Evaluating Process Parameter Interdependencies Based on Knowledge Graphs in Manufacturing

Tom Jeleniewski[1][a], Aljosha Köcher[1][b], Hamied Nabizada[1][c], Jonathan Reif[1][d], Felix Gehlhoff[1][e] and Alexander Fay[2][f]

[1]*Institute of Automation Technology, Helmut Schmidt University, Holstenhofweg 85, Hamburg, Germany*

[2]*Chair of Automation, Ruhr University Bochum, Universitätsstraße 150, Bochum, Germany*

Keywords: Process Parameter Interdependencies, Data Elements, Semantic Web, Ontologies, Web Ontology Language, OWL, Industry 4.0.

Abstract: Formal representations of parameter interdependencies are critical for enabling model-based analysis and reasoning in manufacturing process knowledge graphs. While ontologies based on industrial standards allow for structured semantic descriptions, the computability of embedded mathematical expressions remains a challenging task. This paper presents a SPARQL-driven evaluation framework capable of interpreting and resolving process parameter interdependencies within a knowledge graph. The approach supports an evaluation of nested mathematical expressions, contextual data resolution and computation of process relevant results. The implementation demonstrates how semantic process models can be used for decision support and process optimization tasks.

1 INTRODUCTION

Manufacturing systems are increasingly shaped by demands for individualization, shortened product life cycles, and reduced lot sizes (Jarvenpää et al., 2016). To cope with these dynamics, modern production environments must be designed for flexibility and reconfigurability, allowing them to respond to changing technical and organizational requirements (Afazov, 2013). At the same time, the resulting production processes and system architectures become more complex, both structurally and functionally (Lüder and Schmidt, 2017).

In this context, *Digital Twins* (DTs) have become a key concept for managing complexity and enabling data-driven decision making in engineering. They represent assets by digitally capturing their properties and behavior by means of models, information, and data (Stark et al., 2017). Especially in process design and redesign tasks, an essential part of such DT models consists of quantifiable parameter interdependencies that capture the cause-effect relations governing system behavior (e.g., how cycle time depends on machine speed and part complexity). Accurately representing and evaluating these interdependencies is crucial for tasks such as process optimization, performance estimation, and feasibility assessments.

Semantic Web technologies, such as *Resource Description Framework* (RDF), offer a means to formally describe such interdependencies in a machine-interpretable way in order to test different parameter, material, and resource combinations (Gill et al., 2022).

However, while the *Semantic Web* provides powerful mechanisms for describing and querying structural information, support for quantitative reasoning and mathematical calculations remains limited (Sabou et al., 2020). This becomes a limiting factor in engineering contexts where numeric computations are essential, for example in process optimization, performance estimation, or the evaluation of physical constraints (Hildebrandt et al., 2017).

Embedding mathematical expressions in the knowledge graph keeps formulas bound to their process context and preserves their machine-interpretable semantics. This enables automated evaluation of parameter interdependencies and ad-

[a] https://orcid.org/0009-0007-0360-4108
[b] https://orcid.org/0000-0002-7228-8387
[c] https://orcid.org/0000-0001-8251-837X
[d] https://orcid.org/0009-0001-2079-8967
[e] https://orcid.org/0000-0002-8383-5323
[f] https://orcid.org/0000-0002-1922-654X

Jeleniewski, T., Köcher, A., Nabizada, H., Reif, J., Gehlhoff, F. and Fay, A.
Evaluating Process Parameter Interdependencies Based on Knowledge Graphs in Manufacturing.
DOI: 10.5220/0013937100004000

In *Proceedings of the 17th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2025) - Volume 2: KEOD and KMIS*, pages 86-96
ISBN: 978-989-758-769-6; ISSN: 2184-3228

vanced reasoning such as context-aware retrieval and consistency checks.

Efforts such as *OpenMath-RDF* (Wenzel, 2021) have extended the modeling capabilities of RDF to include mathematical constructs, providing the foundation for representing formulas and interdependencies within knowledge graphs.

Our previous work has demonstrated how process knowledge and parameter interdependencies can be semantically modeled using *OpenMath* expressions, linked to process ontologies according VDI/VDE 3682 and data elements according to DIN EN 61360 (Jeleniewski et al., 2023a). These expressions capture domain-specific dependencies, physical laws, and engineering logic that influence the behavior of production systems.

This paper presents a semantic evaluation framework that enables the operational use of interdependency descriptions within manufacturing knowledge graphs. The approach uses *SPARQL Protocol and RDF Query Language* (SPARQL) queries to extract and interpret formalized mathematical expressions in *OpenMath-RDF* format as well as relevant parameter data embedded in the graph. This semantic evaluation prepares the information required for numeric computation by traversing the symbolic structure, resolving variable bindings. The resulting expression is then passed to an external symbolic math engine for final calculation. By doing so, it enables the application of interdependency knowledge to support automated calculations in engineering workflows.

The remainder of this paper is structured as follows. Section 2 provides an overview of existing research on modeling and evaluating parameter interdependencies. Section 3 provides an explanation of the semantic integration of process and parameter descriptions together with interdependencies as mathematical expressions, which is the basis for the evaluation framework. Section 4 presents the implementation of the SPARQL-driven evaluation framework for knowledge-based computation of process characteristics together with an industrially motivated scenario. Finally, Section 5 concludes the paper and outlines future research directions.

2 RELATED WORK

Early works address parameter interdependencies primarily in the context of simulation-supported process planning and optimization. (Denkena et al., 2011) and (Denkena et al., 2012) present methods for dimensioning and optimizing process chains based on mathematical models. These contributions demonstrate the potential of simulation models to capture process behavior and support decision-making. However, the models lack semantic annotations that would enable reuse, interoperability, or formal reasoning.

