Chiara Grosso Enrico Sandrin
Viet-Man Le (Eds.)

ConfWS 2025
27th International Workshop on Configuration

Bologna, Italy, October 25-26, 2025
Proceedings

Editors' contacts:
chiara.grosso@uniroma1.it, enrico.sandrin@unipd.it, v.m.le@tugraz.at

Preface

The 27th edition of the International Workshop on Configuration (ConfWS 2025) was co-located with the European Conference on Artificial Intelligence (ECAI 2025), hosted by the University of Bologna in Italy. As in previous years, ConfWS 2025 provided a lively forum for researchers and industry professionals interested in all aspects of configuration technologies.

ConfWS 2025 was organized as a two-day event and featured high-quality contributions across all configuration-related research areas. This edition placed particular emphasis on Green Configuration, in line with the EU Green Deal and the EU Agenda 2050, which aim to guide the European community toward a more sustainable future. Researchers and experts from academia and industry shared their contributions on the potential of configuration technologies to support sustainability goals. The program included special sessions on green configuration and sustainability, covering topics such as sustainable configurator applications, efficient reasoning, configuration space learning, integration of large language models (LLMs), and other aspects related to problem solving and optimization.

ConfWS 2025 welcomed participants from academia and industry. A total of 23 papers were submitted for peer review, and 17 were selected for publication in the workshop proceedings after evaluation by at least two independent reviewers per paper. Continuing the workshop's tradition, participants selected the best paper ("Generative Design as a Configuration Problem") and the best student paper ("From 4GL Spreadsheet Computations to Constraint Model Definitions – A Development Process").

We would like to express our sincere gratitude to all authors for their high-quality submissions, the program committee members for their thorough reviews. We also thank the University of Bologna, the ECAI Workshop Chairs, the ECAI Chairs, and Prof. Federico Chesani for their proactive support. Our special thanks go to PMH - Product Management Haag GmbH and to the keynote speaker Dr. Albert Haag for his inspiring contribution and for sharing reflections dating back to his first participation at ECAI in 1982.

Finally, we acknowledge the patronage of the MICS project (Made in Italy – Circular and Sustainable), funded as part of the European Union's Next-Generation EU program (Piano Nazionale di Ripresa e Resilienza – PNRR, Missione 4, Componente 2, Investmento 1.3 – D.D. 1551.11-10-2022, PE00000004). The views expressed in this editorial are solely those of the authors and do not necessarily reflect the official position of the European Union or the European Commission, which cannot be held responsible for any use made of the information contained herein.

December 2025 Chiara Grosso, Enrico Sandrin, Viet-Man Le

Workshop Chairs

Chiara Grosso, Sapienza University of Roma, Italy
Enrico Sandrin, University of Padova, Italy
Viet-Man Le, Graz University of Technology, Austria

Award Chair

Cipriano Forza, University of Padova, Italy

Program Committee

Alexander Felfernig, Graz University of Technology, Austria
Gerhard Friedrich, Alpen-Adria-Universität Klagenfurt, Austria
Gerhard Leitner, Alpen-Adria-Universität Klagenfurt, Austria
Lothar Hotz, Hamburger Informatik Technologie-Center, Germany
Andreas Falkner, Siemens AG Österreich, Vienna, Austria
Lidia Fuentes, Universidad de Málaga, Spain
Yue Wang, Hang Seng University, Hong Kong
José Ángel Galindo, Universidad de Sevilla, Spain
Markus Stumptner, University of South Australia, Australia
Lars Hvam, Technical University of Denmark, Denmark
David Benavides, Universidad de Sevilla, Spain
Cipriano Forza, University of Padova, Italy
Jean-Guillaume Fages, Cosling, France
Albert Haag, Product Management GmbH, Germany
Alois Haselboeck, Siemens AG Österreich, Austria
Richard Comploi-Taupe, Siemens AG Österreich, Vienna, Austria
Tomi Männistö, University of Helsinki, Finland
Abdourahim Sylla, Université Grenoble Alpes, France
Juha Tiihonen, Variantum, Finland
Franz Wotawa, Graz University of Technology, Austria
Rüdiger Dehn, Lino GmbH, Germany
Hao Xu, Renault, France
Sara Shafiee, Technical University of Denmark, Denmark
Roberto Boselli, University of Milano-Bicocca, Italy
Simone DiLeo, Sapienza University of Roma, Italy
Nikola Suzić, University of Trento, Italy
Alessio Trentin, University of Padova, Italy
Elise Vareilles, Toulouse University, IMT Mines Albi, France

Contents

QuickXPlain Explanations for Feature Model Configuration

Alexander Felfernig[†], Damian Garber[†], Viet-Man Le[†] and Sebastian Lubos[†]

Institute of Software Engineering and AI, Graz University of Technology, Graz, Austria

Abstract

Explanations play an important role in the context of feature model (FM) configuration. First, they can assure the interpretability of the calculated solutions (configurations) as a result of a feature model configuration process. Beyond this, explanations can support engineers (developers) of feature models in the identification of issues in the model, i.e., to figure out as to why on a semantic level the feature model does not fully represent the existing product (service) domain knowledge. In this paper, we discuss different basic explanation scenarios in the context of feature model development and feature model configuration. We show how these explanations can be supported on the basis of the concepts of conflict detection and model-based diagnosis.

Keywords

QuickXPlain, Explanation, Feature Model Configuration, Conflict Detection, Model-based Diagnosis

1. Introduction

Feature models (FMs) can be used for the representation of commonality and variability properties of highly-variant artifacts such as physical products and software [1, 2, 3, 4, 5, 6]. Formal representations of feature models such as SAT problems [7] and constraint satisfaction problems (CSPs) [8, 9] are often used to find a solution for a given feature model configuration task and – beyond that – for supporting different types of analysis operations used to assure the well-formedness and semantic correctness of feature models [3, 10]. Compared to the representation as SAT problem, CSPs allow for a more flexible knowledge representation, for example, in terms of a direct representation of logical equivalence properties and implications [4].

Independent of the used FM knowledge representation, it is important to provide users with explanations [4, 11, 12, 13, 14]. Such explanations can serve different purposes ranging from assuring interpretability for the user, increasing the trust level of a user, enhancing a user's domain knowledge, to persuading users to include/exclude specific features into/from an FM configuration [12, 15, 16, 17, 18, 19, 20, 21]. In this paper, we focus on the aspect of *interpretability.* We discuss different explanation scenarios in the context of feature model development and feature model configuration. For example, in the context of feature model development and maintenance, a modeler needs to know the set of features responsible for the violation of a specific property (e.g., void feature models or dead features in feature models). Furthermore, users of an FM configurator are interested as to why specific features have been included but also why other features have been excluded unexpectedly (the counterfactual case). In the following, we discuss different explanation scenarios and show how standard QuickXPlain-style conflict detection [22, 23] and diagnosis [24] algorithms can be applied to generate explanations and corresponding repairs in the case of inconsistencies.

The major contributions of our paper are the following: (1) we formalize basic explanation and repair tasks in FM development and configuration, (2) we show how corresponding explanations and repairs can be determined with existing conflict detection and diagnosis algorithms, and (3) to increase

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

[†]These authors contributed equally.

alexander.felfernig@tugraz.at (A. Felfernig); damian.garber@tugraz.at (D. Garber); v.m.le@tugraz.at (V. Le); sebastian.lubos@tugraz.at (S. Lubos)

0000-0003-0108-3146 (A. Felfernig); 0009-0005-0993-0911 (D. Garber); 0000-0001-5778-975X (V. Le); 0000-0002-5024-3786 (S. Lubos)

understandability, we provide examples of how to generate explanations and repairs. Specifically, we show how to create the mentioned explanations and repairs with the algorithms QuickXPlain [22] and FastDiag [24, 25]. With these contributions, we aim to show different ways of integrating conflict detection and diagnosis as a basis of explanation generation in FM modeling and configuration.

The remainder of this paper is organized as follows. In Section 2, we introduce an example feature model that serves as a working example throughout the paper. Thereafter, in Section 3, we introduce two basic algorithms supporting the tasks of conflict detection and diagnosis. In Section 4, we show how these algorithms can be applied to support different FM-related explanation scenarios. A discussion of threats to validity is provided in Section 5. In Section 6, the paper is concluded with an overview of research issues.

2. Example Feature Model

In the following, we present an example feature model from the domain of *survey software configuration*, which will serve as a running example throughout this paper (see Figure 1).

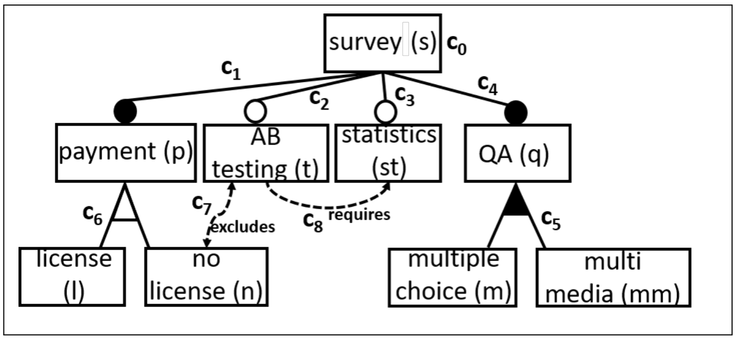


Figure 1: An example feature model (*survey software*).

The features in this model are arranged hierarchically including the following relationships: (1) *mandatory relationships* specify that certain features must be included in every configuration (e.g., the payment feature is required to be included in every configuration), (2) *optional relationships* indicate that certain features may be included, but their inclusion is optional (e.g., the statistics feature can optionally be added to a configuration), (3) *alternative relationships* specify that, within a set of sub-features, exactly one sub-feature must be chosen if the parent feature is included (e.g., one license type must be selected), and (4) *or relationships* require that *at least one* feature from a set of sub-features must be chosen if the parent feature is included (e.g., question answering (QA) can be handled with multiple-choice questions, multimedia-based representations, or both). In addition, *cross-tree constraints* can be used to impose further restrictions: (1) *excludes constraints* between two features prevent both from being included in the same configuration (e.g., if no license is selected as the payment model, ABtesting cannot be included in the same configuration), and (2) *requires constraints* between two features f_a and f_b specify that if f_a is included, f_b must also be included in the final configuration (e.g., the inclusion of the ABtesting feature necessitates the inclusion of statistics).

To enable FM configuration, feature models must be translated into a formal representation. Common approaches for this translation include SAT problems [26, 27], answer set programs (ASPs) [28], and constraint satisfaction problems (CSPs) [8, 29, 30]. In this paper, we adopt CSPs as the formal representation for feature models. For a detailed discussion of the rules governing the translation of feature models into logic-based representations, we refer to [4, 8]. The constraint-based representations we generate in this paper follow these established translation rules.

3. Conflict Detection and Model-based Diagnosis

In Table 1, we introduce a CSP-based formalization of the FM shown in Figure 1. In this formalization, c_0 is the *root constraint* (part of every feature model) that ensures that in each FM configuration at least one feature is included, i.e., no empty feature model configurations are allowed.

Table 1
CSP derived from the feature model in Figure 1. Abbreviated variable names are used, e.g., survey (s).

ID	Description
c_0	$s = true$
c_1	$p \leftrightarrow s$
c_2	$t \rightarrow s$
c_3	$st \rightarrow s$
c_4	$q \leftrightarrow s$
c_5	$q \leftrightarrow (m \vee mm)$
c_6	$p \leftrightarrow (l \wedge \neg n \vee \neg l \wedge n)$
c_7	$\neg(n \wedge t)$
c_8	$t \rightarrow st$

Based on this example CSP (Table 1), we now introduce the concept of a *feature model configuration task* (see Definition 1) and a corresponding *feature model configuration* (see Definition 2) [4].

Definition 1. An *FM configuration task* is defined as a constraint satisfaction problem (F, C), where F is a set of Boolean variables (features) f_i (with domain (f_i) = {true, false}), and $C = REQ \cup KB$ is a set of constraints. Here, $KB = \{c_0 \ldots c_n\}$ represents a set of domain constraints, and $REQ = \{c_{n+1} \ldots c_k\}$ represents a set of user requirements.

In our example, $KB = \{c_0 \ldots c_8\}$ (see Table 1) and $REQ = \{c_9 : t = true\}$, meaning that the user has specified ABtesting (t) to be included in the final configuration.

Definition 2. An *FM configuration* $CONF = \{f_1 = a(f_1), f_2 = a(f_2), \ldots, f_k = a(f_k)\}$ is a set of variable assignments, where $a(f_i)$ is the value assigned to the variable (feature) f_i. A configuration $CONF$ is considered consistent if $(\bigcup \{f_i = a(f_i)\} \in CONF) \cup REQ \cup KB$ is consistent (i.e., a solution exists). A configuration is complete if every variable in F has a corresponding assignment in $CONF$. Finally, $CONF$ is valid if it is both, consistent and complete.

Example 1. An example configuration could be $CONF = \{s = true, p = true, l = true, n = false, t = true, st = true, q = true, m = true, mm = false\}$.

However, there are often situations where a configuration is inconsistent with the constraints in the feature model or the feature model constraints are inconsistent resulting in a void feature model. Also, customer requirements can induce an inconsistency with the FM constraints resulting in a situation where no solution can be identified. To deal with such situations, the concepts of *conflict sets* (see Definition 3) and *diagnoses* (see Definition 4) are fundamental [4]. In the following, these two concepts are formulated with regard to a constraint set C which is inconsistent, i.e., no solution could be found for the constraints in C.

Definition 3. A conflict set is a set $\Gamma \subseteq C$ with inconsistent(Γ). A conflict set is minimal if $\neg\exists\Gamma' : \Gamma' \subset \Gamma$ and Γ' is a conflict set.

QuickXPlain For a minimal conflict set to be resolved, only one constraint needs to be deleted from the conflict set. The term *minimal conflict set* is often used synonymously to the term *minimal unsatisfiable subset* (MUS) [31]. Minimal conflict sets can be determined on the basis of the QuickXPlain algorithm [22] which determines one minimal conflict set (Γ) at a time. Given a set of constraints $C = \{c_1, .., c_k, c_{k+1}, .., c_n\}$ (k is assumed to be $\frac{n}{2}$), if $\{c_1, .., c_k\}$ is inconsistent, the QuickXPlain conflict set search will focus on $\{c_1, .., c_k\}$ and immediately omit $\{c_{k+1}, .., c_n\}$. In many scenarios, all minimal conflict sets need to be determined. In such a situation, QuickXPlain needs to be combined with a corresponding hitting set directed acyclic graph based approach which helps to determine all minimal conflict sets in a systematic fashion [32].

Definition 4. A diagnosis $\Delta \subseteq C$ fulfills: consistent($C - \Delta$). Δ is minimal if $\neg\exists\Delta' : \Delta' \subset \Delta$ and Δ' is a diagnosis.

