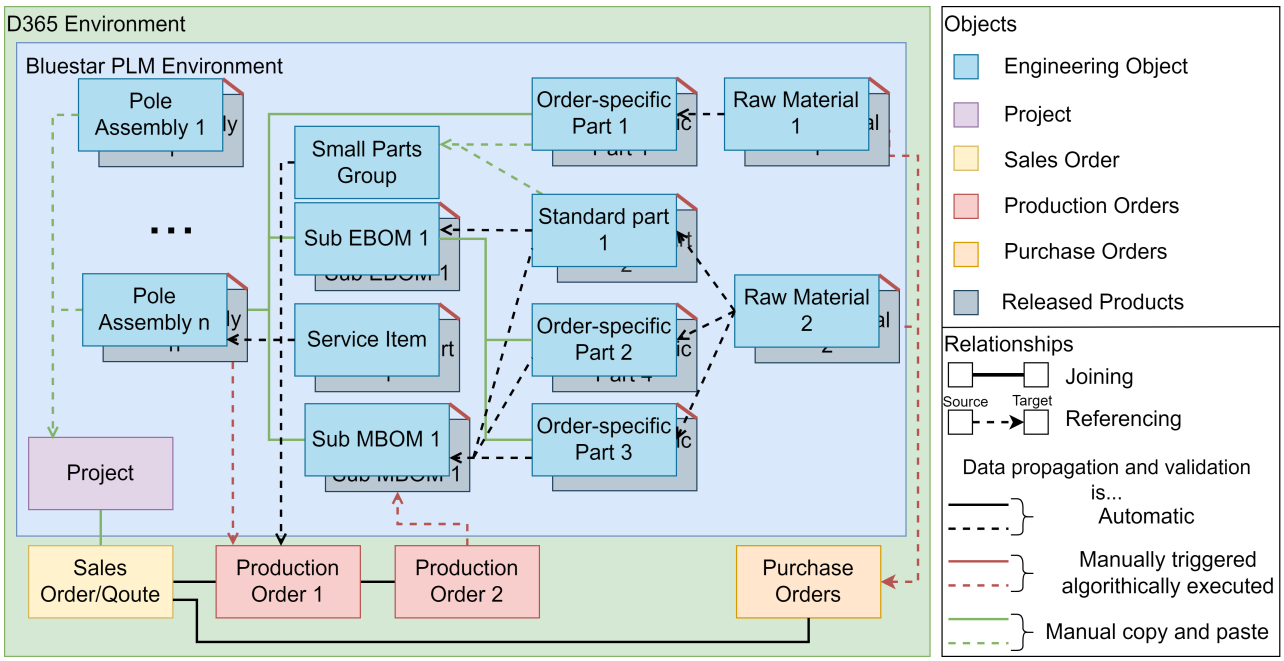


Figure 3.1: An overview of the AS-IS linked objects used in ETO LLC's operations. Showing how one project can be linked to several pole assemblies. The relationships of pole Assembly one have been left out for graphical succinctness.

There are several forms of product documentation in Bluestar PLM, an example can also be seen, the "small parts group", which contains all the small parts related to an order, but may contain parts from several pole assemblies. Because these will be made on the same machine and at the same time, ETO LLC likes to use all of them in one object to create demand for and allocate resources for. However, this object is not used directly in a BOM of any of the poles, as it represents several parts required for several different pole designs. The parts and quantities of the small parts change depending on changes to the design of the poles. To continually update the documentation, the solution must be able to link several engineering objects together, create and update the BOM of a project's small parts group, and there must be a way to determine whether a given object should be included as a small part.

Many of the relationships between objects in F&SCM are automatic or algorithmic, whereas the relationships between Bluestar objects are executed manually by the technical drafters at ETO LLC. It is these relationships that cause the manual data entry and validation that ETO LLC seeks to remove. Therefore, the solution should focus on automatic executions of the relations between objects in Bluestar, with less emphasis on the sharing of data to F&SCM. Thus, the data relations that are currently executed will be analysed in greater depth.

3.3 Intraobject Data Structures at ETO LLC

Firstly, the relations of the main pole assemblies will be analysed, and can be seen in Figure 3.2 shows a simplified relationship between one pole assembly, one of the order-specific parts, a Bluestar Project, and the Sales Order. The impact of a change to one of the sales order's fields can thus be traced through its relationships with other fields. If two fields are connected with a referencing relation, the value of the field is propagated without processing. Fields can also be used in Field Definitions, which are algorithms that produce a field value by processing its inputs.

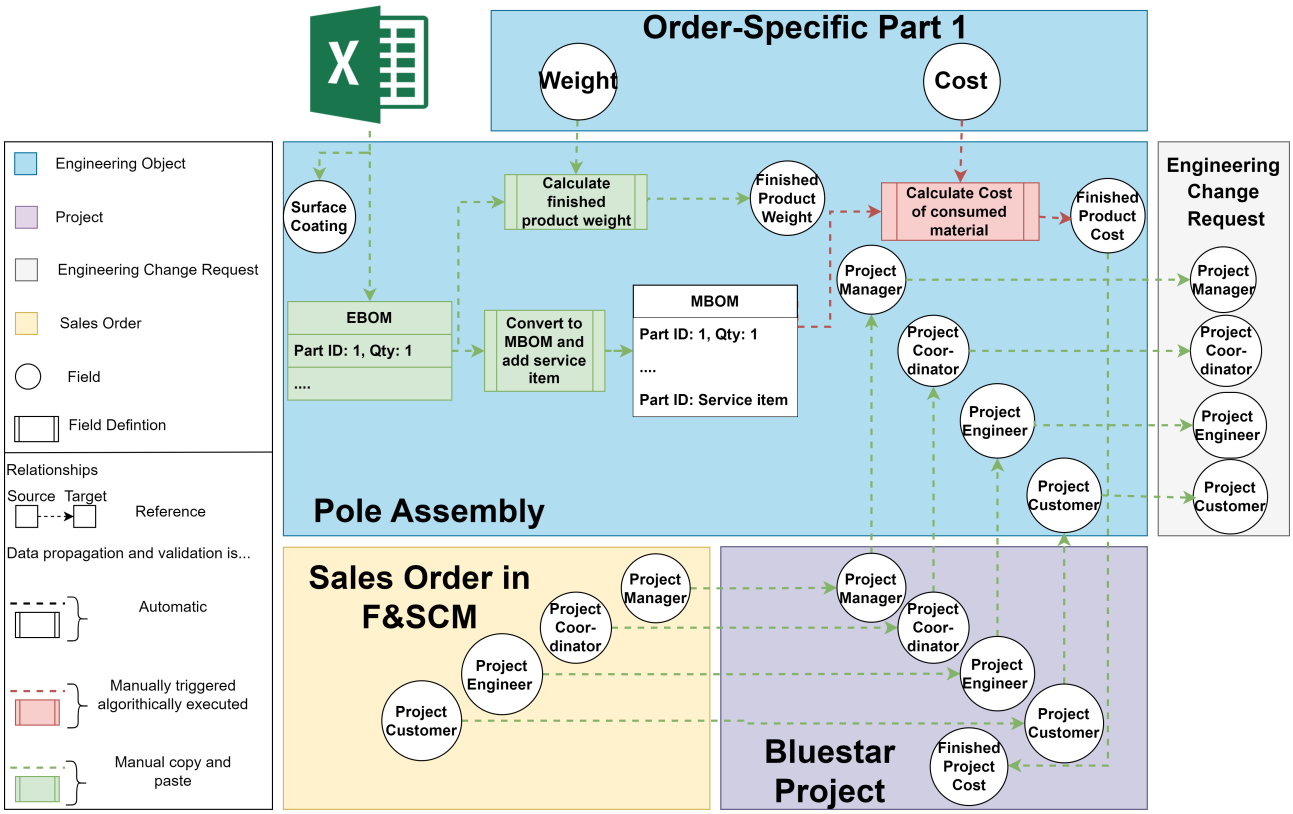


Figure 3.2: The data relationships of the different fields, field definitions, showing relations between objects. The sales order is in F&SCM, all other objects are in Bluestar PLM. The Excel con represents that these data points are first defined in Excel and later entered into Bluestar PLM.

The manual data entry and validation mentioned above are also clearly visible. Especially the four fields defined by the sales order. Whenever a project is created, the technical drafters have to copy this information, especially as there can be many pole assemblies in a project. If this dependence of these fields could be automated by referencing field values on other objects, it would reduce manual workload and allow ETO LLC to set the responsibles for a given change.

The poles' surfaces are treated to increase corrosion resistance, one of the surface treatments is galvanization, but ETO LLC does not have the equipment for galvanization. The external galvanization is represented by a service item, which is added by creating an entry into the BOM table, an entry into the BOM will be called a BOM line. Therefore, when the object's "Surface Treatment" field is set to "Galvanization", the service item should be added to the BOM, while if the field is anything else, the service item line and the operation should not be added. As the service items are not part of a 3D geometric model, they are added to the MBOM. Bluestar currently supports the creation of MBOMs, creating routes, operations, and assigning a resource to perform them. Thus, the solution must be able to create these MBOMS automatically. Below are the steps in creating the galvanization MBOM is given as pseudocode.

Algorithm 1 Create Galvanization MBOM

1: **if** Top Assembly 1.Surface Coating == Galvanized **then**
2: Create MBOM of Top Assembly 1, MBOM-TOP
3: Create new Bluestar Object, named "Galvanized Sub Assembly"
4: **for** each BOM line, $l \in TopAssembly1BOM$ **do**
5: Create new BOM line from "Galvanized Sub Assembly" to l.child
6: **end for**
7: Create BOM line from MBOM-TOP to "Galvanized Sub Assembly"
8: Create BOM line from MBOM-TOP to "Galvanization Service Item"
9: **end if**

The function "Calculate Cost of consumed material" is already implemented into Bluestar PLM, and is a

feature available without customization, however, it is not automatically activated, but must be manually triggered. While the cost calculation function is already implemented, so is the weight calculation function. Both of these are often subject to customization, as "calculate finished product weight" is in the case of ETO LLC, thus it is chosen that the solution should be able to replace these functions instead of using them, to allow for more customizability at the end user.

To perform the galvanization MBOM creation, the solution must be able to represent logical branching, such as if-statements, for-loops, create new MBOMs of a preexisting assembly object, create new objects, and lastly create BOM lines between these objects. Next, the data relationships of an order-specific part will also be investigated. The resulting overview can be seen in Figure 3.3.

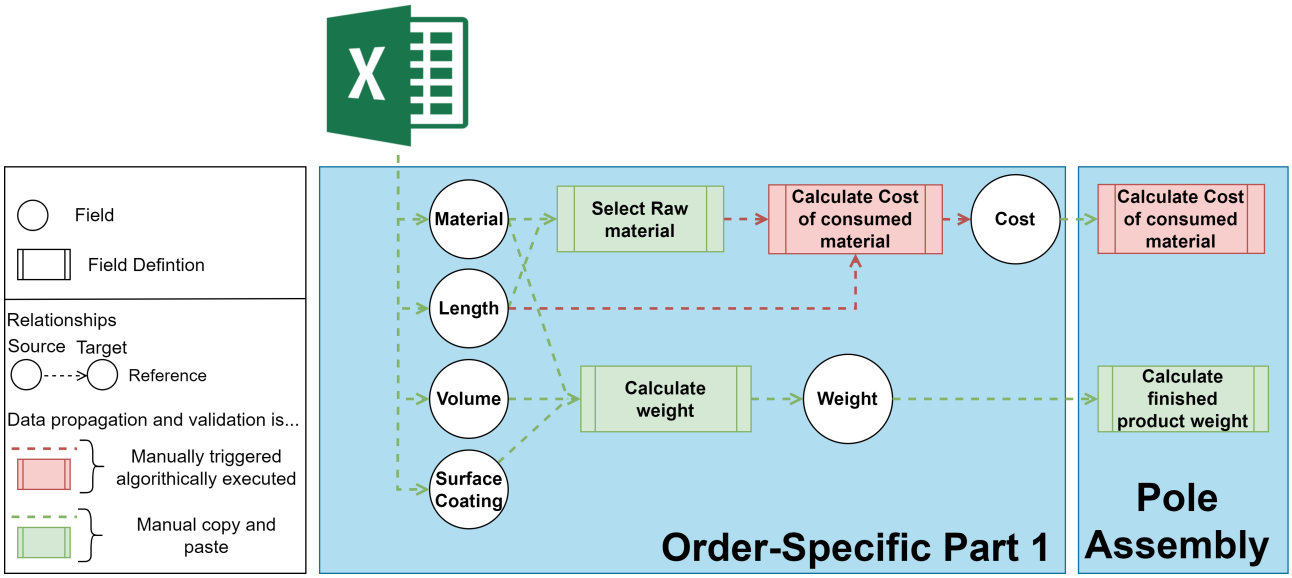


Figure 3.3: The AS-IS relationships between the parts, various fields, field definitions, and the previously analysed assembly.

From the overview, two more field definitions can be seen. To identify the functionality required to perform the field definitions of "Select Raw Material" and "Calculate weight", they will be given as pseudocode below.

Algorithm 2 Calculate Weight

Input: Material, Volume, Surface Coating
Output: Finished Weight

1: **if** Order Specific Part 1.Surface Coating == Galvanized **then**
2: Weight $\leftarrow$ $Material.Density * Volume * 1.08$
3: **return** Weight
4: **end if**

Algorithm 3 Select Raw Material

Input: Material, Length
Output: MBOM Line

```
1: if Material == S355J2 then
2:     if Length ≤ 2meters then
3:         Create MBOM Line, l, from Order-Specific Part 1 to "S355J2MetalPlate"
4:     else if Length ≥ 2meters then
5:         Create MBOM Line, l, from Order-Specific Part 1 to "S355J2MetalCoil"
6:     else
7:         Set Length field on object as Invalid
8:         break
9:     end if
10: else if Material == S355M then
11:     if Length ≤ 2meters then
12:         Create MBOM Line, l, from Order-Specific Part 1 to "S355MMetalPlate"
13:         ...
14:         l.quantity ← Length
```

From these, it can be seen that the solution must be able to access the data fields of other objects and use them in processing, such as arithmetic. Furthermore, it must be able to perform nested statements and loops. Thus the following features must be performed in order for the data relationships at ETO LLC to be represented.

1. Give Bluestar objects read access to fields to other objects.
2. Allow for arithmetic in field definitions.
3. Allow for Logical Branching in field definition execution.
4. Creation and editing of new and existing BOM lines.
5. Creation and editing of new and existing Routes and associated operations.
6. Determine Impact before the performance of a change in field data.
7. Algorithmically generate an MBOM once the EBOM of the product is known.

Lastly, with the data structures in place, a business case for fulfilling the ambitions will be developed and quantified. The last ambition of digital twins will not be performed as it is not within the purview of Bluestar PLM.

3.4 Business Case for ETO LLC

The business case will be built upon the ambitions stated earlier, with one analysis for each ambition. Unless otherwise stated, time estimates for certain actions are given by company representatives from ETO LLC. Furthermore, it is assumed that the manual work can be eliminated.

Automatically create and update derived product documentation in Bluestar

The derived product documentation encompasses two different things, the aforementioned small parts group, and the project cover page. Each of these three will be covered separately.

Small parts group The small parts group is represented in Bluestar as an object with a large number of parts used throughout an entire project. The time spent on creating the small parts group comes from two sources, firstly, each pole's BOM has to be manually checked for small parts, secondly, these must be added to the BOM of the small parts group.

The number of unique pole designs per project varies, but the average is 34, each of whom have about 5 different small parts. Small parts designs are reused across pole assemblies, and thus are only included in the small parts group's BOM once, however, the quantity to be created must be summed for all instances throughout a project. The counting is performed by exporting all the BOMs of the poles into an Excel worksheet and

then summing. The export takes approximately 10 minutes, regardless of size of the project. According to ETO LLC's representatives, it takes about 5 minutes to go through each BOM, and about 1 minute to add the quantities to the running total per small part. At 60 projects per year, that is 30,600 minutes per year.

$$T_{SPGDefinition} = 60\frac{projects}{year} * 34\frac{assemblies}{project}(10\frac{minutes}{assembly} + 5\frac{smallparts}{assembly} * 1\frac{minute}{smallpart}) = 30,600\frac{minutes}{year} \quad (3.1)$$

The presence of errors in these BOMs is negligible, as the BOM is validated before use, which ETO LLC says takes about half the time as creating it. Which is 10,200 minutes per year.

Project cover page The project cover page is used in the drawings sent to the customer. The project cover page includes a list of every unique pole design in a project, along with their quantities. The creation of the cover page is handled by a custom application developed by ETO LLC, which takes another BOM object from Bluestar PLM This BOM is create by taking pole assemblies and their quantities in a project from its sales order. This takes about 12 minutes per assembly, and thus, the time is minutes a year.

$$T_{PCPDefinition} = 60\frac{projects}{year} * 34\frac{assemblies}{project} * 12\frac{minutes}{assembly} = 24,480\frac{minutes}{year} \quad (3.2)$$

Due to the smaller complexity of this task, validation is not normally performed, errors rarely occur, and are usually fixed by the customer.

Reduce the amount of time needed for manual data entry and validation

As discussed above, ETO LLC also performs manual data entry and validation for the various parts and assembly objects themselves. According to the representatives at ETO LLC, it takes about 90 minutes of date entry per pole assembly, including all its design-specific objects and link to standard parts, and it takes about 45 minutes for validation of the same. This includes the time required to define the raw materials and routes, and thus covers both ambitions.

$$T_{ObjectData} = 60\frac{projects}{year} * 34\frac{assemblies}{project} * (90\frac{minutes}{assembly} + 45\frac{minutes}{assembly}) = 19,200\frac{minutes}{year} \quad (3.3)$$

Be able to determine a change's impact on cost, production, and lead time before it is committed to

The impact of achieving this will be evaluated through the greater ease of impact estimation. The process can be seen in Figure 3.4.

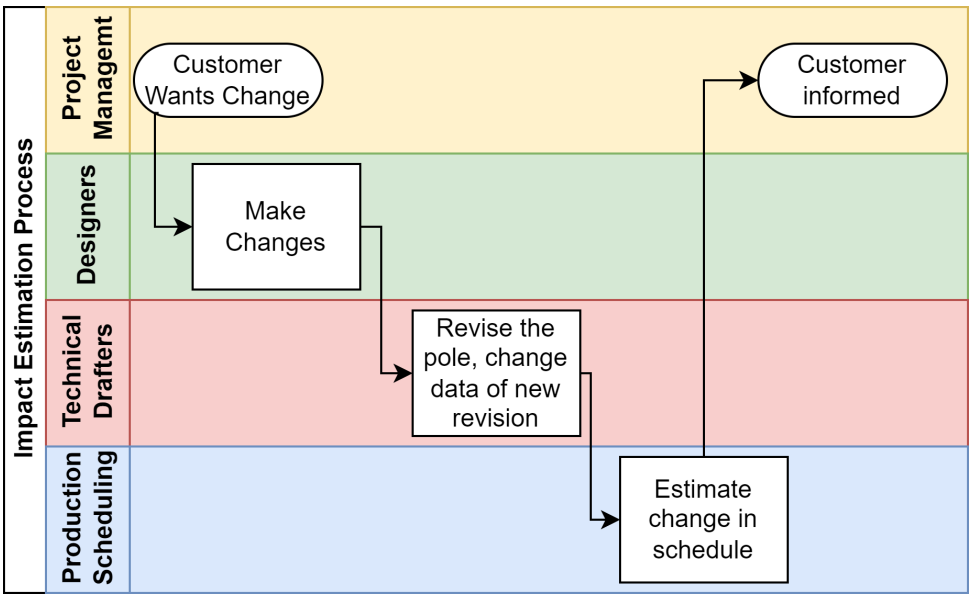


Figure 3.4: The change impact estimation process is the same whether the change is requested internally or externally.

ETO LLC states that about 3% of their designed assemblies need to be changed after approval. The communication with the customer takes about 60 minutes per change. On average, the technical drafters effectively perform 4 full data entry and validation cycles on different objects, taking about 90 and 45 minutes, respectively. The mechanical changes can be derived from the changes in Bluestar PLM, but is performed manually, this takes about 90 minutes. Lastly, the production scheduling takes about 45 minutes. However, this cannot all be performed automatically by Bluestar PLM and would require extending the semantic network to act on data outside of Bluestar PLM, which is outside the scope of this development. Thus, the time that could be reduced is the work performed by the technical drafters, and thus is 8.262 minutes per year.

$$T_{ImpactEstimation} = 60\frac{projects}{year} * 34\frac{assemblies}{project} * 0.03\frac{changes}{assembly} * (90\frac{minutes}{assembly} + 45\frac{minutes}{assembly}) = 44,982\frac{minutes}{year} \tag{3.4}$$

Be able to determine the persons responsible for enacting and approving a given change

When a change is requested, the technical drafters go to the sales order F&SCM, and copy the different responsibles from those to the object representing the change. The time to perform this is rather short at about 3 minutes per change, however, the problems arise when a mistake is made, such as the wrong user being copied or it being forgotten altogether. When the wrong user is copied, the execution of the task is delayed until the wrongly assigned person writes to the project manager, thus delaying it by about 1 day, as this is usually done through email. Next, if the responsibles have not been set at all, the execution will be delayed on average by a week after initiation. Both of these are rather unlikely to happen at about 10% and 5% of changes, respectively. Over a year of projects, this accumulates to 27.54 hours of unnecessary increased lead time.

$$T_{IncreasedLeadTime} = 60\frac{projects}{year} * 34\frac{assemblies}{project} * 0.03\frac{changes}{assembly} * (0.1 * 1\frac{day}{change} + 0.05 * 7\frac{day}{change}) = 27.54\frac{days}{year} \tag{3.5}$$

Several of these estimates result in reduced work for the designers and technical drafters, the projected times savings sum to 129.462 minutes per year of which 92.742 minutes are saved at the technical drafters.

With a theoretical business case evaluated, the solution once finished and a trial at ETO LLC is performed, a validation of the system can be performed by comparing the effect of the trials agains the expected benefits laid out in the theoretical business case above. Next, with the features identified the development of a solution will be performed.

4 Solution Development

From the use cases, several required features have been identified. However, non-technical requirements such as user experience and perceived usefulness among customers is still unknown. It is a priority for Bluestar PLM's users to be able to implement, modify, and maintain their instance of the system themselves. Thus, the system must be designed and implemented in a way that is both usable by and useful for representatives of the companies who have implemented or are implementing Bluestar PLM into their workflows. To ensure this, a number of hypotheses are created and tested using subsequent iterations of the proposed solution. The testing will be performed using feedback from representatives of one or more of the companies represented by the use cases.

4.1 Hypotheses

Given the novelty of the use of semantic networks in Bluestar PLM, a large number of unknowns remain. However, these unknowns are not merely technical but also commercial. For example, it remains unclear whether current customers see value in the implementation of a semantic network, or if they would be willing to pay more for access to it or similar features. These unknowns severely impact the viability of developing, marketing, and maintaining these features. Therefore, several hypotheses are created to explore the usability and usefulness of implementing a semantic network into Bluestar PLM.

