Facilitating Configuration Model Formalization based on Systems Engineering

Eugen Rigger[1] and **Ruth Fleisch**[1] and **Tino Stankovic**[2]

Abstract. Dynamic markets that call for customized products and shorter lead time in product development urge companies to apply technologies such as product configuration systems to leverage performance in product development and its related processes. Product configuration can be referred as a knowledge intensive technology that relies on the application of formalized product knowledge to enable automated reasoning. However, the related knowledge formalization is still considered a major obstacle for the integration of configuration technologies since often being conducted by non-domain experts. In this work, we present a method for the formalization of product configuration related knowledge as early as during development of an engineering system using the systems modelling language SysML. Based on a reference example from literature, it is shown how the required configuration knowledge can be formalized by the systems engineer to avoid costly and error prone knowledge acquisition in later stages of the product lifecycle. The yielded SysML model relies on graphical modelling, only, and can be directly integrated in existing system models. Thereby, the proposed method facilitates formalization of configuration knowledge for engineers and enables reuse of existing models so to save time and reduce errors when formalizing configuration models.

1 INTRODUCTION

Currently, engineering companies are facing two challenges that call for the application of new methods in the engineering and sales of their products: first, the complexity of engineering systems is growing with respect to the number of components and functions of a system as well as the involved disciplines [1]. Second, product development cycles tend to get shorter and customers ask for highly customized products with reduced engineering lead time [2]. In response to these needs, model-based systems engineering (MBSE) and product configuration can be considered as enabling methods that will allow industry to reconcile and meet the two conflicting challenges. MBSE is defined as "the formalized application of modelling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [3]. In this respect, the Systems Modelling Language (SysML)[4] has been established as modelling standard for MBSE and can be considered the most commonly applied language in MBSE [5]. SysML extends the unified modelling language (UML) [6] that is primarily applied in the software engineering domain and features object-oriented modelling of multidisciplinary systems.

Product configuration is applied to support the decision-making process in sales and engineering phases [7] based on reasoning among a set of predefined components and their relations, referred to as the configuration model [8]. Product configuration systems support the product configuration by automated reasoning based on customer requirements and the configuration model. However, a major obstacle in the application of product configuration systems is the knowledge acquisition that is required to enable formalization of the configuration model [9]. Reason for this is the current knowledge acquisition practice where knowledge engineers acquire and formalize the knowledge gained from domain experts [2] which is a costly process that is also considered critical since it is founded upon knowledge sharing and trust [10]. Regarding knowledge formalization, the application of object-oriented modelling supports the formalization of configuration models [11]. Hence, there is potential to integrate MBSE and product configuration so that knowledge that is defined in product development can be reused in later stages of the lifecycle to establish a product configuration system. This reuse of knowledge can facilitate the process of knowledge acquisition or even make it obsolete. However, currently the topics MBSE and product configuration are not linked [12]. Hence, there is a gap that integrates systems modelling and knowledge formalization for configuration models.

In this paper we propose a method that closes this gap by linking MBSE and product configuration using SysML. The method demonstrates how standard modelling language syntax can be applied to formalize dependencies in the systems model to establish a configuration model. A systems model that already captures the constraints and dependencies regarding the product architecture can be adapted and extended so to reuse already formalized knowledge and ensure consistency among models. The resulting configuration model can then be applied in product configuration systems via model transformations [13]. Consequently, the focus and motivation of this work is to enable system engineers/domain experts to contribute to the definition of the configuration model, enable reuse of knowledge for configuration modelling and thereby ensure consistency of models and knowledge among different stages of the product lifecycle. Thus, the knowledge acquisition bottleneck is tackled, in particular for companies applying MBSE. The method presented in this paper is validated based on a case study of a configuration example for personal computers according to [8,11].

The remainder of the paper is structured as follows: Section 2 introduces the related background of MBSE and knowledge

[1] Digital Engineering, V-Research GmbH, Austria, eugen.rigger@v-research.at

[2] Engineering Design and Computing Laboratory, ETH Zurich, Switzerland, tinos@ethz.ch

formalization in the context of product configuration and SysML. Section 3 then introduces the method for modelling configuration knowledge during product development. Section 4 shows the validation of the proposed method based on a case study and discusses attained results. The paper closes by presenting concluding remarks in Section 5.

2 BACKGROUND

In this section, first the terminology of data, information and knowledge is clarified and ways to differentiate the types of knowledge are reviewed and put into context with product configuration. Next, basics of MBSE are introduced and existing efforts for application of the SysML modelling language as a standardized language for knowledge formalization are introduced and reviewed. Based thereon, the research gaps addressed in this work are presented.