FastDiag In each case, a minimal diagnosis consists of a set of constraint which – if deleted from C – assure that $C - \Delta$ is consistent. The term *minimal diagnosis* of often used synonymously to the term *minimal correction subset* (MCS). Furthermore, the complement of a MCS, i.e., $C - MCS$, is denoted as *maximal satisfiable subset* (MSS) [31]. Minimal diagnoses (Δ) can be determined with the FastDiag algorithm [24] which is similar to QuickXPlain in terms of the used divide-and-conquer based search strategy. Given a set of constraints $C = \{c_1, .., c_k, c_{k+1}, .., c_n\}$ (k is assumed to be $\frac{n}{2}$), if $\{c_1, .., c_k\}$ is *consistent*, the FastDiag diagnosis search will focus on $\{c_{k+1}, .., c_n\}$ and immediately omit $\{c_1, .., c_k\}$.

In the following, we show how QuickXPlain can be applied to generate *explanations* in FM development, maintenance, and configuration contexts. Furthermore, we show how FastDiag can be applied to generate *repairs* to recover from unintended (often inconsistent) situations. Note that both algorithms are standard algorithms in the context of conflict detection and diagnosis. For related algorithmic/implementation details we refer to [22, 24].

4. Explanations in FM Development and Configuration

We now introduce a schema of how to apply QuickXPlain [22] and FastDiag [24] for creating explanations and to propose corresponding repair actions where needed. In this context, QuickXPlain(α, β) is assumed to return a minimal conflict set Γ where α represents a set of constraints that can be used for explanation purposes and β represents the background knowledge, i.e., a set of constraints assumed to be correct. Furthermore, FastDiag(ϵ, β) is assumed to return a minimal diagnosis Δ where ϵ represents a set of constraints to be used for diagnosis purposes and β again represents the background knowledge. In our discussion, we distinguish between the phases of (1) *feature model development* (where analysis operations and corresponding explanations play a major role) and (2) *feature model configuration* where users are building their own configurations on the basis of the configuration model (knowledge base) defined by the feature model.

Feature Model Development The major focus of feature model development is to identify the set of features relevant for describing the variability properties of the underlying domain and to integrate constraints that specify relevant commonality and variability properties. In the context of FM development and maintenance, analysis operations play a major role in terms of assuring well-formedness properties of the created models [3]. Indicating the violation of a well-formedness rule defined by an analysis operation is of enormous help [33]. In this context, explanations can help to further advance the state of practice by supporting more in-depth insights into the reasons of the violation of a well-formedness rule.

While being aware of further analysis operations, we provide a selected set of operations specifically in the need of a constraint solver (configurator) support (see Table 2). Analysis operations in the need of such a reasoning support are also in the need of explanations that help to better understand the (negative) outcome of an analysis operation and to trigger repair actions [4]. Each entry in Table 2 consists of the description of the analysis operation (formulated as a corresponding question), a corresponding explanation (*why not*?), and a repair in the case that the analysis operation provides a negative answer.

Table 2
Explanations for FM analysis operations – *cons/incons* are abbreviations of *consistent/inconsistent*.

id	question (analysis operation)	explanation (why not?)	repair
1	is satisfiable(KB)?	$\Gamma \subseteq KB : incons(\Gamma)$	$\Delta \subseteq KB : cons(KB - \Delta)$
2	is life($f_i \in F, KB$)?	$\Gamma \subseteq KB : incons(\Gamma \cup \{f_i = true\})$	$\Delta \subseteq KB : cons(KB - \Delta \cup \{f_i = true\})$
3	is optional($f_i \in F, KB$)?	$\Gamma \subseteq KB : incons(\Gamma \cup \{f_i = false\})$	$\Delta \subseteq KB : cons(KB - \Delta \cup \{f_i = false\})$
4	is irredundant($c_i \in KB, KB$)?	$c_i \notin \Gamma \subset KB : incons(\Gamma \cup \overline{KB})$	$\Gamma \subset KB : incons(\Gamma, \overline{KB})$
5	is generalization(KB_g, KB_s)?	$\Gamma \subseteq KB_g : incons(KB_g \cup \overline{KB_s})$	$\Delta \subset KB_g : cons((KB_g - \Delta) \cup \overline{KB_s})$
6	is satisfiable($CONF, KB$)?	$\Gamma \subseteq CONF : incons(\Gamma \cup KB)$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup KB)$

The analysis operations included in Table 2 are the following:

(1) $issatisfiable(KB)$? helps to figure out if at least one solution can be identified for the given feature model (represented by the knowledge base KB). If no solution can be identified (see also Table 3), i.e., the feature model is *void*, explanations (in terms of minimal conflict sets) Γ can be provided which represent minimal subsets of constraints which are inconsistent [4, 34]. A corresponding repair can be proposed by a diagnosis Δ that represents a set of constraints in KB that – if deleted from KB – help to assure the consistency of the remaining constraints in KB. For determining Γ and Δ, QuickXPlain(KB) and FastDiag(KB) need to be activated. In the example of Table 3, there exists only one minimal explanation (Γ_1) and four related diagnosis Δ_i, i.e., alternative repair options for restoring consistency.

Table 3
Explanation of a void FM (constraint c_π is assumed to be new) with corresponding repair options $\Delta_1..\Delta_4$.

constraint	c_0	c_1	c_4	$c_\pi : \neg(p \wedge q)$
Γ_1	×	×	×	×
Δ_1	×	–	–	–
Δ_2	–	×	–	–
Δ_3	–	–	×	–
Δ_4	–	–	–	×

(2) $is\ life(f_i \in F, KB)$? helps to figure out if at least one FM configuration can be created where the variable (feature) f_i is included [3, 4]. If no such configuration is possible (see Table 4), explanations (in terms of minimal conflict sets) Γ can be provided which represent minimal sets of constraints in KB which are inconsistent with $\{f_i = true\}$. If we want f_i to be true, we need to calculate a diagnosis Δ which represents a minimal constraint set to be deleted from KB to assure that the remaining constraints in KB allow the inclusion of f_i. In our example FM, if the feature $ABtesting(t)$ would have been defined as mandatory, the feature $nolicense(n)$ is not life. An explanation is $\{c_0, c_1, c_2, c_\pi\}$.

Table 4
Explanation of dead feature n (constraint c_π is assumed to be new) with corresponding repair options $\Delta_1..\Delta_4$.

constraint	c_0	c_1	c_2	$c_\pi : \neg(p \wedge t)$
Γ_1	×	×	×	×
Δ_1	×	–	–	–
Δ_2	–	×	–	–
Δ_3	–	–	×	–
Δ_4	–	–	–	×

(3) $is\ optional(f_i \in F, KB)$? helps to analyze if a feature can really be regarded as an optional

feature, i.e., there should be configurations where the feature is excluded [3, 4]. If the feature model represented by KB does not allow the exclusion of f_i (see Table 5), explanations (in terms of minimal conflict sets Γ) can be provided which represent minimal subsets of constraints inconsistent with the exclusion of f_i. A corresponding diagnosis Δ (which represents a resolution of all minimal conflict sets), is a minimal set of constraints that need to be deleted from KB to assure the possibility of having configurations with f_i excluded. In our example FM, if the feature $ABtesting(t)$ would have been defined as mandatory, the feature $statistics(st)$ is not optional anymore. An explanation is $\{c_0, c_2', c_8\}$.

Table 5
Explanation of false optional feature st (c_2‘ is an adaptation of c_2) with corresponding repair options $\Delta_1..\Delta_3$.

constraint	c_0	$c_2' : t \leftrightarrow s$	c_8
Γ_1	$\times$	$\times$	$\times$
Δ_1	$\times$	$-$	$-$
Δ_2	$-$	$\times$	$-$
Δ_3	$-$	$-$	$\times$

(4) the redundancy of a constraint c_i can be explained by the fact that c_i is not part of an irredundant constraint set Γ determined for KB. Making KB irredundant means to delete those elements from KB which are not in Γ [35]. In our example, we would define the statistics feature (st) as mandatory, i.e., $st \leftrightarrow s$, the requires constraint c_8 can be regarded as redundant.

(5) if a generalization between the feature models KB_g (the more general one allowing more solutions [4]) and KB_s (the more specific one representing a solution-wise subset of KB_g) does not exist, an explanation can indicate those elements (constraints) responsible for a situation where KB_g does not represent a superset of KB_s. A related diagnosis Δ indicates those elements that need to be deleted from KB_g such that the mentioned superset property is fulfilled.

(6) if a configuration $CONF$ is inconsistent with the constraints in KB, an explanation indicates constraints (i.e., variable value assignments) in $CONF$ that induce an inconsistency with KB. Furthermore, a diagnosis Δ indicates minimal sets of constraints (assignments) to be deleted from $CONF$ such that consistency with the constraints in KB can be restored [36].

Table 6 provides an overview of how the discussed analysis operations (see Table 2) can be supported by the conflict detection algorithm QuickXPlain [22] and the diagnosis algorithm FastDiag [24]. Note that these algorithms could also be replaced by other alternatives if conflict set and diagnosis minimality is assured by those algorithms. For example, if $issatisfiable(KB)$ is not fulfilled (i.e., the feature model does not allow the identification of a solution), Γ=QuickXPlain($KB, \emptyset$) returns a minimal set of elements of KB as an explanation indicating that those elements are in conflict. Note that more than one such explanation (conflict) could exist in KB. If this is the case, a hitting set directed acyclic graph (HSDAG) based approach [32] can be applied (in combination with QuickXPlain) to identify all related explanations (minimal conflict sets). If a repair proposal is requested, FastDiag can determine a corresponding diagnosis Δ which expresses a minimal set of elements to be deleted from KB in order to be able to restore consistency in KB. In a similar fashion, QuickXPlain and FastDiag can be activated in the other mentioned explanation scenarios.

Table 6
Explanations for FM analysis operations using QuickXPlain [22] and FastDiag [24].

id	question (analysis operation)	explanation (why not?)	repair
1	is satisfiable(KB)?	QuickXPlain($KB, \emptyset$)	FastDiag($KB, \emptyset$)
2	is life($f_i \in F, KB$)?	QuickXPlain($KB, \{f_i = true\}$)	FastDiag($KB, \{f_i = true\}$)
3	is optional($f_i \in F, KB$)?	QuickXPlain($KB, \{f_i = false\}$)	FastDiag($KB, \{f_i = false\}$)
4	is irredundant($c_i \in KB, KB$)?	$c_i \notin$ QuickXPlain($KB, \overline{KB}$)	QuickXPlain($KB, \overline{KB}$)
5	is generalization(KB_g, KB_s)?	QuickXPlain($KB_g, \overline{KB_s}$)	FastDiag($KB_g, \overline{KB_s}$)
6	is satisfiable($CONF, KB$)?	QuickXPlain($CONF, KB$)	FastDiag($CONF, KB$)

Feature Model Configuration Feature model configuration supports the identification of a decision regarding the inclusion or exclusion of specific features in a configuration. In the context of such (often interactive) configuration sessions, explanations can help the user of a feature model configurator to better understand the reason of a specific feature inclusion or exclusion but also as to why specific features have not been included (i.e., an explanation of the counterfactual case) [37, 12, 17]. Example questions that need to be answered (i.e., explained to users) are shown in Table 7.

Table 7
Explanations for FM configuration settings – *cons/incons* are abbreviations of *consistent/inconsistent*.

id	question	explanation	repair
1	why($\{f_i = true\} \in CONF, KB \cup REQ$)?	$\Gamma \subseteq REQ \cup KB : incons(\Gamma \cup \{f_i = false\})$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup \{f_i = false\})$
2	why not($\{f_i = true\} \in CONF, KB \cup REQ$)?	$\Gamma \subseteq REQ \cup KB : incons(\Gamma \cup \{f_i = true\})$	$\Delta \subseteq CONF : cons(CONF - \Delta \cup \{f_i = true\})$
3	why not(REQ, KB)?	$\Gamma \subseteq REQ : incons(REQ \cup KB)$	$\Delta \subseteq REQ : cons(REQ - \Delta \cup KB)$

Table 8 provides a complete listing of the QuickXPlain and FastDiag activations used in the explanation/repair scenarios shown in Table 7.

Table 8
Explanations for FM configuration settings using QuickXPlain [22] and FastDiag [24].

id	question	explanation	repair
1	why($\{f_i = true\} \in CONF, KB \cup REQ$)?	QuickXPlain($KB \cup REQ, \{f_i = false\}$)	FastDiag($KB \cup REQ, \{f_i = false\}$)
2	why not($\{f_i = true\} \in CONF, KB \cup REQ$)?	QuickXPlain($KB \cup REQ, \{f_i = true\}$)	FastDiag($KB \cup REQ, \{f_i = true\}$)
3	why not(REQ, KB)?	QuickXPlain(REQ, KB)	FastDiag(REQ, KB)

(1) given a configuration $CONF$ which includes a variable value assignment $f_i = true$, a corresponding (counterfactual) explanation identifies those elements (the minimal conflict set Γ) in $REQ \cup KB$ which induce an inconsistency with the negation of $f_i = true$ and are therefore responsible for the inclusion of feature f_i [12, 38]. The corresponding activation of QuickXPlain to determine the minimal conflict set Γ is shown in Table 8. In this setting, QuickXPlain identifies a minimal set of constraints in $REQ \cup KB$ that induce an inconsistency with the constraint $f_i = true$ [12]. In this scenario, FastDiag can be applied to identify a diagnosis Δ consisting of elements from $CONF$ which have to be deleted/adapted in such a way that $f_i = true$ becomes part of $CONF$.

(2) given a configuration including a variable value assignment $f_i = false$, a counterfactual explanation identifies those elements (the minimal conflict set Γ) in $REQ \cup KB$ which induce an inconsistency with the negation of $f_i = false$ and are therefore responsible for the exclusion of feature f_i. A corresponding diagnosis Δ determined by FastDiag would indicate those elements to be adapted[1] in $CONF$ such that the inclusion of f_i becomes possible.

(3) if the user requirements in REQ do not allow the determination of a solution, an explanation will indicate a subset of requirements that induce an inconsistency with KB [22]. A diagnosis Δ includes a set of requirements that need to be deleted/adapted to restore consistency. If we assume the existence of the user requirements $REQ = \{c_9, c_{10}, c_{11}\}$, the corresponding explanations are Γ_1 and Γ_2 with the diagnoses Δ_1 and Δ_2 indicating a way to resolve the conflicts in Γ_1 and Γ_2. In this context, Δ_1 is a singleton diagnosis, i.e., a diagnosis consisting of the minimal number of elements needed to resolve all existing conflicts.

[1]Due to the Boolean nature of feature model variables, an adaptation/deletion of a feature always means either to switch from feature inclusion to exclusion or vice-versa.