A more generalized modeling strategy is proposed in (Grigoriev et al., 2013), who identifies and formalizes key process parameters within the aerospace domain and develops a configurable process model calibrated with statistical and empirical data. While the model supports quantitative analysis, it does not integrate standardized semantics.

Other contributions focus on qualitative or descriptive representations of dependencies. (Hoang et al., 2017) introduce a framework for modeling the mutual influence between product, process, and resource parameters to support system adaptation in mechatronic contexts. The method allows engineers to trace cause-effect relations and deduce necessary compensations in process or resource configurations. However, the approach does not allow for quantitative interdependency modeling and does not support automated evaluation or inference.

(Albers et al., 2019) take a design-centric perspective by linking product design features to manufacturing process impacts via a multidimensional matrix representation. This provides engineers with insights into systemic effects of design changes, but again lacks formal semantics or machine-interpretable logic.

To overcome the described limitations, semantic approaches based on ontologies have been widely explored. Ontologies provide a mechanism to formally describe domain knowledge in a reusable, extensible, and interoperable way.

(Liang, 2018) developed an ontology for process planning in additive layer manufacturing, which structures domain-specific knowledge around manufacturing steps, machine setups, and material usage. (Cheng et al., 2016) combine multiple ontologies to model devices, products, and parameters for production lines in an Industry 4.0 context. Both works demonstrate ontology-based modeling capabilities but remain isolated in scope and do not follow industrial standardized terminologies. Consequently, their reuse potential across domains or systems is limited and interdependencies between parameters are not modeled explicitly.

A more structured approach is presented by (Hildebrandt et al., 2020), who outline a methodology for developing industrial ontologies in collaboration with domain experts using standardized terminologies. The method emphasizes the use of *Ontology Design Patterns* (ODPs) aligned with established standards such as VDI/VDE 3682 for

process modeling and DIN EN 61360 for parameter description. This ensures syntactic and semantic interoperability and lays the foundation for linked knowledge representations.

ODPs based on industrial standards have since been applied in various domains. For example, (Köcher et al., 2020) employ them to model capabilities and skills of *Cyber-Physical System* (CPS). (Gill and Fay, 2023) apply an alignment ontology based on widely accepted standards to describe aircraft maintenance processes, showing the generalizability and applicability of the usage of ODPs. An important aspect of modeling interdependencies is the formal representation of mathematical expressions in a way that is compatible with *Semantic Web* technologies. In this way, the formula and its elements can be semantically annotated with respect to their role in the process. However, this is not addressed by the publications mentioned so far.

Rule-based approaches such as *Semantic Web Rule Language* (SWRL) allow rule-based reasoning, but their mathematical expressiveness is restricted to basic arithmetic. Extensions like the combination of SWRL with OpenMath improve coverage (Sánchez-Macián et al., 2007). However, built-in predicates in triple stores cause vendor lock-in and restrict portability.

A frequently applied alternative is to use RDF as a descriptive data layer and implement all mathematical logic in external applications, such as simulation tools or dedicated software frameworks (Sabou et al., 2020). While this option is efficient in terms of computation, it eliminates the semantic binding between formulas, variables, and process parameter descriptions. As a result, interdependencies lose their explicit machine-interpretable meaning, preventing advanced use cases such as consistency verification.

(Marchiori, 2003) was among the first to propose embedding mathematical semantics into the *Semantic Web*, highlighting metadata linkage and queryability as major advantages. (Lange, 2013) provides a comprehensive comparison of *MathML* and *OpenMath* for expressing mathematical knowledge, with a focus on their integrability into RDF-based systems.

MathML supports the layout and structure of mathematical formulas, while *OpenMath* centers on semantics by referencing operators through standardized *Content Dictionaries* (CDs)[1]. (Davenport and Kohlhase, 2009) argues for the harmonization of mathematical ontologies and demonstrates the advantages of *OpenMath* in semantic contexts, particularly due to its extensible design.

The *OpenMath-RDF* vocabulary introduced by (Wenzel, 2021) builds on these principles and allows mathematical expressions to be represented as RDF graphs. It supports nesting, operator typing, and the integration of existing *OpenMath* CDs. Therefore, (Wenzel, 2021) provides essential infrastructure for embedding mathematical knowledge into semantic models.

Building on this foundation, the authors of this paper have developed an infrastructure that connects knowledge modeling according to industrial standards with quantifiable interdependencies expressed as mathematical formulas (Jeleniewski et al., 2023a). Recent work has shown that embedding formulas in the ontology supports systematic consistency checks across process models (Jeleniewski et al., 2025). However, evaluating such interdependencies within complex graph structures and preparing them for computation remains a challenging task, which this paper aims to address.

3 SEMANTIC MODEL FOR PROCESS PARAMETER INTERDEPENDENCIES

To formally represent interdependencies between process parameters in a reusable and machine-interpretable manner, we introduced the *ParX* ontology[2] in (Jeleniewski et al., 2023a). It is implemented in the *Web Ontology Language* (OWL) and aligns structural process knowledge, technical parameter descriptions, and mathematical semantics by integrating multiple domain-specific ODPs into a semantically coherent *alignment ontology*.

3.1 Ontology Structure and Conceptual Foundation

The *ParX* ontology follows a modular design. Each module reflects a domain-specific conceptualization and is implemented as an individual TBox ontology. The alignment ontology imports the respective vocabularies and defines connecting axioms to combine them into a unified ABox-level model.

The following ODPs based on industrial standards are integrated:

- **VDI/VDE 3682**[3] *Formalized Process Description* (FPD) for modeling manufacturing processes, process operators, and state-based input/output semantics (VDI/VDE 3682:1, 2015),

[1] https://openmath.org/cdnames/

[2] https://github.com/hsu-aut/parX

[3] https://github.com/hsu-aut/IndustrialStandard-ODP-VDI3682

- **DIN EN 61360**[4] for the semantic description of technical parameters including types, instances, and associated units of measurement (DIN EN 61360-1, 2018),
- **UNECE**[5] unit of measurement ontology for an additional unit specification of data elements (UNECE, 2010),
- **VDI/VDE 2206**[6] for a structural description of technical systems and components (VDI/VDE 2206, 2021),
- ***OpenMath-RDF***[7] for encoding mathematical expressions in RDF, including operators, variables, and functional application structures according to (Wenzel, 2021).