1. End users at ETO LLC, being their technical drafters and the PLM technical lead, can conceptually understand the purpose and functions of a semantic network.
2. End users at ETO LLC believe they are sufficiently knowledgeable to use the currently selected features if implemented.
3. The selected features are sufficient and necessary for the generation of value for the users.
4. The chosen solution architecture is capable of supporting the identified features for ETO LLC.
5. The solution architecture is generalizable to other customers.
6. End users at ETO LLC, being their technical drafters and the PLM technical lead, can create and maintain a semantic network in Bluestar PLM on their own after training..

Given the high level of uncertainty and the complexity, both technical and commercial, a build-measure-learn loop wil be used to answer the listed hypotheses, several iterative prototypes will be developed and exposed to the company representatives. These prototypes will follow a progressive development flow and will thus become closer to a final commercialized product through each iteration. The technical developments of each iteration will be shown and briefly discussed and then customer feedback will be presented for the given iteration. Lastly, the feedback, related to a specific hypothesis or not, will be used to guide the development of the subsequent iteration.

4.2 Solution Architecture

The chosen architecture must be able to be implemented into the currently existing Bluestar PLM system, and it must also be generalizable to other customers beyond ETO LLC while solving their problems. This has given rise to the following architecture.

It is chosen to limit the data propagation to be pulled by the executing object rather than be pushed from another. This means that an item can retrieve data and process it to make some changes to itself, but it cannot directly set the field values in any other object. Some data entities are a part of the object even though their lifecycle may be partially independent of the object's lifecycle. Examples of these are BOM-lines, which are entries into a BOM table, routes, operations, and links. These can be created after the main object is created, and can be deleted without deleting the object, though deleting the object will delete them. These,

therefore, fall within the object which is assigned to the class, from now on the classified object. The resulting architecture becomes a "Network of Networks". That means each object carries an internal semantic logic network describing its internal structure, and each object is also part of a network of other objects. To create the internal semantic network, several artifact types were added to Bluestar PLM:

* **Data Model**: This is the data object representing the entire semantic network within a given object.
* **Calculation nodes**: Calculation nodes use instructions to algorithmically convert data from field nodes, external object links, and other calculation nodes into a value, which is then recorded in one of the object's fields.
* **Field Nodes**: A field node represents the use of a field that is stored on the object itself.
* **External Object Link**: The external object link is the external object that the data model will use in calculation nodes.
* **Instructions**: A sequence of executable steps, each containing a specific action. Together, a calculation node's instructions comprise the algorithm the calculation node will use.
* **Actions**: The mathematical or logical operator which will be used by an instruction.

This chapter will use a term called "artifacts", in this context, they mean a type of digital record or entry into a database, thus an artifact may be the data model or one of its nodes or actions. These different artifacts comprise the data model. A simple example of a data model within an object can be seen in Figure 4.1.

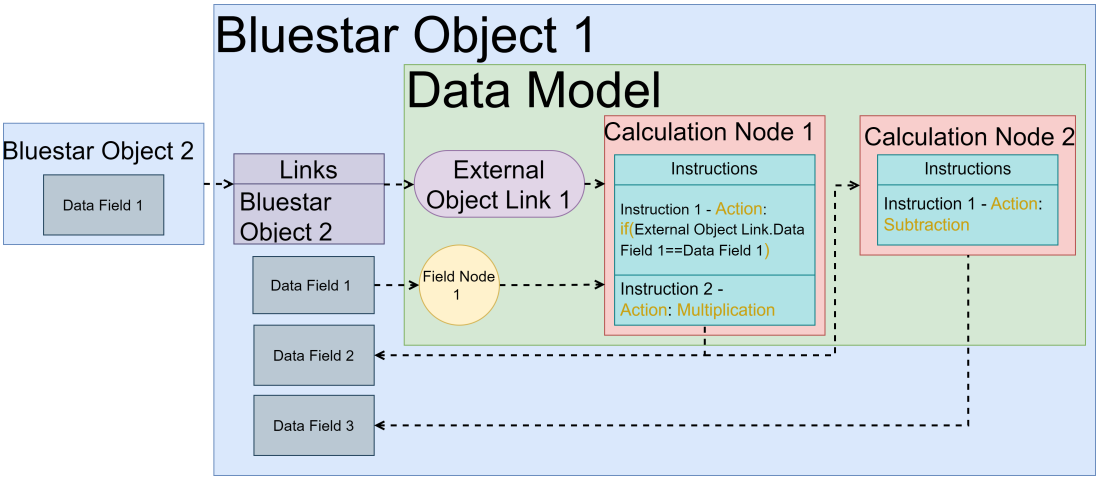


Figure 4.1: The five different kinds of artifacts used to represent the semantic network within an object. The arrows indicate the direction of data dependency.

Next, because the data dependencies vary between objects but are often similar within a class of objects, the architecture will extend one of Bluestar PLM's preexisting features, called classes. Classes are categories to which other instantiated objects can be assigned. An object, such as a project or an engineering object, can only be assigned to one class. Each customer defines their own classes and attributes, but these are shared among all users in a single Bluestar PLM environment. In Figure 4.2, the relationships of the new artifacts with relevant preexisting artifacts.

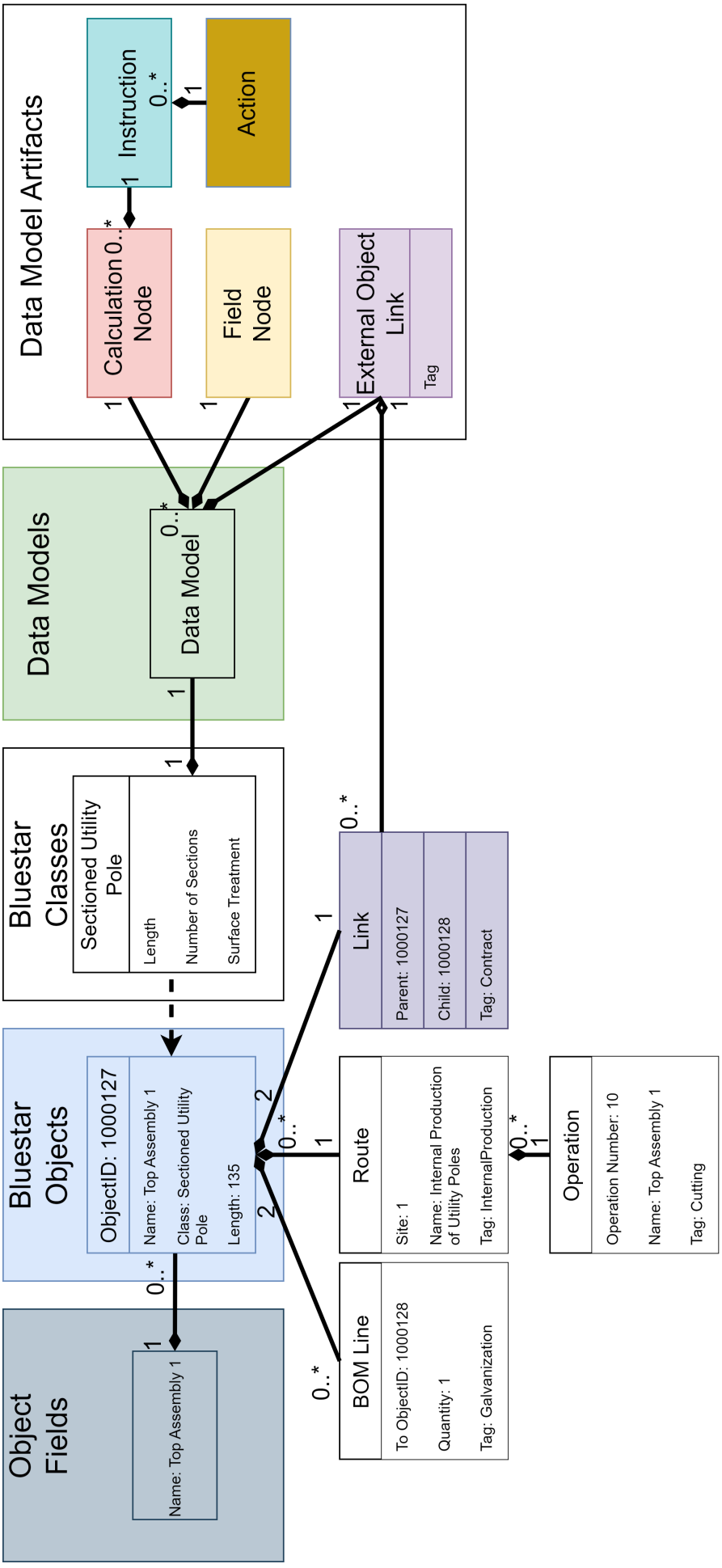
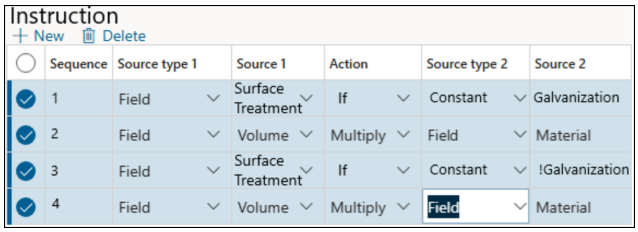
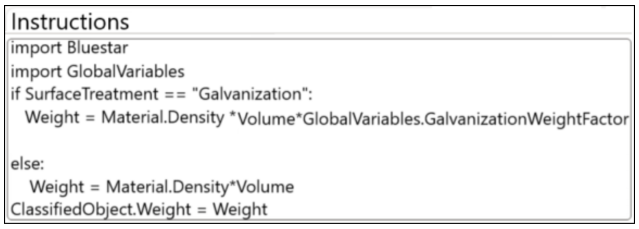


Figure 4.2: An overview of the different concepts mentioned and their relationships formalized using UML. If the artifact type is used in Figure 4.1, they are shown using the same colour here, if the type is not present in Figure 4.1, the type is coloured white.

Two methods of encoding the instructions were considered: a restrictive tabular form and a flexible programmatic form. The tabular form requires formatting each instruction into a row in a table, where the data inputs are specified along with the action that the two sources will be used in. A programmatic encoding would utilize a formalized programming language, such as Python, to describe the actions to be performed. An example of the two encodings representing the conditional calculation of product weight depending on the "Surface Treatment", "Weight", and "Volume" can be seen in Figure 4.3.



(a) A tabular encoding of the conditional weight calculation, by combining the use of two defined actions of if-statements and multiplication.



(b) An encoding of for the conditional calculation of weight using a python syntax.

Figure 4.3

The encoding into tabular form makes it much easier to restrict the ability and flexibility of end users to impact the data in Bluestar in unintended ways since the user is restricted to performing a predetermined set of actions. However, it may also make the encoding of instructions less logically flexible, subsequently, the amount of actions required to support the business logic outside of ETO LLC may be unviable to develop and support. However, the effort required to support this and the business logic is unknown. Even though it may be more scalable, the initial development time for a workable prototype based on programmatic encoding was considered to be significantly larger. Thus, due to the easier coding implementation and the greater protections against mistakes for end users, the Bluestar management prefers the tabular option if it is found to be sufficiently generalizable. Therefore, it is chosen to proceed with a tabular encoding and to estimate the development work required to allow the encoding of the business logic of other Bluestar PLM customers, which can be seen in Section 5.1

As mentioned in Chapter 3, objects in Bluestar PLM are represented in F&SCM by releasing the object into F&SCM from where it can be referred to and impact can be propagated to sales, production, and purchasing orders. Everything that causes a change in F&SCM must be present in F&SCM so a change can never be propagated directly from an engineering object to a commercial record outside of release, as this would be impossible to trace and account for by users who do not have Bluestar PLM access. Therefore, this solution will be scoped to the implementation of features into Bluestar, but it will only be very limited in changing the F&SCM and the broader Dynamics 365.

4.3 First Development Iteration

This iteration focused on answering of Hypothesis *1.* being "End users at ETO LLC, being their technical drafters and the PLM technical lead, can conceptually understand the purpose and functions of a semantic network" To test this hypothesis, a minimally viable product was developed by implementing Features *1.*, *2.*, and *3.* from the previously identified features.

1. Give Bluestar objects read access to fields on other Bluestar objects.
2. Allow for arithmetic in field definitions.
3. Allow for Logical Branching in field definitions.

These features are implemented by extending Bluestar PLM's classes to be able to define a "*Data model*" for a given class, in which the user can define relations and dependencies between a class's attributes, native object fields, and fields of linked objects. These relations are simple arithmetic and logical branching calculations. Thus, this prototype is not able to perform anything related to source files, the creation of new objects, or new relations to existing objects. The data model creation is performed in the graphical user interface seen

in Figure 4.4.

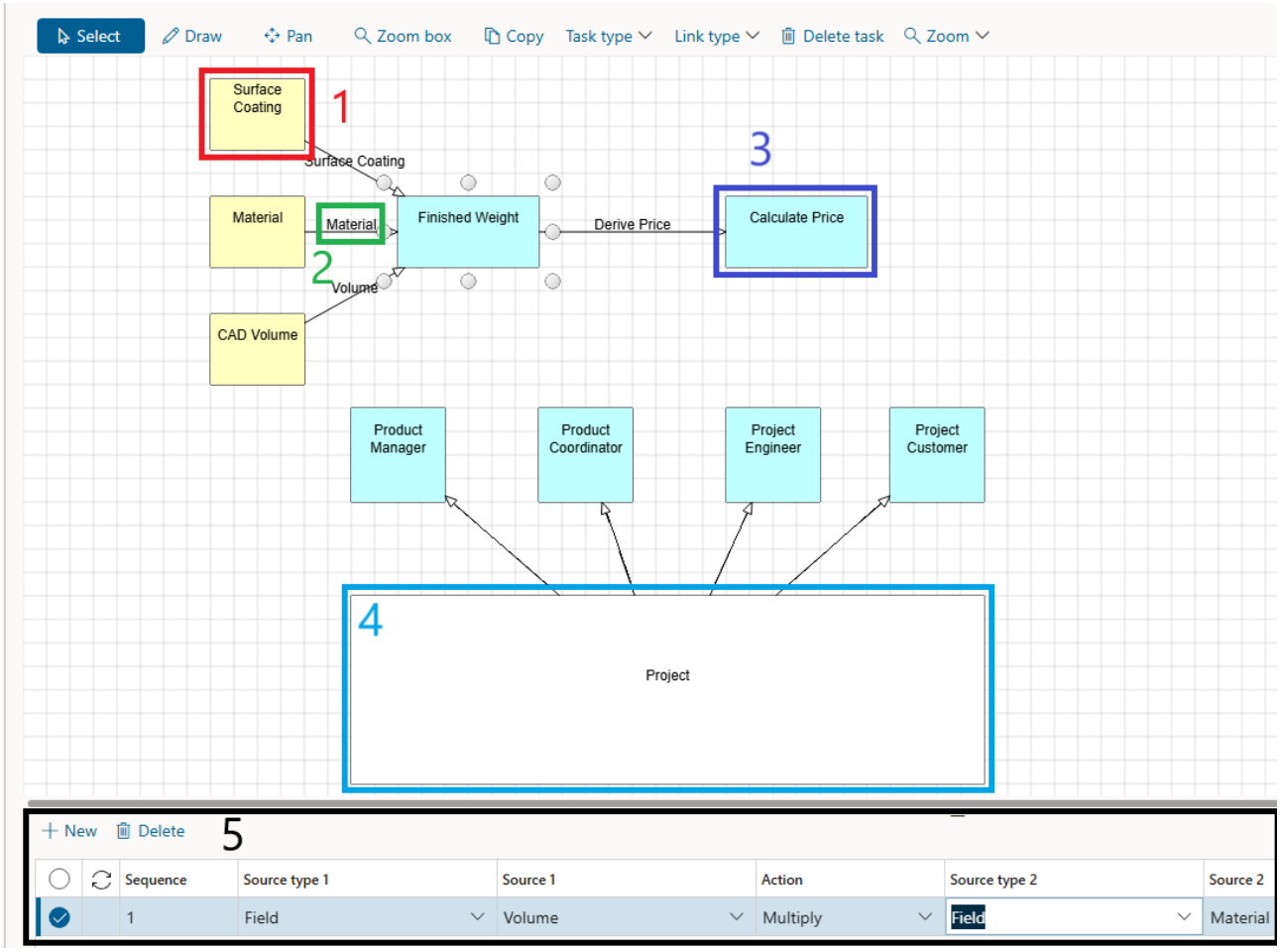


Figure 4.4: The interface created for the definition of a data model by allowing the user to create and define nodes, external object link, and their relations.

The data model interface has five main components, as seen in Figure 4.4, which are used to implement the features mentioned. (1) The three yellow coloured blocks are field nodes and are used to show the successive node's data dependency on a given field. Here, the finished weight is dependent on the field nodes "Surface Coating", "Material", and "CAD Volume". The data dependency is shown explicitly by using (2) the relationships between nodes, each of which can be given a name to describe the meaning of the data dependency between the field and the calculation. This allows (3) the processing nodes to use their relations to field nodes and (4) external object link nodes to generate the values that will be outputted into a selected field or attribute on the classified object. This is generated by the processing nodes using (5) the instructions, where the external object links, field nodes, and the output from previous instructions can be used as inputs into various actions. The instructions are used to implement the arithmetic and logical branching required by the features by implementing six actions to encode the technical and business logic into the data model.

To determine the usability of the first iteration of development, the prototype was shown to four different company representatives of ETO LLC. Two representatives were chosen because they had the most theoretical and managerial experience in using Bluestar and implementing Bluestar and other PLM systems at ETO LLC and other companies, these will be called the *tech leads*. The remaining two were chosen as they were the users currently performing the data copying and validation, and were, therefore, the most familiar with ETO LLC's product data structure, furthermore, they had the most direct experience in using and managing data in Bluestar and thus would be the primary maintainers of a data model. these will be referred to as the *detailers*.

The Representatives of ETO LLC received coaching and a demonstration on the use and suggested purpose of the prototype data model. To test the hypothesis, a simplified product data structure was derived from the

intraobject data structure and then presented to the representative.

Task: The company representatives were then asked to individually create the given data structure seen in Figure 4.5 within a test environment containing the solution prototype.

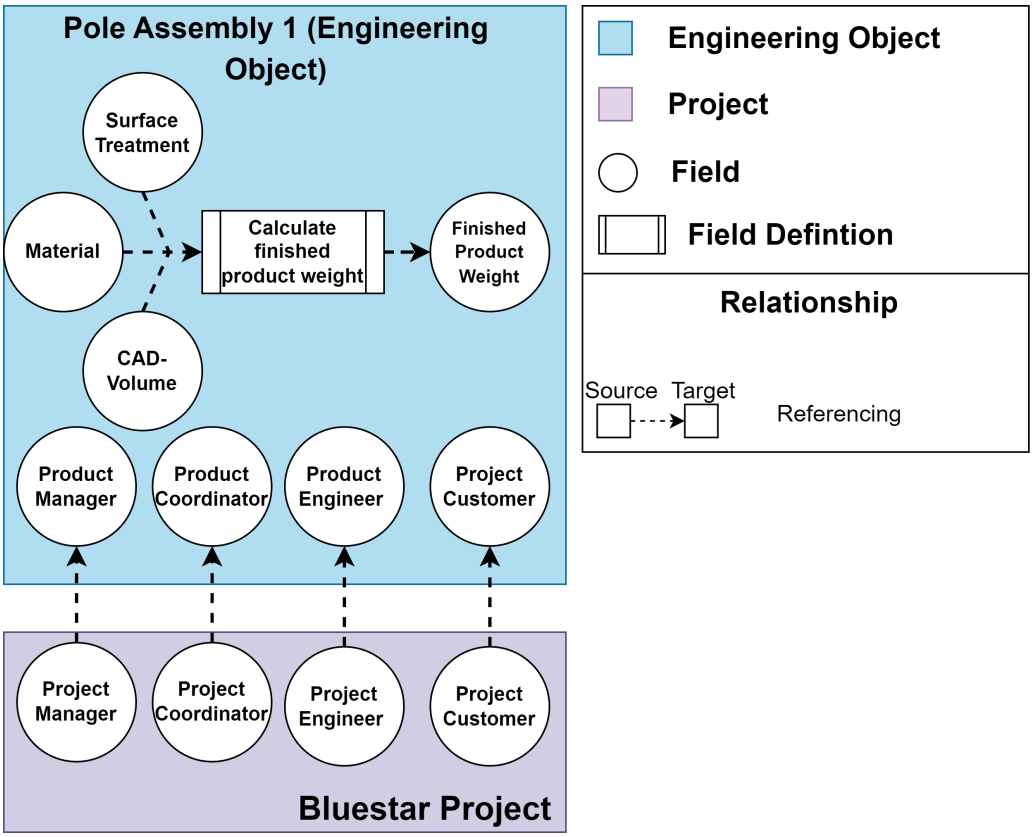


Figure 4.5: The simplified product data structure given to the company representative of ETO LLC.

Result: All four company representatives found the interface to be understandable and legible, though somewhat "clunky" as expressed by the senior tech lead. After approximately 30 minutes of explanation and coaching, the company representatives were able to complete the task, but all did require further coaching, including reminding the representative of previously told information and explanation of terminology. In particular, while the detailers showed more skill in navigating the interfaces, they were much more confused about the purpose of each data model component and consequently took a longer time to complete the task. They found the different kinds of nodes hard to distinguish and use. They found that defining the instructions was particularly challenging due to the lack of a "do nothing" action, the lack of which required concatenating the selected field's value with an empty string to achieve the data references. To accommodate this feedback a "do nothing" action was added and the external object link node was changed to a rounded shape to denote its difference from the others. While the value of the identified features was not surveyed, the company representatives did believe that such features would provide meaningful value if the other identified features were implemented. This issue was further explored in the second development iteration.

4.4 Second Development Iteration

In the second iteration, Hypothesis 2., being "End users at ETO LLC believe they are sufficiently knowledgeable to use the currently selected features if implemented", a simple use case for the Assembly data structure, seen in Figure 4.6, was sent to the same company representatives of ETO LLC as used for the evaluation of the first iteration. The use case was not attempted to be made, this was performed in the third iteration. Instead, the second iteration focused on ETO LLC's company representatives' perceived ability to make use of it to create the data structure with the identified but not yet implemented features.