Table 9
Explanation of inconsistent user requirements $\{c_9, c_{10}, c_{11}\}$ with corresponding repair options $\Delta_1..\Delta_2$.

requirements	$c_9 : t = true$	$c_{10} : n = true$	$c_{11} : st = false$
Γ_1	$\times$	$\times$	$-$
Γ_2	$\times$	$-$	$\times$
Δ_1	$\times$	$-$	$-$
Δ_2	$-$	$\times$	$\times$

5. Threats to Validity

In this paper, we have shown how to apply consistency-based conflict detection and diagnosis to determine corresponding explanations and repair actions. We are aware of the fact that there are related open issues, for example, instead of deleting constraints from an envisioned superset KB_g to assure an entailment relationship with the specialized feature model KB_s, we could also think about adding additional constraints to KB_s. We regard such open repairs as being beyond the scope of this paper, however, this appears to be a challenging topic for future work. In this paper, we did not conduct performance evaluations of the different explanation and repair settings, however, the used algorithms are well-established and have shown to be applicable in various configuration scenarios which we took as a reason for not including another performance evaluation of QuickXPlain and FastDiag.

6. Conclusions

In this paper, we have shown how to apply conflict detection and diagnosis for the creation of consistency-based explanations and corresponding repair actions. We have identified two major application scenarios which are (1) the generation of explanations and repair actions in the context of supporting analysis operations in FM development and maintenance and (2) the support of users in interactive FM configuration sessions. We have shown how to apply/integrate QuickXPlain as standard algorithm for the detection of minimal conflict sets and FastDiag as an algorithm for the identification of minimal diagnoses. In FM development and maintenance, *why not*? explanations help to understand why specific well-formedness rules defined by analysis operations, fail. In FM configuration, *why not*? and *why*? explanations can be used to explain the inclusion or exclusion of specific features but also the reason as to why no solution exists. Open issues for future work include user studies to better understand in which context to provide which explanation or repair, an analysis of further explanation types to be included in feature model design and configuration, and an analysis of the improvement potentials of existing conflict detection and diagnosis algorithms using machine learning techniques.

Declaration on Generative AI

The authors used ChatGPT for language refinement and improving readability. All AI-generated suggestions were carefully reviewed and edited by the authors, who take full responsibility for the content of this publication.

References

[1] M. Acher, P. Temple, J.-M. Jézéquel, J. A. Galindo, J. Martinez, T. Ziadi, VaryLATEX: Learning Paper Variants That Meet Constraints, in: 12th International Workshop on Variability Modelling of Software-Intensive Systems, VAMOS '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 83–88. doi:10.1145/3168365.3168372.

[2] S. Apel, C. Kästner, An overview of feature-oriented software development, Journal of Object Technology 8 (2009) 49–84.

[3] D. Benavides, S. Segura, A. Ruiz-Cortes, Automated analysis of feature models 20 years later: A literature review, Information Systems 35 (2010) 615–636.

[4] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-Driven Design, Analysis and Applications, Springer, 2024. doi:10.1007/978-3-031-61874-1.

[5] J. Galindo, J. Horcas, A. Felfernig, D. Fernandez-Amoros, D. Benavides, Flama: A collaborative effort to build a new framework for the automated analysis of feature models, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC '23, ACM, New York, NY, USA, 2023, pp. 16–19. doi:10.1145/3579028.3609008.

[6] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peterson, Feature-oriented Domain Analysis (FODA) – Feasibility Study, TechnicalReport CMU – SEI-90-TR-21 (1990).

[7] J. Gu, P. Purdom, J. Franco, B. Wah, Algorithms for the satisfiability (sat) problem: A survey, in: DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1996, pp. 19–152.

[8] D. Benavides, P. Trinidad, A. Cortés, Using constraint programming to reason on feature models, in: W. Chu, N. Juzgado, W. Wong (Eds.), Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering (SEKE'2005), Taipei, Taiwan, Republic of China, July 14-16, 2005, 2005, pp. 677–682.

[9] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V.-M. Le, K. Pilsl, M. Enzelsberger, T. N. T. Tran, An overview of machine learning techniques in constraint solving, J Intell Inf Syst 58 (2022) 91–118. doi:10.1007/s10844-021-00666-5.

[10] A. Felfernig, D. Benavides, J. Galindo, F. Reinfrank, Towards Anomaly Explanation in Feature Models, in: ConfWS-2013: 15th International Configuration Workshop (2013), volume 1128, 2013, pp. 117–124.

[11] A. Felfernig, M. Schubert, S. Reiterer, Personalized diagnosis for over-constrained problems, in: 23rd International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, pp. 1990–1996.

[12] A. Felfernig, D. Garber, V.-M. Le, S. Lubos, Causality-based explanations for feature model configuration, in: 19th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS '25, ACM, New York, NY, USA, 2025, pp. 86–90. doi:10.1145/3715340.3715438.

[13] S. Lubos, T. N. T. Tran, A. Felfernig, S. Polat Erdeniz, V.-M. Le, LLM-generated Explanations for Recommender Systems, in: 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '24, ACM, New York, NY, USA, 2024, pp. 276–285. doi:10.1145/3631700.3665185.

[14] S. Lubos, M. Gartner, A. Felfernig, R. Willfort, Leveraging LLMs to Explain the Consequences of Recommendations, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, ACM, 2025, pp. 318–322. doi:10.1145/3699682.3728328.

[15] A. Felfernig, M. Wundara, T. Tran, S. Polat-Erdeniz, S. Lubos, M. E. Mansi, D. Garber, V. Le, Recommender systems for sustainability: overview and research issues, Frontiers in Big Data 6 (2023). doi:10.3389/fdata.2023.1284511.

[16] M. Hentze and T. Pett and T. Thüm and I. Schaefer, Hyper Explanations for Feature-Model Defect Analysis, in: 15th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS'21, Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3442391.3442406.

[17] D. Kramer, C. Sauer, T. Roth-Berghofer, Towards explanation generation using feature models in software product lines, in: 9th Workshop on Knowledge Engineering and Software Engineering, CEUR, 2013, pp. 13–23. URL: https://ceur-ws.org/Vol-1070/.

[18] L. Ochoa, O. González-Rojas, T. Thüm, Using decision rules for solving conflicts in extended feature models, in: ACM SIGPLAN International Conference on Software Language Engineering, Association for Computing Machinery, New York, NY, USA, 2015, pp. 149–160. doi:10.1145/2814251.2814263.

[19] N. Tintarev, J. Masthoff, A survey of explanations in recommender systems, ICDEW '07, IEEE

Computer Society, USA, 2007, pp. 801–810. doi:10.1109/ICDEW.2007.4401070.

[20] T. Tran, S. Polat Erdeniz, A. Felfernig, S. Lubos, M. El Mansi, V. Le, Less is more: Towards sustainability-aware persuasive explanations in recommender systems, in: Proceedings of the 18th ACM Conference on Recommender Systems, RecSys '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 1108–1112. doi:10.1145/3640457.3691708.

[21] T. Tran, A. Felfernig, V. Le, An overview of consensus models for group decision-making and group recommender systems, User Modeling and User-Adapted Interaction 34 (2023) 489–547. doi:10.1007/s11257-023-09380-z.

[22] U. Junker, QuickXPlain: preferred explanations and relaxations for over-constrained problems, in: 19th National Conference on Artifical Intelligence, AAAI'04, AAAI Press, 2004, pp. 167–172.

[23] O. A. Tazl, C. Tafeit, F. Wotawa, A. Felfernig, DDMin versus QuickXplain - An Experimental Comparison of two Algorithms for Minimizing Collections, in: R. Peng, C. E. Pantoja, P. Kamthan (Eds.), 34th International Conference on Software Engineering and Knowledge Engineering, KSI Research Inc., 2022, pp. 481–486. doi:10.18293/SEKE2022-172.

[24] A. Felfernig, M. Schubert, C. Zehentner, An efficient diagnosis algorithm for inconsistent constraint sets, AI for Engineering Design, Analysis, and Manufacturing (AIEDAM) 26 (2012) 53–62.

[25] V. Le, C. Silva, A. Felfernig, D. Benavides, J. Galindo, T. Tran, FastDiagP: an algorithm for parallelized direct diagnosis, AAAI'23/IAAI'23/EAAI'23, AAAI Press, 2023. doi:10.1609/aaai.v37i5.25792.

[26] C. Gomes, H. Kautz, A. Sabharwal, B. Selman, Satisfiability Solvers, Handbook of Knowledge Representation (2008) 89–134.

[27] M. Mendonça, A. Wasowski, K. Czarnecki, SAT-based analysis of feature models is easy, in: D. Muthig, J. McGregor (Eds.), Software Product Lines, 13th International Conference, SPLC 2009, San Francisco, California, USA, August 24-28, 2009, Proceedings, volume 446 of *ACM International Conference Proceeding Series*, ACM, 2009, pp. 231–240.

[28] V. Myllärniemi, J. Tiihonen, M. Raatikainen, A. Felfernig, Using Answer Set Programming for Feature Model Representation and Configuration, in: 16th International Workshop on Configuration, CEUR, Novi Sad, Serbia, 2014, pp. 1–8.

[29] A. Falkner, G. Friedrich, A. Haselböck, G. Schenner, H. Schreiner, Twenty-five years of successful application of constraint technologies at siemens, AI Mag. 37 (2016) 67–80. doi:10.1609/aimag.v37i4.2688.

[30] T. W. F. Rossi, P. van Beek, Handbook of Constraint Programming, Elsevier, 2006.

[31] S. Gupta, B. Genc, B. O'Sullivan, Explanation in constraint satisfaction: A survey, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4400–4407. doi:10.24963/ijcai.2021/601, survey Track.

[32] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57–95.

[33] M. Kowal, S. Ananieva, T. Thüm, Explaining anomalies in feature models, SIGPLAN Not. 52 (2016) 132–143. doi:10.1145/3093335.2993248.

[34] R. Bakker, F. Dikker, F. Tempelman, P. Wogmim, Diagnosing and solving over-determined constraint satisfaction problems, in: Proceedings of IJCAI-93, Morgan Kaufmann, 1993, pp. 276–281.

[35] V. Le, A. Felfernig, M. Uta, T. Tran, C. Silva, WipeOutR: automated redundancy detection for feature models, in: 26th ACM International Systems and Software Product Line Conference - Volume A, SPLC '22, ACM, New York, NY, USA, 2022, pp. 164–169. doi:10.1145/3546932.3546992.

[36] J. White, D. Benavides, D. Schmidt, P. Trinidad, B. Dougherty, A. Ruiz-Cortes, Automated diagnosis of feature model configurations, Journal of Systems and Software 83 (2010) 1094–1107.

[37] C. Dubslaff, K. Weis, C. Baier, S. Apel, Causality in configurable software systems, in: 44th International Conference on Software Engineering, ICSE '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 325–337. doi:10.1145/3510003.3510200.

[38] G. Friedrich, Elimination of spurious explanations, in: 16th European Conference on Artificial Intelligence, ECAI'04, IOS Press, NLD, 2004, pp. 813–817.

From 4GL Spreadsheet Computations to Constraint Model Definitions - A Development Process

Boi Schaefer[1,†], Lothar Hotz[1,*,†] and Kirsten David[2]

[1]Hamburger Informatik Technologie-Center e.V., Vogt-Kölln-Straße 30, 22527 Hamburg
[2]Universität Bielefeld, Universitätsstraße 25, 33615 Bielefeld

Abstract

In this paper, we present an approach for mapping variables and equations given in a tabular application of a 4GL spreadsheet (fourth generation programming language) to a constraint model. We start with a spreadsheet given for a specific application in the area of modernization of buildings. The spreadsheet computes for landlords and tenants the increase of the rent after a modernization is done. These computations shall be part of a platform that enables computation and negotiation of building modernization endeavor. This approach is particularly relevant for configuration systems where domain experts typically express configuration knowledge through spreadsheets, and constraint-based configuration platforms require declarative constraint models. Our development process bridges this gap by providing a systematic methodology to transform 4GL spreadsheet computations into maintainable constraint models, enabling better integration of domain expertise into configuration platforms.

Keywords

4GL Tables, PyChoco, Constraint Model, OR-Tools, SCREAMER

1. Introduction

Constraint models enable computations by declaratively specifying variables with their domains and constraints between them, which are then processed by a constraint solver tool [1]. An application problem is defined through variables with appropriate domains and constraints between the variables as a Constraint Satisfaction Problem (CSP) [2]. One way to provide the necessary computations of an application task to developers of a constraint model is to give a formal representation of variables and equations in a spreadsheet from a fourth-generation language (4GL) [3][1] such as Microsoft EXCEL. This challenge is particularly prevalent in configuration systems, where domain experts often express configuration knowledge through familiar spreadsheet interfaces, while the underlying platform requires formal constraint models to enable automated reasoning, optimization, and validation. Thus, in a platform, the spreadsheet interface is replaced by a user interface and the computations are done with a constraint system in the backend.

While several approaches exist for converting spreadsheets to other formats, existing research has predominantly focused on data migration and format transformation rather than constraint model generation. For example, Harris and Gulwani [4] develop methods for automatically restructuring spreadsheet tables, while Shigarov et al. [5] present rule-based approaches for converting arbitrary spreadsheet tables into relational database formats. These approaches demonstrate sophisticated data transformation capabilities but primarily address format conversion and data migration scenarios, preserving data content while losing the underlying computational logic embedded in spreadsheet formulas. Direct code generation techniques can maintain computational aspects but sacrifice the

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

*Corresponding author.
†These authors contributed equally.
boi.schaefer@hitec-hamburg.de (B. Schaefer); lothar.hotz@hitec-hamburg.de (L. Hotz); kirsten.david@uni-bielefeld.de (K. David)
https://hitec-hamburg.de (B. Schaefer); https://kogs-www.informatik.uni-hamburg.de/~hotz/ (L. Hotz); https://www.uni-bielefeld.de/fakultaeten/rechtswissenschaft/forschung/forschungsstellen/fir/intelmod/ (K. David)
0000-0001-7370-7726 (L. Hotz)

[1]https://en.wikipedia.org/wiki/Fourth-generation_programming_language

declarative nature essential for constraint-based reasoning. Rule-based systems, while capable of capturing logical structures, struggle with complex mathematical relationships and multidirectional computation capabilities provided by constraint-based configuration systems.

Converting spreadsheet computations to constraint models presents unique challenges that extend beyond traditional data transformation approaches. Main questions arise such as: What are the variables, domains, and constraints? How are variables grouped? What are input variables? Which are computed? How are the computations being represented, with integer or mixed-integer programming? A further aspect is the dynamic generation of the CSP, depending on the input values of the user which might occur in the spreadsheet. Additionally, constraint solvers are typically back-end libraries without user interfaces, which requires integration with front-end components for user interaction.