Each ODP corresponds to a self-contained and reusable vocabulary rooted in industrial standards. The *ParX* alignment ontology acts as a semantic bridge between them, following best practices for ontology integration (Hildebrandt et al., 2020; Gangemi and Presutti, 2009).

3.2 Process and Parameter Modeling

Process descriptions are formalized by using the class `VDI3682:ProcessOperator`, which is responsible for transforming entities of the super type `VDI3682:State` from a prior state to a subsequent state. The classes `VDI3682:Product`, `VDI3682:Information`, and `VDI3682:Energy` are sub types of `VDI3682:State` and are used to specify the state-describing element. A process operator is assigned to a corresponding resource via `VDI3682:isAssignedTo`, pointing to an instance of `VDI3682:TechnicalResource`, which is responsible to perform the process step.

To model parameter semantics, the class `DINEN61360:DataElement` is used. Each data element is linked to:

- a `DINEN61360:TypeDescription` that provides semantic annotations, including the expected `UNECE:Unit`,
- and a `DINEN61360:InstanceDescription`, which holds, for example, actual values.

Units are modeled as individuals of the `UNECE:Unit` vocabulary, ensuring syntactic validity and semantic comparability of data elements across the graph.

[4] https://github.com/hsu-aut/IndustrialStandard-ODP-DINEN61360

[5] https://github.com/hsu-aut/IndustrialStandard-ODP-UNECE-UoM

[6] https://github.com/hsu-aut/IndustrialStandard-ODP-VDI2206

[7] https://openmath.org/om-rdf/

3.3 Representation of Interdependencies

Interdependencies between process parameters are expressed as mathematical relations using the *OpenMath-RDF* vocabulary (Wenzel, 2021). Each formula is modeled as an instance of `OM:Application`, which references an `OM:Operator` (e.g., `arith1:divide`) and a list of arguments modeled as RDF collections (`rdf:List`). The ordered nature of these list structures, using `rdf:first` and `rdf:rest` predicates, ensures deterministic traversal of mathematical arguments, maintaining the correct sequence for non-commutative operations. Arguments can be constants, variables or sub-applications, which in turn are expressed as `OM:Application`.

Each variable is an instance of `OM:Variable` and is semantically linked to a `DINEN61360:DataElement` via the object property `ParX:isDataFor`. This creates a clear and machine-interpretable binding between symbolic mathematical constructs and domain-specific process data.

The *ParX* alignment ontology defines the semantic relations that integrate process structure, parameter data, and mathematical logic across the involved vocabularies. It introduces dedicated object properties to enable these cross-domain associations. A class diagram of this ontology is illustrated in Figure 1.

The property `ParX:isDataFor` associates mathematical variables with corresponding parameter instances, represented as `DINEN61360:DataElement` individuals. To associate a mathematical expression with a specific process step, `ParX:hasInterdependency` is used to assign an `OM:Application` instance to a `VDI3682:ProcessOperator`. Furthermore, `ParX:expectsUnit` declares the expected unit for each `OM:Variable`, enabling unit consistency verification during evaluation and integration.

3.4 Decomposition and Composition of Interdependencies

In complex manufacturing systems, it is often not feasible to specify a comprehensive interdependency formula for an entire process operator, especially when process knowledge is distributed across multiple engineering domains or abstraction levels. For example, a manufacturing process may span multiple departments, with specialized teams using entirely different technologies. As each process technology relies on distinct domain expertise, formalizing interdependencies across the entire process chain as a single expression becomes increasingly challenging.

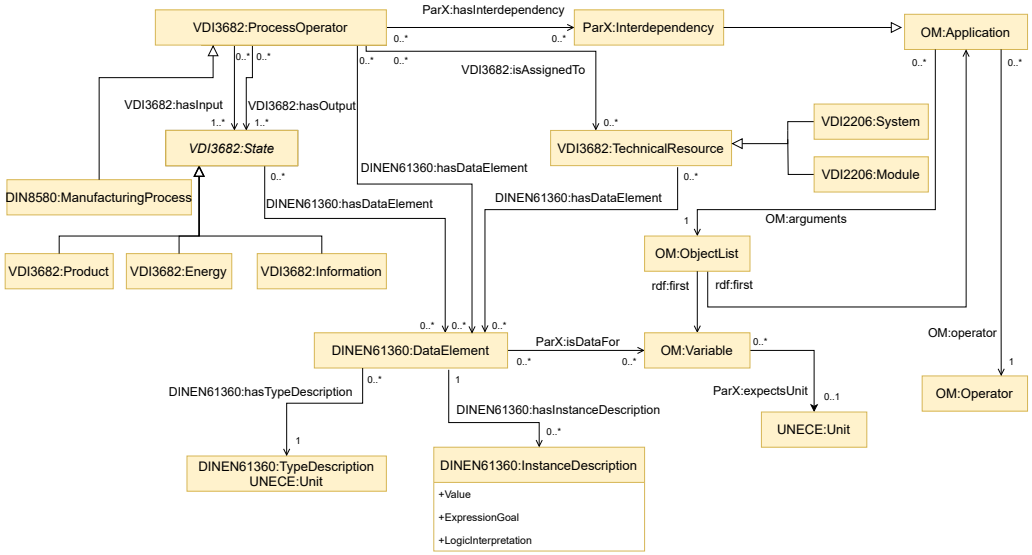

Figure 1: Class diagram of the *ParX* alignment ontology illustrating the integration of VDI/VDE 3682, DIN EN 61360 and *OpenMath-RDF* ontology design patterns (based on (Jeleniewski et al., 2023a)).