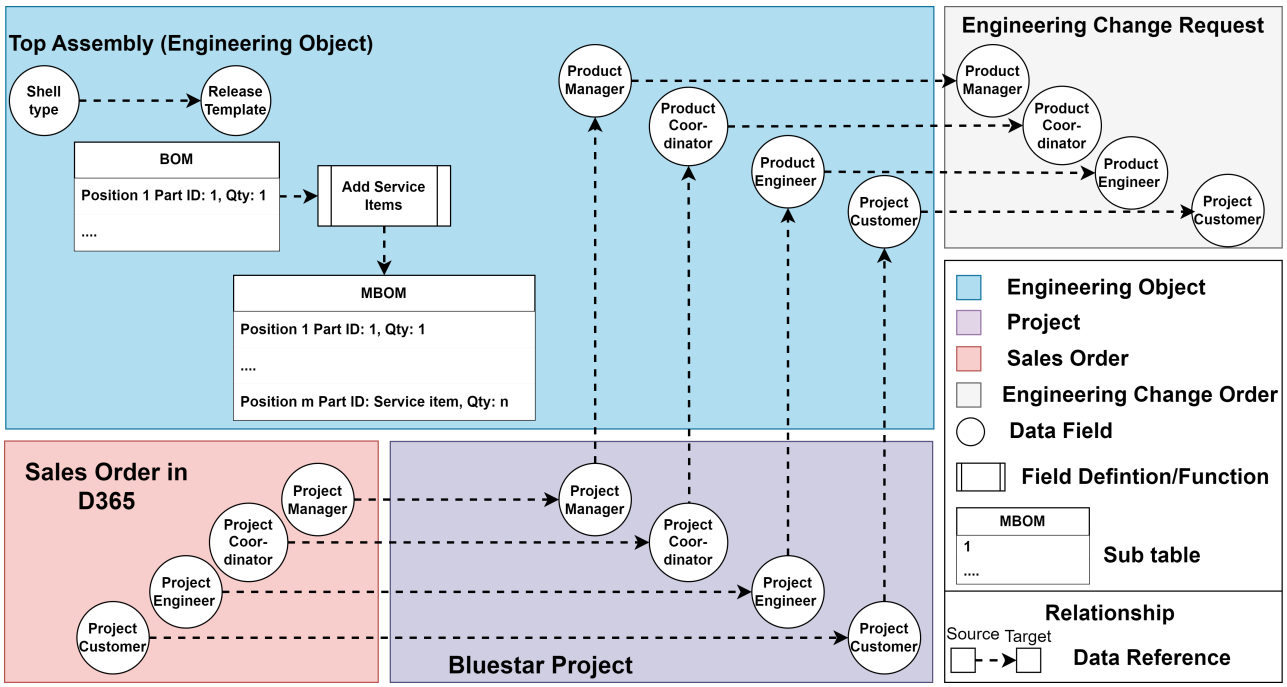


Figure 4.6: The use case was created and exposed to company representatives of ETO LLC to gauge their perception of value for their current challenges.

Task: The company representatives were asked to score each of the features on a survey about each of the seven features: Each representative was asked to rate their belief in their own and the wider organization's technical ability to make use of the feature, marked below as "Usability". The ratings were made between 1 and 5, with 1 being completely unmaintainable and 5 being very easily usable, with a score below the middle point of 3 representing a belief that they do not currently have the necessary knowledge and ability to use the features. The averages of the four company representatives of the answers can be seen in Table 4.1.

Result:

Table 4.1: Rating of usability of suggested features by representatives of ETO LLC on a 5-point scale.

Feature	Usability
Give Bluestar objects read access to fields to other objects.	4
Allow for arithmetic in field definitions.	4
Allow for Logical Branching in field definition execution.	4
Creation and editing of new and existing BOM-lines.	4
Creation and editing of new and existing Routes and associated operations.	3.25
Determine Impact before the performance of a change of field data.	2.75
Algorithmically generate an MBOM once the EBOM of the product is known	2

All features except impact estimation and MBOM generation were determined to be usable by the company representatives. Thus, these features appear to be self-describing enough that the tech leads and detailers could use them without further significant training. Thus, it was decided to develop the usable features before the more user-challenging ones. The more user-challenging features must still be developed if the ambitions of ETO LLC is to be fulfilled, however, the ability to create the required training material should be explored to increase the perceived usability. Due to time constraints, this will not be performed before the turn-in of this thesis.

To implement the usable features, several new instruction actions were created, the development specifica-

tion for the interface, their behaviors, and special cases are described in Appendix B. In total, six new actions are added: three for creating new BOM-lines, ROutes, or Operations, and three further to edit pre-existing BOM-lines, routes, and operations.

All three of these are collections of fields and are thus much more complex to interact with than a single isolated field on an object. Importantly, all three are fully owned by the classified object, as routes by default in Bluestar are product-specific, and the operations are route-specific. An object can have an arbitrary number of BOM-lines and routes, and routes can have an arbitrary number of operations, the uniqueness and preferability will be solved using the tagging described in Section 4.2. Furthermore, due to their greater complexity, the creation and editing cannot be performed through the tabular encoding previously decided upon as the number of fields in the table, which would have to be shown and editable at all times, would make the interface vastly more cumbersome and less intuitive and intelligible for the user. Thus, a choice of tabular and programmatic encoding is implemented this, and the user interface was further developed as can be seen in Figure 4.7.

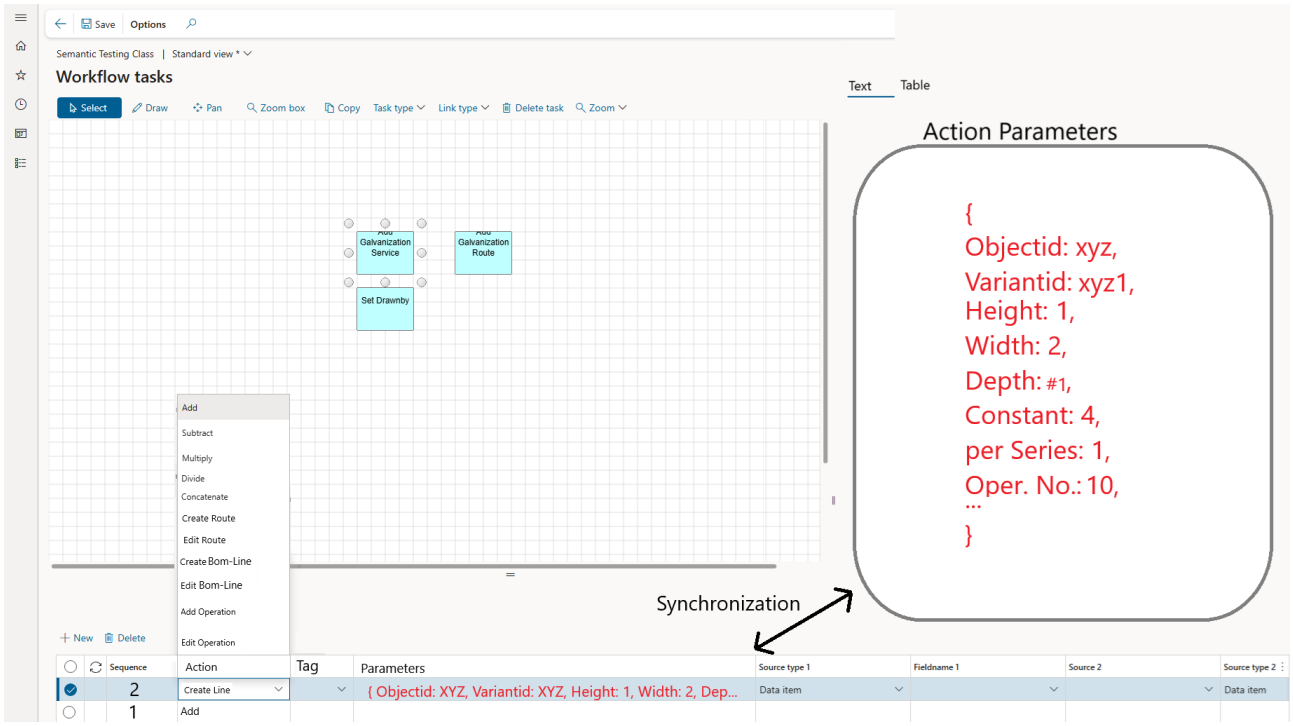


Figure 4.7: The new interface showing the action parameters text field, which will be used to define the action to be undertaken during execution.

Another development is the creation of an "Action Parameters" field. To reduce the interface complexity of the tabular instruction encoding, a multiline text field is created. This text field is then synchronized into a field of the same name in the instruction table. The action parameters are the values for a BOM line, Route, or operation. The current values will then be overwritten with the ones specified in the action parameters. Once the creation and editing of the BOM lines, routes, and operations were developed, the prototype was used to test the next two hypotheses.

4.5 Third Development Iteration

The third iteration will test Hypothesees three and five being "The selected features are sufficient and necessary for the generation of value for the users" and "The chosen solution architecture is capable of supporting the identified features for ETO LLC" respectively. To test this, the two intraobject use cases, described in Section 3.3, will be combined and are illustrated in Figure 4.8. **Task:** This data structure will then be attempted to be made by the company representatives used in the first and second iteration.

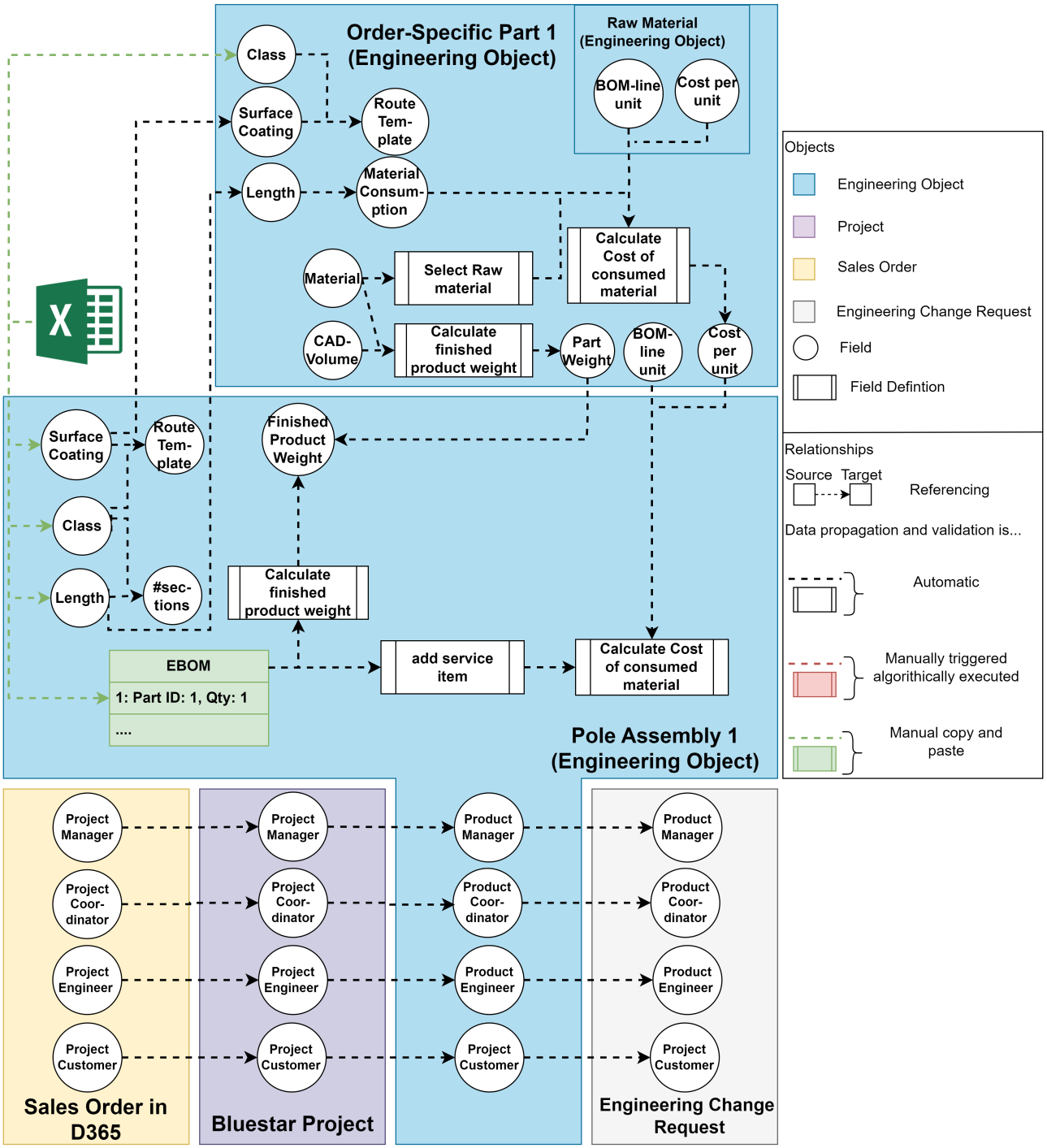


Figure 4.8: The data structure for the third iteration was created by combining the two data structures at ETO LLC described previously.

Result: Using the developed features, the company representatives were able to complete the assignment by creating the data model, though the representatives continually required further coaching, including on topics previously covered in iteration 1. Furthermore, two more challenges were discovered.
Firstly, ETO LLC has some custom developments for their F&SCM environment, which heavily restricts the naming and number of fields in their F&SCM products. In particular, they can only have 20 unique data fields other than those originally provided by an out-of-the-box F&SCM environment. While this particular issue is specific to ETO LLC, many Bluestar implementation projects result in engineering or data representation concessions because of the way the products must be represented in F&SCM. Often, an engineering department may want extra class attributes, which are not relevant commercially. These are often not added as they will be synchronized into F&SCM to the confusion of production or sales. Thus, another feature is required to be implemented for the prototype to be used at ETO LLC:

8. Algorithmically determine how an Engineering Object is represented in F&SCM.

Secondly, during the creation of the data model, many of the representatives' frustrations were caused by the co-evolution of two or more data models. As each data model is created to work independently of the others, it is also defined on its own. Therefore, the representatives were often confused about which part of the data structure they were working on. At ETO LLC, the set of classes used in the data structure is the same between instantiations. That is the assemblies, parts, projects, and so on, each of which has different classes, but the classes of projects used for their products are the same. Thus, the same data models are often used together and will also be defined together. To accommodate this, the data model interface will be modified to support the definition of multiple data models simultaneously.

However, due to time constraints, particularly caused by the developer's unavailability for the remainder of this thesis before submission, the results of feature development will not be described here. Similarly, the solution is not sufficiently developed to be able to test hypothesis six, being "End users at ETO LLC, being their technical drafters and the PLM technical lead, can create and maintain a semantic network in Bluestar PLM on their own after training." As this requires all the requested features. Furthermore the impact of using the prototype at ETO LLC will not be evaluated or compared to the theoretical business case from Chapter 3 as the prototype is not suffieciently finished.

Nevertheless, it is concluded that the current architecture is capable of supporting the necessary features for representing the data structure at ETO LLC, thus confirming Hypothesis four. Moreover, as the main cause of frustration is to be alleviated by allowing multiple data models to be defined simultaneously, no further functionality was requested by the company representatives, thus confirming hypothesis four. Hypothesis five will be explored by analysing two other customers of Bluestar PLM and then identifying the required features beyond those already developed, which be performed later in Section 5.1.

5 Discussion

This chapter presents a critical discussion of the further development following the development and implementation of the Semantic Web within Bluestar PLM. It explores the practical challenges of the implementation and addresses any challenges encountered during deployment. The discussion also reflects on how the results align with the initial objectives and highlights opportunities for future improvement.

5.1 Usefullness Beyond ETO LLC

For full commercial development of the semantic networks to be viable, it must also be useful beyond the specific use case for ETO LLC outlined in Chapter 3. Therefore, two further companies with different process patterns, data structures, and digital ambitions are outlined, and any new feature requirements will be identified. Lastly, the current architecture's ability to support features will be evaluated by the developer who performed the previous development.

CTO Company - Variant Creating

Some other customers have much more well-defined product structures and therefore they make use of the product configuration system to generate the required specifications and files from the user input and customer requirements. These customers fall into the Configure-to-order or CTO process pattern, where a new product is often created for or used in a new customer order, but the data on the configuration is derived through logic set by the configuration model. This is the case for the company representing this use case, CTO AG.

CTO AG produces clothing rack systems for industrial clients. The AS-iS data structure can be seen in Figure 5.1, some of the parts in the product structure are standard, however, because of the use of configuration masters, much of the data of the individual object is created using the rules from the configuration masters.

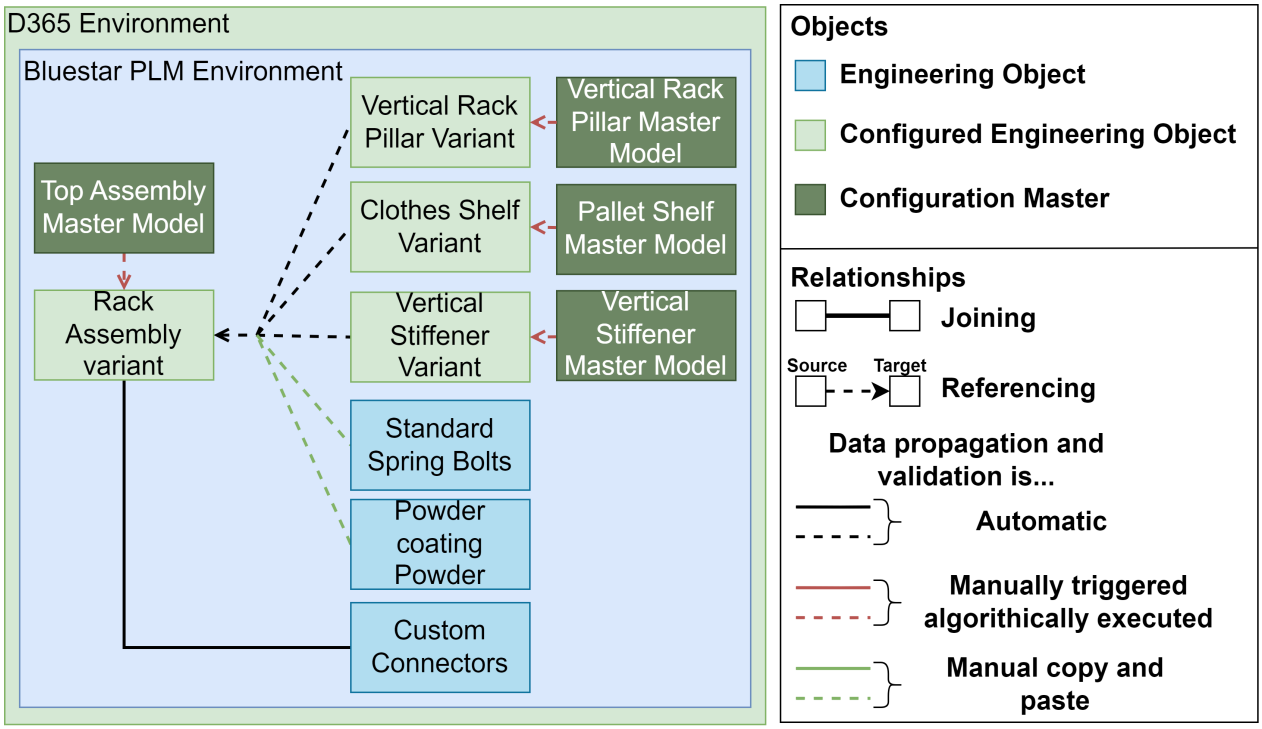


Figure 5.1: The different objects in a rack assembly variant and their associated master models used for configuration. The interactions with production equipment have been left out for succinctness.

The design of these racks follows predetermined rules captured in the master models, but individual dimensions, connection pieces, and aesthetic elements are liable to change. As CTO AG has a long history of using this system, it has tens of thousands of preexisting rack assemblies and parts that are reused during configuration. While engineering files and drawings may be unchanged, the company's production is continually developed through greater automation and new process technologies. Thus, the production route and operations of a given variant may change through time. Due to the number of variants, it is not viable to manually maintain each variant.

Similarly, it is not viable to manually determine which of the tens of thousands of variants are no longer producible. Currently, the designers and engineeers, and CTO AG will inspect each part of the BOM structure to determine whether or not it is possible to make a given location.

Thus, CTO AG's ambitions are to...

1. maintain routes and operations on pre-existing variants when production resources change.
2. identify variants no longer producible due to new product constraints

To explore how these ambitions could be realized, the data structures of one of the parts from Figure 5.1 at CTO AG will be investigated. A hypothetical TO-BE Bluestar representation of the data structure of the vertical rack pillar can be seen in Figure 5.2. Data dependencies unrelated to the identified ambitions have been left out.

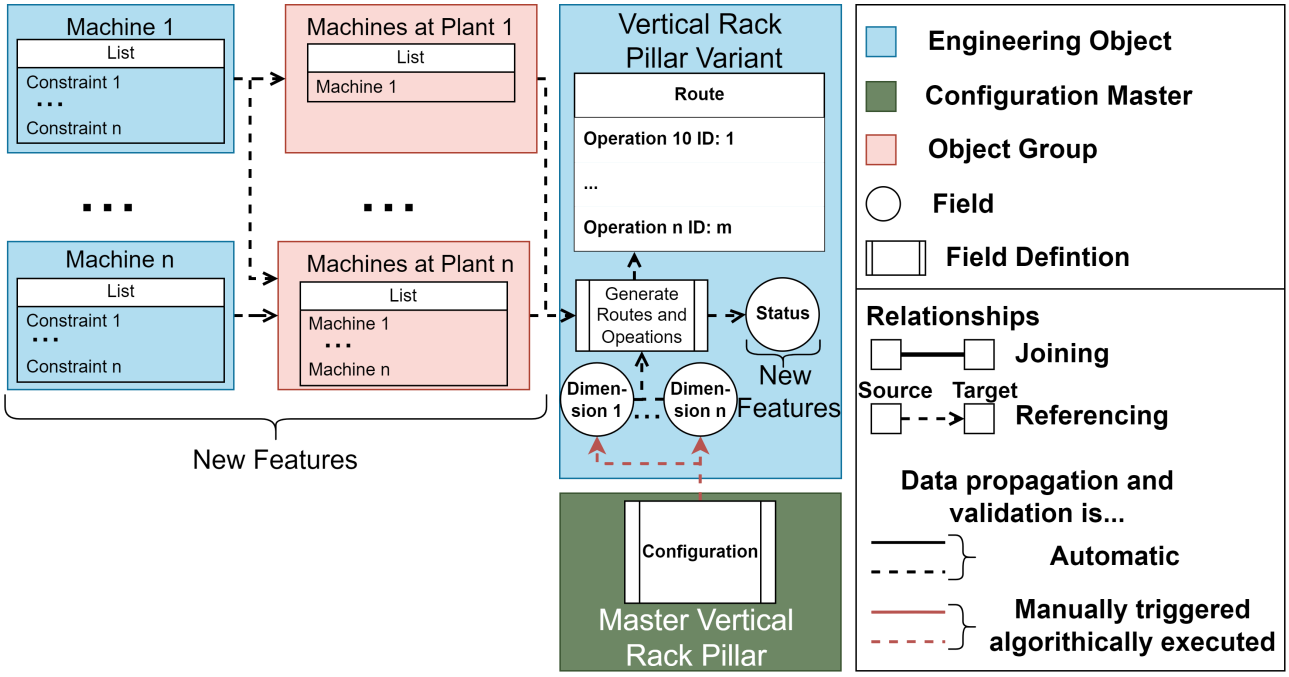


Figure 5.2: The interactions between different objects used to generate the routes and operations for a given variant.

As seen in Figure 5.2, the different machines are categorized into different groups depending on availability at a given plant. Each machine has a set of constraint for the different dimensions, the routes and operations for a given variant then depend on the machines' constraints. If no valid routes can be made, the variant is not producible with the current resources.

This would require the ability to represent machines, their constraints, and group machines in Bluestar. Furthermore, it requires a semantic agreement between the machines' constraints and the dimensions.ith the data structure and business Processes outlined, a business case for the use and implementation of model-based features can be made.

Business Case

The central problem for the company is the maintenance of variants as the business develops and routes and products change. The variants have been made previously and may have been used in customer orders.

If the required changes to Bluestar are made, the production preparation team would not have to perform manual producibility validation for each design, and therefore save time. Secondly, the non-producible variants could be hidden from designers and sales personnel such that new projects do not use them, thus saving time outside of production preparation, by removing redesigns.