In this paper, we present our approach of a development process (Section 3) that leads from a spreadsheet to a constraint model. First, we describe in Section 2 our application in the area of building modernization. This application is used throughout the paper for demonstrating our approach, especially, because the application comes with a spreadsheet developed by a domain expert. This task is also relevant for configuration tasks as the spreadsheet computation may also be used for representing configuration knowledge, as well as the dynamic aspect of the CPS generation relates to configuration. Then, we elaborate a pre study with a programming language and constraint solver that enable fast developments. This step has answered the question, if in principle the spreadsheet can be mapped to a constraint solver (see Section 4). In the next step, we select an appropriate constraint solver (see Section 5). The modeling and implementation was done with the selected OR-Tools solver and we analyzed the use of integer modeling vs. mixed-integer programming (see Section 6). Section 7 presents the overall architecture and interfaces, as Section 8 discusses the approache and Section 9 concludes.

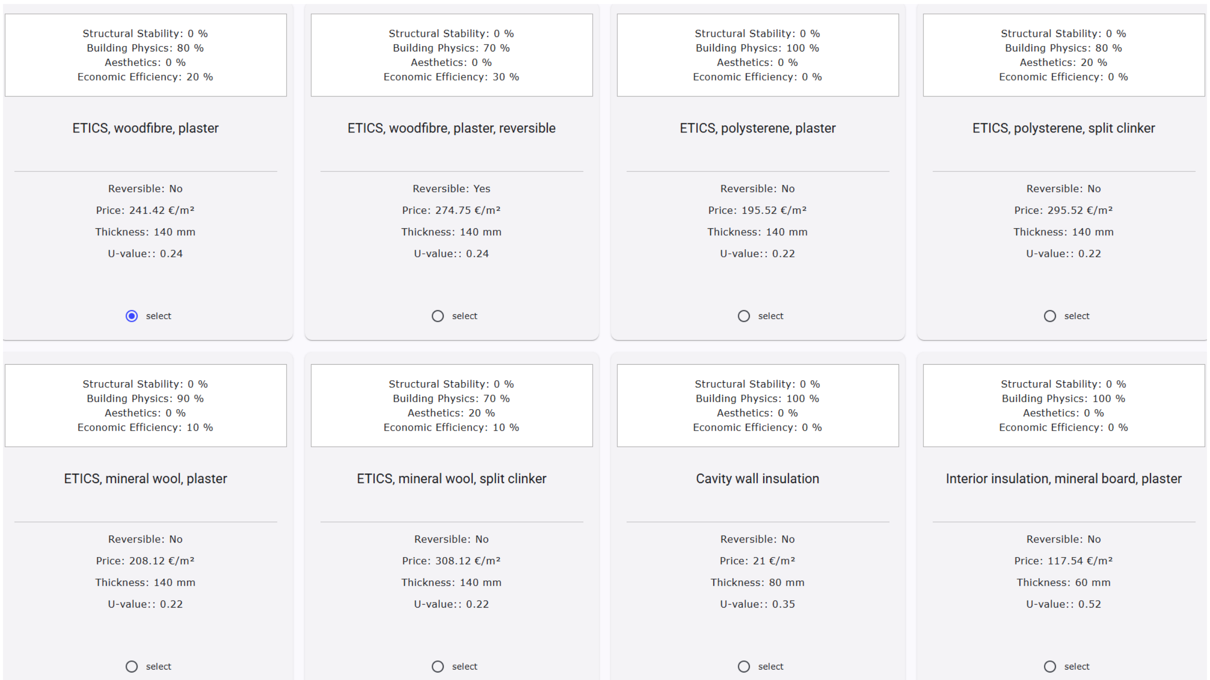


Figure 1: Catalog of construction measures for modernization with percentage indications.

2. Application Description

This research is part of the Intelligent Modernization Platform (IntelMOD) project, which aims to offer a tool to tenants and landlords to initiate modernization negotiations based on eight building functions: Stability (Standsicherheit), Moisture Protection (Feuchteschutz), Thermal Insulation (Wärmeschutz), Sound Insulation (Schallschutz), Fire Protection (Brandschutz), Daylight Access (Tageslicht), Aesthetics

(Ästetik), and Ecological and Economical Efficiency (Wirtschaftlichkeit) which the so called *Functional Cost Splitting (FK)* provides [6]. The FK is a recommended course of action for determining fair and traceable rent adjustments following energy-efficient modernization. It can be used in practice by landlords and tenants as a transparent basis for communication and calculation for object- and even measure-specific rent increases for modernizations to avoid or resolve disputes while simultaneously optimizing the degree of sustainability of the relevant measures. With the framework of FK, the essential step towards organizing the necessary information, data, and regulations of modernization knowledge has already been taken.

2.1. Functional Cost Splitting

The Functional Cost Splitting of the IntelMOD platform determines the rent increase relevant and non-rent increase relevant cost components of modernization measures according to German rental law in three successive steps.

2.1.1. Step I: Project Data and Weighting of Improvements Planned with the Measure

The first step captures project data and defines a percentage weighting of the eight basic functions of a building component: Stability, Moisture Protection, Thermal Insulation, Sound Insulation, Fire Safety, Daylight Access, Aesthetics, and Ecological and Economical Efficiency. Initially, basic building data including total floor area of the building, the size of the affected apartment and total costs of the modernization measure are entered. The system distinguishes between three states: the ACTUAL state before modernization, the planned MOD state of the modernization measure, and the resulting NEW state after implementation.

Function Level Structure

ACTUAL State - Function Level 1 (Input): At the first level, four main functions are weighted: Stability, Building Physics, Aesthetics, Ecological and Economical Efficiency

ACTUAL State - Function Level 2 (automatically calculated): Computations for building physics weighting which is automatically divided equally among the five sub-functions: Moisture Protection, Thermal Insulation, Sound Insulation, Fire Safety, and Daylight Access.

MOD State - (Input): For modernization planning, all eight individual functions are directly weighted according to specific modernization objectives.

NEW State - (automatically calculated): The target state is calculated as a weighted average between ACTUAL and MOD state for all eight individual functions and represents the intended function distribution after modernization.

2.1.2. Step II: Building Data and Assessment of Improvements Actually Achieved with the Measures

In the second step, for each of the eight individual functions, it is entered whether an actual improvement is achieved through the planned modernization measure. This assessment is made through binary Yes/No decisions for each function as further input variables.

2.1.3. Step III: Building Data and Assessment of the Fulfillment Grade (Under-fulfillment) of Functions of the Existing Building Component for Estimation of Required Maintenance Costs

The third step captures the current condition of the existing building component through detailed assessment. For each of the eight individual functions, the fulfillment grade of the ACTUAL state is determined by entering existing damages and their extent and effects on the respective function as next input variables. The completion grade assessment is performed by entering percentages of the damage condition in relation to the affected area and functional impairment. The system automatically calculates the maintenance-relevant cost component that is attributable to the repair of existing defects. This calculation is only done for the selected damage assessment, that is, the fields in the spreadsheet are only computed if other fields are selected by the user (*dynamic field activation*), hence, dynamic constraint creation (see below).

2.1.4. Final Cost Calculation

The system automatically calculates all relevant cost indicators based on the inputs from the three steps. From the determined rent increase relevant cost component, the annual and monthly cost components are calculated for both the entire building and per square meter and for the affected apartment. The automatic calculations include proportional distribution by floor area and consider the function weightings and fulfillment grades determined in the previous steps. The system ensures that rent increase relevant components and non-rent increase relevant components are correctly separated.

2.1.5. Representation of the FK

The FK was developed in a 4GL spreadsheet with the tool EXCEL by Kirsten David [6]. Users can use these spreadsheet for computing the modernization cost as depicted above. There by, they provide the input variables. Through 4GL equations such as `=IF('Bewertung Verbesserung NEU'!L26='Bewertung Verbesserung NEU'!J24; 'Definition Funktions-Soll NEU'!N41; "keine Verbesserung")` or `=(R25+R33)/2` computations are executed by the spreadsheet. The MOD state is depicted with weighted values (s.a). Of course those might be difficult to provide, hence, a catalog of construction measures for modernization is given to the user where each construction measure maps to specific weights for the functions (see Figure 1). In total, the calculations of the FK are specified as requirements using EXCEL spreadsheet tables. The EXCEL spreadsheet contains not only prescribed equations but also configuration-dependent formulas. Different building configurations require different calculation paths. For instance, buildings with certain damage patterns activate additional worksheets with specific formulas, while buildings without those damages skip these calculations entirely. This conditional logic creates a dynamic calculation structure that varies based on the input configuration.

3. Conceptual Approach

For an application in a backend of an Internet platform the spreadsheet cannot be used and has to be converted into a program. We did not consider a procedural program in a high-level programming language as implementation target, because we do not want to loose flexibility in the computations as well as ensure easy maintenance. Instead, we choose constraint programming as an implementation approach that ensures a declarative description of the problem through variables with appropriate domains, constraints, and a hidden algorithm for solving the problem implemented in a solver. Thus, this approach separates the knowledge of the calculation from the actual computation itself. Further advantages when using domains, i.e., intervals, ranges can be taken into account for input variables (e.g., weight ranges) which are computed by the solver (leading to, e.g., ranges of rent increase). Also the multidirectional feature of constraint solvers is enabled by this approach and will be used in the future for reverse computing from potential rent increase to needed modernisation construction measures.

The steps of our approach are: pre study, tool selection, CSP implementation, architecture, and user interface. They are presented in the following sections.

```
(defstruct (IST-funktionen-gewichtung-bestehendes-bauteil-fkt-ebene-1
            (:conc-name fgbb-IST-fkt-1-)
            (:print-function print-fgbb-IST-fkt-1))
  standsicherheit
  bauphysik
  aesthetik
  wirtschaftlichkeit
  summe)

(defun create-a-bestehendes-bauteil-IST-fkt-ebene-1 ()
  (let ((ss (an-integer-betweenv 0 *upperbound* "IST-fkt-1-ss"))
        (bp (an-integer-betweenv 0 *upperbound* "IST-fkt-1-pb"))
        (ae (an-integer-betweenv 0 *upperbound* "IST-fkt-1-ae"))
        (wk (an-integer-betweenv 0 *upperbound* "IST-fkt-1-wk"))
        (su (an-integer-betweenv 0 *upperbound* "IST-fkt-1-su")))
    (assert! (=v su (+v ss bp ae wk)))
    (assert! (=v su *upperbound*))
    (make-IST-funktionen-gewichtung-bestehendes-bauteil-fkt-ebene-1
     :standsicherheit ss
     :bauphysik bp
     :aesthetik ae
     :wirtschaftlichkeit wk
     :summe su)))
```

Figure 2: SCREAMER/COMMON LISP implementation for the IST/ACTUAL functions of Level 1. A `defstruct` groups variables of one row describing the actual values. A function (`create-a-bestehendes-bauteil-IST-fkt-ebene-1`) defines constraint variables and constraints of that row.

4. Pre Study

We first developed a prototypical implementation of the Functional Cost Splitting (FK) in COMMON LISP [7] using the constraint system SCREAMER [8]. Creating a machine-readable structure, the FK had to be converted into a digital and machine-readable format. For this purpose, the existing process was analyzed and translated into standardized constraints, which enable formal computation. A prototype for the FK automation was implemented based on constraints in SCREAMER, successfully realizing significant portions of the FK.

In scientific processing and optimization, spreadsheet data can often be represented using constraints. This method allows for an explicit modeling of relationships between spreadsheet expressions and variables. Variables in this context refer to parameters or functions dictated by the problem structure (e.g., weights for Stability, Thermal Insulation). Constraints, on the other hand, represent the conditions that these variables must satisfy (e.g., the sum of all values must be exactly 100%).

The grouping of variables depends on the characteristics of the problem to ensure an overview, efficient modeling, traceability, and maintainability. The use of large language models (LLMs) for creating a constraint model from the spreadsheet was not considered sustainable due to deficiencies in maintainability and scalability of the resulting code, which contained, e.g., number-based filed names such as $R23$.

To address these issues, the SCREAMER COMMON LISP implementation was chosen. This provides a powerful platform for realizing constraint models (see Figure 2 for an example for the implementation of ACTUAL functions of Level 1). A test-driven approach was employed to ensure the correctness of the implementation – particularly in cases involving mutual computations between variables.

The developed model focuses on the application of the FK and includes:

Variables: Functional parameters such as weights for Stability, Thermal Insulation, etc., i.e., the fields in the 4GL spreadsheet.

Domains: Value ranges of these variables (e.g., 0% to 100%).

Constraints: Specifications and technical requirements (e.g., the sum of all values must be exactly 100%).

The advantages of constraint modeling lie in its efficiency for solving complex problems, its adaptability, as well as the comprehensibility and verifiability of the results.

As result of the pre study, we developed a semantically meaningful grouping of variables which directly corresponds to rows in the spreadsheet. Furthermore, the identification of the input variables, constraints, and output variables was done. Through a test-driven approach, we could show the computability of the spreadsheet through a constraint solver.

5. Tool selection

We did not use the COMMON LISP as a basis for the platform implementation, because of less available programming skills in this sector. Instead, we used a more known programming language, i.e., PYTHON.

We compared the CSP solvers PyChoco [9][2], Pyomo[3], and OR tools[4]. These three solvers were selected for comparison to evaluate different approaches in constraint and optimization programming. PyChoco represented the category of specialized constraint solvers with established Java foundation and PYTHON interface. Pyomo stood for flexible mathematical optimization frameworks that support various solver backends. OR-Tools represented integrated optimization platforms that combine multiple solver types (CSP, MIP, LP) in a unified environment. Through this deliberate selection, we could systematically evaluate three different philosophies according to the defined criteria and identify the optimal solution for our FK system.

Comparison criteria are:

1. Age and Latest Version: Evaluating the maturity and current updates of each solver is essential to ensure reliability and sustainability.
2. Programming Language Support: The availability of interfaces in desired programming languages (e.g., PYTHON) was a critical factor for ease of integration.
3. Embedded vs. External Solver Connection: Whether the solver can be seamlessly integrated into existing workflows or requires external setup.
4. Versatility and Range of Functions: Assess the ability to handle various problem types, such as CSP, linear programming (LP), mixed-integer programming (MIP), and routing problems.
5. Documentation and Community Support: The availability of comprehensive documentation and active developer communities for troubleshooting and updates.
6. Literature and Resources: Access to academic papers, tutorials, and case studies that support learning and implementation.

OR tools emerged as the preferred choice cause of several advantages:

1. Regular Updates and Active Maintenance: The tool receives consistent updates, ensuring it remains current and reliable.
2. Multi-Language Support: Particularly strong support for PYTHON, which aligns with modern development trends and ease of use.
3. Powerful Integrated Solvers: OR tools offer robust solvers directly embedded within the platform, streamlining the problem-solving process.