To address this, the *ParX* ontology supports the decomposition of high-level process operators into finer-grained subprocesses, for which interdependencies can be formalized. Conversely, it allows the composition of these local expressions into a global interdependency that approximates or represents the behavior of the parent process operator.

Interdependency Decomposition: If no interdependency can be defined at a higher level of abstraction, the process may be decomposed into subordinate subprocesses. Following the concept of VDI/VDE 3682, each of these subprocesses is again modeled as a `VDI3682:ProcessOperator`, allowing interdependencies to be specified at a more detailed level. Each subprocess represents a more specific operation or transformation step and may be linked to its own interdependency expression using `ParX:hasInterdependency`. On the level of smaller functional units, parameter interdependencies are typically easier to define and manage due to their reduced complexity and well contained scope.

Interdependency Composition: If all subprocesses that contribute to a higher-level process are described with formalized interdependencies, it becomes possible to derive an overarching interdependency expression by composing the mathematical content of the subordinate expressions. This is achieved by symbolically substituting the outputs of one subprocess into the inputs of the next.

Through recursive resolution and replacement, nested formulas can be merged into an evaluable expression that characterizes the input-output behavior of the entire composed process.

This approach allows system architects and engineers to work with manageable and validated fragments of process knowledge while enabling automated reasoning and evaluation at higher abstraction levels. It supports both top-down modeling (starting from an abstract specification and refining into subprocesses) and bottom-up synthesis (aggregating detailed subprocess knowledge into a global view).

As a result, the decomposition and composition mechanisms offer a method to represent and evaluate parameter interdependencies across process description hierarchies.

3.5 Integration and Instantiation of Interdependencies

The integration of interdependencies into the ontology follows a structured method introduced in (Jeleniewski et al., 2023b), which enables the systematic formalization of informal process knowledge and its instantiation within the semantic model.

Two main types of input are required for this integration:

- A process model structured according to VDI/VDE 3682, describing process operators, their associated inputs, outputs, and assigned technical resources. Such models can be created using dedicated modeling tools and transformed

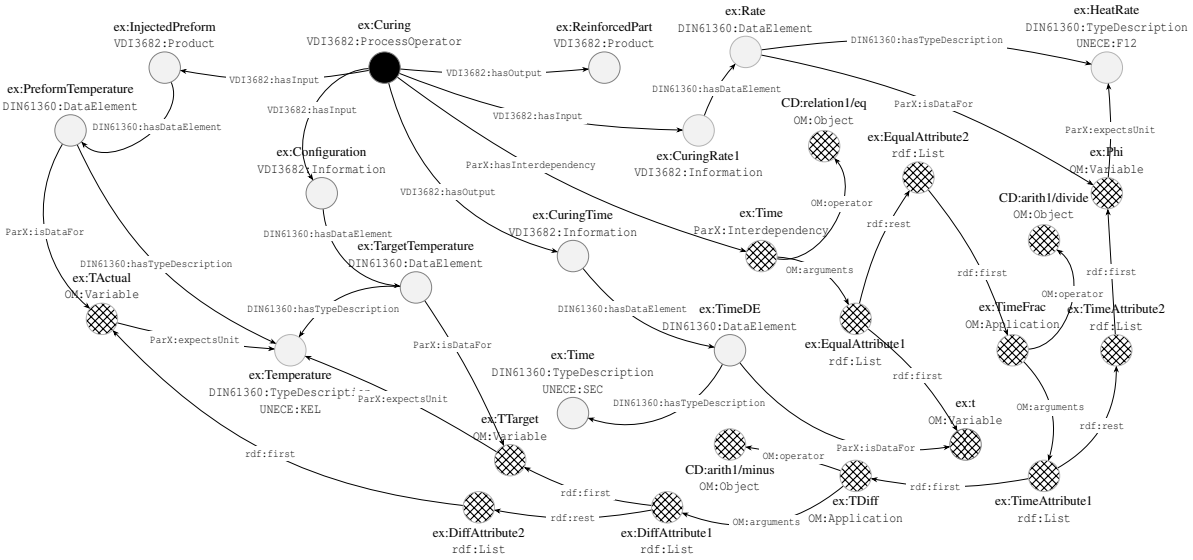

Figure 2: Simplified excerpt of a *curing* process step in the ontology, focusing on interdependencies, data element bindings, and unit semantics.

into OWL, e.g. using the *fpb-owl-mapper* [8], which converts JSON-based FPD models generated via *FPB.js* [9] (Nabizada et al., 2020) into RDF/OWL representations conforming to the *VDI/VDE 3682* ODP.

- A mathematical expression describing the interdependency between input and output parameters, which must be represented in a form that is compatible with the semantic process and parameter description model. To ensure this level of integration, the expression must be encoded using the *OpenMath-RDF* format. Since manually creating *OpenMath-RDF* structures is technically demanding, a dedicated parsing tool is provided: the *openmath-rdf-parser* [10]. It allows users to enter formulas as human-readable strings (e.g., $F = m \cdot a$) and automatically transforms them into *OpenMath*-compliant RDF graphs.

Starting with the identification of the relevant `VDI3682:ProcessOperator`, the process model is analyzed to collect associated input and output parameters, as well as any assigned technical resources. If an interdependency between these parameters is known, the corresponding mathematical expression can be integrated in the ontology as *OpenMath-RDF*. The involved variables are then aligned to data elements according to the concept described in Section 3.3.

If no suitable interdependency is available at the given abstraction level, the process can be decomposed into subprocesses where interdependencies are known (cf. Section 3.4). Through application of the integration procedure, these subprocesses are progressively enriched with interdependency knowledge.

