CONTO: An Ontology-based Approach for Interoperable Configuration Knowledge

Stefan Bischof[1,*], Andreas Falkner[1], Erwin Filtz[1], Patrik Schneider[2], Simon Steyskal[1] and Mihaela Topa[3]

[1]Siemens AG Österreich, Vienna, Austria
[2]Siemens AG, Munich, Germany
[3]Siemens SRL Romania, Bucharest, Romania

Abstract

CONTO (CONfiguration ONTOlogy and TOols) addresses the challenges of vendor lock-in and costly evaluations in product configuration systems through an ontology-based semantic framework that establishes interoperability without reinventing existing solver technologies. Our approach provides dual representations–instance-based Configuration Vocabulary and OWL DL-based formalism–with tooling to transform product models into programs for various platforms. By decoupling modelling from configuration processing, CONTO creates an abstraction layer that preserves existing investments while enabling integration with emerging AI tools. Our implementation shows practical applications supporting both commercial and open-source configurators.

1. Introduction

CPQ (Configure, Price, Quote) solutions provide the means for organizations to automate the sales process by allowing customers to choose between a manifold of product variants (configure step) and calculate/show marked dependent prices of configured products (price/quote step) to them. However, organizations face a significant challenge: selecting optimal configuration systems requires substantial resources, while the risk of vendor lock-in for configuration tools threatens long-term adaptability and cost-effectiveness. Vendor lock-in occurs due to different approaches to describe configuration knowledge, which can include rule-based, constraint satisfaction-based, and proprietary (legacy) approaches.

This paper introduces CONTO (CONfiguration ONTOlogy and TOols), our semantic framework for product configuration: an approach that establishes interoperability by uplifting domain-specific product specification to an domain-independent, exchangeable semantic model based on two possible representations. Our framework provides a comprehensive semantic model for product lines and configurations, supported by tools for the creation, maintenance, and processing of these models. Through a system of vendor-specific exporters, our framework transforms product models into the appropriate formats required by diverse configuration systems.

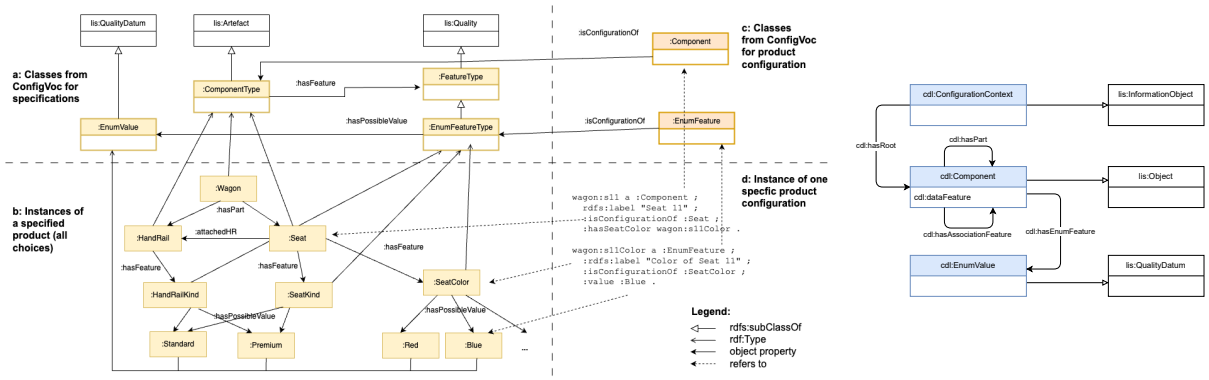
The significance of this approach lies in the creation of an abstraction layer that effectively decouples product modelling from configuration processing. This separation enables organizations to maintain flexibility in their configuration ecosystem while preserving existing technological investments. Furthermore, our framework facilitates integration with emerging AI technologies for automated model generation, with the aim of reducing manual modelling efforts.

In this paper, we present the conceptual foundations of our framework (Section 3), outline the current tooling that allows us to create a semantic model for product lines (Section 4), and discuss limitations and future work (Section 5).