[2] https://github.com/choco_solver/pychoco
[3] https://pyomo.readthedocs.io/en/latest/
[4] https://github.com/google/or-tools

4. Versatility in Problem Types: It supports a wide range of optimization problems, including CSP, LP, MIP, and routing challenges.
5. Extensive Documentation: Comprehensive documentation is available to guide users through implementation and troubleshooting.
6. Large Developer Community: A vibrant developer community provides active support and shares knowledge, enhancing the tool's ecosystem.
7. Special Recognition: OR tools have received significant recognition in the field, including multiple wins in the MiniZinc Challenges, a prestigious competition for CSP solvers.

In conclusion, after evaluating these criteria, OR tools were selected due to their comprehensive functionality, strong PYTHON support, active development, and proven track record in solving complex optimization problems.

```python
class ISTFunktionenGewichtungBestehendesBauteilFktEbene1:
    """
    Class representing the existing functions weighting for an existing
        building component at function level 1.
    """

    def __init__(
    self,
    standsicherheit,
    bauphysik,
    aesthetik,
    wirtschaftlichkeit,
    summe,
    ):
    # Initialize the attributes with the provided variables
    self.standsicherheit = standsicherheit
    self.bauphysik = bauphysik
    self.aesthetik = aesthetik
    self.wirtschaftlichkeit = wirtschaftlichkeit
    self.summe = summe
```

Figure 3: Part 1/2: Class definition for grouping variables. See Figure 2 for the COMMON LISP implementation.

6. OR-Tools Implementation

6.1. Technical Implementation as Constraint Model

We implemented the FK as a CSP with OR Tools. The constraint model comprises 23 fixed input fields, hence, variables, for Step I and II, complemented by a dynamic number of input fields ranging from 0 to 330 variables depending on the extent of damage assessment selected in Step III. The constraint structure follows a similar pattern with 130 fixed constraints and an additional 0 to 288 dynamic constraints that scale with the complexity of the damage evaluation process.

The primary constraints ensure mathematical consistency by requiring that function weightings in each state sum to exactly 100% and that fulfillment grades remain within the valid range of 0% to 100%. Other constraints are related to mathematical computations for the rent increase. The model generates 8 primary output variables as illustrated in the corresponding Figure 6, representing the final cost allocation and rent increase calculations derived from the functional cost splitting methodology.

```python
1 def create_a_bestehendes_bauteil_IST_fkt_ebene_1(model):
2         """
3         Creates an existing building component at function level 1 (IST FKT 1).
4
5         Parameters:
6         - model: The CP-SAT model where variables and constraints are added
7
8         Returns:
9         - An instance of ISTFunktionenGewichtungBestehendesBauteilFktEbene1
              containing the variables for IST FKT 1
10        """
11        # Define integer variables for each function, ranging from 0 to 100
12        ss = model.NewIntVar(0, 100, "IST_fe1_ss")
13        bp = model.NewIntVar(0, 100, "IST_fe1_bp")
14        ae = model.NewIntVar(0, 100, "IST_fe1_ae")
15        wk = model.NewIntVar(0, 100, "IST_fe1_wk")
16        su = model.NewIntVar(0, 100, "IST_fe1_su")
17
18        # Add constraints to ensure the sum of all function values equals the
              total sum variable
19        model.Add(ss + bp + ae + wk == su)
20        # The total sum should be exactly 100 (percent)
21        model.Add(su == 100)
22
23        # Return an instance of the IST FKT 1 class with the defined variables
24        return ISTFunktionenGewichtungBestehendesBauteilFktEbene1(ss, bp, ae, wk,
               su)
```

Figure 4: Part 2/2: Modeling variables (lines 12 to 16) and two constraints (lines 19 and 21) for the IST/ACTUAL functions of Level 1. See Figure 2 for the COMMON LISP implementation.

The constraint variables are grouped into classes, which typically correspond to rows in the FK spreadsheets. Some classes for Step I are `Building`, `ISTFunktionenGewichtungBestehendesBauteilFktEbene1`, `ISTFunktionenGewichtungBestehendesBauteilFktEbene2`, and `MODFunktionenGewichtungBestehendesBauteilFktEbene1`. Figures 3 and 4 provide the model for class `ISTFunktionenGewichtungBestehendesBauteilFktEbene1`, which combines all variables, their domains, and two constraints for the four weighted functions of the ACTUAL State Function Level 1 which is equivalent to the implementation Figure 2 shows.

6.2. Number Representation

The requirements have demonstrated that the domains of the constraint variables exhibit diverse data types. Percentages associated with FK functions are represented using integer values, as decimal places would not make sense here since the user is intended to estimate these percentages subjectively. Conversely, real estate calculations, including rent increases, necessitate real numbers.

In selecting a constraint tool, one has the option between utilizing integer or mixed-integer programming for processing integers and/or real values. Initially, we employ the integer algorithm, which required a systematic scaling procedure for decimal values. Monetary values (such as modernization costs) were multiplied by a scaling factor of 10,000 to ensure sufficient precision in the cent range. Percentage values (such as under-fulfillment) were scaled by a factor of 100 to obtain one decimal place (e.g. 42.7% becomes 4270 in the solver).

Before constraint solving, all necessary input values are scaled accordingly. Calculations proceed within integer domain, with results subsequently scaled back for output purposes. This procedure ensures both the required numerical precision and compatibility with integer constraint solvers.

Precise determination of calculations for the first two table sheets was achieved after incorporating rounding in the EXCEL calculation for specific test cases.

While integer programming convinces through deterministic results and optimized solver performance, it requires complex scaling procedures for decimal values. Mixed Integer Programming (MIP) offers advantages through direct processing of both integer and continuous variables, thereby avoiding scaling artifacts and simplifying the modeling approach. For future extensions of the FK system, particularly when integrating optimization objectives and interval inputs, MIP could represent a more efficient alternative.

Figure 6 presents an example of the final result of the increase in rent when modernizing.

6.3. Dynamic CSP Generation

As discussed in Section 2, some fields of the spreadsheet are dynamically activated or computed when other fields are filled out. Hence, the constraint satisfaction problem is not the same for each building. Depending on the damage assessment, more or less variables and constraints are present. For example, if moisture damage is identified, additional variables for moisture protection calculations and their corresponding constraints are included in the CSP model. Hence, depending on the input values given by the user, the variables and constraints are selected by a generation module, and the constraint model is created and solved.

6.4. Optimization Aspects

Beyond constraint satisfaction, our approach naturally supports optimization objectives that extend the capabilities of the original spreadsheet implementation. The FK calculations can be enhanced to optimize for various criteria, including cost minimization where the system identifies the minimum cost configuration that achieves required function improvements, and benefit maximization where function improvements are maximized within given budget constraints. The constraint framework also accommodates multi-objective optimization scenarios that balance cost considerations, function improvements, and sustainability metrics simultaneously. Furthermore, Pareto optimization capabilities enable the identification of trade-off solutions between conflicting objectives, providing decision makers with a comprehensive view of available alternatives.

The constraint model structure facilitates these optimization extensions through its clear separation of variables, constraints, and objective functions. This architectural separation enables straightforward integration of optimization goals without requiring fundamental changes to the underlying constraint model. The declarative nature of the constraint representation allows for dynamic objective function specification, where different optimization criteria can be applied to the same underlying model based on user preferences or specific modernization scenarios. These optimization capabilities represent a significant advancement over the original spreadsheet approach, which is limited to single-point calculations and cannot explore solution spaces or identify optimal configurations.

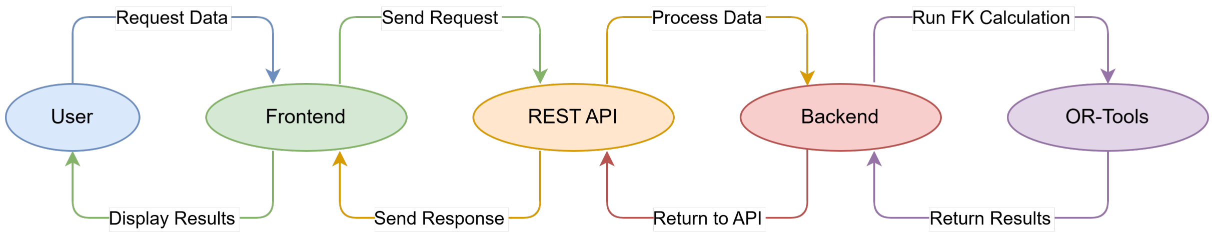


Figure 5: Architecture, including the constraint solver in the backend.

7. Architecture and User Interface

System components divide into the main components:

Front-end: User interface for entering and displaying results. Sends REST requests to the back-end.

REST API: Interface between the front and back end. Defines endpoints for data transfer.

Back-end with OR tools: Processes requests and solves CSP models. Implements functional cost splitting.

The interface data is modeled in the JSON format where key value pairs directly map to variables of the CSP.

The following interactions between the components occur (see Figure 5):

- The User sends request data ("Request Data") via the web client of the user to the Frontend.
- The Front-end processes the request and sends it ("Send Request") to the REST API.
- The REST API receives the request and forwards the data ("Process Data") to the Back-end for processing.
- The Back-end initiates the Functional Cost Splitting (FK) calculation ("Run FK Calculation") and calls OR-Tools for this purpose.
- OR-Tools performs the constraint programming calculations and returns the results ("Return Results") back to the Back-end.
- The Back-end processes these results and forwards them ("Return to API") to the REST API.
- The REST API sends the response ("Send Response") to the Front-end.
- The Front-end prepares the data and displays the results ("Display Results") to the User.

An ontology (not depicted, see [10]) specifies the necessary classes to model the knowledge database, covering legal, construction-related, and FK-related aspects. The FK calculations were provided to the front-end via an API and corresponding interfaces were defined.

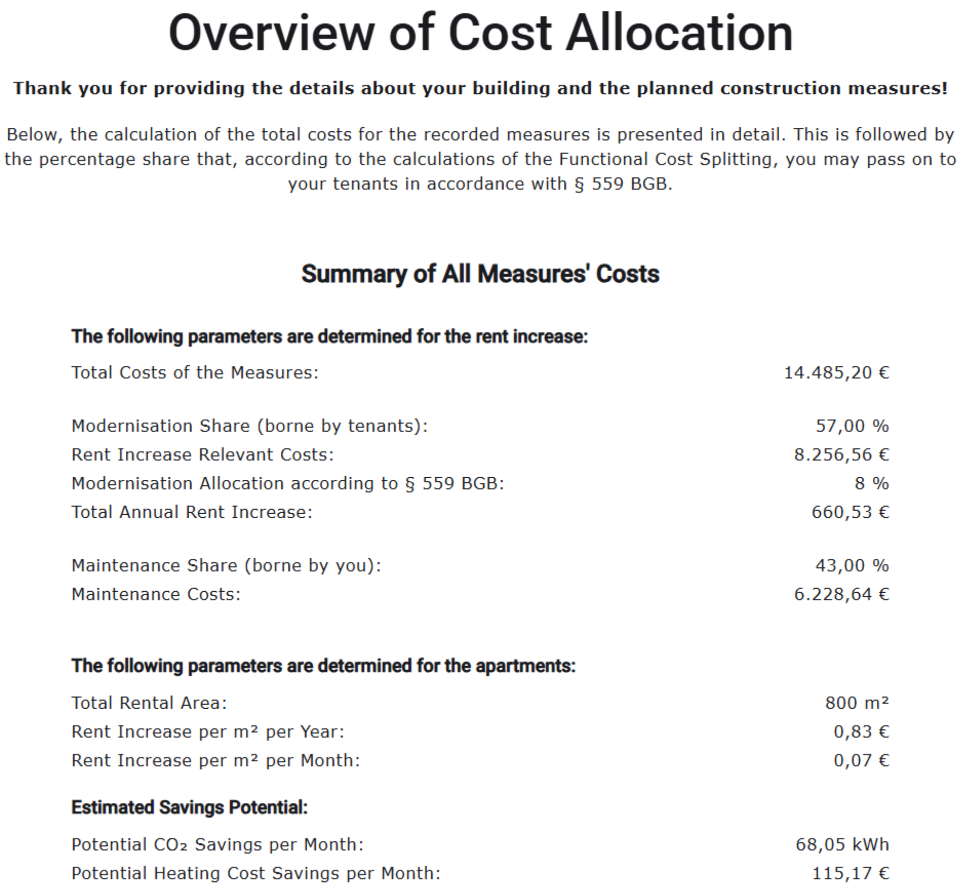
A prototypical user interface enables the entry of building-specific data, the cost of modernization, as well as the input of the weightings for the functions (see Figure 1). The prototype is available on the Internet[5].

8. Advantages and Limitations

Our constraint-based approach offers several significant advantages over alternative methods for converting spreadsheet computations to executable models. The systematic mapping process preserves the semantic structure of the original spreadsheet, maintaining domain expert knowledge intact throughout the transformation. Unlike procedural implementations that hardcode computational logic, constraint models provide a declarative nature that separates problem description from solution algorithms, enabling greater flexibility and maintainability. The declarative approach naturally supports multi-directional computation capabilities, allowing for reverse calculations such as determining required modernization measures from desired rent increase level. Additionally, the constraint framework seamlessly integrates optimization objectives, enabling extensions for cost minimization, benefit maximization, and multi-objective optimization scenarios. The dynamic CSP generation capability effectively handles conditional logic and varying problem complexity based on input configurations, adapting the constraint model size and complexity to the specific building assessment requirements.

However, the approach also presents several limitations that must be acknowledged. The development process requires substantial manual analysis and modeling effort, making it labor-intensive compared to an automated conversion tools, e.g., with LLMs. Effective variable grouping and constraint identification

[5]https://mieter.intelmod.hitec-hamburg.org/



Overview of Cost Allocation

Thank you for providing the details about your building and the planned construction measures!

Below, the calculation of the total costs for the recorded measures is presented in detail. This is followed by the percentage share that, according to the calculations of the Functional Cost Splitting, you may pass on to your tenants in accordance with § 559 BGB.

Summary of All Measures' Costs

The following parameters are determined for the rent increase:	
Total Costs of the Measures:	14.485,20 €
Modernisation Share (borne by tenants):	57,00 %
Rent Increase Relevant Costs:	8.256,56 €
Modernisation Allocation according to § 559 BGB:	8 %
Total Annual Rent Increase:	660,53 €
Maintenance Share (borne by you):	43,00 %
Maintenance Costs:	6.228,64 €
The following parameters are determined for the apartments:	
Total Rental Area:	800 m²
Rent Increase per m² per Year:	0,83 €
Rent Increase per m² per Month:	0,07 €
Estimated Savings Potential:	
Potential CO_2 Savings per Month:	68,05 kWh
Potential Heating Cost Savings per Month:	115,17 €

Figure 6: Result of the computations providing the increase per m^2 per month and further information in 8 variables (the two "total" variables are input variables and the last two are not computed by the CPS).

demand deep understanding of both the application domain and constraint programming principles, limiting the approach's accessibility to domain experts without technical programming knowledge. The resulting system's performance and capabilities are fundamentally constrained by the chosen constraint solver's limitations and computational efficiency.