This structured approach enables the construction of semantically enriched process knowledge graphs and is explained in more detail in (Jeleniewski et al., 2023b).

An exemplary graph is shown in Figure 2. The figure illustrates a process description that includes both process parameters and an integrated representation of interdependencies used to calculate the curing time of a curing process step (highlighted in black) as part of the *Resin Transfer Molding* (RTM) process. During curing, the already injected preform is transformed into a reinforced product.

To enable this, input information such as the cooling rate and target temperature, as well as the current temperature of the injected preform, is provided.

The elements of the integrated formula (see (1)), which represent the interdependencies between curing time and process parameters (required temperature T_{target}, the actual temperature T_{actual}, and the curing rate ϕ_{cure}) are highlighted by crosshatched nodes in the graph.

$$t_{\text{cure}} = \frac{T_{\text{target}} - T_{\text{actual}}}{\phi_{\text{cure}}} \tag{1}$$

[8] https://github.com/hsu-aut/fpb-owl-mapper

[9] https://fpbjs.net

[10] https://github.com/aljoshakoecher/openmath-rdf-parser

4 INTERDEPENDENCY EVALUATOR

Understanding and applying parameter interdependencies in a reliable and reusable manner requires not only formalized representations of these interdependencies, but also mechanisms to evaluate them in specific process contexts. For this purpose, we propose an evaluation framework that computes output values from process models by interpreting mathematical expressions embedded in the process knowledge graph. The evaluator is available on *GitHub*[11].

The approach enables reusable, fully semantic-based evaluation without requiring external scripts for formula implementations.

A key design decision is the complete reliance on SPARQL queries for traversing and resolving the mathematical expressions encoded in *OpenMath-RDF*. By leveraging the queryable graph structure, the evaluator avoids manual evaluation routines (e.g., identifying applicable formulas, collecting relevant parameter or data values and manual execution of calculations). Instead, the logical structure of the *OpenMath-RDF* model itself guides the evaluation and finally, the computation of interdependency descriptions.

4.1 Workflow and Implementation

The evaluation framework enables the evaluation of formally described parameter interdependencies by computing mathematical expressions embedded in the process knowledge graph. It is implemented as a *Node.js* application and interacts with a SPARQL-compliant triple store as graph database, where semantic process models and parameter data are hosted.

The evaluation process follows a structured multi-stage workflow, which is guided by SPARQL queries. Each stage incrementally resolves the symbolic expressions and integrates instance data from the graph to obtain a final numerical result. In Figure 3, the workflow performed by the framework is illustrated.

Formula Retrieval: The evaluation workflow begins by identifying the mathematical expression that defines a specific output parameter for a given `VDI3682:ProcessOperator`. This is achieved through a SPARQL query that selects expressions associated via the `ParX:hasInterdependency` relation and additionally checks whether the expression represents an equality, indicated by the `om:operator` being `CD:relation1/eq` (see Listing 1).

[11] https://github.com/jelenito/ParX-evaluator

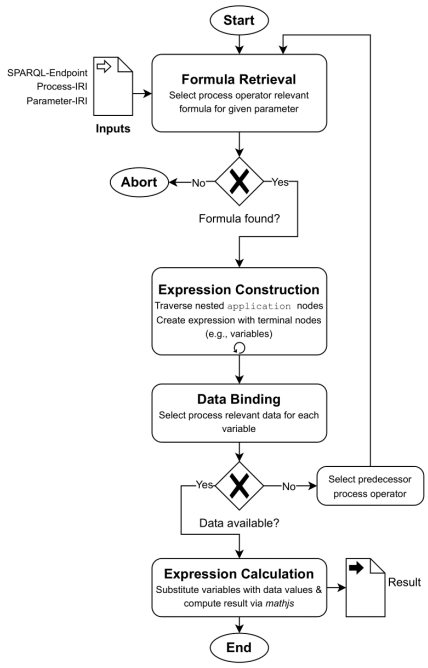

Figure 3: Workflow performed by the evaluation framework.

The query then navigates to the left-hand side of the equality (first element of the `rdf:List` of arguments) and verifies that it is a variable assigned to the target parameter (`DINEN61360:DataElement`) using the `ParX:isDataFor` property. This ensures that only those expressions are retrieved in which the specified parameter is explicitly defined as the variable being solved for.

Listing 1: SPARQL-based formula retrieval.

```
SELECT ?formula WHERE {
  <${processUri}>
  parx:hasInterdependency ?formula .
  ?formula om:arguments ?argList;
        om:operator CD:relation1/eq .
  ?argList rdf:first ?LHS .
  ?LHS a om:Variable .
  ?de parx:isDataFor ?LHS .
  FILTER(str(?de) = "${dataElementIri}"
      )}
```

Expression Construction: Once the corresponding formula has been retrieved, the evaluator constructs a symbolic representation of the expression by recursively traversing `OM:Application` nodes via SPARQL queries. Each application node specifies a mathematical operator via a reference to a *OpenMath* CD and a set of arguments encoded as RDF lists. The evaluator follows these references and expands the expression tree by recursively resolving each ar-

gument. This process continues until all branches of the expression reduce to terminal nodes (such as `OM:Variable`), which represent the leaves of the tree. The resulting symbolic tree accurately reflects the hierarchical structure of the mathematical expression and forms the basis for subsequent variable substitution.

Data Binding: Each `OM:Variable` in the expression is assigned to a `DINEN61360:DataElement` via `ParX:isDataFor`. The evaluator queries the associated `DINEN61360:InstanceDescription` to retrieve available numerical values. These are substituted into the expression tree, forming the basis for further symbolic evaluation.

Recursive Fallback Resolution. A particular challenge addressed by the framework is the recursive resolution of missing data values. If no directly assigned data element provides a value for a required variable, the evaluator identifies preceding formulas that compute the corresponding data element as an output. Evaluation is then recursively triggered for these upstream processes, allowing multi-step dependencies to be resolved without external orchestration logic.