Assuming that CTO AG fulfills 100 orders a year, each containing 25 different variants, that 20% of all variants are not producible anymore, that each validation takes 30 minutes, and that redesigns take approximately 2 hours. Then the lost time across can be found by Equation 5.1, which comes to 195.000 minutes per year, of which 75.000 minutes is production preparation and 120.000 minutes in design and sales.

$$T_{Lost} = 100\frac{Orders}{Year} * 25\frac{Variants}{Order} * (30\frac{minutes}{variant} + 0.2 * 240\frac{minutes}{variant}) = 195.000minutes \tag{5.1}$$

The implementation of the suggested features would therefore save CTO AG 1,5625 full-time employees every year. For brevity, the business case here only estimates the impact of the new features thus, other gains mirroring those expected at ETO LLC are entirely possible.

Mass Producing Companies - Workflow Defined

The last use case is based on a mass-producing company, MTS ApS, which makes roller blades for the consumer market. There is a significant number of variants, but these are designed and produced ahead of the point of sale and without any input from the customer. The roller bladess vary primarily in three ways: the mold used to create the roller blade, called the technical norm, the materials which are layered and put into the mold, defined by a technical specification, and lastly, the size, which is set for a given variant. This results in a hierarchical structure in which some data of a given variant is defined for the individual variant, some is derived from the technical specification, which in turn inherits some from the technical norm.

The specification and development of these variants is very time-consuming, highly cross-functional, and is performed each year to keep the collection fresh. Therefore, the engineering and aesthetic designers must continually collaborate on a large number of variant development projects, each at different levels of completion. To structure this, the company makes extensive use of Bluestar PLM's task and workflow objects to describe and schedule the current tasks. Currently, a spreadsheet and to contain the results and conclusions of the completed tasks for each development project is passed between departments which is then gradually filled out. This, however, leads to lost work when a spreadsheet file is lost or the wrong version is updated. Furthermore, an engineer must interpret the resulting structure from the spreadsheet and then implement it into Bluestar PLM, which is both time-consuming and prone to errors during both development and interpretation.

Thus, MTS ApS's ambitions are to...

1. Use task completion to execute or inform predefined functions to create new objects or relations.
2. Sequentially apply rules throughout product development.

These ambtiions can also be derived from the data structure seen in Figure 5.3, here the data structure includes a Bluestar workflow and associated tasks. These are used to define not specific variants but rather the technical specification, which acts as a meta model for each of the roller blades.

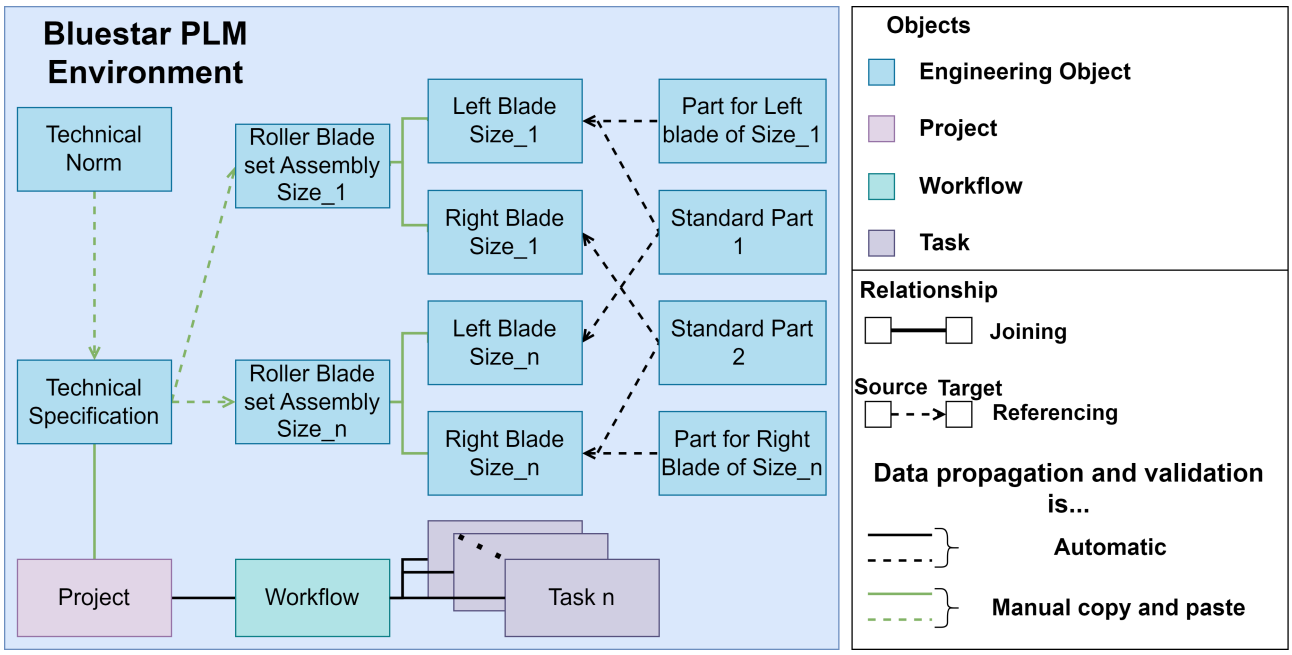


Figure 5.3: An overview of the AS-IS data structure representing the relations between various objects at MTS ApS.

In effect the project and workflow are just passthroughs for the choices made in each task to be applied to the technical specification. This yields the intraobject use case data structure can be seen in Figure 5.4.

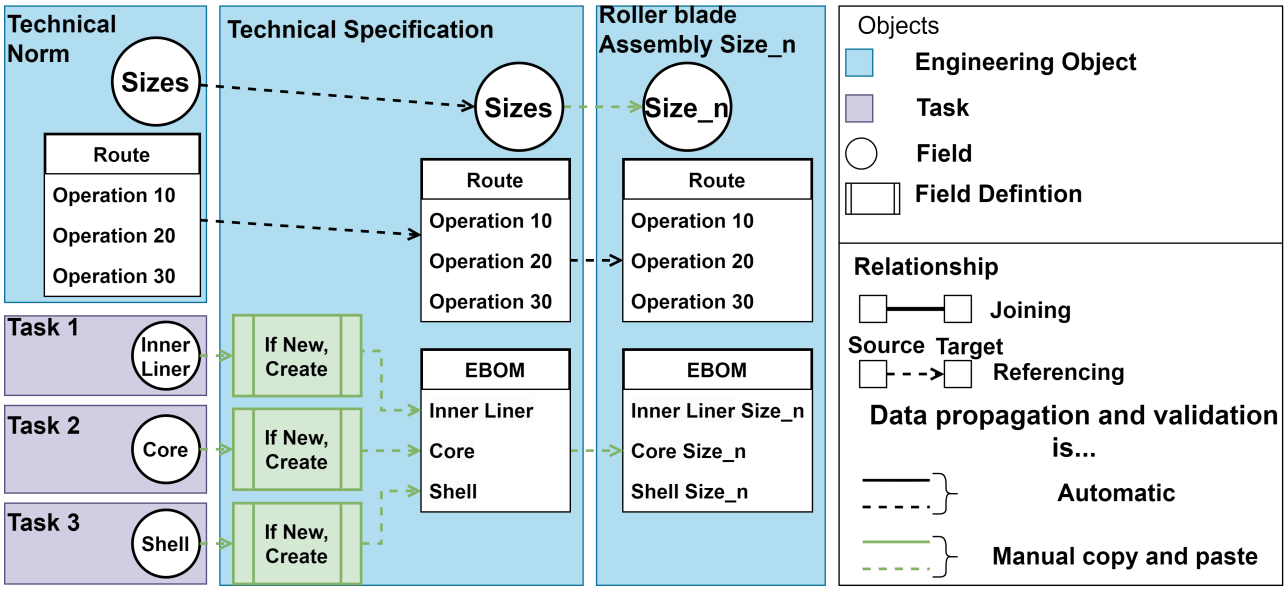


Figure 5.4: The interactions between the tasks, technical norm, technical specifications and the resulting roller blade assemlby variant. The data structure is shown as if all currently identified features were implemented.

Even if all the currently identified features were implemented, much of this data structure would still be manually executed, because Bluestar PLM lacks the ability to easily use the choices made in a task outside of the workflow itself. Furthermore the ability to trigger some action depending on the state of a given task is currently possible and has not been covered in the other previously identified features.

Business Case

The company has five different business areas for different kinds of roller blades, which in total become nine different product lines, which in turn contain 80 different technical specifications, all of whom is redesigned every year. Each technical specification has an average of five different sizes, although this varies by technical norm. In total, there are 10 different technical norms across all technical specifications. Thus, on average,

every year the company runs 80 roller blade development projects and creates 400 new roller blade variants with individual designs. This effort is performed by a team of 25 people, who also handle engineering change requests, prototyping, and production process development.

With 400 variants, each containing a right and a left, each containing two levels of sub BOMs, the number of BOMs becomes 2400. If each BOM takes approximately 15 minutes, and an escaping error is made in 1% of BOMs, due to wrongful interpretation of the project spreadsheet, and 60 minutes of work is lost to rectify an escaping mistake in a BOM. The total BOM time is calculated as in Equation 5.2.

$$T_{BOMDefinition} = 2400 \ \ BOM \cdot (15 \ \ \frac{min}{BOM} + 0.01 \ \ \frac{Mistakes}{BOM} \cdot 60 \ \ \frac{min}{Mistake}) = 37.440 \ \ min \tag{5.2}$$

However, the BOM of a variant can be calculated based on the technical specification by filtering it based on the size of the roller blade and right/left chirality of the roller blades. Thus, this time and associated errors could be effectively removed if the conclusions made in the development spreadsheet were automatically applied to the technical specification that the project is developing.

Next, the use of a spreadsheet to track the project results and conclusions have led to several problems, including the loss of work when the spreadsheet is lost and an old version must be found and the same work must be remade. Assuming each development project requires input from 10 people and consists of 50 tasks, each requiring 60 minutes of work to finish, then if each person completes 2 of the 50 tasks before passing the spreadsheet on, then losing the spreadsheet will lose 120 minutes of work and the time spent trying to find it. Thereby, there will be 25 handovers between project collaborators, and there is a % chance for the spreadsheet to be lost for each handover. Thus, the time lost for project handover is calculated by Equation 5.3.

$$T_{LostWork} = 80 \ \ projects \cdot 25 \ \ \frac{handover}{project} \cdot 0.05 \ \ \frac{losses}{handover} \cdot (120 \ \ \frac{min}{loss} + 15 \ \ \frac{min}{loss}) = 13.500 \ \ min \tag{5.3}$$

With these further customers analyzed and their business cases shown, the features required to support them can be seen in Table 5.1. The compatibility of each feature was then evaluated by the responsible Bluestar PLM developer.

Table 5.1: The new features which must be developed for the business cases to be realized and for the data models to support the data structures at CTO AG and MTS ApS.

Feature	Use Case	Compatible with current Architecture
Represent and Evaluate Constraints of production equipment	CTO AG	If semantics match, yes
Set the status of an object	CTO AG	Yes
Create and reference dynamic lists of objects	CTO AG	Yes
Reference tasks choices in a workflow	MTS ApS	Yes
Trigger actions through workflow and task progression	MTS ApS	Yes

With the new features identified and their ability to be implemented into the current architecture the penultimate hypothesis being "The solution architecture is generalizable to other customers" is therefore accepted. The architecture is generalizable to other customers.

5.2 Further Development for Current Features

The hypothees have been investigated some architectural characteristics were discovered during development, which were not solved due to time constraints. These issues relate particularly to the practical functioning of semantic webs implemented alongside ordinary PLM features, such as Class inheritance hierarchy, multi-class structures revisioning, engineering change management, and will be discussed here.
Though it is unclear whether this increase in interrelatedness between classes causes greater complexity and rigidity in defining changing the preceding classes to be worthwhile.

Multi-Class Data Models

As discovered during the third development iteration, the users would like the ability to define data models together, as some were often, if not exclusively, used together. However, with the current architecture binding the data model to the class, this makes the management of the data model more troublesome if the model is also used in other situations and with other sets of classes than the one it is currently being defined for. Thus, a user could make changes to a class' data model, which impacts interactions beyond the scope they are currently considering or even know about. Implementers of Bluestar PLM often use the rule of thumb that a class is a category of objects that have the same defining properties and rules. Nevertheless, the desired behavior is unclear if the objects of the given class have different rules and interactions in different situations?

One may argue that the class should be split into several sub-classes, one for each situation the class may be used in, such that each sub-class can contain their own data model. However, this would result in objects that are functionally and technically identical, but used in different contexts to have different classes. This may seem acceptable, except when one considers that many users may not know the wider context of the physical object they are interacting with. How would the worker determine the class of the physical object? How would they know which classes contain functionally identical objects? In such an implementation, the classes become unwieldy for anyone not directly involved with their definition and management, as they may not know of the various other situations that the same object may be used in.

Another suggestion is to separate the data model from the class itself, and break the rule of similar properties and rules within a class. Thereby, the classes can have the same product properties, but have different rules and interactions in different situations. This could allow the EOSs of different objects to define multi-class data models by giving each class one or more data models used in different situations. These multi-class data models would represent an ontology, which could then be instantiated in its entirety into a semantic network.

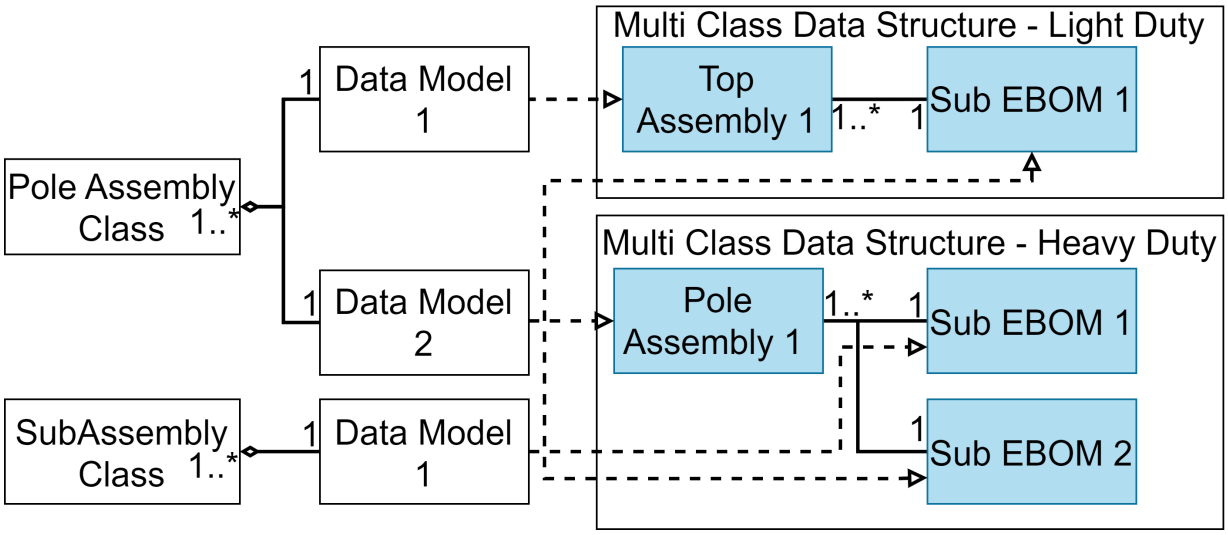


Figure 5.5: An example of the relationships between the classes, objects, data models, and the multi-class data models shown in UML.

Semantic Networks' Interaction with Changes

A further consideration is the unresolved problem of data referencing through revisions and approvals. If an assembly uses the weight field of an object representing a part, the assembly object is then approved and locked for changes, but if the weight of the part is changed, then either the assembly's weight will have to be changed, or it will be outdated. Allowing the weight property on the assembly to change does not solve this problem. Since other nodes in the assembly objects' semantic network may cause changes based on the weight, those subsequent nodes will be out of date. Similarly, today Bluestar PLM allows the creation of links and routes which include approved objects because neither links or routes can change the underlying approved object. Semantic Networks change this and thus such actions would suffer the same problems as referencing fields on other objects. This could lead to incongruities in internal object data or incongruities in data in different objects.

One potential solution is to specifiy the nodes in the data model into nodes that can be impacted after approval and those that cannot, such as seen in Figure 5.6. Through this, the effect of changes can be allowed to be propagated to some but not all of the objects' fields and attributes. In this way, Bluestar can allow sensitive nodes to be updated, while others are not. Moreover, by anticipating changes made after approval, the design data model may sometimes be able to segregate the two, leading to dynamic and static clusters of fields within the object.

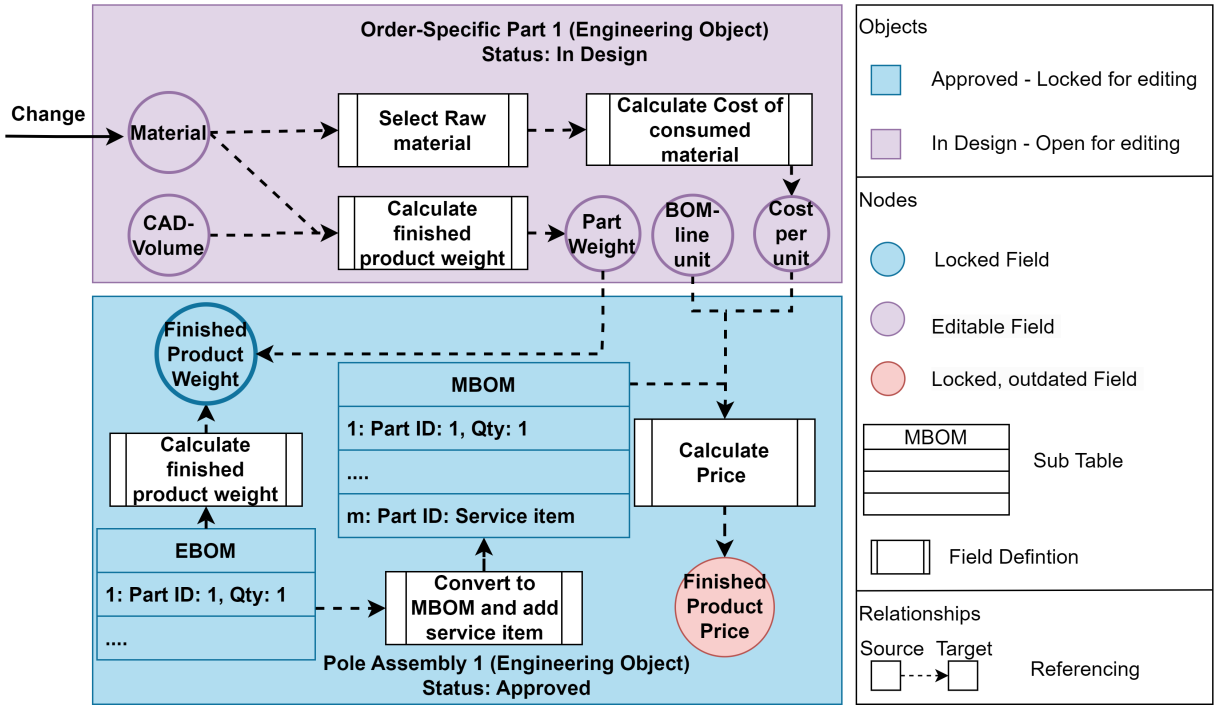


Figure 5.6: An example of the relationships between the classes, objects, data models, and the multi-class data models shown in UML.

Furthermore, if objects are deeply intertwined, the management of them as separate objects may be inpractical. Another solution could be to revise and manage all objects with are linked together with a joining relationship as a singular object. In Figure 5.7 the set of objects which could be managed together through joining relationships is given for the interobject data structure at ETO LLC.

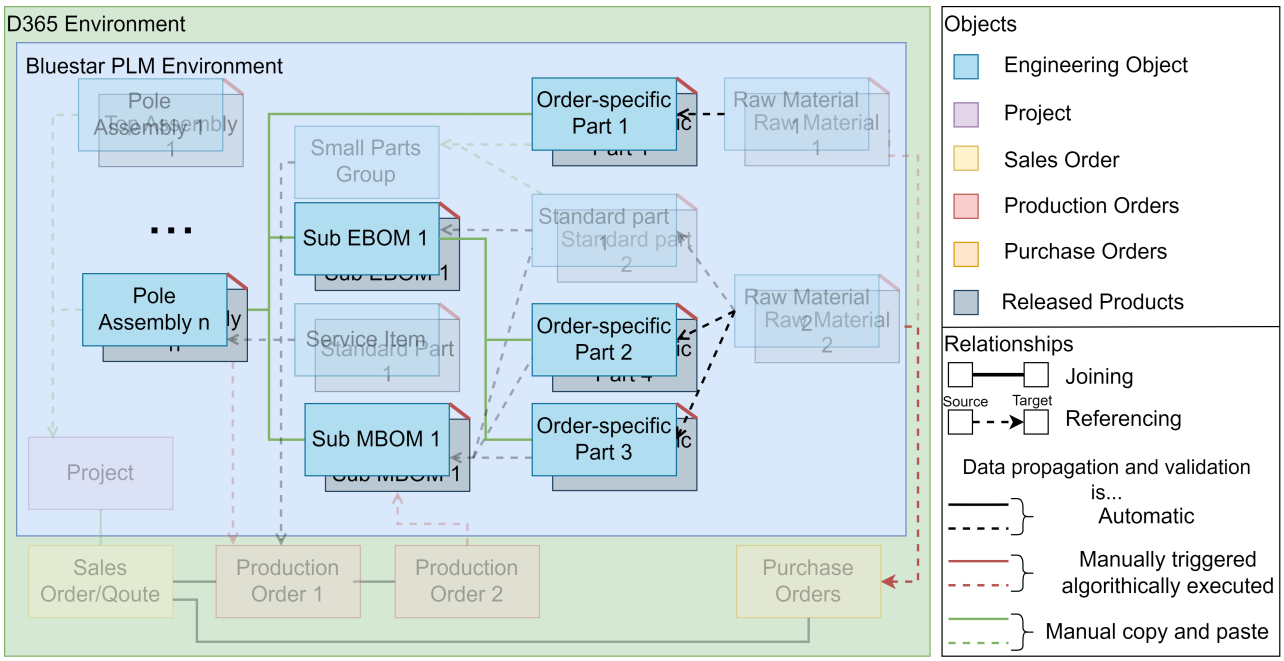


Figure 5.7: The joined data structure is created by removing the reference relations from Figure 3.1. The removed objects have been made semi-transparent.

Managing these items together would ensure that the data models can run to their fullest extent, and thus that the data of the different objects is internally consistent, while also allowing the objects to be locked once approved.

5.3 Interactions with CAx

The implementation discussed throughout this thesis has focused on integrating and interacting with Bluestar PLM and F&SCM. However, much of the technical design and analysis work is performed using various domain-specific applications. Virtually all users make use of MCAD software, such as Inventor, or ECAD software, such as Altium or EPLAN. Some also use analysis tools such as Ansys or Abaqus, others use more production-related CAx such as MasterCam or RobotStudio. Thus, there is both a large number of CAx applications used in different industries and for entirely different purposes, yet the output from these applications is critical.