When compared to alternative approaches, our method provides better maintainability and flexibility than direct code generation but requires more initial development investment. Compared to retaining the original spreadsheet format, our approach enables integration into larger software systems and supports advanced optimization capabilities, though it sacrifices the immediate usability that domain experts experience with familiar spreadsheet interfaces.

9. Conclusion

The paper presents a development process for a constraint model starting from a 4GL spreadsheet. We use this process to implement computations for a building modernization platform with constraints. Main subtasks are the identification of the variables and their domains as well as the constraints. Furthermore, grouping of variables and constraints in rows of the spreadsheet facilitates a transparent implementation. The computations are based on integer and mixed-integer programming. A dynamic aspect of the spreadsheet was mapped onto dynamic constraint model generation, depending on the user's input. Finally, we could completely map the spreadsheet to the constraint model and solve modernization related rental increases. Further work will encompass the usage of ranges for the input variables.

Acknowledgments

This research was funded by the Federal Ministry for Economic Affairs and Climate Action (Bundesministerium für Wirtschaft und Klimaschutz), Germany, within the "Joint project: EnOB: IntelMOD - Intelligent modernization platform based on functional cost splitting; sub-project: Infrastructure of the modernization platform" ("Verbundvorhaben: EnOB: IntelMOD - Intelligente Modernisierungsplattform auf Basis des Funktionalen Kostensplittings; Teilprojekt: Infrastruktur der Modernisierungsplattform."), FKZ 03EN1094D.

Declaration on Generative AI

During the preparation of this work, the authors used the LLM models DeepSeek and Claude exclusively for translation (German to English) and for literature and research searches. The LLM models were not employed to generate core content of the paper, i.e., they were not involved in analysis, methodology, results, or conclusions. Additionally, the code discussed in the paper is written by the authors.

References

[1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann, 2014.

[2] L. Hotz, A. Felfernig, M. Stumptner, A. Ryabokon, C. Bagley, K. Wolter, Configuration Knowledge Representation & Reasoning, in: A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen (Eds.), Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann Publishers, 2014, pp. 59–96.

[3] J. Martin, Fourth Generation Languages, Volume I: Principles, Prentice-Hall, 1985.

[4] W. R. Harris, S. Gulwani, Spreadsheet table transformations from examples, in: Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM, 2011, pp. 317–328.

[5] A. O. Shigarov, A. A. Mikhailov, Rule-based spreadsheet data transformation from arbitrary to relational tables, Information Systems 71 (2017) 123–136.

[6] K. David, Funktionales Kostensplitting zur Ermittlung von Mieterhöhungen nach energetischen Maßnahmen - Eine Handlungsempfehlung auf Basis theoretischer und empirischer Untersuchungen, doctoralthesis, HafenCity Universität Hamburg, 2019. URL: https://repos.hcu-hamburg.de/handle/hcu/505.

[7] G. L. J. Steele, COMMON LISP: the Language, Digital Press, 1984.

[8] J. M. Siskind, Screaming Yellow Zonkers, Technical Report, M.I.T. Artificial Intelligence Laboratory, 1991.

[9] Choco Team, PyChoco: Python binding for Choco-solver, https://github.com/chocoteam/pychoco, 2025.

[10] L. Hotz, K. Wilke, Structuring Legal Text as Preparation for Chat-Bot Use, in: Artificial Intelligence for Digital Public Services, STAF Workshop, CEUR Workshop Proceedings, 2025. To appear.

The Task Assignment Problem for Safety-Critical Networks Considering Communication and Criticality

Franz Wotawa[1,*,†], Julian Proenza[2], Manuel A. Barranco[2] and Alberto Ballesteros[2]

[1]*Graz University of Technology (TU Graz), Institute of Software Technology, Inffeldgasse 16b/2, A-8010 Graz, Austria*
[2]*Universitat de les Illes Balears, Cra. de Valldemossa, km 7.5. 07122 Palma, Illes Balears, Spain*

Abstract

The task assignment problem is well-known and of great practical importance. In a previous work, we presented a corresponding answer set programming model and provided an initial experimental evaluation, demonstrating its practical applicability. However, this model falls short in several aspects, making it less than entirely suitable for safety-critical networks. In this paper, we extend the model providing means for representing critical tasks and communication. In particular, we introduce predicates for capturing communication among tasks and their limitations caused by networks, i.e., the available bandwidth. We also provide a preliminary experimental evaluation of the new model, demonstrating its feasibility for minor problem instances.

Keywords

Configuring computing nodes, ASP models for configuration, Experimental analysis

1. Introduction

Knowledge-based configuration, i.e., the composition of elements and parts to fulfill customers' needs, has garnered considerable attention. There has been a lot of research and applications reported in scientific literature, ranging from service configuration [1], governance systems [2], product configuration [3], considering hardware and software in the automotive domain [4], to green configuration in scheduling [5], only to give the most recent examples. Modelling for configuration is often based on logic, see, e.g., [6], and most recently, answer set programming (ASP) for representing models used for configuration and reasoning to obtain valid configuration has gained more attention, e.g., see [7, 3, 8, 9].

In this short paper, we continue work on system configuration, focusing on configuring networks comprising nodes for executing pre-defined tasks. As already mentioned in our previous paper [10], the underlying problem is related to scheduling and shift designs [11] and has also been considered in previous work, e.g., [12]. Furthermore, the underlying problem can be seen as a variant of the well-known knapsack problem [13, 14]. In contrast to our previous paper, we extend the underlying model to bring it closer to the intended application area, which are highly reliable networks for hard real-time systems, where faults occurring during operation need to be mitigated within a pre-defined time see e.g., [15]. Mitigation depends on the type of fault, e.g., a fault in a network node, where tasks need to be reallocated. Such a (re-) configuration needs to be fast, such that no computational real-time requirements are violated. It is worth noting that machine learning has already been suggested [16] to solve re-configuration during operation.

In particular, we introduce concepts for handling communication among nodes and their limitations. We simplify communication by considering only one bus where all nodes are connected to enable allocated tasks to communicate with each other. The formal representation allows us to state which task is communicating with another and also the costs of communication in terms of required bandwidth. In

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
†Authors are listed in reverse alphabetical order.
wotawa@tugraz.at (F. Wotawa); julian.proenza@uib.es (J. Proenza); manuel.barranco@uib.es (M. A. Barranco); a.ballesteros@uib.cat (A. Ballesteros)
https://www.tugraz.at/ (F. Wotawa)
0000-0002-0462-2283 (F. Wotawa)

addition, we also capture the challenge of critical tasks. A task is considered critical if its non-execution would cause a safety-relevant effect. Such a task is replicated in a safety-critical network, and voting mechanisms are applied to ensure that it is always executed. Examples of safety-critical tasks include a brake controller that must always enable braking when requested by a driver of a vehicle.

In addition to the extended model, we conduct a first limited experimental evaluation based on several instances of task assignment problems. The evaluation utilizes the answer set programming solver `clingo` [17] and answers the question whether the extended model is appropriate for being used in the context of safety-critical networks, i.e., whether it is fast enough to assign tasks whenever required.

We structure this paper as follows. First, we introduce the underlying configuration problem. Afterward, we discuss the extensions of the model and present an answer set programming solution, followed by the experimental evaluation. Finally, we conclude this short paper.

2. Problem description

In this section, we define the task-to-node assignment problem, or short *task assignment problem*. We start summarizing the problem and its related constraints from our previous paper [10]. We assume k computing nodes $n_1, \ldots, n_k$ and n tasks $t_1, \ldots, t_n$. For each node n_i, we know the maximum number of tasks $c(n_i)$ it can hold and the available memory $m(n_i)$. For each task t_j, we know its memory consumption $m(t_j)$. In the following, we use this knowledge to formulate several constraints that need to be considered when assigning tasks to nodes. For the constraints, we assume a function $assigned(n_i)$ that returns a set of tasks that is assigned to a node n_i.

1. *Memory limitations:* The required memory by the task shall never exceed the available memory of the node.
$$\forall i \in \{1, \ldots, k\} : \left(\sum_{t_j \in assigned(n_i)} m(t_j) \leq m(n_i) \right) \tag{1}$$
2. *Task limitations:* The number of tasks assigned to a node shall never exceed its capabilities.
$$\forall i \in \{1, \ldots, k\} : (|assigned(n_i)| \leq c(n_i)) \tag{2}$$
3. *Global memory limitations:* The required memory of all tasks shall never exceed the memory provided by all nodes.
$$\sum_{j=1}^{n} m(t_j) \leq \sum_{i=1}^{k} m(n_i) \tag{3}$$
4. *Global task limitations:* The number of available tasks shall never exceed the sum of the number of tasks of all nodes.
$$n \leq \sum_{i=1}^{k} c(n_i) \tag{4}$$

A solution to the *tasks assignment problem* is an assignment of all tasks to nodes such that $\forall j \in \{1, \ldots, k\} : \exists i \in \{1, \ldots, n\} : t_j \in assigned(n_i)$, there is no tasks assigned to two different nodes, i.e., $\forall i, j \in \{1, \ldots, k\}, i \neq j : assigned(n_i) \cap assigned(n_j) = \emptyset$, and all constraints are fulfilled. Such an assignment is a valid one and may not exist due to limitations regarding available memory or the total node capacity. We may also consider optimality criteria such as minimizing the number of nodes where we assign tasks.

In addition to this original `task assignment problem`, we now add further information and constraints to represent the safety-critical network more appropriately. In particular, networks are for communication. Communication channels impose further constraints due to resource limitations. For example, there is only a maximum bandwidth available. If too many tasks are communicating at the same time, the bandwidth might not be sufficient. To simplify communication, we now only consider

Table 1
Predicates used to specify nodes, tasks, and their corresponding knowledge from [10].

Predicate	
node(n)	specifies that n represents a node
tcapacity(n,c)	maximum number of tasks c that a node n can hold
mcapacity(n,m)	maximum memory m provided by node n
task(t)	specifies that t is a task
memory(t,m)	memory m required by task t

a bus, where all messages have to pass through. Hence, there is a limitation of the bus in terms of bandwidth BW. We now need to specify the communication needs of each task. We assume that not necessarily each tasks need to communicate with each other. Hence, we introduce a function com that maps a potentially empty set of tasks to a given task, and a function bw that maps a task t and any tasks from $com(t)$ to a necessary bandwidth. However, this bandwidth is only required if the two tasks are not assigned to the same node. If two tasks are in the same node, there is no need to use the bus for communication. Obviously, the required total bandwidth needed for communication shall never exceed the bandwidth of the bus BW.

$$\sum_{j \in \{1,\ldots,n\}} \sum_{\substack{t \in com(t_j) \wedge \\ \nexists i \in \{1,\ldots,k\} : \\ \left(\begin{array}{c} t \in assigned(n_i) \wedge \\ t_j \in assigned(n_i) \end{array} \right)}} bw(t_j, t) \leq BW \tag{5}$$

In addition to communication, we may also want to state that several tasks should never be allocated to the same node. Critical tasks are examples. Such tasks may replicate each other in behavior and are used to add fault tolerance. For simplification purposes, we only introduce a predicate $distinct$ for any two tasks stating that both are not allowed to be assigned to the same node:

$$\begin{array}{c} \forall j \in \{1,\ldots,n\} : \forall j' \in \{1,\ldots,n\}, j \neq j' : distinct(t_j, t_{j'}) \rightarrow \\ \nexists i \in \{1,\ldots,k\} : \big(t_j \in assigned(n_i) \wedge t_{j'} \in assigned(n_i)\big) \end{array} \tag{6}$$

The task assignment problem comprising the additional constraints 5 and 6 is the *extended task assignment problem.*

3. Implementation

After outlining the task assignment problem in the last section, we present a solution using answer set programming where we rely on the syntax of the `clingo` solver [17], which is similar to the Prolog language. For more information regarding answer set programming (ASP), we refer to introductory literature, e.g., [18]. Note also that we do not discuss the ASP model in detail, except for the new addition. The details are described in our previous paper [10]. In Table 1, we summarize the predicates necessary to specify the original task assignment problem.

To find solutions for this problem, we further introduced a predicate `select` that takes a task `T` as the first parameter and a node `N` as the second. The ASP solver selects tasks for nodes such that no constraint is violated. To be self-contained, we summarize the `clingo` source code comprising additional predicates for handling memory and task requirements:

```
% Generate a selection of a node for each task
{ select(T,N) : node(N) } = 1 :- task(T).
```

```
% Constraints
% No 1: Do not exceed the max. number of tasks
noTasksAssigned(M,N) :-
   M = #count { T : select(T,N)}, node(N).
:- noTasksAssigned(M,N), tcapacity(N,C), M>C.

% No 2: Do not exceed the max. memory of a node
memRequired(M,N) :-
   M = #sum { NM,T :select(T,N), memory(T,NM)},
   node(N).
:- memRequired(M,N), mcapacity(N,C), M > C.

% Global constraints
totalCapacity(C) :- C=#sum {T,N :tcapacity(N,T)}.
totalNrTasks(C) :- C=#count {T :task(T) }.
totalMemReq(C) :- C=#sum {M,T :memory(T,M)}.
totalMem(C) :- C=#sum {N,CN :mcapacity(CN,N)}.
:- totalCapacity(C), totalNrTasks(TC), C < TC.
:- totalMemReq(Ctask), totalMem(Cnode),
   Ctask > Cnode.
```

It is worth noting that this implementation corrects two shortcomings of the original one, which may have led to some results that should not have been consistent solutions to the task assignment problem. In the following, we now discuss the extension to this model to allow computing solutions for the *extended task assignment problem*.