2. Related Work

The need to represent product configuration models using ontologies was identified in the 1980s. Over time, the work started with process descriptions that identified the most important aspects of the product

SEMANTiCS'25: International Conference on Semantic Systems, September 3–5, 2025, Vienna, Austria
*Corresponding author.
0000-0001-9521-8907 (S. Bischof); 0000-0003-3445-0504 (E. Filtz); 0009-0003-8803-9523 (M. Topa)



(a) Example of instance-based approach (b) OWL DL-based approach

Figure 1: Ontology visualisation of the two different CONTO representations

configuration [1]. Around the turn of the millennium, first works regarding a general ontology for product configuration were presented, already defining a class hierarchy of concepts (e.g., components, attributes, constraints and their relations) using OWL [2]. Later, the idea of generic models is refined to more domain-specific configuration models and also adding additional rule languages such as Semantic Web Rule Language (SWRL) to represent constraints [3, 4]. All these developments have recently been reviewed and a new meta-model for product configuration has been proposed [5]. Industry is slowly adopting product configuration ontologies, which need to be adjusted to the specific industrial domain and meet their requirements while keeping the advantages of Semantic Web technologies, which allow for the easier integration of product data across platforms used in configuration tasks [6, 7].

3. Requirements and Proposed Framework/Ontology

The goal of CONTO is a tool-independent product model representation that includes also configured products of these models. As shown in Fig. 2, CONTO acts as the mediator between domain-specific product model representation that is often maintained in legacy tools and formats, and vendor-specific representations such as those required by state-of-the-art tools such as the Tacton configurator or open source constraint languages such as MiniZinc [8]. With CONTO, we propose an ontology-based representation of an 'universal' product model. The general elements of such a product model were introduced by Soininen et al. [2] and include (a) kind-of- as well as part-of-hierarchies of model components (including cardinalities), (b) attribute features with value ranges for the data types: string, enum, doubles, and decimal, including defaults, (c) association features, (d) feature cardinalities, (e) various types of constraints (such as logic-based, and rule-based), and (f) clear distinction between product line specifications and (real) configured products. Based on the above requirements and features, we provide an instance-based and an OWL-DL-based starting point for a product modeler. The starting points represent two different modeling paradigms that are currently not interchangeable, since in the former the constraints are defined as RDF-based tables and in the latter as OWL axioms. However, both approaches are aligned with the ISO-standard industrial data ontology (IDO), thus use the same upper ontology and classes.[1]

Figure 2: CONTO workflow

Instance-based Representation In this representation, product specifications are defined as instances (i.e., the assertions), where the ontology (i.e., the terminology) provides the basic classes/properties as guidance. The base ontology is called *Configuration Vocabulary* (ConfigVoc) and is bounded by the expressivity of RDFS. A modest extension of ConfigVoc is provided by the 'Config Lite Ontology', which adds lightweight reasoning using axioms such as inverse roles, role chains, and restrictions.

[1]Industrial Data Ontology: https://rds.posccaesar.org/ontology/lis14/ont/core/

With ConfigVoc, our aim is to capture two important 'dimensions' of product configuration, namely, the first dimension of (a) product-independent vs. (b) product-dependent configuration model, and the second dimension of (c) product line specifications vs. (d) configured products. In Fig. 1a, we show the dimensions in an example, where the classes of (a,c) are given by the ConfigVoc ontology and (b) are the instances of our wagon product line specification. Note that in (b) we define the full variant space via constraints; hence, product configuration tools will take this information as an input, whereas (d) is an example of a successful run of a configuration tool. ConfigVoc has over 90 classes and over 100 object/data properties. In the following, we describe the usage of the most important classes and properties: (a) *ConfigurationContext* is the starting point for any model and usually defines different product lines, thus introducing modularity into product lines; (b) *ComponentTypes* defines the structural elements of a product line capturing part-of hierarchies or more complex associations. The product hierarchy of ComponentTypes is unfolded by the `hasPart` property; (c) *FeatureTypes* introduce the product features (e.g., color) linked to a specific ComponentType and define the possible variant space by the associated values. Several subclasses of FeatureType allow value typing, for instance, the EnumFeatureType only allows only certain EnumValues as possible choices; (d) *Object properties* connect the above classes via domain/range and include `hasRoot` (ConfigurationContext/ComponentType), `hasFeature` (ComponentTyp/FeatureType), and `hasPossibleValue` (EnumFeatureType/EnumValue).