Therefore, a way to utilize and record the results of the CAx applications while the information is not stored in files, and without media breaks between data producers and consumers. Other PLM platforms, like 3DEXPERIENCE, have solved this by merging the CAx applications and PLM functionality into one platform. This is not possible for Bluestar PLM, because it is not acceptable to the current users to change their CAx applications into new ones that are merged with the Bluestar PLM.

Currently, the Bluestar PLM supports the storing of CAx-files into objects, one object will have one source file and a separate drawing file, from which further files are derived, such as creating a PDF of the drawing. This means that media breaks occur when multiple CAx applications are used to define one object, such as when using additional CAx for analysis, or when different CAx are used for different parts of the lifecycle, such as for ETO LLC in using different CAD-applications for bidding and order-stage design. Moreover, it creates media breaks between the product-defining files and the consuming process or resource-defining files. An example of the media breaks of ETO LLC's data structure can be seen in Figure 5.8.

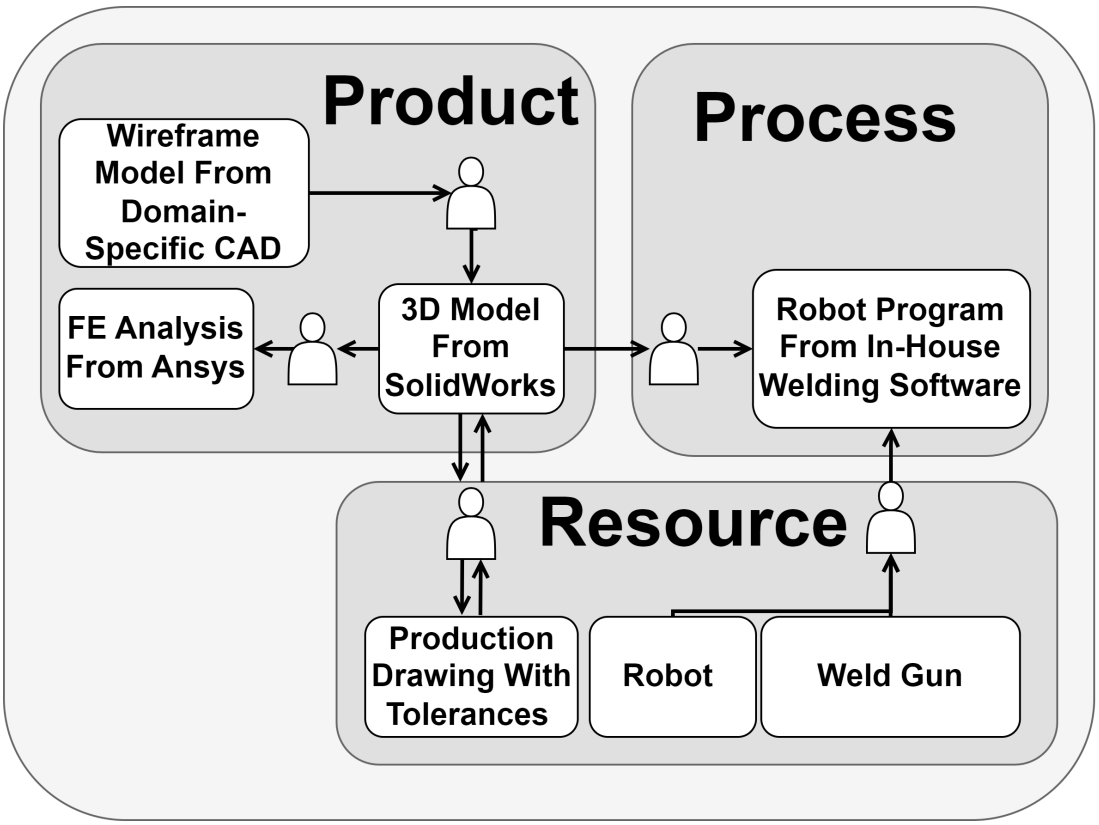


Figure 5.8: The various files associated with the product, process and resource at ETO LLC, the human busts represent the need for a person to perform synchronization and validation.

For Bluestar to transition from an object-based to a model-based platform, Bluestar must be able to resolve those media breaks to allow for the data dependencies between the files to be encoded and executed. Thus, Bluestar PLM must be able to perform or replace the following data dependencies between files.

1. Dependence between two files contained within the same object.
2. Dependence between two files contained within two different objects.

Currently, files can be "checked in" to Bluestar PLM from certain CAx, primarily from MCAD, ECAD, and Microsoft Office. These files are then attached to a new or existing object, and properties of the files are used to update the object's name, EBOM, and so on. These files can then be "checked out" from Bluestar PLM to their respective application for editing and be checked back in once this is completed.

One way to achieve the effect of model-based data structures is to let each CAx behave and output files as it currently does, and to move these resulting files into Bluestar. But to allow the customer to represent and update the dependent files when checking in. Thereby, a user working on an object can make a change to the 3D CAD model of a product, this model is used to derive a STEP-file, which is used for FEA. At a minimum, this would allow Bluestar to signify that a dependent file and associated field values are outdated. Furthermore, if each step of updating the analysis results using the new 3D model and checking these results into Bluestar PLM can be performed algorithmically. Then the effect of a model-based data structure can be created while still using file-based CAx applications.

However, the relationship between different CAxs is not easily generalizable across product structures. Consider robotic welding, where the trajectory and welding parameters would naturally change if the geometry or material of the components were to change. Thus, there is a data dependency between the CAD model and the robotic welding program. Similarly, the product may also be defined from multiple files, among them software, which may even be the same CAx as seen in Figure 5.8. where the PPR-SOS has been broken into its constituent systems, which in turn have been broken down further. Here, the PPR-structure contains several files from the same CAx but in different parts of the PPR-structure.

To allow for the results of the CAxs into the data model, the source files can be represented as data sources,

which could then be used to derive the value of other nodes, similar to the effect of referencing other objects in the data model. Moreover, as shown in Figure 5.8, some files are dependent on others. Thus, the dependent files, the target, must also be updated when the preceding file, the source, is updated. This would enable a data structure like the one seen in Figure 5.9, where the FEA file is dependent on a STEP file derived from a CAD file generated by Inventor CAD.

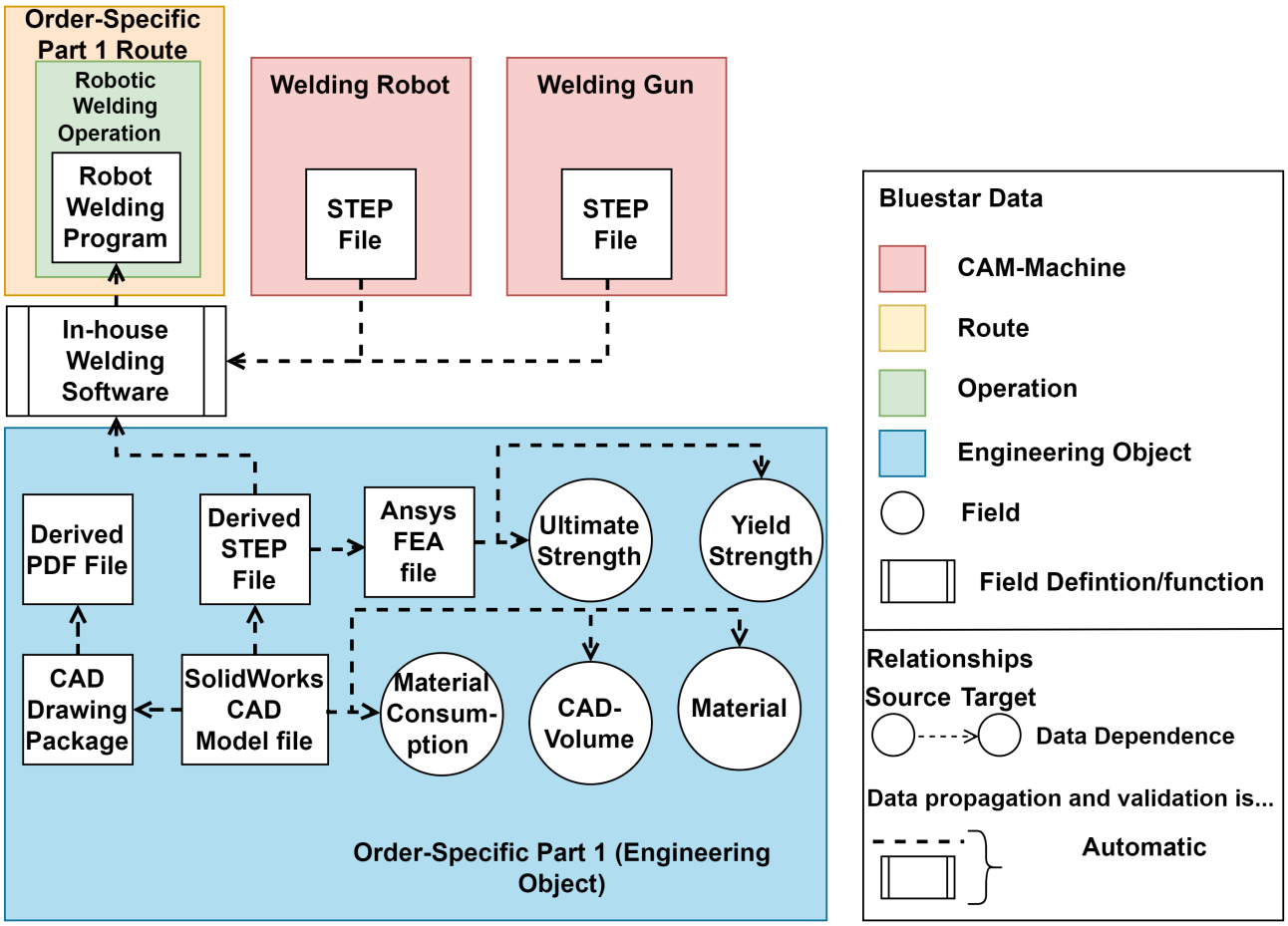


Figure 5.9: The data dependencies of each object and associated files representing the PPR structure shown in Figure 5.8.

Creating an add-in-less model-based data structure may not be achievable, as each CAx application will encode its files differently. Bluestar will have to decode these files to synchronize data into fields in Bluestar PLM. Even if the CAx application was able to output a standardized file format like STEP, a semantic mismatch may still occur, as the relation of a given STEP file to one or more objects cannot be generalized across all Bluestar PLM customers.

However, realizing MBE requires overcoming interoperability challenges, current PLM/CAD APIs and exchange standards must be able to carry PMI and semantic data. The lack of mature standards for fully defining all product information is a known barrier that existing standards, such as STEP, require human effort to fully align data[57]. Achieving a model-based data structure therefore requires Bluestar PLM to develop an add-in for the given CAx, which further requires that the CAx has an API that allows Bluestar PLM to:

1. Derive an application-neutral file format from the source CAx application.
2. Replace a referenced application-neutral file format in the target CAx application.
3. Heal potential file encoding artifacting to make the application-neutral file format compatible with the target CAx application.
4. Rerun or recalculate the analysis after the replacement.

5. Extract data fields from the CAx application and check them into Bluestar PLM.

In summary, it is not viable for Bluestar PLM's users to cease their use of existing CAx applications. Objects in Bluestar may be defined from several related source files, transition to a model-based data structure will require Bluestar to be able to manage these file relations and dependencies. These relations and dependencies are highly dependent on the particular data producers and consumers of the given use case, both regarding the company and the product. Therefore, these dependencies should be represented and executed in the data model of the given object. This has not been considered during the implementation of the semantic Networks in Bluestar.

The approach taken throughout this thesis is that PDMt must create an interface for the definition execution of a given modelling language to define the data model. However, another approach may be to define the data model in currently available MBSE tools, such as Cappella, and then create a representation or interpretation of the Cappella model in Bluestar, an example of this with a previously used data structure can be seen in Figure 5.10. For this to replace the currently implemented data model, the modelling language used would have to be formal, such that the model can be made executable algorithmically.

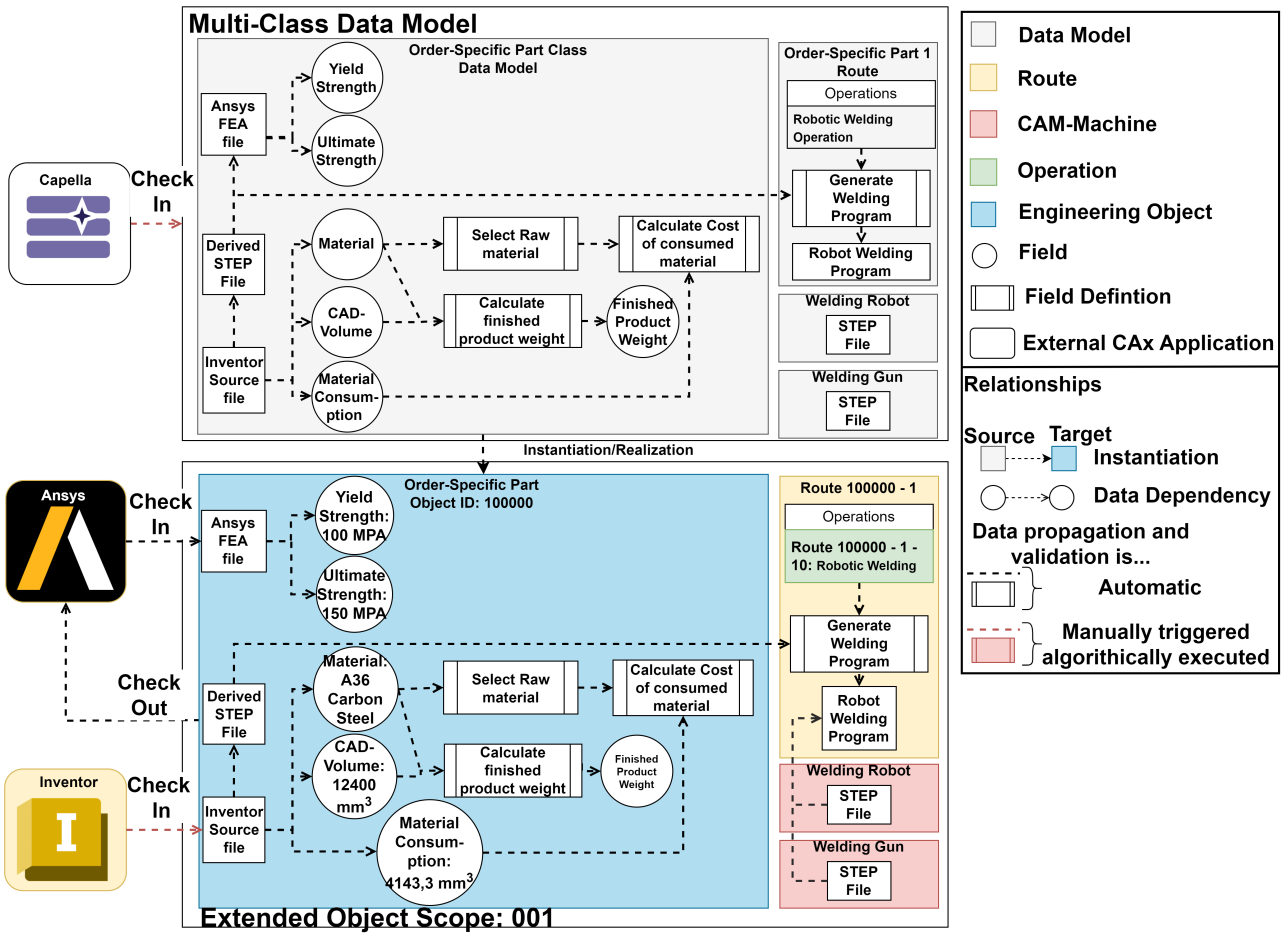


Figure 5.10: An example for the integration of CAx with the data models and the definition of data models through CAx, here shown with Capella.

This would allow the users to create their models in a proven tool, and in a language with extensive documentation and other users, which aids in the implementation by not requiring PDMt to make and maintain such documentation and training material. It would however, require that an add-in for this is created, as well as a mapping from the artifacts defined in Cappella to the data model in Bluestar PLM, this ultimately may prove more burdensome than recreating an existing modelling language in Bluestar PLM.

6 Conclusion

These challenges are exemplified well in the troublesome implementation of Bluestar PLM at the company ETO LLC. Who wanted to be able to represent their internal technical and business logic in Bluestar PLM, but does not currently do so today, as it would require bespoke development or be too time-consuming to be viable.

A number of technologies related to digital models, including MBSE, MBE, PPR, and Digital Twins, are investigated as potential solutions. The competitive advantage of Bluestar is the ability to transfer data between Bluestar PLM and Microsoft Dynamics, which is enabled by semantic networks. Moreover, they are a useful technology to represent the interactions and interrelations of digital models. This leads to the final problem formulation: "*How can a CAx-neutral semantic network be implemented into Bluestar PLM to fulfill ETO LLC digital ambitions?*".

ETO LLC's operational data structure is analysed, and it is found that a large number of manual transformations and validations are performed for each product. These validations can be explained as simple algorithmic rules, but they cannot be automated today without custom software packages from Bluestar PLM. To fulfill ETO LLC's ambitions, both new kinds of objects and the automation of Bluestar PLM's functions are required. The functions required are found to include simple arithmetic, logical branching, and the creation and manipulation of new and preexisting objects.

Bluestar PLM's customers are high-variance low-volume producers, thus, the use of any feature in Bluestar PLM is conditioned on the feature not increasing the time to design each product. For this reason, it is prohibitively time-consuming to design a new semantic network for each product. However, there is often a very large carryover of technical and business logic between products designed independently of one another. Therefore, it is decided that the semantic network owned by Bluestar PLM's classes and then added to the objects of the class.

Because neither Bluestar PLM's developers nor its customers have experience with designing and developing such features, an iterative development method is used to answer six hypotheses. This results in the creation of a new kind of Bluestar entity called data models, which are used to represent the semantic network. It is found that the users at ETO LLC have the conceptual ability to perform the required modelling and see value in several of the required features. Some features are found to be hard to understand by the company representatives, and therefore may require significant documentation and training to make use of them, which is counter to Bluestar PLM's strategy, where the customers perform the implementation by themselves with as little help as possible. A of the delivery of this thesis the implementation is not fully complete due to constrained amounts of available developer time, however, the specifications for the development has been created.

The impact of the features on the data structures at ETO LLC is found to be significant and the data structure is found to be creatable within the features specified, and the amount of time spent on data validation would be significantly reduced if the features were implemented. Next, the ability to generalize the architecture across different Bluestar customers with different process patterns and data structures is discussed, and several new features are identified. Bluestar PLM's customers rely on multiple CAx applications for much of their technical work, it is not acceptable for customers to change their CAx applications. To allow for model-based data structures, the output of these applications must update the object's data and any dependent files. Since the relationship between different file types is different between company and product, the relationship between object data and CAx files must be explicitly defined. The ability to create the data model through existing MBSE languages and tools is explored, and an implementation is suggested.

Bibliography

[1] Wenting Zou, Saara A. Brax, and Risto Rajala. "Complexity in Product-Service Systems: Review and Framework". In: *Procedia CIRP* 73 (2018). 10th CIRP Conference on Industrial Product-Service Systems, IPS2 2018, 29-31 May 2018, Linköping, Sweden, pp. 3–8. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2018.03.319. URL: https://www.sciencedirect.com/science/article/pii/S2212827118305006.

[2] Angelo Corallo et al. "Model-based enterprise approach in the product lifecycle management: State-of-the-art and future research directions". In: *Sustainability* 14.3 (2022), p. 1370.

[3] Simon P Frechette. "Model based enterprise for manufacturing". In: *44th CIRP international conference on manufacturing systems*. Omnipress Madison, USA. 2011, p. 6.

[4] Werner Kritzinger et al. "Digital Twin in manufacturing: A categorical literature review and classification". In: *IFAC-PapersOnLine* 51.11 (2018). 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018, pp. 1016–1022. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.08.474. URL: https://www.sciencedirect.com/science/article/pii/S2405896318316021.

[5] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. "Digital twin for smart manufacturing, A review". In: *Sustainable Manufacturing and Service Economics* 2 (2023), p. 100017. ISSN: 2667-3444. DOI: https://doi.org/10.1016/j.smse.2023.100017. URL: https://www.sciencedirect.com/science/article/pii/S2667344423000099.

[6] Paweł Stączek et al. "A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment—A Case Study". In: *Sensors* 21.23 (2021). ISSN: 1424-8220. DOI: 10.3390/s21237830. URL: https://www.mdpi.com/1424-8220/21/23/7830.

[7] Jun Ma et al. "A digital twin-driven production management system for production workshop". In: *The International Journal of Advanced Manufacturing Technology* 110 (2020), pp. 1385–1397.

[8] Alim Yasin et al. "A roadmap to integrate digital twins for small and medium-sized enterprises". In: *Applied Sciences* 11.20 (2021), p. 9479.

[9] Mojtaba Mahmoodian et al. "Development of digital twin for intelligent maintenance of civil infrastructure". In: *Sustainability* 14.14 (2022), p. 8664.

[10] Haidar Hosamo Hosamo et al. "A Digital Twin predictive maintenance framework of air handling units based on automatic fault detection and diagnostics". In: *Energy and Buildings* 261 (2022), p. 111988.

[11] Muhammad Hassan, Marcus Svadling, and Niclas Björsell. "Experience from implementing digital twins for maintenance in industrial processes". In: *Journal of Intelligent Manufacturing* 35.2 (2024), pp. 875–884.

[12] Dominik Heber, Marco Groll, et al. "Towards a digital twin: How the blockchain can foster E/E-traceability in consideration of model-based systems engineering". In: *DS 87-3 Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 3: Product, Services and Systems Design, Vancouver, Canada, 21-25.08. 2017*. 2017, pp. 321–330.

[13] *IOS Press Ebooks - Product Avatar as Digital Counterpart of a Physical Individual Product: Literature Review and Implications in an Aircraft*. [Online; accessed 2025-05-09]. URL: https://ebooks.iospress.nl/doi/10.3233/978-1-61499-544-9-657.

[14] Emilie Kathrine Clausen. *After-sales Redesign and Repair » Bluestar PLM*. [Online; accessed 2025-05-12]. Nov. 2024. URL: https://bluestarplm.com/modules/after-sales-redesign/.

[15] Abdul-Rahman Al-Ali et al. "Digital twin conceptual model within the context of internet of things". In: *Future Internet* 12.10 (2020), p. 163.

[16] Michael Grieves and John Vickers. "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems". In: *Transdisciplinary perspectives on complex systems: New findings and approaches* (2017), pp. 85–113.

[17] Dimitris Mourtzis. *Design and operation of production networks for mass personalization in the era of cloud technology*. Elsevier, 2021.

[18] Jessica Ulmer et al. “Usage of digital twins for gamification applications in manufacturing”. In: *Procedia CIRP* 107 (2022), pp. 675–680.

[19] Anders Jepsen. “Comparing Text-based Onboarding Methods for Virtual Reality Interactive Showcasing”. In: (2023). URL: https://projekter.aau.dk/projekter/files/535200078/IxD_MSc_Thesis_VR_Onboarding_Anders_Jepsen_Final_2.0.pdf.

[20] Maulshree Singh et al. “Digital Twin: Origin to Future”. In: *Applied System Innovation* 4.2 (2021). ISSN: 2571-5577. DOI: 10.3390/asi4020036. URL: https://www.mdpi.com/2571-5577/4/2/36.