Let us handle the communication between nodes first. For stating communication needs between tasks, we introduce a predicate `comReq\3` that states communication needs between two tasks and the required bandwidth. Hence, this predicate more or less captures the functions com and bw. For example, `comReq(t1,t2,10)` states that there is a message transfer from task `t1` to `t2` requiring a bandwidth of 10. What we need to formalize is the communication need and a constraint stating bandwidth violation. For the former, we introduce a predicate `comNeed\3` that summarizes communication needs between two tasks depending on their assignment to nodes. If they are at the same node, there is no communication bandwidth required. Otherwise, it is stated as defined in `comReq`. The following rules capture this behavior:

```
comNeed(T1,T2,0) :- select(T1,N), select(T2,N).
comNeed(T1,T2,B) :-
   select(T1,N1), select(T2,N2),
   N1!=N2, comReq(T1,T2,B).
```

Based on `comNeed`, we can now specify the sum of the communication bandwidth required, which we define as follows, utilizing the predicate `comRequired` for the whole bus:

```
comRequired(M) :-
   M=#sum {B,T1,T2 :comNeed(T1,T2,B),
                    task(T1), task(T2)}.
```

Finally, we state the communication constraint that the communication required is not allowed to exceed the total bandwidth provided by the bus (which is 50 in this particular case):

```
comChannel(50).
:- comChannel(B), comRequired(M), M>B.
```

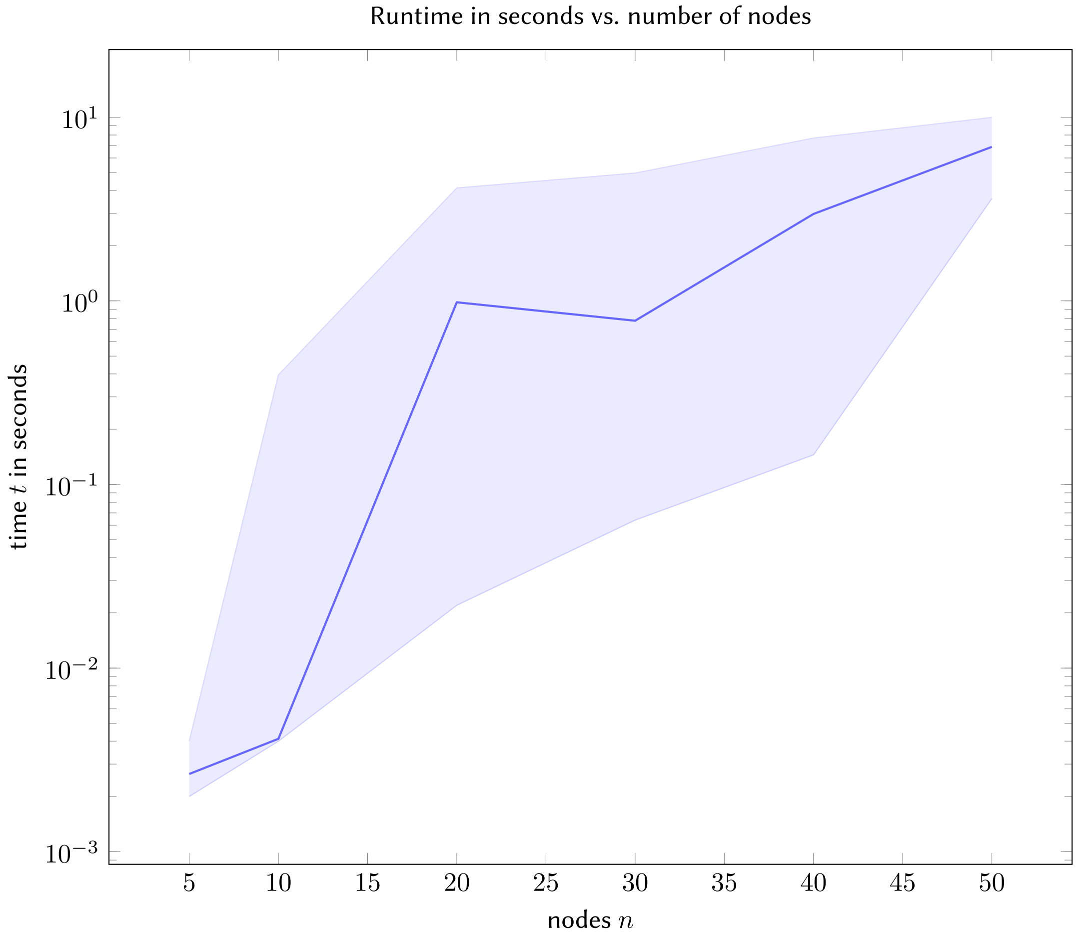


Figure 1: Solving runtime of satisfiable and unsatisfiable instances

For handling information about critical nodes, we introduce the `distinct\2` predicate to state that two tasks should never be assigned to the same node, e.g., `distinct(t1,t2)` states that tasks `t1` and `t2` has to be assigned at different node. Stating this constraint is straightforward:

```
:- select(T1,N), select(T2,N), distinct(T1,T2).
```

It is worth noting that this model is still a simplified representation of the task assignment for safety-critical networks. But it covers certain important aspects, which have not been considered before.

4. Experimental evaluation

Similar to the experimental evaluation in our previous paper [10], we want to investigate the runtime behavior of the ASP solver `clingo` when using systems comprising a different number of tasks and nodes. In particular, it is interesting to know how many nodes can be handled within a fixed time span of, e.g., 0.01 or 0.1 seconds. In addition, we are interested in the effects of the additional constraints on the runtime.

Experimental setup: We used a Java program for generating model instances automatically, where we ranged the number of nodes from 5, 10, 20 to 100 and the number of tasks randomly between the number of nodes and its double. The capacity of each node was randomly set from 1 to 10. The memory provided by each node was randomly chosen from 20, 40, 60,…, 200. The memory required by every task was randomly set to 10, 20, or 30. Moreover, we randomly selected whether a task communicates

Nodes n	SAT + UNSAT	SAT
5	0.003	0.003
10	0.041	0.041
20	0.982	0.752
30	0.779	0.361
40	2.973	2.298

Table 2
Average runtime in seconds of satisfiable and unsatisfiable instances in comparison with satisfiable instances only.

with another and also whether two tasks are distinct. We obtained two different test sets, each of size 110, considering two different probability settings. We conducted the experiments using an Apple MacBook Pro, with an Apple M1 CPU comprising 8 cores and 16 GB of main memory, running under macOS Sequoia Version 15.5. For computing solutions, we relied on `clingo` version 5.7.1 and applied the standard setup. Note that this setup (with the exception of the underlying operating system) is the same as used in our previous paper [10] to allow for a comparison.

Experimental results: After generating the problem instances, we ran `clingo` to compute one solution, i.e., we ran `clingo` using the prompt `clingo –time-limit=10 –outf=2` where we set a time limit of 10 seconds and obtained all results in JSON format. What we observed is that the underlying new constraints impact the runtime. For the first test set, we exceeded the time limit 76 times. From the remaining instances, 29 were satisfiable and 5 were unsatisfiable. For the second test, the number of instances where we could not establish a solution drops to 67. The number of satisfiable instances increases to 43, and no unsatisfiable instance was obtained.

Figure 1 depicts the minimum, maximum, and average runtime for all satisfiable and unsatisfiable runs for each category where data was available. In comparison with the results from our previous paper [10], we see a big difference. Only smaller instances comprising less than 10 nodes can now be configured in less than 0.1 seconds, which was 20 in our other publication. Hence, the additional constraints have a substantial impact, which is also visible by the high number of instances that cannot be analyzed within the 10-second boundary.

We further compared the average runtime of all satisfiable and unsatisfiable instances with the one obained considering only satisfiable instances. Table 2 summarizes the results where we only consider nodes where enough instances for a comparison remain. We see that when considering unsatisfiable instances the runtime increases on average, which is in line with results obtained in our previous paper.

Threats to validity: The presented results are from an initial experimental evaluation. The experimental setup is limited, not considering the entire range of potential parameters. Due to the time boundary set, there are many instances where satisfiability or unsatisfiability cannot be assigned. The setup also does not allow for answering several interesting questions, like the responsibility of certain constraints for the increased runtime.

5. Conclusions

In this paper, we extend an existing model for the task assignment problem, considering constraints for communication and also for tasks that should not run on the same computing node. We further present the results of an initial experimental evaluation, which show that there is an impact on the overall runtime, potentially limiting its practical use to minor instances. However, the current evaluation is limited, and further experimental evaluations and a more in-depth analysis are necessary. In future work, we aim to address the questions regarding the influence of specific constraints on the overall runtime and develop a more sophisticated test set that considers a wider variety of parameters.

Acknowledgments

The work was supported by the Austrian Science Fund (FWF) Cluster of Excellence Bilateral AI under contract number 10.55776/COE12.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check, Paraphrase, and Reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] E. M. Strøm, T. M. Münsberg, L. Hvam, Identifying potential applications of service configuration systems in a logistics company, in: Proc. of the 25th Intern. Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 60–66. URL: https://ceur-ws.org/Vol-3509/paper9.pdf.

[2] S. Muñoz-Hermoso, D. Benavides, F. J. D. Mayo, Multi-level configuration in smart governance systems, in: Proc. of the 25th Intern. Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 67–74. URL: https://ceur-ws.org/Vol-3509/paper10.pdf.

[3] R. Comploi-Taupe, G. Friedrich, T. Niestroj, Dynamic aggregates in expressive ASP heuristics for configuration problems, in: Proc. of the 25th Intern. Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 75–84. URL: https://ceur-ws.org/Vol-3509/paper11.pdf.

[4] F. Jost, C. Sinz, Challenges in automotive hardware-software co-configuration, in: Proc. of the 26th Intern. Workshop on Configuration (ConfWS 2024), Girona, Spain, September 2-3, 2024, volume 3812 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 17–20. URL: https://ceur-ws.org/Vol-3812/paper2.pdf.

[5] C. M. Moya, C. Pérez, M. A. Salido, Developing an algorithm selector for green configuration in scheduling problems, in: Proc. of the 26th Intern. Workshop on Configuration (ConfWS 2024), Girona, Spain, September 2-3, 2024, volume 3812 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 41–49. URL: https://ceur-ws.org/Vol-3812/paper6.pdf.

[6] A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner, Consistency based diagnosis of configuration knowledge-bases, in: Proceedings of the Tenth International Workshop on Principles of Diagnosis, Loch Awe, 1999.

[7] S. Mishra, Product configuration in answer set programming, Electronic Proceedings in Theoretical Computer Science 345 (2021) 296–304. URL: http://dx.doi.org/10.4204/EPTCS.345.46. doi:10.4204/eptcs.345.46.

[8] N. Rühling, T. Schaub, T. Stolzmann, Towards a formalization of configuration problems for asp-based reasoning: Preliminary report, in: Proc. of the 25th Intern. Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 85–94. URL: https://ceur-ws.org/Vol-3509/paper12.pdf.

[9] R. Comploi-Taupe, A. A. Falkner, S. Hahn, T. Schaub, G. Schenner, Interactive configuration with ASP multi-shot solving, in: Proc. of the 25th Intern. Workshop on Configuration (ConfWS 2023), Málaga, Spain, September 6-7, 2023, volume 3509 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 95–103. URL: https://ceur-ws.org/Vol-3509/paper13.pdf.

[10] F. Wotawa, Using answer set programming for assigning tasks to computing nodes, in: Proc. of the 26th Intern. Workshop on Configuration (ConfWS 2024), Girona, Spain, September 2-3, 2024, volume 3812 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 64–67. URL: https://ceur-ws.org/Vol-3812/paper9.pdf.

[11] M. Abseher, M. Gebser, N. Musliu, T. Schaub, S. Woltran, Shift design with answer set programming, in: Logic Programming and Nonmonotonic Reasoning - 13th International Conference, LPNMR 2015, Lexington, KY, USA, September 27-30, 2015. Proceedings, volume 9345 of *Lecture Notes in Computer Science*, Springer, 2015, pp. 32–39. URL: https://doi.org/10.1007/978-3-319-23264-5_4. doi:10.1007/978-3-319-23264-5_4.

[12] M. Nica, B. Peischl, F. Wotawa, A constraint model for automated deployment of automotive control software, in: Proceedings of the Twentieth International Conference on Software Engineering & Knowledge Engineering (SEKE'2008), San Francisco, CA, USA, July 1-3, 2008, Knowledge Systems Institute Graduate School, 2008, pp. 899–904.

[13] V. Cacchiani, M. Iori, A. Locatelli, S. Martello, Knapsack problems – an overview of recent advances. part i: Single knapsack problems, Computers & Operations Research 143 (2022) 105692. URL: https://www.sciencedirect.com/science/article/pii/S0305054821003877. doi:https://doi.org/10.1016/j.cor.2021.105692.

[14] V. Cacchiani, M. Iori, A. Locatelli, S. Martello, Knapsack problems – an overview of recent advances. part ii: Multiple, multidimensional, and quadratic knapsack problems, Computers & Operations Research 143 (2022) 105693. URL: https://www.sciencedirect.com/science/article/pii/S0305054821003889. doi:https://doi.org/10.1016/j.cor.2021.105693.

[15] A. Ballesteros, M. Barranco, J. Proenza, L. Almeida, F. Pozo, P. Palmer-Rodríguez, An infrastructure for enabling dynamic fault tolerance in highly-reliable adaptive distributed embedded systems based on switched ethernet, Sensors 22 (2022) 7099. URL: https://doi.org/10.3390/s22187099. doi:10.3390/S22187099.

[16] R. Rotaeche, A. Ballesteros, J. Proenza, Speeding task allocation search for reconfigurations in adaptive distributed embedded systems using deep reinforcement learning, Sensors 23 (2023) 548. URL: https://doi.org/10.3390/s23010548. doi:10.3390/S23010548.

[17] M. Gebser, R. Kaminski, B. Kaufmann, T. Schaub, Multi-shot asp solving with clingo, Theory and Practice of Logic Programming 19 (2019) 27–82. doi:10.1017/S1471068418000054.

[18] T. Eiter, G. Ianni, T. Krennwallner, Answer set programming: A primer, in: Reasoning Web. Semantic Technologies for Information Systems: 5th International Summer School 2009, Brixen-Bressanone, Italy, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 40–110. URL: https://doi.org/10.1007/978-3-642-03754-2_2. doi:10.1007/978-3-642-03754-2_2.

Test-driven Generation of Constraint Satisfaction Problems Using Large Language Models

Christian Bähnisch[1,†], Lothar Hotz[1,*,†], Alexander Felfernig[2] and Sebastian Lubos[2]

[1]*Hamburger Informatik Technologie-Center e.V., Hamburg, Germany*
[2]*Graz University of Technology, Graz, Austria*

Abstract
Constraint Satisfaction Problems (CSPs) are a foundational technology used to solve a wide range of real-world problems. A critical factor of the success of constraint-based systems is the efficient acquisition of knowledge, where domain experts and knowledge engineers must reach an agreement on the correctness of the evolving knowledge base as early as possible. In this paper, we introduce a novel approach to automate the generation of CSP-based knowledge bases by following a test-driven approach. We demonstrate how large language models (LLMs) can be leveraged to generate knowledge bases where validation is performed on the basis of pre-defined basic test routines. This approach reduces the overhead typically associated with knowledge base development and maintenance.

Keywords
Constraint Satisfaction Problems, Large Language Models, Knowledge Acquisition, Automated Generation, Test-Driven Development

1. Introduction

Knowledge acquisition for constraint-based systems is a complex and often time-consuming task. It involves formalizing sometimes intangible domain knowledge into structured models consisting of variables, domain definitions, and corresponding constraints [1, 2, 3]. Through their ability to compute and reason about possible solutions, constraint-based systems are particularly popular to support tasks such as knowledge-based configuration [4] and scheduling [5].