Constraints are the most important means to define (technical) requirements and reduce the state space induced by all possible features and values. Since constraints can come in various flavours, the most common are logic-based, formula-/rule-based, and table-based. The focus of this work is on the latter, and we represent a *TableConstraint* using *TableRow(s)* that are made of *TableCell(s)* that refer to *FeatureType(s)*. One TableConstraint resembles a spreadsheet but allows multiple cell values and negation, and can be rewritten into propositional formulas. This offers a more generic and universally translatable representation that can be seamlessly exported to various configurators without being tied to any validation framework such as SHACL[2]. The table shown in Fig. 3 is as a propositional formula: $(seatKind(Special) \wedge (seatColor(Red) \vee seatColor(Blue)) \vee \neg seatKind(Special)$.

:SeatKind	:SeatColor
:Special	:Red , :Blue
not :Special	*

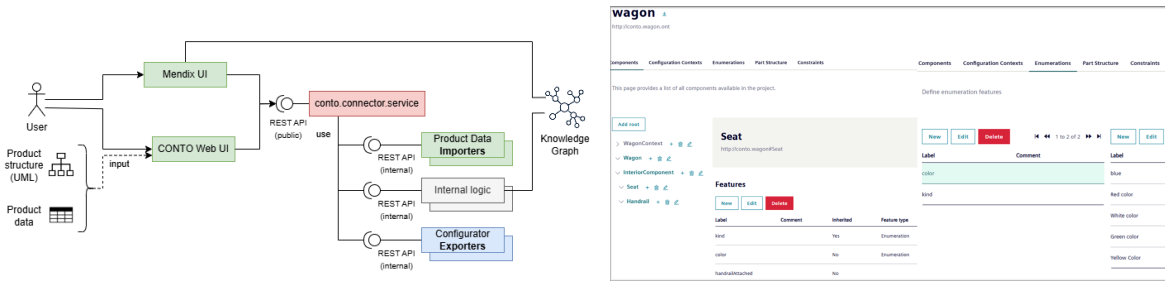
Figure 3: Example of constraint table

OWL DL-based Representation OWL 2 DL provides native representations for many features essential to product configuration. In our approach, we leverage OWL 2 DL as a second representation for both product lines and configurations, building upon McGuinness and Wright's concept of using Description Logics for product configuration [9]. In this representation, the TBox (terminological box) encapsulates the product line model, while the ABox (assertional box) represents concrete configurations.

Our semantic model maps key product configuration concepts to OWL constructs: classes (*components*), inheritance of classes (*kind-of hierarchy*), data properties (*attribute features*), custom data types (*attribute feature ranges*), object properties (*association features*), property cardinality restrictions (*feature cardinalities*), value (number) ranges, *enumerations* are represented as enumerated classes, and *enumeration values* as named individuals. *Table constraints* are expressed through class expressions (a disjunction of conjunctions), while part-of hierarchies use dedicated object properties. We address two representational challenges: *formula constraints* (beyond OWL's native capabilities and still a work-in-progress) and *default values* (represented as annotation properties to enable UI display while acknowledging OWL's monotonic semantics). Fig. 1b illustrates the ontology structure.

This OWL-based representation offers several benefits. We leverage OWL's established formalization rather than creating a new logical framework. OWL reasoners can perform consistency checking for both product lines and configurations, and satisfiability testing on the product line model. In certain scenarios, the reasoner can automatically derive appropriate values. We can apply transformations to simplify or optimize constraints, or translate them to other representations. The resulting configurations have efficient representations. The approach supports component and feature libraries through standard

[2]SHACL standard, https://www.w3.org/TR/shacl/



(a) (b)

Figure 4: (a) CONTO tools architecture and (b) user interface

OWL import mechanisms, and OWL's native versioning features facilitate version management of product models. This representation establishes a solid foundation for our product configuration framework while maintaining compatibility with existing semantic web technologies and tools.

4. Tooling

CONTO Tools facilitate the creation of ontologies for product lines utilizing ConfigVoc. These product ontologies function as primary data sources for exporters, which subsequently generate models compatible with various configurator engines. Product configuration is then performed through dedicated configuration tools or solvers. Fig. 2 shows the product line data flow.

Importers and User Interfaces for Modelling Product lines can be represented through multiple formalisms, including UML-based representations, feature models, domain-specific languages, matrix-based representations, and decision diagrams. CONTO Tools offers an API for importing product structure from an UML diagram and product variants, as well as table and formula constraints from spreadsheets. Fig. 4a depicts the architecture.

Product parts are defined as UML classes and are organized in a hierarchy (*part-of*) using UML composition. Part features are defined as class attributes, each with a type (*long*, *double*, *decimal*, *string*, *boolean* or *enum*), and may have a domain and a default value. Inheritance (*type-of*) is supported to define attributes common to multiple parts. Constraints and variants require context, so they are associated with product parts. They are defined using component attributes. Formula constraints can be defined using a basic constraints definition language, whereas table constraints and variants are represented as tables.