[21] Martin Eigner et al. “Holistic definition of the digital twin”. In: *International Journal of Product Lifecycle Management* 13 (Jan. 2021), p. 343. DOI: 10.1504/IJPLM.2021.119527.

[22] Eduardo Silveira da Trindade, Cristiano André da Costa, and Vinicius Costa de Souza. “Digital twin for product design collaboration: a systematic literature review”. In: *The International Journal of Advanced Manufacturing Technology* (2025), pp. 1–17.

[23] Martin Eigner. *System lifecycle management : engineering digitalization (engineering 4.0)*. eng. 1st ed. 2021. Wiesbaden, Germany: Springer Fachmedien Wiesbaden GmbH, 2021. ISBN: 3-658-33874-1.

[24] Arman Shahbazian et al. “eQual: informing early design decisions”. In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2020). URL: https://api.semanticscholar.org/CorpusID:219814203.

[25] Haider A Al-Fedhly and Waguih ElMaraghy. “Design methodology framework for cyber-physical products”. In: *International Journal of Industry and Sustainable Development* 1.2 (2020), pp. 64–75.

[26] Aditya Akundi and Viviana Lopez. “A Review on Application of Model Based Systems Engineering to Manufacturing and Production Engineering Systems”. In: *Procedia Computer Science* 185 (2021). Big Data, IoT, and AI for a Smarter Future, pp. 101–108. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.05.011. URL: https://www.sciencedirect.com/science/article/pii/S1877050921010917.

[27] Sanford Friedenthal, Regina Griego, and Mark Sampson. “INCOSE model based systems engineering (MBSE) initiative”. In: *INCOSE 2007 symposium*. Vol. 11. sn. 2007.

[28] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[29] Yaroslav Menshenin et al. “Analysis of MBSE/PLM integration: from conceptual design to detailed design”. In: *Product Lifecycle Management Enabling Smart X: 17th IFIP WG 5.1 International Conference, PLM 2020, Rapperswil, Switzerland, July 5–8, 2020, Revised Selected Papers 17*. Springer. 2020, pp. 593–603.

[30] Saulius Pavalkis. “Towards Industrial Integration of MBSE into PLM for Mission-Critical Systems”. In: *INCOSE International Symposium*. Vol. 26. 1. Wiley Online Library. 2016, pp. 2462–2477.

[31] Martin Eigner et al. “System lifecycle management: Initial approach for a sustainable product development process based on methods of model based systems engineering”. In: *Product Lifecycle Management for a Global Market: 11th IFIP WG 5.1 International Conference, PLM 2014, Yokohama, Japan, July 7-9, 2014, Revised Selected Papers 11*. Springer. 2014, pp. 287–300.

[32] Eclipse Foundation. *Is Capella a SysML Tool ?* [Online; accessed 2025-05-14]. URL: https://mbse-capella.org/arcadia_capella_sysml_tool.html.

[33] Detlef Gerhard et al. “MBSE-PLM integration: initiatives and future outlook”. In: *IFIP International Conference on Product Lifecycle Management*. Springer. 2022, pp. 165–175.

[34] Yana Brovar et al. “Overview of MBSE and PLM Integration from a Traceability Perspective: A Mechatronic Case Study”. In: *IFIP International Conference on Product Lifecycle Management*. Springer. 2024, pp. 356–370.

[35] Pierre De Saqui-Sannes et al. “A taxonomy of MBSE approaches by languages, tools and methods”. In: *IEEE Access* 10 (2022), pp. 120936–120950.

[36] Siemens Software. *Fully integrated MBSE workflow with Teamcenter and Simcenter #HowToTeamcenter - YouTube*. [Online; accessed 2025-07-16]. URL: https://www.youtube.com/watch?v=xwXw7BAZt3Q.

[37] Kunwu Zhang et al. “Advancements in Industrial Cyber-Physical Systems: An Overview and Perspectives”. In: *IEEE Transactions on Industrial Informatics* 19.1 (2023), pp. 716–729. DOI: 10.1109/TII.2022.3199481.

[38] T. Tolio et al. “SPECIES—Co-evolution of products, processes and production systems”. In: *CIRP Annals* 59.2 (2010), pp. 672–693. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2010.05.008. URL: https://www.sciencedirect.com/science/article/pii/S0007850610001952.

[39] Ge Wang et al. “A product-process-resource based formal modelling framework for customized manufacturing in cyber-physical production systems”. In: *International Journal of Computer Integrated Manufacturing* 35.6 (2022), pp. 598–618. DOI: 10.1080/0951192X.2021.1992662. eprint: https://doi.org/10.1080/0951192X.2021.1992662. URL: https://doi.org/10.1080/0951192X.2021.1992662.

[40] Tasnim A Abdel-Aty and Elisa Negri. “Conceptualizing the digital thread for smart manufacturing: A systematic literature review”. In: *Journal of Intelligent Manufacturing* 35.8 (2024), pp. 3629–3653.

[41] Yuchu Qin et al. “Status, Comparison, and Issues of Computer-Aided Design Model Data Exchange Methods Based on Standardized Neutral Files and Web Ontology Language File”. eng. In: *Journal of computing and information science in engineering* 17.1 (2017). ISSN: 1530-9827.

[42] K. Agyapong-Kodua, Csaba Haraszkó, and István Németh. “Recipe-based Integrated Semantic Product, Process, Resource (PPR) Digital Modelling Methodology”. In: *Procedia CIRP* 17 (2014). Variety Management in Manufacturing, pp. 112–117. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2014.03.118. URL: https://www.sciencedirect.com/science/article/pii/S2212827114003540.

[43] Jan Duda, Sylwester Oleszek, and Krzysztof Santarek. “The Impact of PLM Systems on the Digital Transformation of Manufacturing Companies”. In: *International Scientific-Technical Conference MANUFACTURING*. Springer. 2024, pp. 151–164.

[44] Teppo Salmia. *Model-Based Definition – First Step Towards Model-Based Enterprise - IDEAL GRP*. [Online; accessed 2025-07-15]. June 2022. URL: https://idealgrp.com/model-based-definition-first-step-towards-model-based-enterprise/.

[45] Till Pfeiffer et al. “Concept and implementation path for model-based impact analysis in product development”. In: *Procedia CIRP* 119 (2023), pp. 1005–1010.

[46] Harald Bauer, Alexander Schoonmann, and Gunther Reinhart. “Approach for model-based change impact analysis in factory systems”. In: *2017 IEEE international systems engineering symposium (ISSE)*. IEEE. 2017, pp. 1–7.

[47] Kamran Goher, Essam Shehab, and Ahmed Al-Ashaab. “Model-Based Definition and Enterprise: State-of-the-art and future trends”. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 235.14 (2021), pp. 2288–2299.

[48] Kamran Goher et al. “Uncertainties in adoption of model-based definition and enterprise for high-value manufacturing”. In: *Advances in Manufacturing Technology XXXIV*. IOS Press, 2021, pp. 355–361.

[49] Yingfeng Zhang et al. “A model-driven dynamic synchronization mechanism of lifecycle business activity for complicated and customized products”. In: *Procedia CIRP* 83 (2019), pp. 748–752.

[50] Xiumao Yang et al. “MBD Attributes Template Method of Aeronautical Products”. In: *MATEC Web of Conferences*. Vol. 139. EDP Sciences. 2017, p. 00017.

[51] Zijie Ren, Jianhua Shi, and Muhammad Imran. “Data Evolution Governance for Ontology-Based Digital Twin Product Lifecycle Management”. In: *IEEE Transactions on Industrial Informatics* 19.2 (2023), pp. 1791–1802. DOI: 10.1109/TII.2022.3187715.

[52] Stefan Biffl and Marta Sabou. *Semantic web technologies for intelligent engineering applications*. Springer.

[53] Marta Sabou, Fajar J Ekaputra, and Stefan Biffl. “Semantic web technologies for data integration in multi-disciplinary engineering”. In: *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects* (2017), pp. 301–329.

[54] Muhammad Yahya, John G Breslin, and Muhammad Intizar Ali. “Semantic web and knowledge graphs for industry 4.0”. In: *Applied Sciences* 11.11 (2021), p. 5110.

[55] Min-Jung Yoo, Clément Grozel, and Dimitris Kiritsis. “Closed-Loop Lifecycle Management of Service and Product in the Internet of Things: Semantic Framework for Knowledge Integration”. In: *Sensors* 16.7 (2016). ISSN: 1424-8220. DOI: 10.3390/s16071053. URL: https://www.mdpi.com/1424-8220/16/7/1053.

[56] Yongxin Liao et al. “Formal semantic annotations for models interoperability in a PLM environment”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 2382–2393.

[57] Thomas D Hedberg Jr et al. “Defining requirements for integrating information between design, manufacturing, and inspection”. In: *International journal of production research* 60.11 (2022), pp. 3339–3359.

A Development Tickets for First Iteration

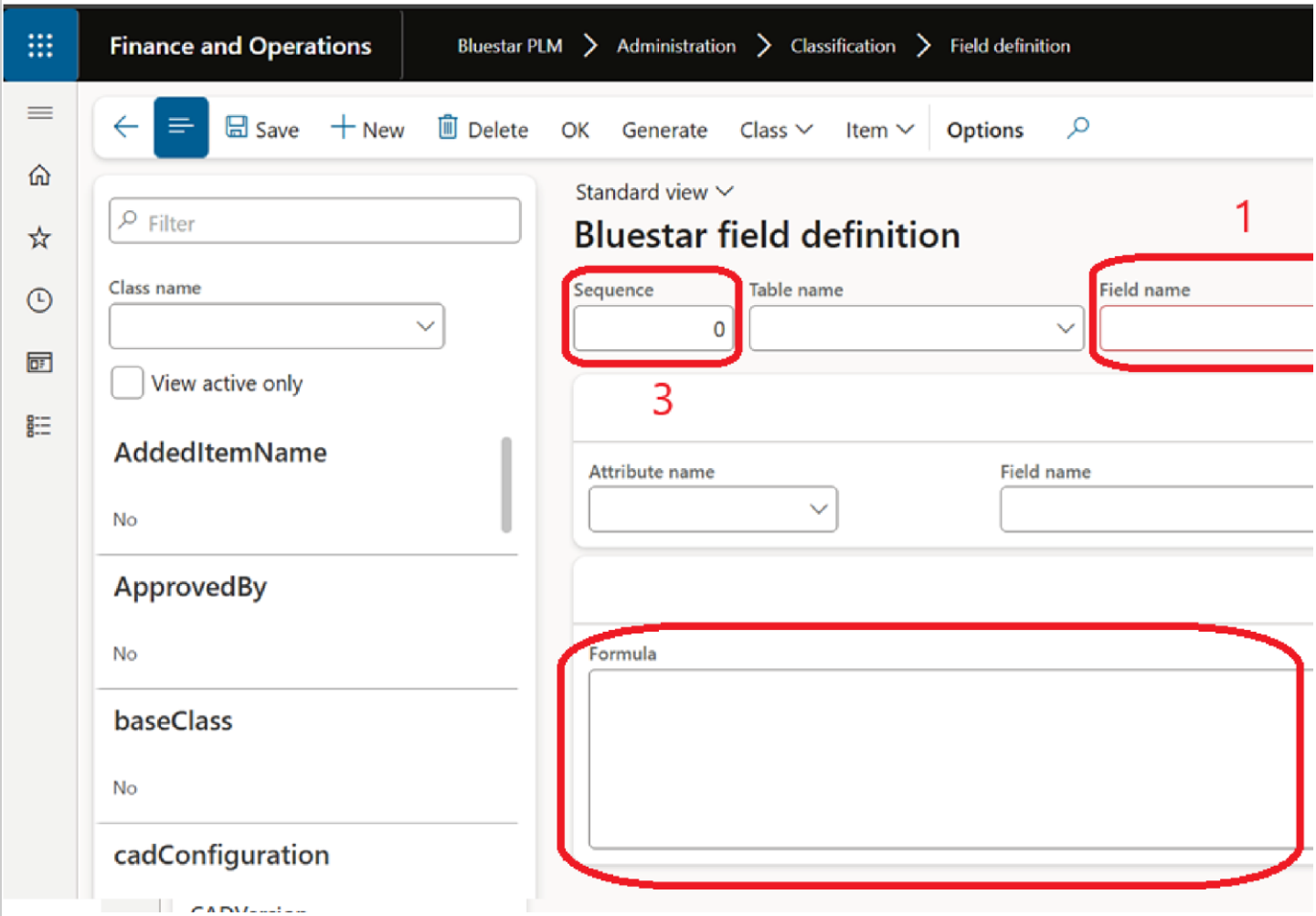
Description

As part of my master's thesis work on representing engineering knowledge as a semantic web, I would like to use fields and attributes from Bluestar's engineering obje

Thus the object's data is to be constructed as a network, in an operational environment this would be performed by the customer. To allow this, I would like to use the

For this feature to be sufficiently flexible, it requires the Field definitions to be able to perform simple branching via If-statements and perform addition, subtraction, mul

So i suggest making three main changes. As seen in the picture below



1: The fields available to be defined should include the assigned class' attributes.

This also means that the output of the field definition must be compatible with the attribute's data type. Thus the allowed output data type should be visible as well.

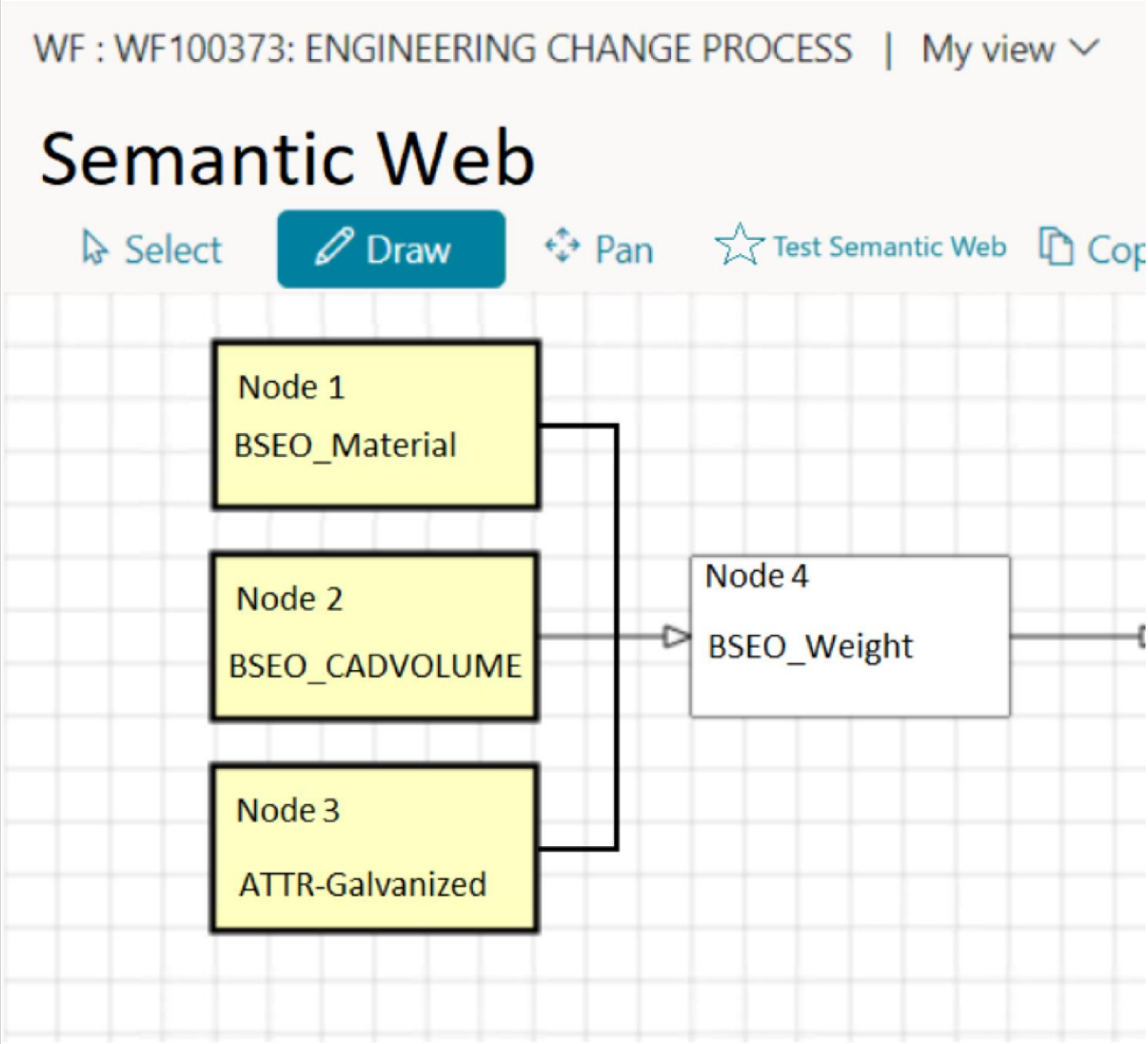
2: To Perform the processing and create the required output i suggest adding the following operators/functions to the field definition formulas. since all the field values

1. Addition, subtraction, multiplication and division of integers and reals
2. Int2Str, Str2Int, - Functions to convert the arguments passed to it from int to string and vice versa
3. Real2Str, Str2Real - Functions to convert the arguments passed to it from real to string and vice versa
4. Real2Int, Int2Real - Functions to convert the arguments passed to it from int to real and vice versa
5. If-statements - to be compatible with the logical statements such as those used in workflow rules.
6. The ability to traverse and access entries in a list, as described in PDMRD-349 SCOPE DEFINITION Create an overview form of Product semantic web - Jira|ht in the form of a list. So with two links of the link tag "Work Instruction" one link to object 200000 and a link to object 200001 if i write work_instruction.ObjectID is

In general this would make the field definition resemble a function call rather than the string concatenation it is now.

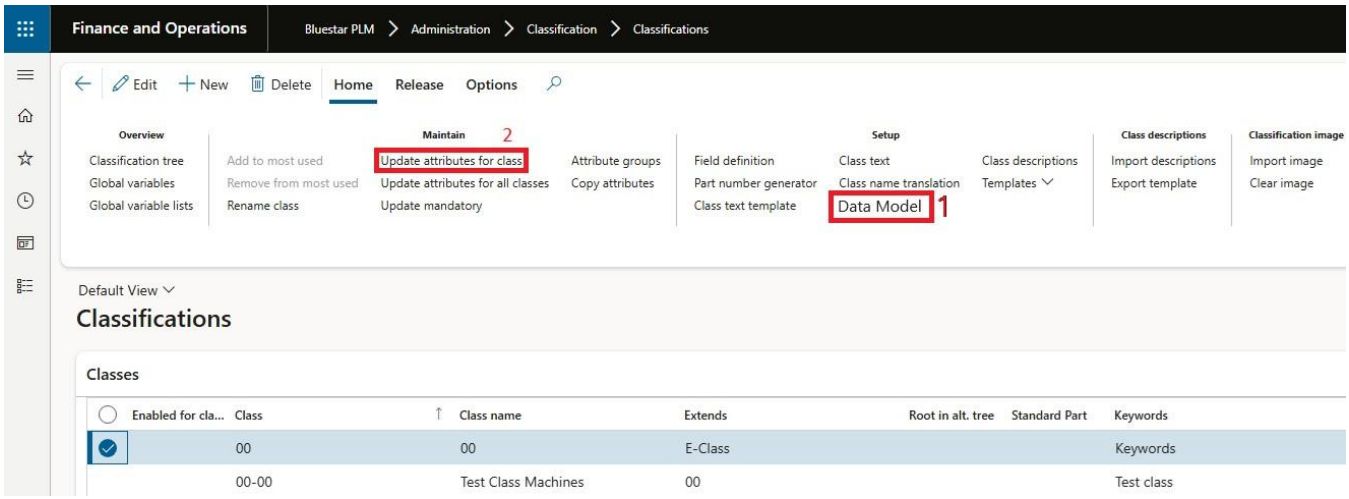
3: In accordance with another development ticket the order of evaluation of field definitions becomes important when succeeding field definitions rely on the output of

An example of the suggested syntax of the field definition can be seen below.



Description

As a part of representing products as semantic webs and the use of field definitions for this purpose, the user will need a way to view field definitions for a given class in a network form, showing the relation between other object and classified object's fields while being able to add to and edit the existing field definitions.

1. 

   The data model of a class replaces the "field definitions" and should allow the user to specify how an object can be linked to other record types and what the relationship between them is.
2. The "Update attributes for class" should also apply the changes of the data model as defined by the user. And the changes to the data model should also set the "Modified" flag for the class, even if no changes to the attributes have been made.

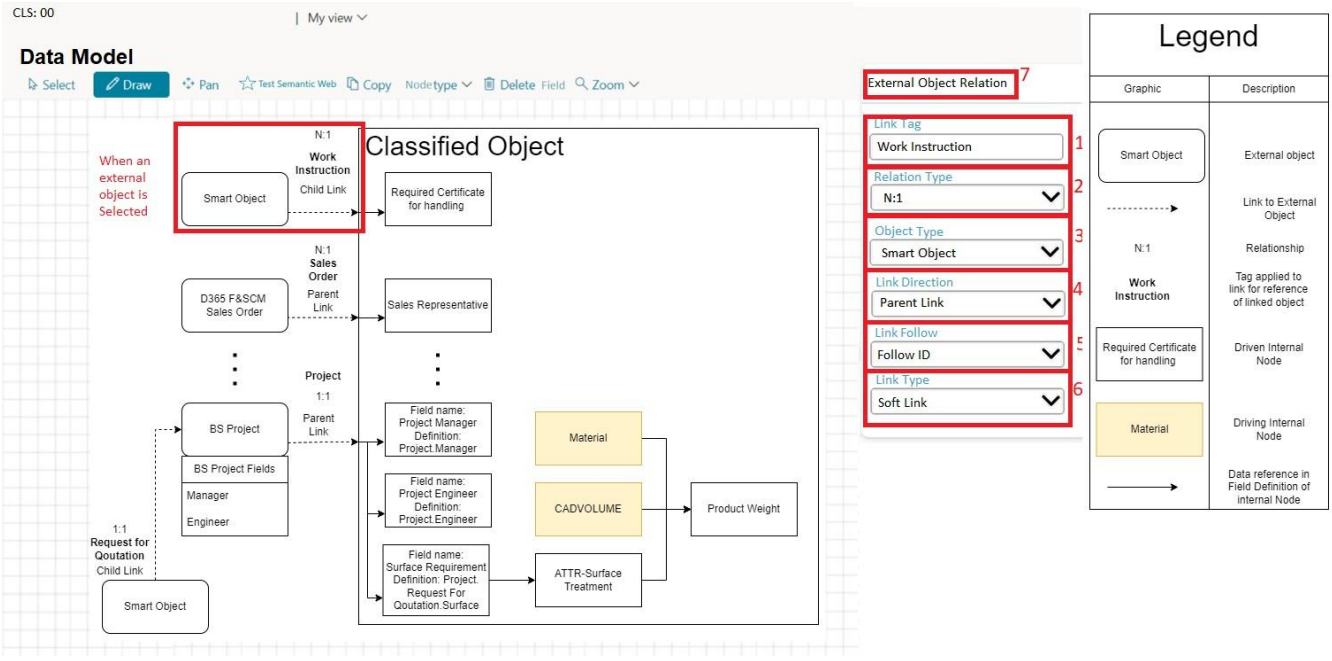
The data model form has four functionalities, which will be described in detail below

1. Defining relationships with other object types
2. Applying a relation to a link
3. Defining relationships between Object fields and attributes
4. Testing the data model

Defining relationships with other object types

This feature describes how an object that has been assigned to the selected class should be able to be linked to other kinds of objects and what the link represents. A suggestion for the interface, based on the workflow task form, can be seen below; the red rectangles and the legend are annotations and

should not be included in the interface.