2.1 Types of Engineering Knowledge

Knowledge can be classified as tacit and formal knowledge [14] whereas the first refers to expert knowledge and intuition, formal engineering knowledge corresponds to information embedded in data, such as design guidelines, SysML and CAD models etc. To further distinguish the terms “knowledge”, “information”, and “data” in this work, the definition by the VDI 5610 [15] which is aligned with other definitions found in literature, e.g. [16,17] is applied:

- “Data are objective facts, they cannot be interpreted without context and further backgrounds. They are to be taken as “raw material”.
- Information are structured data with relevance and purpose, which can be put into a context, categorized, calculated and corrected.
- Knowledge is linked information, which enables to draw comparisons, to establish links and to make decisions. “

In the context of computational support to automate tasks of product development which is referred to as design automation, various ways of structuring engineering knowledge are presented in literature. The work from [18] differentiates process, task and product knowledge. Product knowledge refers to the knowledge about the specifics of the product, such as its architecture, its components, and the related dependencies. Process knowledge captures the knowledge on how to apply information in the context of the product. The task knowledge considers knowledge on algorithms and rules to update the product model. Similarly, [19] distinguish between procedural and declarative knowledge, where procedural refers to design process knowledge and declarative to product knowledge. In the context of product configuration systems that feature strict separation of the configuration model and the related reasoning [20], product knowledge can be considered the basis for the configuration model. A configuration model captures all information about a product so that a constraint satisfaction problem (CSP) can be formalized. The CSP forms the basis of a product configuration systems which then deduces configurations based on user input [8]. A basic CSP is described by a set of variables, the related domains (ranges) and constraints that limit the combination of variables.

2.2 SysML-based Knowledge Formalization

Regarding the usage of a standardized language for formalization of engineering knowledge, SysML has recently been addressed by multiple approaches identified in literature. SysML has evolved since 2007 as a standardized model language to support model-based systems engineering (MBSE) [21,22]. SysML as an object-oriented modelling language aims to support communication and understanding of formalized knowledge [22]. The language provides the full semantic foundation for documentation of system requirements, behavior, structure, and parametric relations. As a standardized language, SysML features reuse of models to avoid loss of knowledge between projects and reduce cost and risk in design [1]. In the context of formalization of a configuration model, the definition of hierarchical structures and relations among these are of relevance. Figure 1 depicts the diagrams that are considered relevant for the formalization of configuration models.

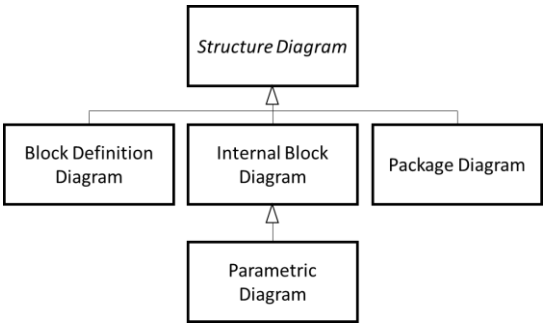

Figure 1. Excerpt of SysML diagram taxonomy [4]

First, *package diagrams* (PKG) can be used to organize the model in containers referred to as packages. Packages aggregate other model elements and are described by a name and URL (Uniform Resource Locator) making them uniquely identifiable and accessible so to foster reuse of models. For instance, model libraries or distinct parts of a system model such as its functional or logical architecture are organized within separate packages. The *block definition diagram* (BDD) enables the definition of hierarchical relations among the entities of a system called *blocks*. *Blocks* are used to describe types of physical entities such as components but also conceptual entities such as the functions of a system. *Blocks* are described based on *part*, *value* and *constraint properties*. *Value properties* account for quantifiable characteristics of a block such as weight, speed, etc. and can be typed by primitive types such as integers, reals or strings. Default values or default range of values can be assigned to value properties upon definition of a block. *Part-of associations* are used within block definition diagrams among blocks to define the *part properties* describing the composition of a block. In this context, *multiplicities* can be used to indicate the total amount of child blocks of a specific type, e.g. a car consist of four wheels. To model variance in a system, *generalizations* can be used. For example, “SUV” or “Van” refer to two different variants of the block “car” that both inherit the properties of the more general parent block. For the case that a block is used as a blueprint so to be reused by its specialized variants, blocks can be typed as *abstract*. This is particularly useful for definition of model libraries so that its elements can be reused among various models [23]. The *internal block diagram* (IBD) is used to detail the internal structure of a specific block, to for

example define interfaces and connections among nested blocks. However, in the context of configuration modelling, the *parametric diagram* (PAR) is more applicable since it enables the formalization of constraints among properties of blocks. For this purpose, SysML provides *constraint blocks* for the formalization of equations and rules. *Constraint blocks* can be assigned as *constraint properties* to *blocks*. Using a PAR, the *constraint property* can be linked to the block's properties using binding connection. Figure 7 shows an example of a parametric diagram representing a constraint property of type *CalculatePrice* with rounded edges. The constraint property belongs to the owning block *HDUnit* denoting the boundary of the PAR. Using *binding connections*, the PAR puts into context the constraint property, HDUnit's value property *price* and the value properties *price* that belong to *hdcontroller* and *hdisk*. To customize SysML for specific domains, *stereotypes* can be introduced to define new modelling elements. This can be used to define a domain specific language for parametric CAD [24].

Relevant approaches in the context of design automation feature systematics for definition of model-libraries for reuse [23,25], formalization of parameter synthesis tasks [26], formalization of CAD configuration tasks [24], formalization of simulation-based design tasks [27] [28], or neutral modelling of simulation models that can be then translated to the format of the desired simulation tool [29]. Reason for the interest in SysML for design automation task formalization is the aspect of integration of formalizations to MBSE processes, and its means to support communication and understanding of formalized knowledge. However, presented approaches rely on customizations of SysML based on newly introduced stereotypes hindering the integration to industrial environments where models are defined following the SysML standard. Further, focus is put on technological aspects of the customization of the language rather than the possibility for integration to workflows of engineers. For example, in [26] stereotypes reflecting the characteristics of a mathematical optimization problems are introduced. Hence, for successful application, engineers require fundamental knowledge about mathematical modelling.