In addition to the CONTO tools we also create a user interface as not everybody is familiar working with UML or other diagrams evolving with the ontology. In addition, it also limits the potential pitfalls that can occur when giving users the possibility to use the full potential of diagrams. Users can create and modify product models with the functionality implemented in the UI, which is aligned with the ontology and therefore reduces the risk of introducing wrong data. A snippet of the current version of the UI is shown in Fig. 4b. It allows users to enter components, enumerations, part structure and constraints, which are directly stored in a knowledge graph. Furthermore, the UI supports the export of product configuration models into RDF as well as dedicated exporters defined in the CONTO tools.

Exporters and Configuration Engines for Configuration CONTO represents the product line independently of the business domain and any configurator vendor, so the logic of exporters is based solely on ConfigVoc. The CONTO API provides a list of all supported configurator engines. The current implementation generates models for both the commercial Tacton configurator (TCX models) and the open-source Constraint Programming Language MiniZinc [8]. However, it can be extended to other commercial tools such as Configit, Encoway, feature models like Universal Variability Language, or logic programming languages such as Answer Set Programming. It could also send product data to

CPQ systems, such as TactonCPQ, if APIs are available. Although various configurator engines are supported, not all engines support all features of CONTO. Conversely, CONTO does not support all the features of every existing configurator engine.

5. Conclusions & Future Work

CONTO provides semantic models and tools that transform product models into executable configuration programs while decoupling product modeling from configuration execution. This decoupling improves flexibility and reduces vendor lock-in risks.

We recognize that our approach has limitations. Product configuration systems often contain specialized features that don't generalize well across platforms, such as procedural attachments. We made the design decisions in developing CONTO to prioritize declarative, semantic product models with broadly applicable features, while intentionally excluding specialized capabilities that would compromise cross-platform compatibility. This deliberate scope management ensures that CONTO remains versatile across different configuration environments.

Our future work focuses on four directions: (a) extending the ontologies, (b) developing exporters for additional configurators, (c) creating a tool-independent product configurator interface, and (d) leveraging AI systems to enhance product modelling using CONTO's formal structure. We hope that CONTO represents a significant advancement toward interoperable product configuration systems that adapt to diverse organizational needs while maintaining semantic integrity.

Declaration on Generative AI

During the preparation of this work, the authors used Sonnet 4.0 in order to: paraphrase and reword, improve writing style, and grammar and check spelling. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] S. Mittal, F. Frayman, Towards a generic model of configuraton tasks., in: IJCAI, volume 89, Citeseer, 1989, pp. 1395–1401.

[2] T. Soininen, J. Tiihonen, T. Männistö, R. Sulonen, Towards a general ontology of configuration, Artif. Intell. Eng. Des. Anal. Manuf. 12 (1998) 357–372.

[3] D. Yang, R. Miao, H. Wu, Y. Zhou, Product configuration knowledge modeling using ontology web language, Expert Systems with Applications 36 (2009) 4399–4411.

[4] M. M. Amini, T. Coudert, E. Vareilles, M. Aldanondo, Integration of Ontologies and Constraint Satisfaction Problems for Product Configuration, in: 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2021, pp. 578–582.

[5] E. K. Abbasi, T. Leclercq, P. Heymans, A meta-model for product configuration ontologies, in: Proceedings of the 26th ACM International Systems and Software Product Line Conference – Volume B, ACM, 2022, pp. 166–173.

[6] A. Smirnov, N. Shilov, A. Kashevnik, T. Jung, M. Sinko, A. Oroszi, Ontology-driven product configuration – industrial use case, in: Proceedings of the International Conference on Knowledge Management and Information Sharing, SciTePress, 2011, pp. 38–47.

[7] H. Haav, R. Maigre, A semantic model for product configuration in timber industry, in: Databases and Information Systems X, volume 315 of *Frontiers in Artificial Intelligence and Applications*, 2018, pp. 143–158.

[8] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, G. Tack, MiniZinc: Towards a standard CP modelling language, in: Principles and Practice of Constraint Programming, 2007, pp. 529–543.

[9] D. L. McGuinness, J. R. Wright, Conceptual modelling for configuration: A description logic-based approach, AIEDAM 12 (1998) 333–344.