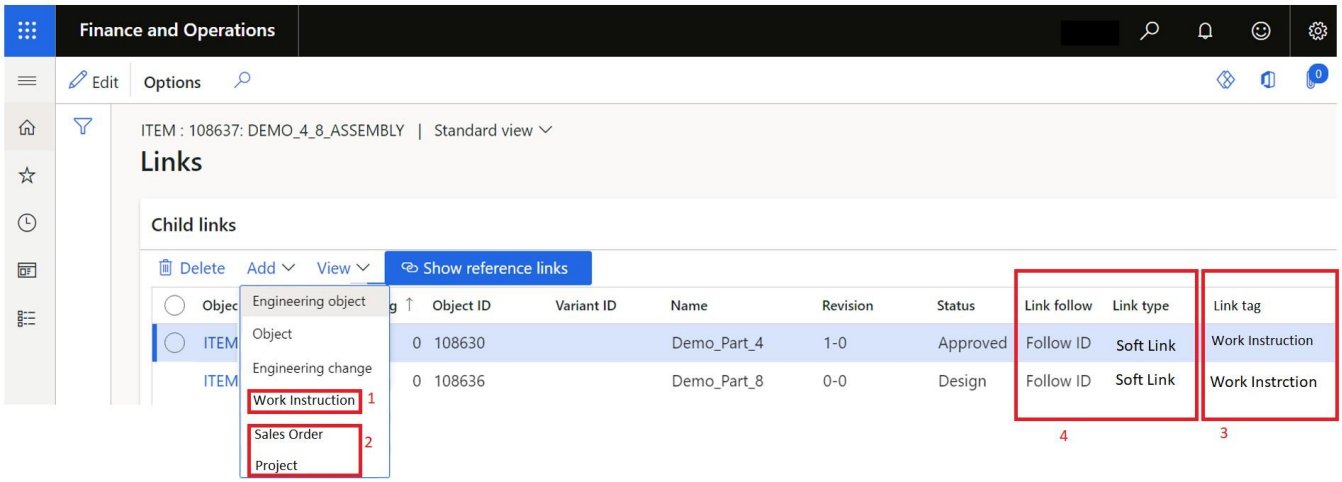


1.

Link Tag: a String variable, must be Unique within the current data model, filling this field is mandatory as this field defines the ID for referencing the linked object.

2. **Relation Type**: Single dropdown. Options are (1:1 and N:1). This defines how many instantiations of a given external object link may be made to each object of the class. If 1:1, then the classified object may only have ONE link with this link tag. If N:1, the link tag can be applied to any number of links during link creation.
3. **Object Type**: This defines the data table that can be linked to for this external object relation. The external object relation allows objects of this type to be assigned as the given external object relation.
4. **Link Direction**: Defines whether the external object link is a parent or child of the linked object.
5. **Link Follow**: should set the link follow of the link, and should have the same options as the manually defined link.
6. **Link Type**: should set the link type of the link, and should have the same options as the manually defined link.
7. This menu should be opened when the rounded square of the external object or its link is selected.

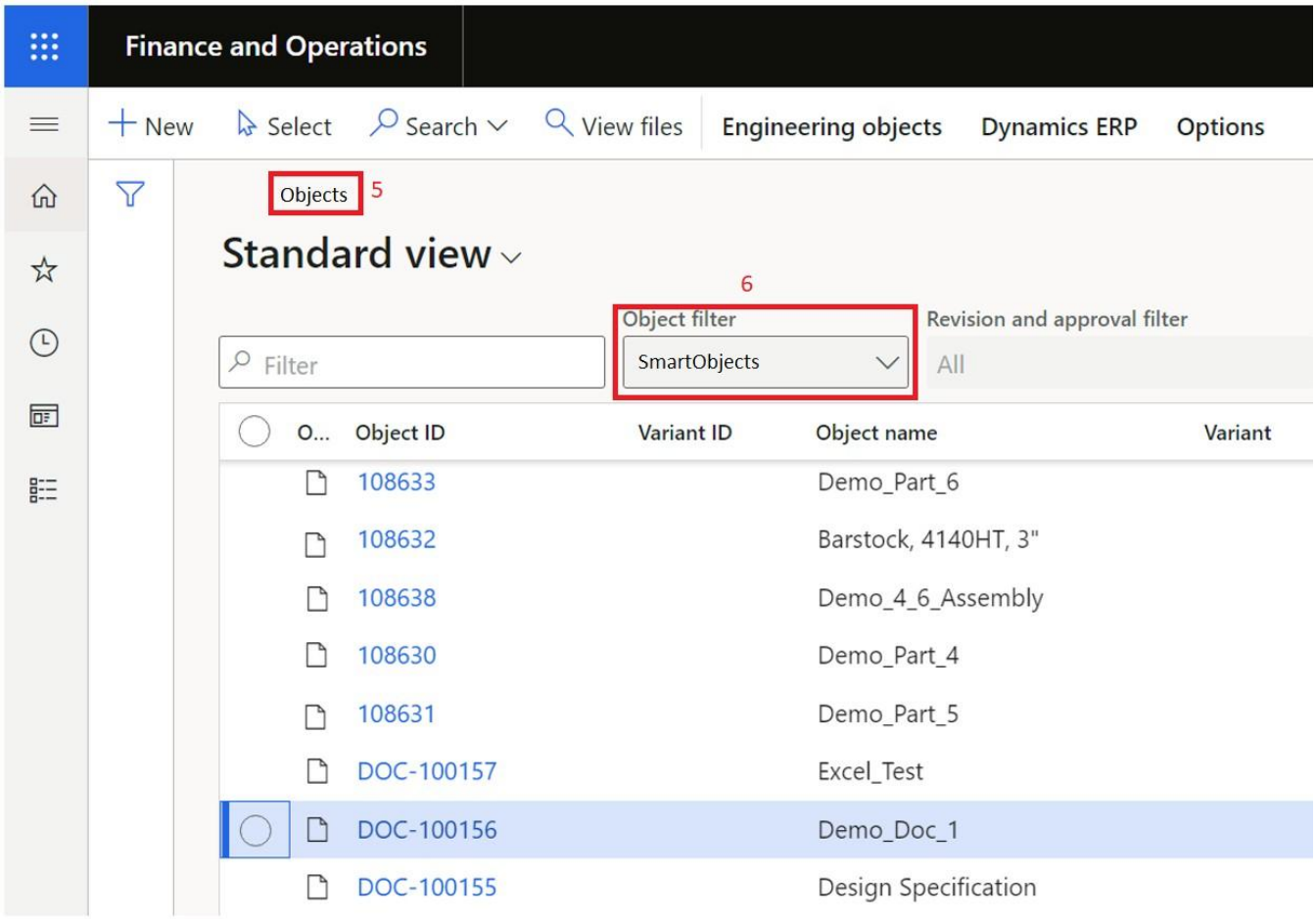
This interface defines an “external object relation,” which is a template for links between the classified object and others; thus, when creating a link, the user selects the link tag of the external object relation, which opens the object selection form filtered for objects of the selected object type. Once the external object relations have been defined, an object, such as an engineering object, is assigned to the given class. Objects previously assigned to the class should be changed by using the “update attributes for class”. **Applying a relation to a link**



1.

The Link Tag is visible in the “Add” category.

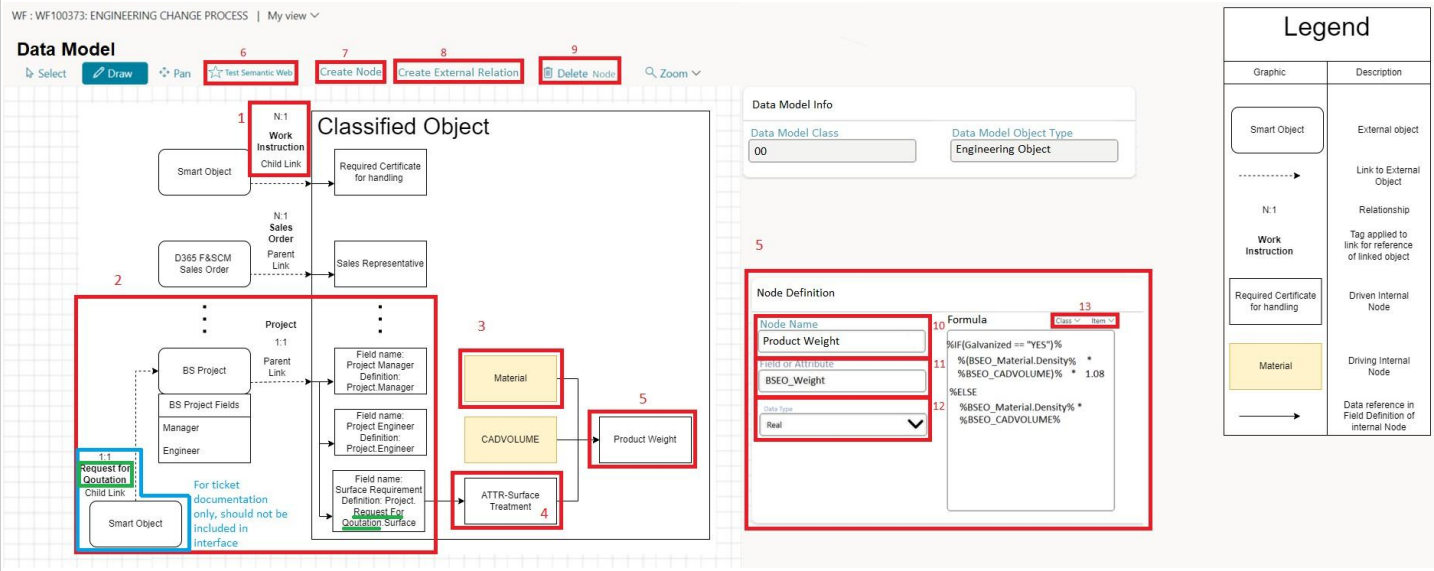
2. So are the other link tags defined in the data model of the class.
3. The Linked tag can be seen as a column.
4. The Link Follow is set to “Follow ID” and Link Type is “Soft Link” because these are defined in the external object relation.



1. The search shows any object type and is filtered using point 2
2. The objects are filtered to only show the object type allowed by the external object relation, in this case “SmartObjects” because this is defined by the data model as seen above.

Once “Select” is clicked, the link is created, and the fields are set from the external object relation.

Defining relationships between Object fields and attributes

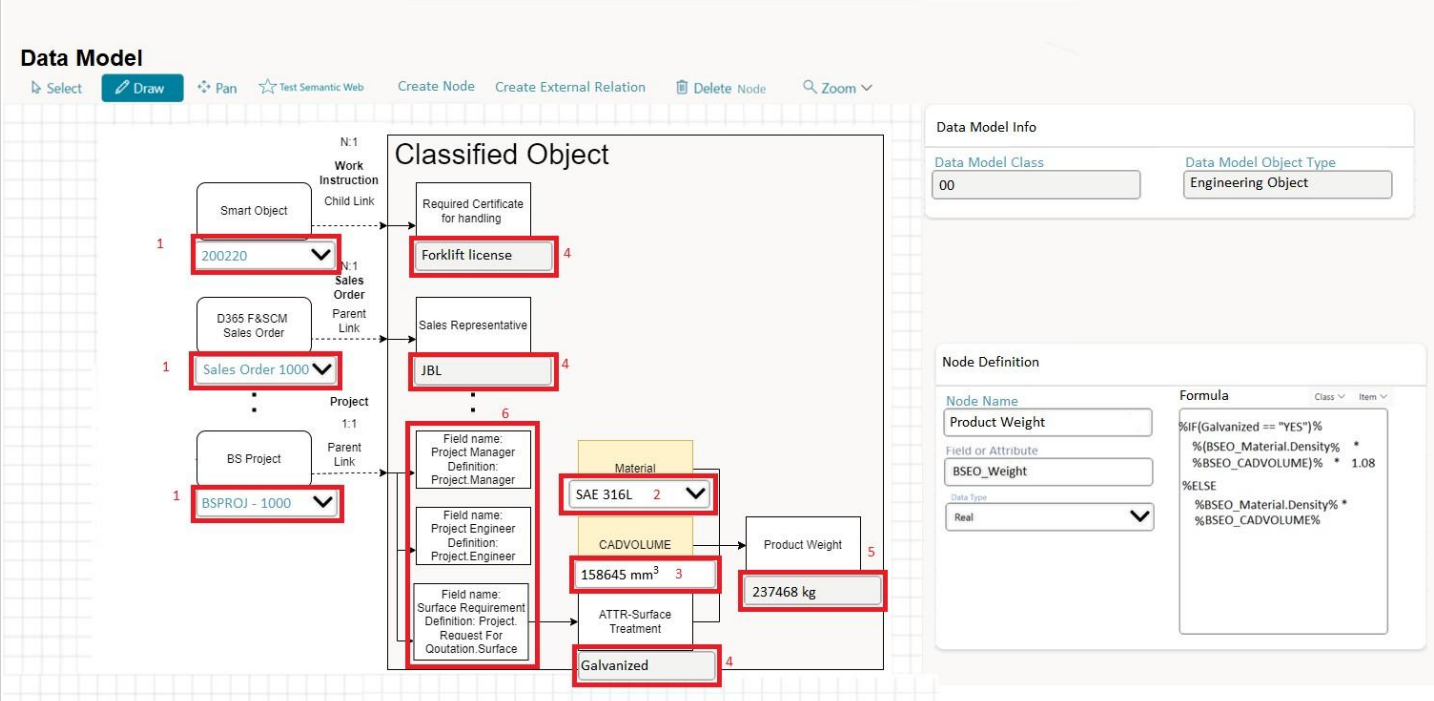


1. **Link Definition**: The external object relation’s “Relation type”, “Link Tag”, and “Link Direction” are shown above the relation.
2. **Referencing Data Across Links**: A Node’s definition can include the fields and attributes from other objects linked to the classified object. This is performed through the node definition.
3. **Driving and Driven Nodes**: Nodes without inputs are driving; those with inputs are driven. The driving nodes are yellow, and the driven should be shown in white.
4. **Interaction with Attributes**: Nodes can represent both ordinary data fields from the data model’s object type and the attributes of the class.
5. **Node Definition**: When selecting a node, the definition fasttab. This includes the formula section, which defines how to generate the value of the associated field and attribute. The syntax and evaluation of this field are described in https://bluestarplm.atlassian.net/issues/PDMRD-348?filter=-1&jql=reporter%20%3D%20currentUser%28%29%20ORDER%20BY%20updated%20DESC.
6. **Test Data Model**: Described below

7. **Create Node**: This function should allow the user to draw a node similar to tasks. Once drawn, the new node is selected, and the **Node Definition** fasttab is opened.
8. **Create External Relation**: This function should allow the user to create a new “External object relation” which can be drawn similar to a task in the workflows form, once drawn, the new external object relation (the rounded square) is selected and the external object relation fasttab as seen above in **Defining relationships with other object types** point 7.
9. **Delete Node**: This should delete the currently selected node and the data references of it, and the node's incoming references.
10. **Node name:** This field specifies the name of the node as it is displayed in the data model.
11. **Field or Attribute**: This field defines which field or attribute the node should put the output of the formula into.
12. **Data Type**: The field defines the return data type of the node's formula.
13. Class and Item Dropdowns are the same as the ones in the current field definition screen.

Testing the data model

Given the complexity of the data model, the user would have to be able to test it.



1. **Testing External Object Relationship**
2. **Testing the values of nodes with dropdowns**
3. **Testing the values of nodes with Writein**
4. **Evaluated Node values'**
5. **Final Node**
6. These nodes should also have their values, but it has not been showed above due to space and graphical constraints

B Development Tickets for Second Iteration

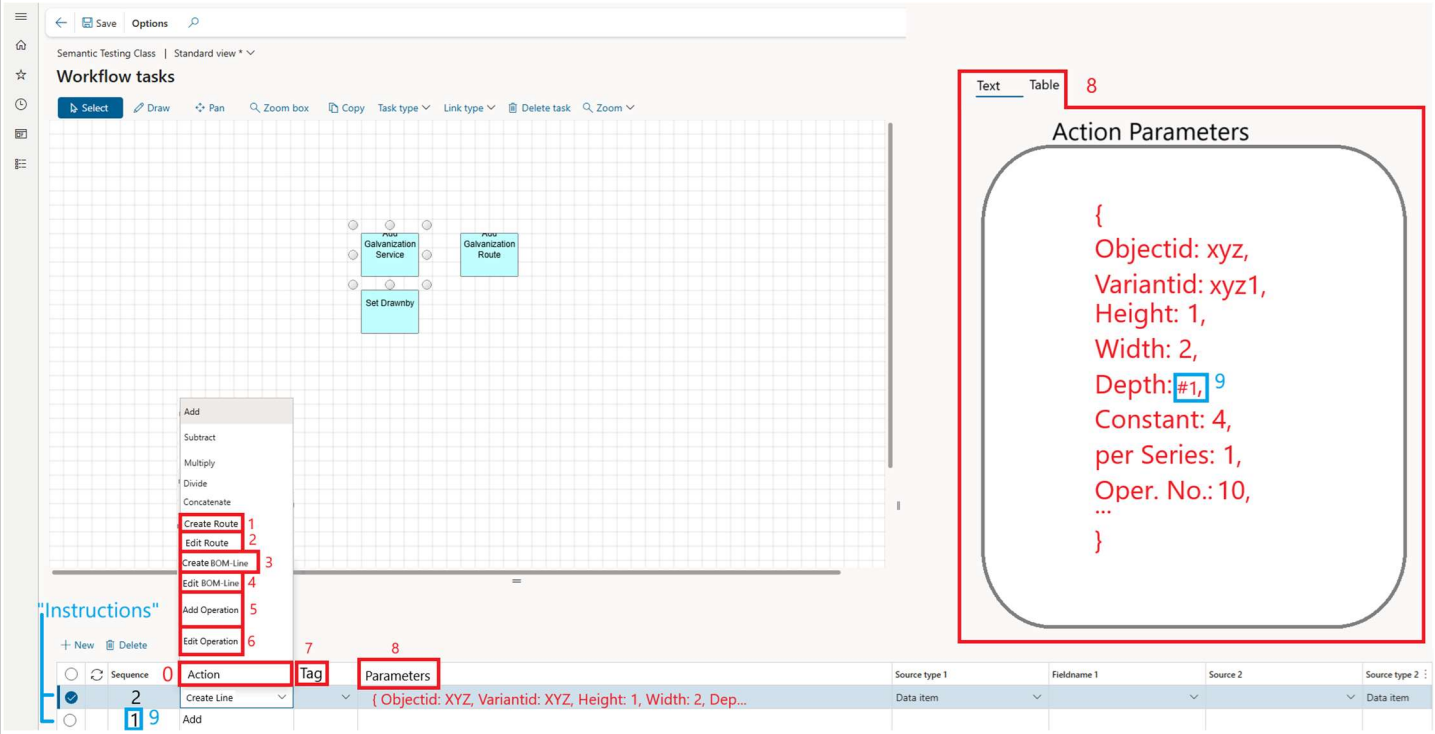
Description

Some customer use cases require for the nodes in the semantic web to be able to create and edit various records in the BOM lines and routes tables. Thus the needed functionality is the ability to Create and edit Routes, operations and BOM-lines as specified in **Image 1**.

As a data model can only pull data from data sources, and evaluate the defined field definitions, the BOM and routes can only be created in the assigned object, No new records or changes to current routes are possible outside of the classified object.

During testing of the data model through the “test data model” function the resulting BOM-lines and routes should not be created permanently.

Image 1



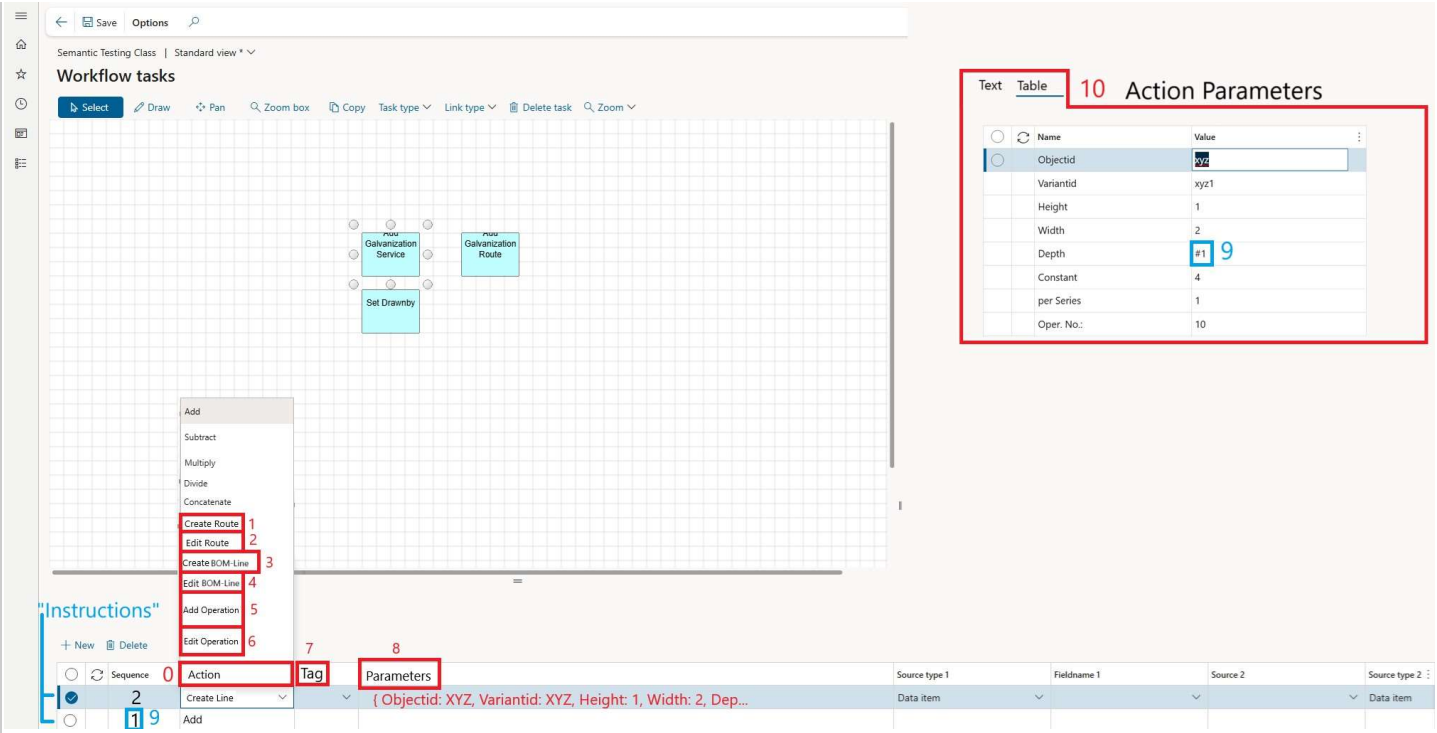
The table at the bottom of the interface is the “instructions” which will be run when the given node is evaluated. This table will be referred to as the “instructions table”.