Expression Calculation: The evaluator distinguishes between semantic graph traversal and symbolic computation. While SPARQL provides basic query and arithmetic functionality, it lacks mechanisms for symbolic evaluation, dynamic recursion, or expression tree manipulation (Graux et al., 2020).

Hence, only graph access and structural analysis are handled via SPARQL, whereas mathematical processing is delegated to *mathjs*. *mathjs*[12] is a well documented open-source mathematics library for *JavaScript* that supports symbolic parsing and numeric computation of mathematical expressions.

Once all variables are resolved, the symbolic expression is converted into the required *mathjs* format and passed to the *mathjs* engine for final evaluation. *mathjs* engine parses and computes the result based on the defined mathematical operators and numeric inputs.

The framework can be integrated as a component into broader engineering environments or knowledge-based systems. Its architecture provides a maintainable and extensible foundation for future capabilities such as constraint checking, explanation facilities, or domain-specific evaluation logic.

[12] https://mathjs.org/

4.2 Application Example: RTM Process

In cooperation with an industrial partner from the field of composite manufacturing for aerospace components, a simplified example of an RTM process has been modeled. The scenario represents a condensed and illustrative segment of a process chain (see Figure 5), used to demonstrate the capabilities of the proposed evaluation framework. To ensure clarity and focus, only a selected subset of interdependencies and process operators is considered in this example.

RTM is a widely used closed-mold process for manufacturing fiber-reinforced plastic components. In this process, dry fiber preforms are placed into a mold cavity, and a thermoset resin is injected to infiltrate the fibers. Once the mold is filled, curing is initiated to solidify the composite structure. Process performance depends on geometric, material, and injection parameters, which exhibit various interdependencies that can be formalized and evaluated.

The selected example comprises three consecutive `VDI3682:ProcessOperator` instances, each connected via data elements and mathematical expressions modeled using *OpenMath-RDF*. For improved readability, the full formulas, which are described in this section, are omitted in Figure 5 and are instead indicated after the instances of `ParX:Interdependency` (crosshatched nodes). An exemplary connection of variables within a formula including assigned data elements can be seen in Figure 2.

Tool Setup. The first process operator determines the required resin volume based on geometric properties of the component and the tool. The `VDI3682:TechnicalResource` *RTM-Tool* provides the mold cavity area A_{cav}, and the desired laminate thickness h_{lam} is supplied as a `VDI3682:Information` input. The output is the configured Tool annotated with the filling volume V. The process operator is aligned to an interdependency described by (2).

$$V = A_{\text{cav}} \cdot h_{\text{lam}} \tag{2}$$

Resin Preparation. The second process operator describes the resin preparation. The aligned interdependency description describes the resin shot mass using the computed volume and the material density ρ_{res} of the selected resin. The output is a prepared resin shot with the mass computed by the interdependency described by (3).

$$m_{\text{res}} = \rho_{\text{res}} \cdot V \tag{3}$$

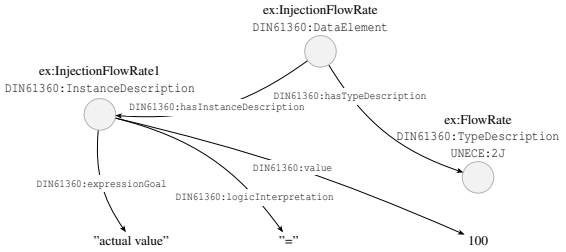

Figure 4: Examplary data element description for data binding.

Resin Injection. The third process operator represents the injection of the resin into the mold. It takes the resin shot (with mass m_{res}) and the injection flow rate $\dot{V}_{\text{inj}}$ as input. The injection time t_{inj} can be computed with the aligned interdependency description (see (4)).

$$t_{\text{inj}} = \frac{m_{\text{res}}}{\rho_{\text{res}} \cdot \dot{V}_{\text{inj}}} \tag{4}$$

This example demonstrates a typical use case during early-stage process planning, where a user queries the system for the required injection time t for a new composite component. The evaluator begins by identifying the mathematical expression linked to the injection operator and attempts to resolve all input variables using the knowledge graph.

In this exemplary use case (`ex:ResinInjection`), the flow rate $\dot{V}_{\text{inj}}$ and resin density ρ_{res} are directly available via linked `DINEN61360:DataElement` instances, but the resin shot mass m_{res} is not. An exemplary data element assignment for $\dot{V}_{\text{inj}} = 100\text{cm}^3$ can be seen in Figure 4. The unit cm^3 is assigned via the classification as `UNECE:2J`.

To resolve this missing value, the evaluator recursively identifies the preceding `ex:ResinPreparation` operator, retrieves its associated expression, and checks whether its inputs are available via SPARQL querying.

Again, the resin volume V required to calculate m_{res} is not directly given. Thus, the evaluator continues to the `ex:ToolSetup` operator and resolves the volume expression using the cavity area A_{cav} and laminate thickness h_{lam}, both of which are connected to data elements within the technical resource and input information.

Once V is calculated, it is passed upstream to compute m_{res}, which in turn enables the evaluation of t_{inj}. Through this recursive traversal, the evaluator derives the requested output value solely from the available data for A_{cav}, h_{lam}, ρ_{res}, and $\dot{V}_{\text{inj}}$.

This illustrates how the proposed framework enables dynamic resolution and stepwise evaluation of parameter interdependencies within process models, supporting transparency, reusability, and automated decision-making.