Regarding the domain of product configuration, [12] show that MBSE and product configuration are currently not linked an suggest the usage of SysML models reflecting the modular product architecture as a basis for development of the configuration model. However, the suggested approach by [12] lacks details regarding the integration of the system model and the configuration model, such as consistency management.

Consequently, means to enable engineers to formalize product knowledge in a reusable, intuitive manner that also ensures consistency of models among the product lifecycle are missing. In a literature review on knowledge levels required for the formalization of design automation tasks, [30] proposed to use SysML as a standardized language providing the full semantics required for design automation task definition. Thus, in this work, a method is proposed that facilitates knowledge formalization for usage in product configurators following the SysML standard.

3 CONFIGURATION MODEL FORMALIZATION USING SysML

In this section the method combining MBSE and product configuration is presented. SysML semantics are overloaded to account for the specifics of configuration modelling. Thus, the formalization of a configuration model can take place early in product development, and integrated to a (MBSE) process and the related models. Thereby, a system model corresponding to the "single source of truth" can be established integrating all relevant information about the system for later processing during the product lifecycle [1]

In the following, first, means to organize the configuration model within a system model are introduced. Next, the semantics for the formalization of a configuration model are introduced.

3.1 Organize the Configuration Model

To organize and structure the knowledge for establishing a configuration model, the following packages are defined: the package "product architecture" captures the information related to the hierarchy of the system and used blocks. The package "interconnections" is used to aggregate the definitions of constraints among the blocks' properties. Within the package "model libraries", model libraries can be defined or integrated to foster the reuse of model elements. Further, the usage of abstract blocks for instantiation of model libraries enables formalization of constraints among abstract blocks rather than the specific children blocks. Thereby, the formalization of generic CSPs [8] is rendered feasible. Figure 2 shows the package diagram as a starting point for formalizing the configuration model. It is shown how the package "Configuration" aggregates the three packages used to define the configuration model and how the package itself is integrated to the system model. Therefore, the information about the system can be captured within one model comprehensively. The organization of the system model depicted in Figure 2 follows the structure proposed in [22] enriched by the package "Configuration".

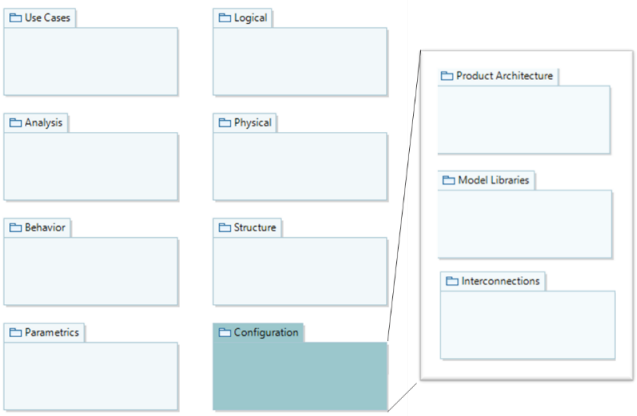

Figure 2. PKG depicting the structure of a configuration model within a systems model.

3.2 Define the Configuration Model

In the following it is shown how the packages depicted in Figure 2 can be enriched to fully define a configuration model by using existing SysML syntax, only. First, the concept of model libraries is briefly introduced so that the definition of generic components is enabled (3.2.1). Afterwards, the instantiation of product architectures is shown including the formalization of related variables and constraints directly within the product architecture using relations on a block definition diagram (3.2.2) and by

defining parametric relations using constraint blocks and parametric diagrams (3.2.3).

3.2.1 *Model Libraries*

A model library can be defined using a block definition diagram. Once a model library is instantiated, it can be reused among different projects. In this work, the systematics of model library definition as proposed in [23,25] are applied as illustrated in Figure 3. In particular, the concepts of inheritance are applied to define abstraction hierarchies [31]. With this respect, SysML provides *generalization relationships* between blocks as well as the definition of *abstract elements*. This enables the definition of abstract components where generic constraints can be defined upon. The information regarding variance of a product can be already captured within the system model for description of product families [32] and therefore can be directly reused within the configuration model.

Table 1. Model elements used for definition of component libraries.

Model elements	Model type	Symbol	Meaning
Block	Block	«Block» Child 1	Modular unit of a system
Abstract Block	Block	«Block» LibraryElement	The block cannot be instantiated and is used for inheriting properties, only.
Specialization	Specialization Relation		Inheritance of parent block's properties to child block

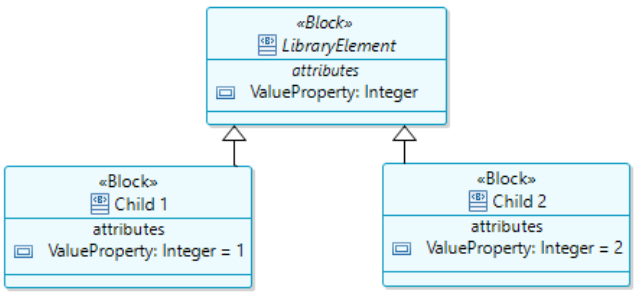

Figure 3. Generic example of component library illustrating the modelling concepts shown in Table 1.