1. New addition of options to the Action Coloumn in the Instructions-table to allow the creation and editing of Routes, Operations and BOM-lines.
2. The “create Route”-action should create a new route on the classified object, if the classified object is not an item or a BOM engineering object, then the instruction is skipped. The “Tag” coloumn should be used to be able to refer to the route after creation. The parameters field further descriubed in “8” should be used to define the values of the route upon creation. Thus the “Create Route”-action also automatically includes an “edit Route” action to ingest and execute the “Action Parameters”
3. The “edit Route” action should change the route which is refered to by the tag, given in the tag field, if no tag is supplied the instruction is skipped.

The edit Route-action can only change the fields of the route which is allowed to be changed manually by a user.

4. The action “Create BOM-Line” will create a new BOM-line on the classified object, if the classified object is not an item or a BOM engineering object, then the instruction is skipped. The addition of a BOM-line should not create a ref-link to the object referenced in the BOM-line. The position number of the BOM-line is chosen as the next available number, as with a manually created BOM-line. If no VariantID is supplied the non-variant version of the object is used, if the searched for item does not exist the instruction is skipped. If no specific revision is given the newest Approved revision of the item should be used, if no approved version exists then the newest.
5. “Edit BOM-Line” action should change the BOM-line which is refered to by the tag, given in the tag field, if no tag is supplied the instruction is skipped, there is no BOM-line with the tag, the instruction is skipped. The edit Route-action can only change the fields of the route which is allowed to be changed manually by a user. If the position specified in the “Action Parameters” is not available the instruction is skipped.
6. “Add Operation” will add an operation to the Route last created, or edited, this operation will contain the tag specified in the “tag” field. This action will use the action parameters similarly to the previous create and edit actions. The Operation Number for the new operation will be the same as the one applied if manually created, thus 10 higher than the currently highest operation number.
7. The “Edit Operation” action will use the value in the “Tag” field to search for an operation in the Route last created, or edited, to edit, the edits to the operaitons field values will be given by the “Action Parameters”.
8. The “Create Route”, “Create BOM-line” and “Add Operation” actions will use the value in the “Tag”-field to create a new record with that tag, whereas the “Edit”-actions will use the tag to identify previously created routes, BOM-lines and added Operations. For “Add”, “Subtract”, “Multiply”, “Divide”, and “Concatenate” the “Tag”-field does nothing, and may be hidden.
9. The “Action paramaters” is a new coloumn in the instructions table, due to the length of the field it should be synchronized with a multiline textbox on the right of the instructions table. Before an action of a given instruction is chosen the “Action parameters”-field and mulitline textbox is empty, once the action is set, such as “Create Line” the manually editable fields of a BOM-line is generated as JSON format as seen in the image above. The format should be fieldName: Value, similar to the format used by the configurator service queue. The values of a given BOM line should not be loaded, but should be empty. The JSON should be convertable to a table as seen in (10) below, similar to the domain-value table of the configurator service queue. The user should be able to move back and forth between the two views. Once the user clicks out of the field the value in the “Actions Parameters” multiline-textbox or table should be synchronized into the “Parameters”-field in the Instructions table. if the user clicks into one of the fields of an instruction the instructions “Action Parameters” should be loaded into the Multiline-textbox. If the mandatory field values for the given action is not fulfilled, then the instruction is skipped,
    1. for Routes the required fields are 1. Company
    2. Site, if the selected site does not exist in the company, the instruction is skipped.
    3. Name
2. For BOM-lines this is
1. ItemID
3. For Operations this is
1. Operation as defined by the Operations list from “Production Control/Setup/Routes/Operations in F&SCM
2. Route group as defined by “Cost management/Manufacturing accounting policies setup/Route groups” in F&SCM
10. Referencing the output of a previous instructions is performed by “#<Sequence>”, If the sequence referered to does not exist, then nothing is done to the field refered in the “action parameters”.

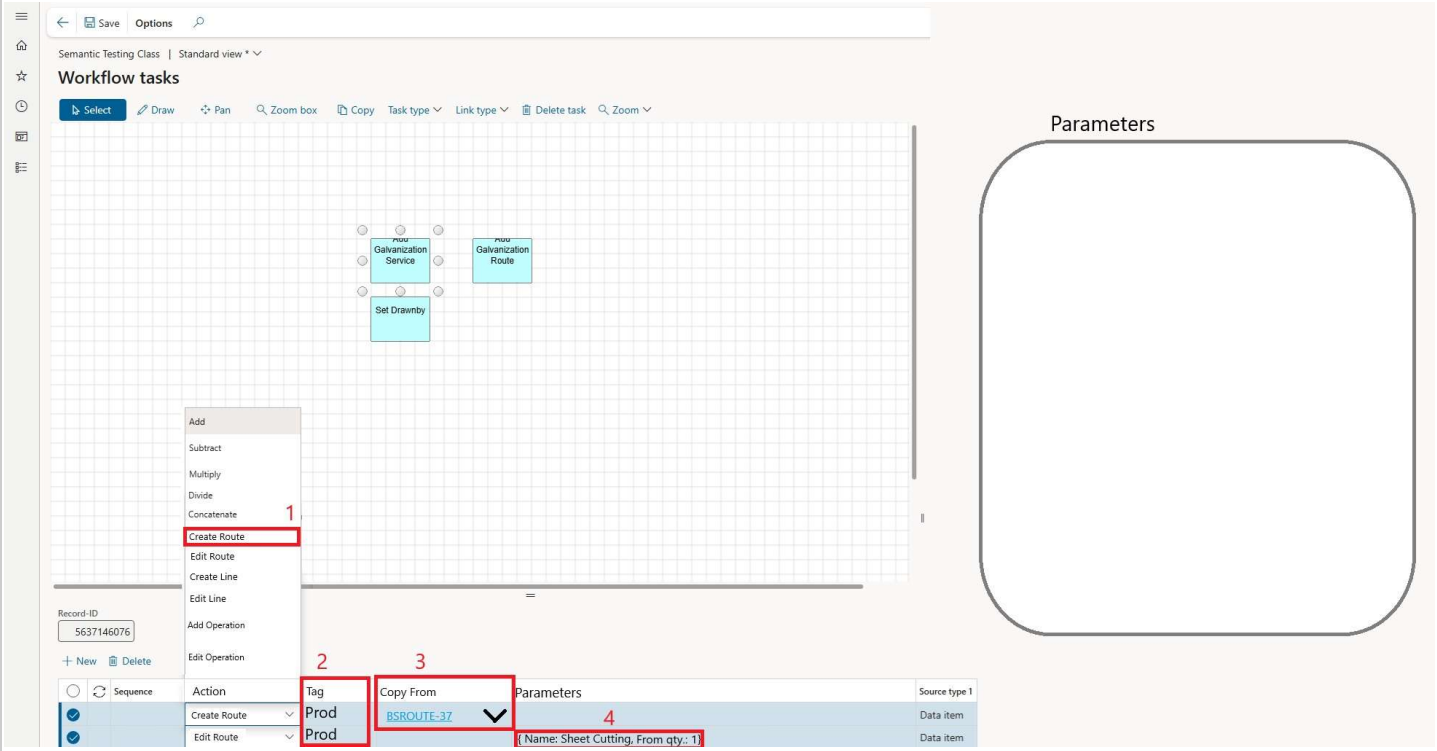
Image 2



1. Alternative Table view of the JSON format for the Action Parameters. Which should parse the JSON and convert it into a table of fields and their specified values.

Below a set of concrete examples will be given.

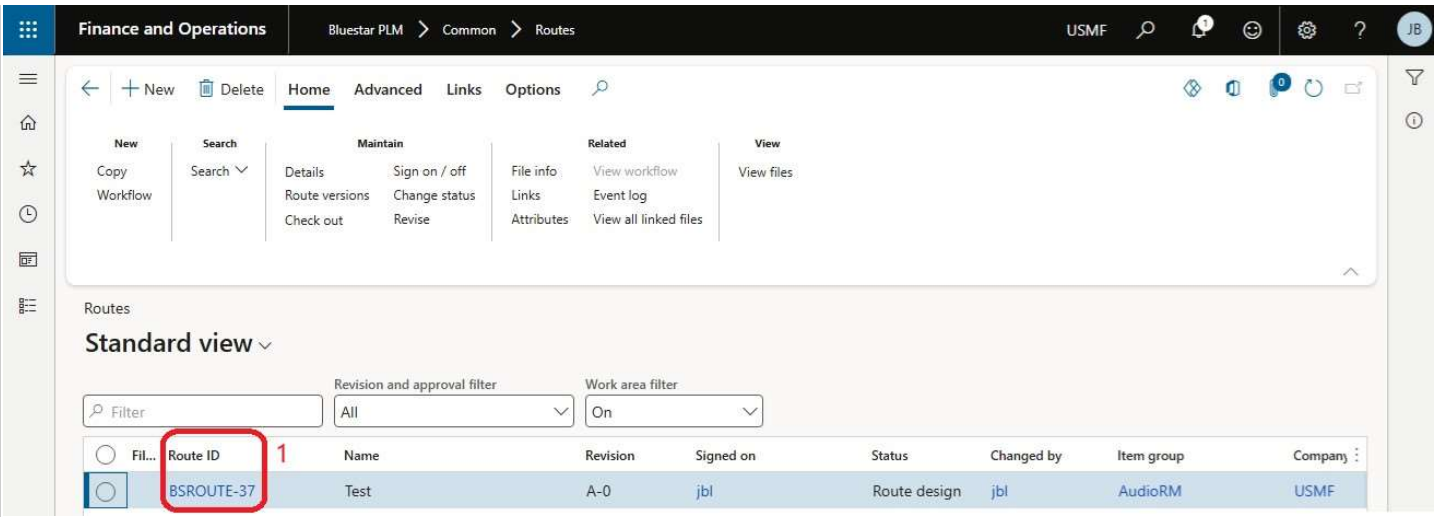
Image 3



1. First the action “Create Route” is selected
2. Second the tag “PROD” is applied to the instruction which will be set on the created route on execution.
3. A new Coloumn is to be added to the instruction table if the action is set to be “create Route” or “Create BOM Line”, for routes the user can choose between previously existing routes which have been marked as “Yes” in their “Template”-field, the selected route will then be copied to the classfied object on execution along with its operations and field values. If the action “Create BOM Line” is selected the user can select between engineering object BOM objects which have been set as “yes” in their “Template”-field, on execution of the instruction the selected BOM's lines is then copied to the classified object. Since the field values are derived from the object specified in the “Copy from” field, the Action Parameters are not specified and are not used if previously specified.
4. Once the route has been created a later instruction can edit it be referring to it via its “Tag” and by specifying the values in the “Parameters”

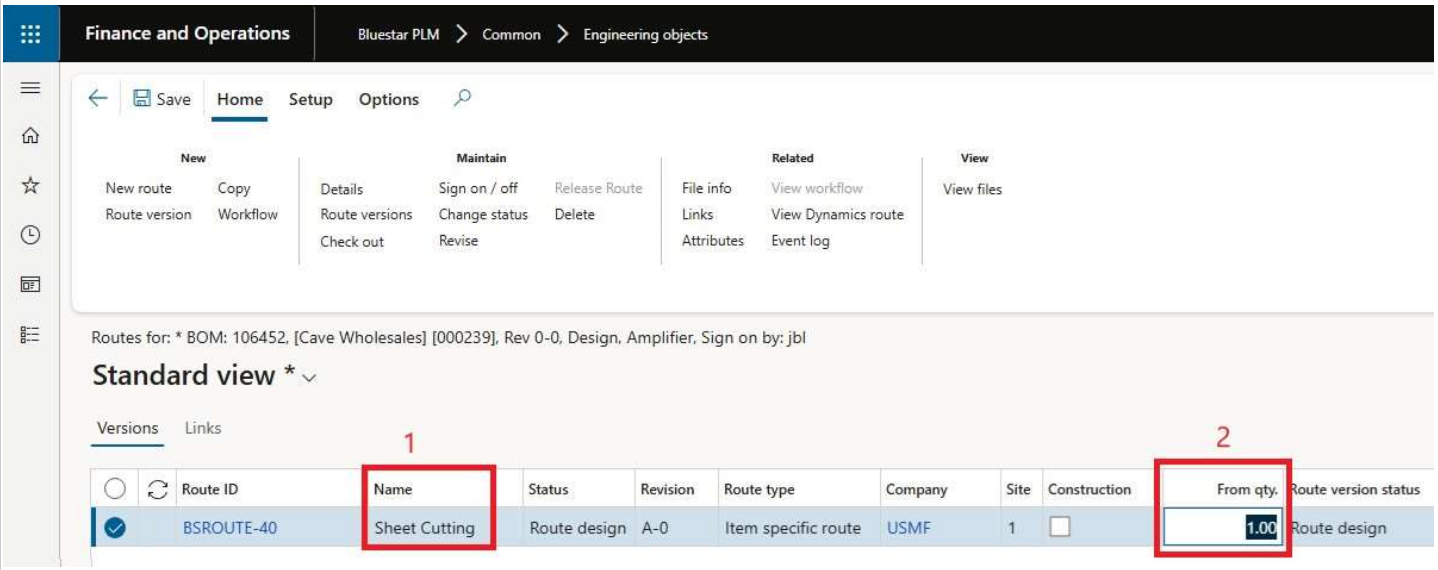
If the Line, route or BOM tag is already exists then the instruction should be skipped, if the tag the “edit Route/Line/Operation” does not exist the instruction should be skipped

Image 4



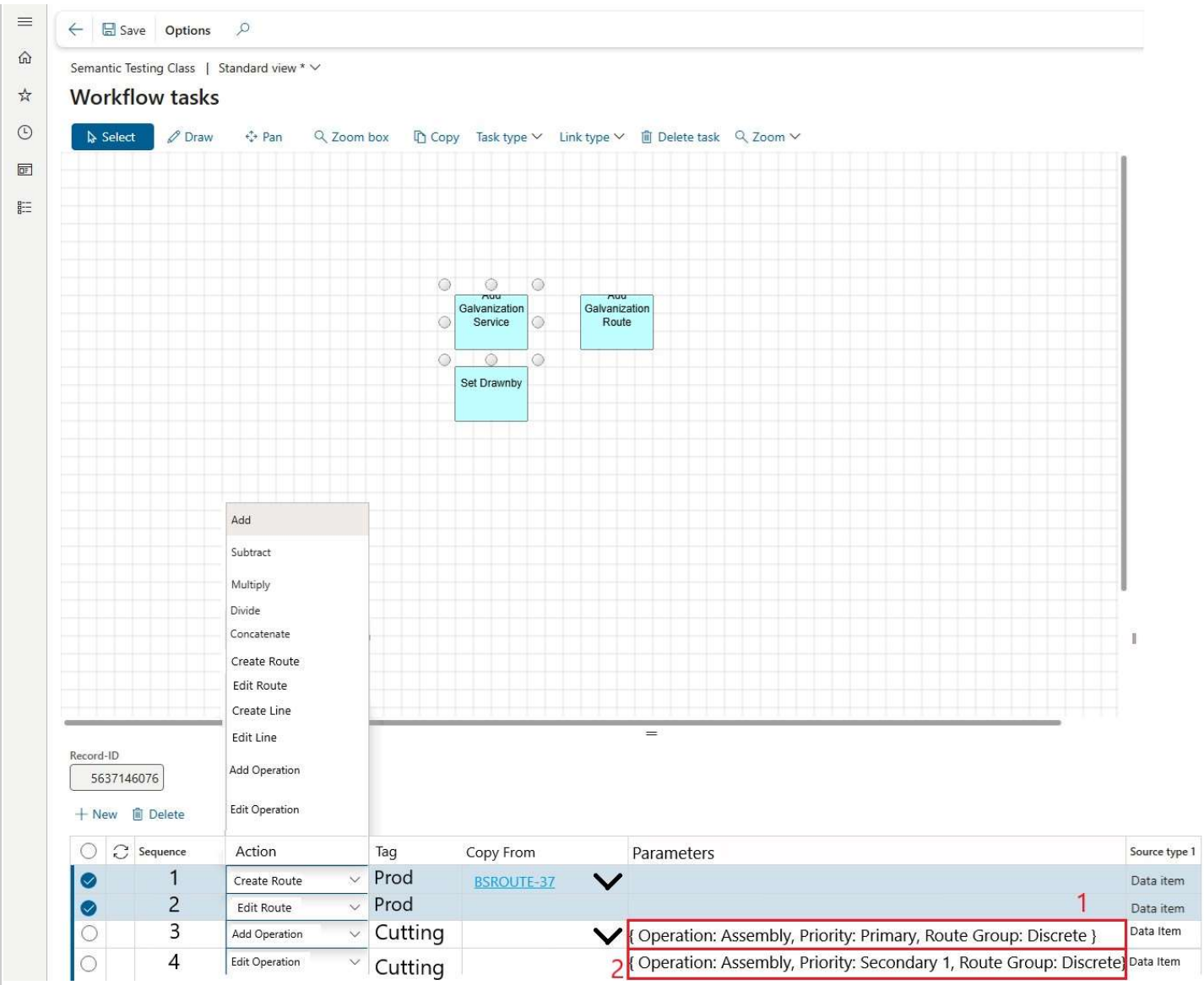
1. Here the Route whose route ID has been referred to in the first instruction in the image above.

Image 5



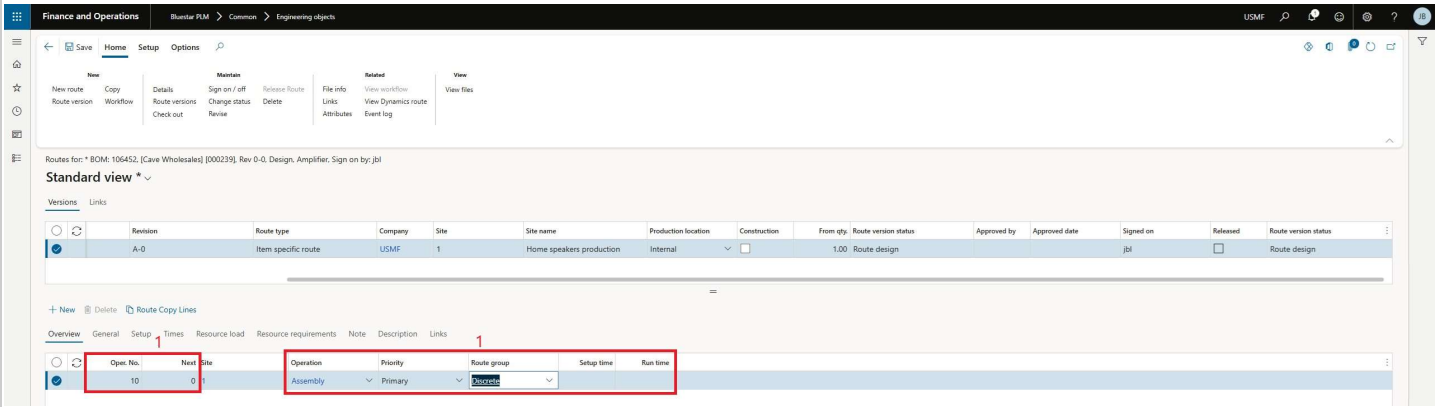
1. + 2. Because of the second “edit Route” operaition the new route has recieved a new name and from qty as specified in the action parameters in 4. of **Image 3**

Image 6



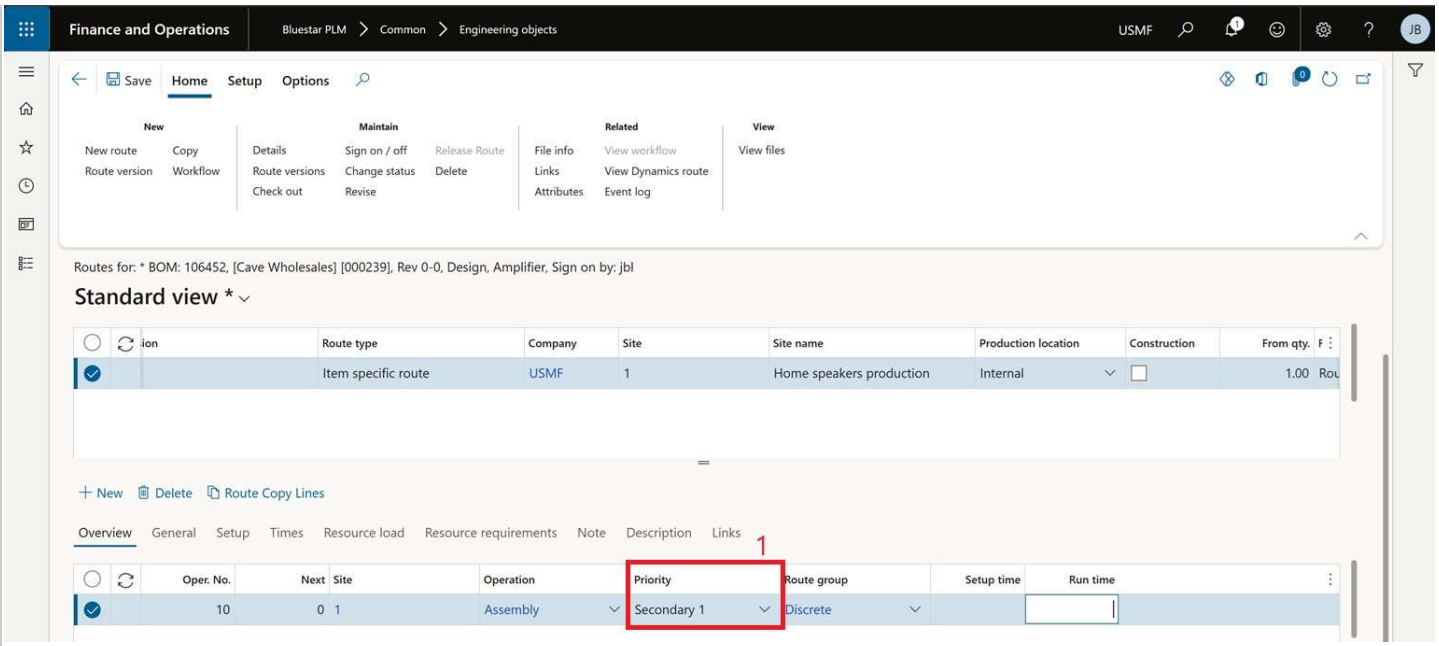
1. Later an “Add Operation” instrcution is added to the node, this will be added to the Route referred to by the tag “PROD” because it is the latest added route in this node. The operation will carry the “Cutting” tag. Operation cannot be created from a copy this the values of the operation are given by the “Action Parameters” field.
2. Last an “Edit Operation” will be used to edit the operation carrying the “Cutting” tag, and will carry out the field changes given in the “Action Parameters” field.

Image 7



1. On execution of the third instruction of **Image 4**

Image 8



1. On execution of the Fourth instruction of **Image 4**, the originally specified values of the Operation is overwritten with those specified in (2) of **Image 6**

Generated at Thu Jul 31 21:34:19 UTC 2025 by Johannes Bach Larsen