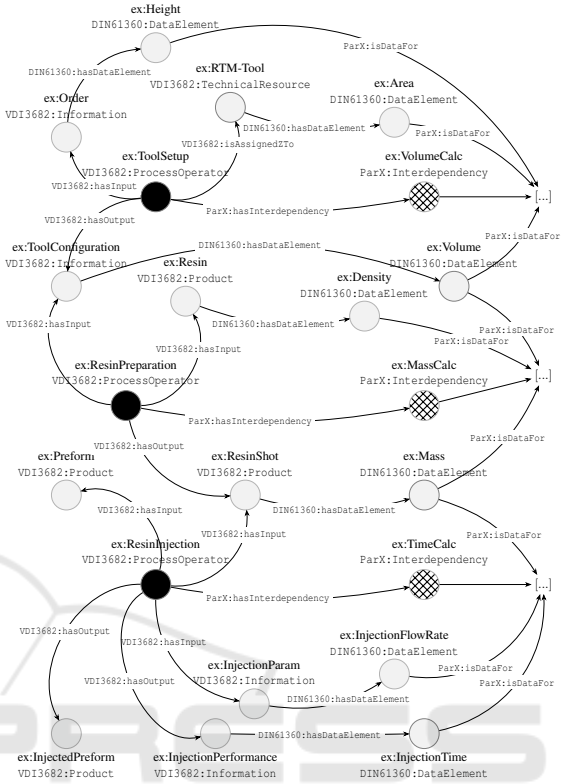

Figure 5: Exemplary excerpt of parameter interdependencies across three process operators (black) in an RTM process.

5 CONCLUSION AND FUTURE WORK

This paper presents a semantic modeling and evaluation framework for representing process parameter interdependencies in manufacturing systems based on knowledge graphs. The *ParX* ontology integrates structural process descriptions, parameter semantics, and formalized mathematical expressions using ODPs derived from industrial standards such as VDI/VDE 3682, DIN EN 61360, and *OpenMath-RDF*. This alignment enables a machine-interpretable, reusable representation of process knowledge, capable of capturing domain-specific dependencies in a standardized form.

Building upon this model, a SPARQL-driven evaluation framework has been introduced that allows mathematical interdependencies to be resolved and computed directly based on the ontology. The approach separates semantic traversal and symbolic

computation by using SPARQL queries for graph navigation and delegating the mathematical evaluation to an external computation engine. This architecture allows for flexible evaluation workflows and supports recursive resolution of upstream dependencies along process chains.

The framework enables automated parameter evaluation, can support design-time decision support, and offers a mechanism to evaluate interdependencies across multiple levels of process abstraction. The proposed approach demonstrates how semantic technologies can be extended beyond structural modeling to actively support quantitative reasoning in manufacturing engineering contexts.

Furthermore, the current evaluation framework focuses primarily on value calculation. Restriction checking against predefined constraints during runtime evaluation has not yet been fully integrated.

As part of future work, the evaluation framework will be extended to incorporate constraint-based validation mechanisms, allowing computed outputs including intermediate results to be automatically verified against parameter restrictions. This extension will strengthen the integration between interdependency evaluation and restriction checking, and support advanced reasoning capabilities for knowledge-based process analysis and optimization.

Although the proposed approach has shown no observable response delays in the presented scenarios, it must be acknowledged that performance may decrease when the number of triples and the complexity of concatenated formulas increase. As SPARQL query execution time is inherently dependent on graph size, the scalability remains a limitation and subject of future work.

To further increase accessibility and practical adoption, it is also planned to encapsulate the evaluation logic into a dedicated software package (e.g., an `npm` module), enabling its seamless reuse and integration into engineering environments and web-based tools.

ACKNOWLEDGEMENTS

This contribution originates from the projects *LaiLa* and *iMOD*, funded by *dtec.bw – Digitalization and Technology Research Center of the Bundeswehr* which we gratefully acknowledge. *dtec.bw* is funded by the *European Union – NextGenerationEU*.

REFERENCES

Afazov, S. M. (2013). Modelling and simulation of manufacturing process chains. *CIRP Journal of Manufacturing Science and Technology*, 6(1):70–77.

Albers, A., Stürmlinger, T., Mandel, C., Wang, J., de Frutos, M. B., and Behrendt, M. (2019). Identification of potentials in the context of Design for Industry 4.0 and modelling of interdependencies between product and production processes. *Procedia CIRP*, 84:100–105.

Cheng, H., Zeng, P., Xue, L., Shi, Z., Wang, P., and Yu, H. (2016). Manufacturing Ontology Development Based on Industry 4.0 Demonstration Production Line. In *2016 Third International Conference on Trustworthy Systems and their Applications (TSA)*, pages 42–47. IEEE.

Davenport, J. H. and Kohlhase, M. (2009). Unifying Math Ontologies: A Tale of Two Standards. In Carette, J., Dixon, L., Coen, C. S., and Watt, S. M., editors, *Intelligent Computer Mathematics*, volume 5625 of *Lecture Notes in Computer Science*, pages 263–278. Springer Berlin Heidelberg, Berlin, Heidelberg.

Denkena, B., Behrens, B.-A., Charlin, F., and Dannenberg, M. (2012). Integrative process chain optimization using a genetic algorithm. *Production Engineering*, 6(1):29–37.

Denkena, B., Henjes, J., and Henning, H. (2011). Simulation-based dimensioning of manufacturing process chains. *CIRP Journal of Manufacturing Science and Technology*, 4(1):9–14.

DIN EN 61360-1 (2018). Standard data element types with associated classification scheme - Part 1: Definitions - Principles and methods (IEC 61360-1:2017).

Gangemi, A. and Presutti, V. (2009). Ontology Design Patterns. In Staab, S. and Studer, R., editors, *Handbook on Ontologies*, pages 221–243. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gill, M. S. and Fay, A. (2023). Utilisation of semantic technologies for the realisation of data-driven process improvements in the maintenance, repair and overhaul of aircraft components. *CEAS Aeronautical Journal*.