3.2.2 *Product Architecture*

The product architecture in MBSE is defined using a block definition diagram and captures all aspects regarding the hierarchical organization of a system. Hence, the product architecture can be directly reused for defining the configuration model. According to the object-oriented paradigm of SysML [22], the hierarchical structure of a system (the product architecture) requires a main block that aggregates the parts (i.e. subsystems / - assemblies / - components). In the context of the configuration model, a main block *ConfigurationModel* is introduced so to separate the configuration model from the remaining system model and serve as a starting point. Further, characteristic input variables are assigned to the *ConfigurationModel* itself, so that subsequently constraints can be defined using these variables. For example, a customer might want to specify a maximum price when configuring a car. The block *ConfigurationModel* is then linked to the block "System" using a part-of association.

Variables and dedicated types of constraints can be declared directly within the product architecture using the modelling elements listed in table 2: *Abstract blocks* are used to indicate selection of components, *multiplicities* to define variability in the number of a specific block and *value properties* to specify (discrete) variables and the corresponding solution domain by specifying the default value. Value properties of SysML blocks that are not assigned any value or a specific value, respectively, are considered as parameters and are not subject to the CSP. Within Figure 4, the block *LibraryElement* refers to a component that needs to be selected from the component library according to requirements. The *DiscreteVariable* of *Child1* refers to a variable with indicated domain by assigning a value range to the value property. The *multiplicity* of *Child2* shows that this block needs to be in the system at least once but can also be prevalent in a larger amount within the final configuration, depending on customer requirements. It must be noted that variable declaration can be combined, e.g., a library element possibly contains a variable which is to be determined when configuring the product.

Table 2. Key model elements used on BDD for definition of product architectures including constraints.

Model elements	Model type	Symbol	Meaning
Abstract Block	Block	«Block» LibraryElement	Indication of a discrete variable for component selection. Works in conjunction with model libraries, only.
Multiplicity	Part-of association	[1..*]	Indication of parts as discrete variables with multiplicity as lower and upper bound of variable
Variable range	Value property	a,b,c,d,…	Indication of (discrete) variable domain.

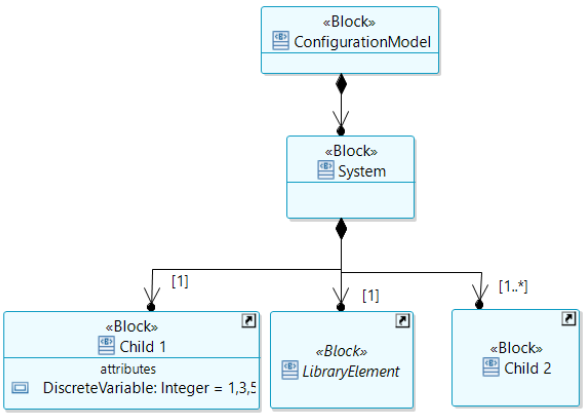

Figure 4. Generic example of product architecture as well as possible constraints using concepts shown in Table 2.

3.2.3 *Interconnections*

To define constraints related to interconnection of value and part properties of the blocks, parametric diagrams can be used. Thereby, constraints for the configuration model can be defined based on linking constraint blocks to specific value and part properties, e.g. if each block contains a value property indicating the cost, a constraint can be defined stating the cost of all active components needs to be below a specific threshold. For each interconnection, first a *constraint block* is defined specifying all variables used within the constraint by assigning value and part properties to the constraint as well as a constraint expression. Next, a parametric diagram depicting the constraint block is instantiated that is then linked to the package "interconnections" to provide the required overview of formalized constraints. Depending on the type of constraint properties, the main different types of constraints can be formalized as follows:

1. Parametric relations for two or more value properties.
2. Decision structure matrices [33] are used to define compatibility among part properties: a *.csv* table (comma-separated values) with semicolon as separator and constraints expressions. Thereby, all possible instances of the part properties build the column and row headings. Incompatibilities are indicated based on "0", required combinations by "x". For the case of no entry, combinations of the part properties are allowed.
3. Decision tables [34] are applied to formalize if-then relations among different types of properties. In the rows, first all conditions are listed followed by the properties that need to be excluded ("0") or are required ("x").

Examples of the formalization of interconnections are provided in the following section where a case study for configuration of personal computers is introduced. With the suggested types of constraints, the provided case study can be formalized. However, the list does not claim completeness and additional types will need to be added for special cases.

4 CASE STUDY

In [8] a configuration model of a personal computer (PC) is described using UML and for some of the constraints a textual representation is applied. In the following, the formalization of the configuration model using SysML syntax, only, is presented. Similar to Section 3, the model is introduced by first showing the package structure of the configuration model in relation to the personal computer system model. Following this, the model libraries are presented as well as the product architecture. Finally, representative interconnections are depicted detailing the three different types of constraints according to Section 3.2.3. All models were defined using the Papyrus 5.0.0 open-source modelling environment.

4.1 Organizing the PC Configuration Model

In a first step, the PC configuration model is integrated to the existing system model of a personal computer as depicted in Figure 2. Since the configuration model is integrated on the same level as the system model, reuse of elements from the system model is enabled.