Gill, M. S., Reif, J., Jeleniewski, T., Weigand, M., and Fay, A. (2022). Application potentials of semantic technologies for digital twins in aircraft design, manufacturing and maintenance. In *dtec.bw-Beiträge der Helmut-Schmidt-Universität / Universität der Bundeswehr Hamburg: Forschungsaktivitäten im Zentrum für Digitalisierungs- und Technologieforschung der Bundeswehr dtec.bw - Band 1*. Universitätsbibliothek der HSU/UniBw H.

Graux, D., Sejdiu, G., Stadler, C., Napolitano, G., and Lehmann, J. (2020). MINDS: A Translator to Embed Mathematical Expressions Inside SPARQL Queries. In Blomqvist, E., Groth, P., de Boer, V., Pellegrini, T., Alam, M., Käfer, T., Kieseberg, P., Kirrane, S., Meroño-Peñuela, A., and Pandit, H. J., editors, *Semantic Systems. In the Era of Knowledge Graphs*, volume 12378 of *Lecture Notes in Computer Science*, pages 104–117. Springer International Publishing, Cham.

Grigoriev, S. N., Kutin, A. A., and Turkin, M. V. (2013). Modelling Complex Production Processes in Aerospace Industry based on Dimensional Analysis. *Procedia CIRP*, 7:473–478.

Hildebrandt, C., Glawe, M., Müller, A. W., and Fay, A. (2017). Reasoning on Engineering Knowledge: Applications and Desired Features. In Blomqvist, E., Maynard, D., Gangemi, A., Hoekstra, R., Hitzler, P., and Hartig, O., editors, *The Semantic Web*, volume 10250 of *Lecture Notes in Computer Science*, pages 65–78. Springer International Publishing, Cham.

Hildebrandt, C., Köcher, A., Küstner, C., Lopez-Enriquez, C.-M., Müller, A. W., Caesar, B., Gundlach, C. S., and Fay, A. (2020). Ontology Building for Cyber–Physical Systems: Application in the Manufacturing Domain. *IEEE Transactions on Automation Science and Engineering*, 17(3):1266–1282.

Hoang, X.-L., Marks, P., Weyrich, M., and Fay, A. (2017). Modeling of interdependencies between products, processes and resources to support the evolution of mechatronic systems. *IFAC-PapersOnLine*, 50(1):4348–4353.

Jarvenpää, E., Siltala, N., and Lanz, M. (2016). Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. In *2016 IEEE International Symposium on Assembly and Manufacturing (ISAM)*, pages 120–125. IEEE.

Jeleniewski, T., Nabizada, H., Reif, J., Gehlhoff, F., and Fay, A. (2025). Consistency Verification in Ontology-Based Process Models with Parameter Interdependencies. In *2025 IEEE 30th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE.

Jeleniewski, T., Nabizada, H., Reif, J., Köcher, A., and Fay, A. (2023a). A Semantic Model to Express Process Parameters and their Interdependencies in Manufacturing. In *2023 IEEE 32nd International Symposium on Industrial Electronics (ISIE)*, pages 1–6. IEEE.

Jeleniewski, T., Reif, J., and Fay, A. (2023b). Integrating Interdependencies in Semantic Manufacturing Process Description Models. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4. IEEE.

Köcher, A., Hildebrandt, C., Vieira da Silva, L. M., and Fay, A. (2020). A Formal Capability and Skill Model for Use in Plug and Produce Scenarios. In *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1663–1670. IEEE.

Lange, C. (2013). Ontologies and languages for representing mathematical knowledge on the Semantic Web. *Semantic Web*, 4(2):119–158.

Liang, J. S. (2018). An ontology-oriented knowledge methodology for process planning in additive layer manufacturing. *Robotics and Computer-Integrated Manufacturing*, 53:28–44.

Lüder, A. and Schmidt, N. (2017). Challenges of Mechatronical Engineering of Production Systems: An Automation System Engineering View. In Ghezzi, L., Hömberg, D., and Landry, C., editors, *Math for the Digital Factory*, volume 27 of *Mathematics in Industry*, pages 93–114. Springer International Publishing, Cham.

Marchiori, M. (2003). The Mathematical Semantic Web. In Asperti, A., Buchberger, B., and Davenport, J. H., editors, *Mathematical Knowledge Management*, volume 2594 of *Lecture Notes in Computer Science*, pages 216–223. Springer Berlin Heidelberg, Berlin, Heidelberg.

Nabizada, H., Köcher, A., Hildebrandt, C., and Fay, A. (2020). Offenes, webbasiertes Werkzeug zur Informationsmodellierung mit Formalisierter Prozessbeschreibung. In *Automation 2020*, pages 443–454. VDI Verlag.

Sabou, M., Biffl, S., Einfalt, A., Krammer, L., Kastner, W., and Ekaputra, F. J. (2020). Semantics for Cyber-Physical Systems: A cross-domain perspective. *Semantic Web*, 11(1):115–124.

Sánchez-Macián, A., Pastor, E., de López Vergara, J. E., and López, D. (2007). Extending SWRL to Enhance Mathematical Support. In Marchiori, M., Pan, J., and Marie, C., editors, *Web Reasoning and Rule Systems*, volume 4524 of *Lecture Notes in Computer Science*, pages 358–360. Springer Berlin Heidelberg, Berlin, Heidelberg.

Stark, R., Kind, S., and Neumeyer, S. (2017). Innovations in digital modelling for next generation manufacturing system design. *CIRP Annals*, 66(1):169–172.

UNECE (2010). Recommendation No. 20: Codes for Units of Measure Used in International Trade.

VDI/VDE 2206 (2021). Development of mechatronic and cyber-physical systems.

VDI/VDE 3682:1 (2015). Formalised Process Descriptions - Concept and Graphic Representation.

Wenzel, K. (2021). OpenMath-RDF: RDF encodings for OpenMath objects and Content Dictionaries. In *31st OpenMath Workshop*.