4.2 Defining the PC Configuration Model

Figure 5 shows the model library that is used for defining the components of the PC that need to be selected based on customers' requirements. It shows that different variants for the hard disk (HDisk), the related controller (HDController), the central processing unit (CPU), the operating system (OS), the motherboard (MB) and screens exist.

Figure 6 shows the product architecture of the PC. Multiplicities define the domain of decision variables of the amount of each child component. The value properties in the configuration model (maxPrice, usage, efficiency) denote customer requirements which are linked to properties of other blocks via interconnections as detailed below.

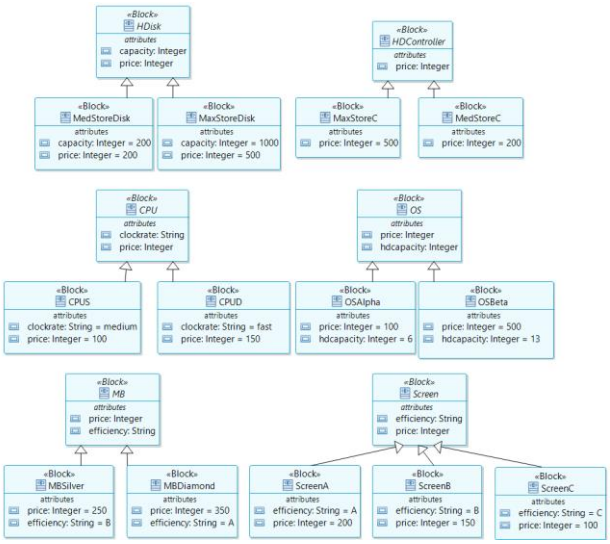

Figure 5**.** Model library of the PC configuration example.

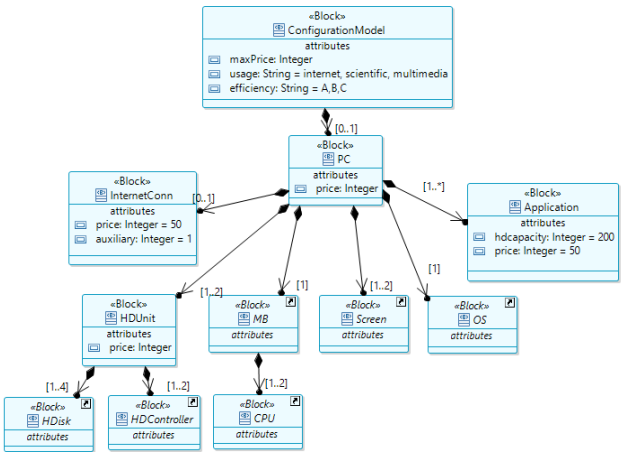

Figure 6. Product architecture of the configuration model.

Regarding the definition of interconnections among the blocks' properties, Figure 7 shows an example of a parametric relation: The defined constraint links several value properties denoting that the price of the hard disk unit (HDUnit) is defined by the prices of the hard disks and controllers. It must be noted that multiplicities are considered implicitly by multiplying the value properties with the amount defined by the multiplicities of the owning blocks.

Similarly, the computation of the price of the PC can be defined as the sum of the prices of the hard disk units, the motherboard, the CPUs, the operating system, the screens, applications, and the internet connection (InternetConn). Additional interconnections are: One linking the price limit defined by the customer (maxPrice) with the price of the PC, one ensuring that the capacity of all hard disks in total is larger than the required capacities (value properties hdcapacity) of the operating system and selected applications, and two others equating the efficiency of the PC with the efficiency of the motherboard and of the screens, respectively. Interconnections among two part properties are used to denote restrictions and requirements regarding the selection of CPUs, motherboards and operating systems. Figure 8 shows the definition of the decision table for CPUs and motherboards. The constraint property highlights the formalization of the constraint as a *.csv* table. Finally, interconnections between a value property and part properties are applied to link the customer input regarding the selection of efficiency classes to the respective motherboards and screens. In this case, the .csv-table lists the instances of the value properties as column headings and all possible instances of the part properties. Figure 9 show the respective parametric diagram. Similarly, internet and multimedia usage are linked to the internet connection so to define that for these cases an internet connection must be prevalent. Further, for the case of scientific usage, a CPU of type CPUD needs to be selected.

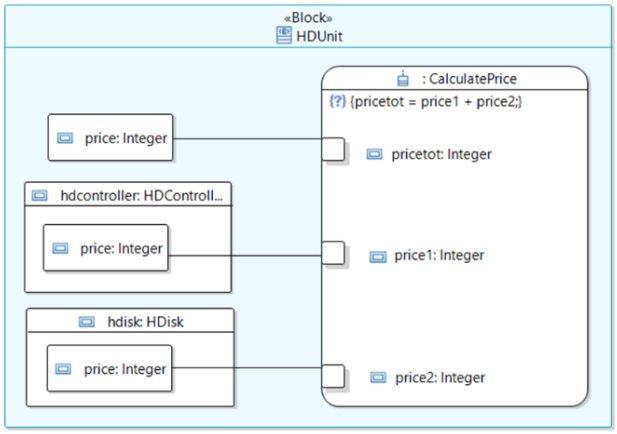

Figure 7. Interconnection among value properties defining the price of the HDUnit.

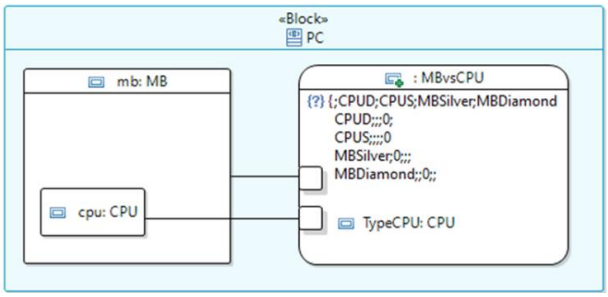

Figure 8. Interconnection among two part properties defining compatibilities among blocks.

5 DISCUSSION

The discussion of the proposed method for formalization of configuration models using SysML is based on the findings from the case study presented in Section 4 and split into two parts: First, the knowledge representation yielded by application of SysML is reviewed according to the nine criteria introduced in [8]. Second, the integration within MBSE is critically reviewed.

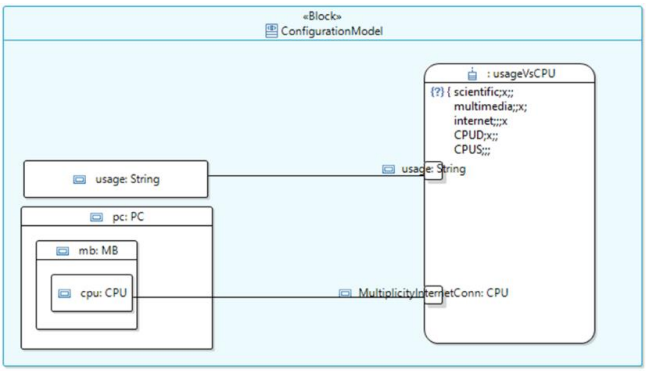

Figure 9. Interconnection among value and part properties.

5.1 Benchmark of Knowledge Representation

In the following the criteria for assessment of knowledge representations for configuration models [8] are introduced using italic letters. Each criterion is then discussed separately:

Standard graphical modelling concepts?

The approach presented in this work builds upon the SysML, an established standard. Recent reviews show that (1) model-based systems engineering is gaining popularity [35], and (2) SysML is the dominating modelling language for MBSE [5]. The case study shows that a configuration model can be fully formalized using SysML-based graphical modelling, only. No coding is required for formalization of additional constraints. Thereby, the presented approach adds up to existing formalization relying on UML.

Component-oriented modelling?

Being an object-oriented modelling language developed for the modelling of complex systems, SysML enables the representation of hierarchical component structures facilitating the communication of configuration models. Additionally, the integration of the configuration model directly within the system model fosters reuse of components mitigating efforts and reducing errors for formalization of configuration models.

Automated consistency maintenance?

Commercial modelling applications provide basic support for model validation. However, integration of support for the formalization of configuration models requires model transformations to other representations where evaluation of constraints is rendered feasible. This topic is considered a potential line of future work.

Modularization concepts available?

The concept of package diagrams allows to organize a model in modules. Further, the presented approach presents a concept for organizing the configuration model according to the type of knowledge that is being formalized: product architecture, model libraries and interconnections.

Support of easy knowledge base evolution and maintenance?

SysML is intended to support formalization and communication of complex knowledge. Therefore, the proposed approach aims to

enable knowledge engineers to reuse essential product knowledge from the system model. Preliminary usability studies with engineers from industry indicate the potential of the approach to enable domain experts to formalize essential parts of a configuration model themselves. Therefore, the proposed approach fosters collaboration of domain experts and knowledge engineers.

Model-based knowledge representation?

Using SysML, the configuration model is represented in a declarative manner that needs to be transformed to a representation that can be linked to reasoning mechanisms. Hence, the configuration model and problem-solving logic are strictly separated as required in a model-based approach [36].

Efficient reasoning?

Using model transformations, the SysML configuration model can be transferred to various representations such as CSP models. However, model transformation potentially yields sub-optimal formalizations leading to losses in performance when compared to formalizations done directly within the solver's environment.

Able to solve generative problem settings?

No, however, when using nested configuration models, generative problem configuration scenarios where the constraints are added to the configuration model on demand could be enabled. Future work needs to elaborate on concepts enabling nested configuration when using SysML modelling.

Able to provide explanations?

Since the configuration model needs to be transformed to a representation for solving the configuration problem, explanations are potentially enabled when using appropriate representations [8].

5.2 Systems Engineering Integration

The introduced method builds upon SysML syntax without modifications or extensions of the modelling language. Therefore, the configuration model can be defined reusing essential parts of the system model. When combining MBSE and product configuration, domain experts implicitly formalize parts of the configuration model and knowledge engineers can directly build upon the system model, respectively. The integration of already validated knowledge allows saving time and reducing errors in modelling. Further, having one representation for different stages of the product lifecycle ensures consistency of data, such as propagation of modifications in product families after engineering change requests. However, in the presented approach the organization of models within packages needs to be strictly followed so that reasoning among models is confined to the content available within the package. In this respect, future work needs to elaborate on means to assure the correct formalization of configuration models in existing system models. For example, processes need to be elaborated to define collaboration among domain experts and knowledge engineers.

Considering model transformations, a next step corresponds to the elaboration of transformations from the SysML configuration model to different problem-solving domains such as CSP or Boolean satisfiability problem [8]. In this context, future work also needs to address the comprehensiveness of the proposed method for formalization of configuration models based on additional examples from engineering industry as well as the integration of means to facilitate knowledge base debugging [37]. Regarding the latter, a key challenge is to establish a link the two representations, so that identified faulty constraints within the computational model can be correctly modified within the SysML model. Finally, future work should investigate the extension of the proposed modelling techniques so to enable knowledge formalization for design automation methods in general. This will facilitate integration of computational methods in product development so to enable a vision of digital engineering, where computational methods can be rapidly defined and integrated by engineers themselves.

6 CONCLUSION

In this paper a method to integrate MBSE and configuration modelling is presented. Based on SysML, the method builds upon a standardized language and enables integration of system and configuration models so to ensure consistency of product knowledge along the product lifecycle. Thereby, reuse of models is enabled to save time when modelling, reduce errors when relying on already validated models and facilitate collaboration of domain experts and knowledge engineers. Evaluation of the proposed method based on a reference example from literature shows that a full configuration model can be defined using existing SysML syntax, only. The application of the method does not require the use of coding techniques for formalization of the configuration model. Future work needs to investigate the broader validation of the approach based on additional use cases as well as the assessment of the usability with domain experts and knowledge engineers. The focus of these investigations will be on the extension of the method towards the formalization of design automation tasks and debugging of knowledge bases. The vision is that facilitated knowledge formalization will foster the application of design automation in industrial practice.

ACKNOWLEDGEMENTS

We would like to thank the referees for their comments, which helped improve this paper considerably.

This work has been partially supported and funded by the Austrian Research Promotion Agency (FFG) via the "Austrian Competence Center for Digital Production" (CDP) no. 881843, the K2 centre InTribology, no. 872176 as well as ASID, no. 856326.

REFERENCES

[1] B. Beihoff, C. Oster, S. Friedenthal, Paredis, Christiaan, D. Kemp, H. Stoewer, D. Nichols, J. Wade, A World in Motion – Systems Engineering Vision 2025, INCOSE, San Diego, California, 2014.

[2] C. Forza, F. Salvador, Product information management for mass customization, Palgrave Macmillan New York, 2007.

[3] INCOSE-TP-2004, Systems Engineering Vision 2020, INCOSE, 2004.

[4] OMG SysML, (n.d.). http://www.omgsysml.org/.

[5] J. Lu, G. Wang, J. Ma, D. Kiritsis, H. Zhang, M. Törngren, General Modeling Language to Support Model-based Systems Engineering Formalisms (Part 1), INCOSE Int. Symp. 30 (2020) 323–338. https://doi.org/10.1002/j.2334-5837.2020.00725.x.

[6] Object Management Group, UML, UML - Unified Model. Lang. (n.d.). https://www.omg.org/spec/UML/About-UML/.

[7] L. Hvam, N.H. Mortensen, J. Riis, Product customization, Springer, Berlin, 2008.

[8] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, Configuration Knowledge Representation and Reasoning, in: Knowl.-Based Config., Elsevier, 2014: pp. 41–72. (2015).

[9] K. Kristjansdottir, S. Shafiee, L. Hvam, C. Forza, N.H. Mortensen, The main challenges for manufacturing companies in implementing and utilizing configurators, Comput. Ind. 100 (2018) 196–211. https://doi.org/10.1016/j.compind.2018.05.001.

[10] M. Asrar-ul-Haq, S. Anwar, A systematic review of knowledge management and knowledge sharing: Trends, issues, and challenges, Cogent Bus. Manag. 3 (2016). https://doi.org/10.1080/23311975.2015.1127744.

[11] A. Felfernig, G.E. Friedrich, D. Jannach, UML AS DOMAIN SPECIFIC LANGUAGE FOR THE CONSTRUCTION OF KNOWLEDGE-BASED CONFIGURATION SYSTEMS, Int. J. Softw. Eng. Knowl. Eng. 10 (2000) 449–469. https://doi.org/10.1142/S0218194000000249.

[12] S. Florian, S. Lea-Nadine, K. Dieter, MBSE-basierte Produktkonfiguratoren zur Analyse der Modularisierung bei der Entwicklung modularer Baukastensysteme, in: R.H. Stelzer, J. Krzyzwinski (Eds.), Entwerf. Entwick. Erleb. Produktentwicklung Des. 2019 Band 2, TUDpress, Dresden, 2019: pp. 55–70.

[13] M. Brambilla, J. Cabot, M. Wimmer, Model-Driven Software Engineering in Practice: Second Edition, Synth. Lect. Softw. Eng. 3 (2017) 1–207. https://doi.org/10.2200/S00751ED2V01Y201701SWE004.

[14] S.K. Chandrasegaran, K. Ramani, R.D. Sriram, I. Horváth, A. Bernard, R.F. Harik, W. Gao, The evolution, challenges, and future of knowledge representation in product design systems, Comput.-Aided Des. 45 (2013) 204–228. https://doi.org/10.1016/j.cad.2012.08.006.

[15] Verein Deutscher Ingenieure, VDI 5610 - Blatt 1 - Wissensmanagement im Ingenieurswesen - Grundlagen, Konzepte, Vorgehen, 2009.

[16] B.J. Hicks, S.J. Culley, R.D. Allen, G. Mullineux, A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design, Int. J. Inf. Manag. 22 (2002) 263–280. https://doi.org/10.1016/S0268-4012(02)00012-9.

[17] J. Stjepandić, W.J.C. Verhagen, H. Liese, P. Bermell-Garcia, Knowledge-Based Engineering, in: J. Stjepandić, N. Wognum, W.J.C. Verhagen (Eds.), Concurr. Eng. 21st Century, Springer International Publishing, Zürich, 2015: pp. 255–286. 10.1007/978-3-319-13776-6_10 (2015).

[18] D. Baxter, J. Gao, K. Case, J. Harding, B. Young, S. Cochrane, S. Dani, An engineering design knowledge reuse methodology using process modelling, Res. Eng. Des. 18 (2007) 37–48. https://doi.org/10.1007/s00163-007-0028-8.

[19] J.H. Panchal, M.G. Fernández, C.J.J. Paredis, J.K. Allen, F. Mistree, A Modular Decision-centric Approach for Reusable Design Processes, Concurr. Eng. Res. Appl. 17 (2009) 5–19. https://doi.org/10.1177/1063293X09102251.

[20] J. Tiihonen, A. Felfernig, An introduction to personalization and mass customization, J. Intell. Inf. Syst. 49 (2017) 1–7. https://doi.org/10.1007/s10844-017-0465-4.

[21] S. Friedenthal, R. Griego, M. Sampson, INCOSE model based systems engineering (MBSE) initiative, in: INCOSE 2007 Symp., 2007. (2016).

[22] S. Friedenthal, A. Moore, R. Steiner, A practical guide to SysML: the systems modeling language, Morgan Kaufmann, (2014).

[23] B. Kruse, A Library-based Concept Design Approach for Multi-Disciplinary Systems in SysML, Dissertation, ETH Zürich, 2016.

[24] P. Klein, J. Lützenberger, K.-D. Thoben, A Proposal for Knowledge Formailization in Product Development Processes, in: Proc. Int. Conf. Eng. Des. ICED15, Design Society, Milano, Italy, 2015: pp. 261–272.

[25] S. Wölkl, Model Libraries for Conceptual Design, Dissertation, TU München, (2012).

[26] A.A. Shah, C.J.J. Paredis, R. Burkhart, D. Schaefer, Combining Mathematical Programming and SysML for Automated Component Sizing of Hydraulic Systems, J. Comput. Inf. Sci. Eng. 12 (2012) https://doi.org/10.1115/1.4007764.

[27] R.S. Peak, R.M. Burkhart, S.A. Friedenthal, M.W. Wilson, M. Bajaj, I. Kim, Simulation-Based Design Using SysML Part 1: A Parametrics Primer, in: INCOSE Int. Symp., Wiley Online Library, San Diego, California, 2007: pp. 1516–1535. http://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2007.tb02964.x

[28] R.S. Peak, R.M. Burkhart, S.A. Friedenthal, M.W. Wilson, M. Bajaj, I. Kim, Simulation-Based Design Using SysML Part 2: Celebrating Diversity by Example, in: INCOSE Int. Symp., Wiley Online Library, San Diego, California, 2007: pp. 1536–1557. http://onlinelibrary.wiley.com/doi/10.1002/j.2334-5837.2007.tb02965.x

[29] C. Bock, R. Barbau, I. Matei, M. Dadfarnia, An Extension of the Systems Modeling Language for Physical Interaction and Signal Flow Simulation, Syst. Eng. 20 (2017) 395–431. https://doi.org/10.1002/sys.21380.

[30] E. Rigger, T. Stanković, K. Shea, Task Categorization for Identification of Design Automation Opportunities, J. Eng. Des. (2018). https://doi.org/10.1080/09544828.2018.1448927.

[31] C.L. Dym, R.E. Levitt, Knowledge-based systems in engineering, McGraw-Hill, New York, (1991).

[32] H.P.L. Bruun, N.H. Mortensen, U. Harlou, M. Wörösch, M. Proschowsky, PLM system support for modular product development, Comput. Ind. 67 (2015) 97–111. https://doi.org/10.1016/j.compind.2014.10.010.

[33] T.R. Browning, Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities, IEEE Trans. Eng. Manag. 63 (2016) 27–52. https://doi.org/10.1109/TEM.2015.2491283.

[34] E. Hering, Entscheidungstabelle nach DIN 66241, in: Softw.-Eng., Vieweg+Teubner Verlag, Wiesbaden, 1984: pp. 18–25. https://doi.org/10.1007/978-3-322-86222-8_3.

[35] A.M. Madni, M. Sievers, Model-based systems engineering: Motivation, current status, and research opportunities, Syst. Eng. 21 (2018) 172–190. https://doi.org/10.1002/sys.21438.

[36] S. Mittal, F. Frayman, Towards a Generic Model of Configuraton Tasks., in: IJCAI, Citeseer, (1989) pp. 1395–1401.

[37] L.L. Zhang, Product configuration: a review of the state-of-the-art and future research, Int. J. Prod. Res. 52 (2014) 6381–6398. https://doi.org/10.1080/00207543.2014.942012.