In the field of software engineering, large language models (LLMs) have become invaluable tools, assisting developers in a range of tasks, from code completion and review to generating test cases and documentation [6]. LLMs such as ChatGPT[1] and Copilot[2] have shown significant potential in supporting software development tasks, including debugging and refactoring. Particularly, the combination of LLMs with compilers or interpreters has facilitated self-evaluating workflows, where generated code is immediately tested, and errors are fed back to the model for iterative improvement [7, 8, 9]. Such test-driven approaches have proven to be effective in refining code and maintaining syntactic correctness [6].

A simple example of applying LLMs in the context of code generation is GPT-Engineer[3], which exploits natural language prompts to generate game logic with a corresponding HTML-based user interface. By integrating a large language model with a compiler or interpreter, the model can self-correct by using error messages from failed executions – an approach supported by tools such as LangCHAIN.[4]

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
†These authors contributed equally.
christian.baehnisch@uni-hamburg.de (C. Bähnisch); lothar.hotz@uni-hamburg.de (L. Hotz); alexander.felfernig@tugraz.at (A. Felfernig); sebastian.lubos@tugraz.at (S. Lubos)
0000-0001-7370-7726 (L. Hotz)

[1]https://chatgpt.com/ (accessed: May 2025)
[2]https://copilot.microsoft.com/ (accessed: May 2025)
[3]https://github.com/AntonOsika/gpt-engineer (accessed: May 2025)
[4]https://www.langchain.com/ (accessed: May 2025)

system:
You are an agent designed to solve constraint problems, you write python code to solve and verify constraint problems. Your code is run and verified by the user. If you get an error, debug your code and try again. You only use the constraint solver pychoco. You conclude your python code with a variable "solutions" and verify it with the method call verify_solutions(solutions, <problem_name>, where <problem_name> is given to you as the first word of the instructions. "solutions" should be a list of all solutions to the constraint problem. If verification fails, you correct your solution.
Use the following example as a template for solving constraint problems with pychoco:
User: ints10. find numbers one to 10.
Assistant:

````
```python
from pychoco.model import Model
from test_tool import verify_solutions

model = Model()
x = model.intvar(1, 10, "x")

solver = model.get_solver()
solutions = [s.get_int_val(x) for s in solver.find_all_solutions()]
verify_solutions(solutions, "ints10")
```
````

user:
tsp. solve the traveling salesman problem, use the following problem instance:

```
# Number of cities
C = 4

# Distance matrix
D = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
```

the solution should be a list of valid solutions, each solution being a list of integers representing the cities to be vistited.

Figure 1: General prompt including an example for the LLM on how to call the verification method. The specific user prompt provides a problem instance. (Human created.)

Inspired by these advancements, we investigate in which way large language models can support knowledge acquisition and engineering for constraint satisfaction problems (CSPs). The research questions we aim to answer are the following:

1. Can LLMs generate reliable knowledge models?
2. What kind of prompts are effective?

In our earlier work [10], we introduced first ideas on applying test-driven principles to the generation of constraint satisfaction problems using LLMs. The current paper extends that work in two major ways: (i) we employ more recent LLM technology (GPT-4.1), which provides improved stability and reasoning capabilities compared to the models used previously, and (ii) we enhance the verification process, in particular for more demanding problems such as the traveling salesperson problem (TSP), where we introduce refined verification routines that guide the model towards handling optimality aspects. We use similar example problems in both papers.

In this paper, we propose a test-routine-guided approach for the generation of knowledge bases representing constraint satisfaction problems. In our approach, basic test routines act as a background knowledge oracle, guiding an LLM to automatically generate and validate generated CSPs. Based on

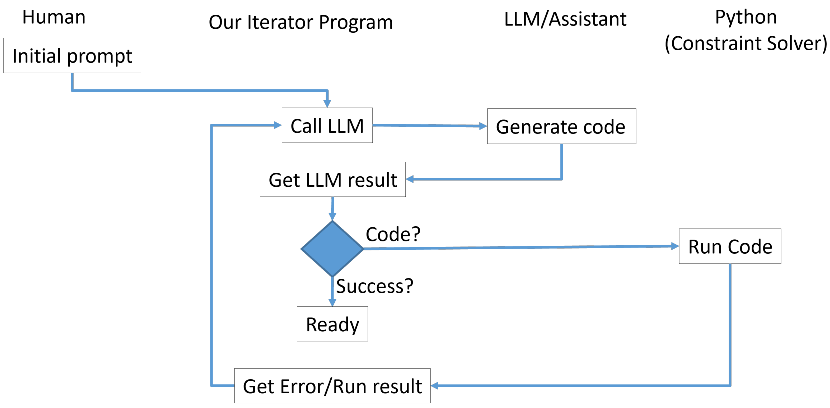


Figure 2: Architecture iterating through LLM-generated code. Verification is done by enforcing the LLM using the method `verify_solutions`, which is then executed with Python. For termination, a maximum number of iterations can be given (not shown).

a provided test routine that acts as a validation checkpoint, an LLM generates the knowledge base. This approach not only supports the automated generation of constraint models but also ensures the correctness of the knowledge base by leveraging test-driven validation.

The major contributions of this paper are the following:

1. We show how to integrate LLMs into knowledge acquisition processes for constraint-based systems.
2. Based on test-driven development, we are able to generate CSPs which – on a semantic level – conform completely with the underlying domain knowledge.
3. We show the applicability of the proposed approach with different commonly known CSPs.

The remainder of this paper is organized as follows. In Section 2, we review the state of the art in knowledge generation. In Section 3, we present our approach for test-routine-guided CSP generation. Section 4 details our verification method. Section 5 presents details of the implementation and the results of some solved CSPs. A discussion of the evaluation results of our generation approach is provided in Section 6. In Section 7, we conclude the paper by summarizing the research results and indicating different future research directions.

2. State of the Art in Knowledge Base Generation

Many real-world problems can be defined as constraint satisfaction problems (CSPs) requiring knowledge engineers to translate domain-specific constraints into corresponding formal representations. This modeling process is often complex and labor-intensive and triggers a need for enhanced knowledge acquisition support [11]. We now provide an overview of the existing state of the art in the automated generation of knowledge bases. These approaches range from the generation of knowledge bases from graphical descriptions, the learning and generation of knowledge bases from examples, and finally, the LLM-based knowledge base generation which is a discipline of specific relevance for the work presented in this paper [12].

Generating Knowledge Bases from Graphical Descriptions. The modeling of domain knowledge on the basis of graphical models has a long tradition in knowledge engineering. Examples thereof are the graphical representation of ontologies on the basis of the PROTÉGÉ[5] environment or the representation of variability properties in terms of feature models in the FEATUREIDE environment.[6] Such feature models can then be translated into corresponding constraint-based or SAT-based representations [13].

[5] https://protege.stanford.edu/ (accessed: May 2025)
[6] https://featureide.de/ (accessed: May 2025)

Listing 1: Verification method for the TSP problem. Other verifications are removed. (Human created.)

```python
def verify_solutions(
    solutions: List, problem: Literal["queen"] | Literal["tsp"] | Literal["simple"]
) -> bool:
    # Code for other problems removed.

    if problem == "tsp":
        try:
            # valids = [test_tsp.is_valid_route(sol) for sol in solutions]
            # if sum(valids) == 2 and len(solutions) == 2:
            #     print("All solutions are correct")
            # else:
            #     print(f"{sum(valids)} of the {len(valids)} solutions are correct. Expected 2 solutions")

            evals = [test_tsp.eval_route(sol) for sol in solutions]
            cost = sum([s[0] for s in evals])
            ham = sum([s[1] for s in evals])
            start = sum([s[2] for s in evals])
            correct = sum([(s[0] and s[1] and s[2]) for s in evals])
            if correct == 2 and len(solutions) == 2:
                print("All solutions are correct")
                return
            else:
                print(f"\n{correct} solutions are correct,  {len(solutions) - correct} are not correct. Expected 2 correct solutions:")
                print(f"{cost} solutions have minimal cost\n{ham} solutions are hamiltonian\n{start} solutions start with city 0.")
            return
        except:
            raise ValueError(f"invalid input: {solutions}")
    raise ValueError(f"unkown constraint problem: {problem}")
```

Although graphical knowledge representation help to reduce the knowledge acquisition bottleneck, the construction of graphical models is still a manual process [14].

Learning Knowledge Bases. Learning knowledge bases (KBs) from examples is a key research area in artificial intelligence which focuses on an automated acquisition of structured representations of domain knowledge. One related aspect is the learning of CSPs from examples. Learning CSPs involves inferring constraints based on positive and negative examples, often using constraint acquisition techniques that query an oracle or analyze example datasets to build consistent constraint models [15]. Such approaches have shown to be effective in domains such as scheduling, configuration, and automated diagnosis.

With the goal to optimize search performance, *learning search heuristics* for CSP solvers has become a relevant topic. The efficiency of CSP solvers largely depends on heuristics for variable and value selection during search. Machine learning methods have been developed to predict or adapt these heuristics by extracting informative features from problem instances thus leveraging existing solving experience from the past [16]. Portfolio-based solver frameworks, such as SATzilla, use learned models to dynamically select or combine heuristics and algorithms. With this, they significantly improve solver performance across heterogeneous CSP instances [17].

LLM-based Knowledge Generation. With the rise of large language models (LLMs), automated LLM-based knowledge generation has the potential to become a key approach in the context of knowledge

Listing 2: Generated constraint model for the TSP problem. The specific task is given as it was given in the prompt (Lines 5 to 11). LLM created the code. It took 5 iterations with 3 syntactic errors and 1 verification error (see chat in the Addendum). Part 1/2.

```
1  ‘‘‘python
2  from pychoco.model import Model
3  from test_tool import verify_solutions
4
5  C = 4
6  D = [
7      [0, 10, 15, 20],
8      [10, 0, 35, 25],
9      [15, 35, 0, 30],
10     [20, 25, 30, 0]
11 ]
12
13 max_cost = max(max(row) for row in D)
14
15 def get_arc_cost_expr(model, from_var, to_var, arc_cost_var):
16     # add table constraint for (from_var, to_var, arc_cost_var)
17     tuples = []
18     for from_city in range(C):
19         for to_city in range(C):
20             tuples.append([from_city, to_city, D[from_city][to_city]])
21     model.table([from_var, to_var, arc_cost_var], tuples).post()
22
23 model = Model()
24 tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
25 model.all_different(tour).post()
26 model.arithm(tour[0], "=", 0).post()
```

engineering processes [12, 18, 19, 20, 21]. Suchanek and Luu [22] argue, as we do, for combining LLMs with data-driven and knowledge-based methods to leverage the strengths of both. A key strategy in our work is grounding LLM outputs through test cases that verify the intended semantics of generated CSPs. Nayak and Timmapathini [23] follow a similar path by using LLM to identify objects and relations in structured data. A major commonality with the work presented in this paper is the idea of applying test cases as background knowledge, which helps to assure the correctness of the generated knowledge bases. Our approach advances the state of the art by providing generalized test routines which help to assure global consistency.

Ahmed and Choudhury [24] explore LLMs for the generation of optimization problems, aiming to simplify this task for domain experts by reducing the need for deep mathematical expertise. Using datasets of problem definitions and solutions, they demonstrate how LLMs can translate textual descriptions into formal optimization models. Unlike their human-in-the-loop fine-tuning approach, we focus on automated feedback using both syntactic validation (e.g., checking for compile errors raising exceptions) and semantic validation via test cases.

LLMs have also been applied in strategic reasoning contexts, including economic simulations and game theory [25]. A related method is the "Program of Thoughts" approach [26], which uses an LLM and a Python interpreter to solve numerical problems, though it lacks discussion on verifying the resulting code. Logical reasoning applications of LLMs include both fine-tuning models for specific tasks [27] and automated prompt engineering [28]. Pan et al. [29], for instance, improve code generation by using errors from constraint and SAT solvers as prompt feedback. Our work builds on this idea by incorporating test cases to further enhance CSP output on the semantic level. Other work, such as Acher et al. [30], has explored the usage of diverse prompts to generate configuration knowledge, including support for multiple programming languages. However, these efforts rely on manual execution without iterative correction mechanisms.

Finally, the automatic creation of CSPs (as proposed in this paper) can be seen as a specialized form of ontology construction, where concept hierarchies are generated using LLM prompts tailored to specific knowledge queries. For instance, Funk et al. [31] show how prompts like "What are the most important subcategories of category A?" can guide the creation of structured concept trees.

3. Approach for Test-Driven Generation of Constraint Satisfaction Problems

Main aspects of our methodology are the description of the input, the prompt, and the overall architecture. The verification methods is described in Section 4.

3.1. Input Description

We start with a textual description of a problem at hand, for which a constraint model shall be generated.

- This textual description can be the name of a commonly known task such as the N-queen problem, Magic Square, Map Coloring, or the Traveling Salesperson Problem (TSP) [10]. Using commonly known means, here, a description of the problem exists in the Internet and, hence, was used when the LLM had been trained.
- Or, the textual description is a natural language description of restrictions (e.g. *There are five houses. The Englishman lives in the red house. …*) that must be fulfilled such as given for a Zebra Puzzle[7].
- Or, a general description of an abstract problem with a concrete task instance is given.

In this paper, we concentrate on names as input, thus, expecting a commonly known meaning of a constraint satisfaction problem.

In our current implementation, the generated models are expressed in Python using the PyChoco API [32, 33], which defines the formal language available for constraint modeling. As a result, the LLM must produce syntactically correct Python code that makes use of Choco's modeling primitives. This choice ensures that solutions can be verified automatically but also means that the approach is bound to the expressive power and syntax of this solver. While we mention natural language descriptions and abstract problem definitions as possible inputs, in practice the current approach is most effective for well-known benchmark problems with concise descriptions (such as N-Queens or TSP). Larger-scale configuration problems are beyond the current scope. We therefore position our approach as particularly suitable for small to medium-sized CSPs or as a prototyping tool, while the scalability to industrial-size configuration tasks remains open for future work.

3.2. Prompting

We provide a prompt for a given problem in the usual manner where we describe the general task of the LLM as a $system - prompt$ (see Fig. 1 "system:")[8], and one example of the code to be generated (one shot learning). Important for the verification is here that the representation of the solution is specified in the prompt (here a Python list) and the demand to call the method `verify_solutions` with the problem (here the name) parameterized. Note that we do not mention a specific tool in the prompt that would be needed to let the LLM generate a tool call, we simply say: $write\ python\ code$.

[7] https://en.wikipedia.org/wiki/Zebra_Puzzle (accessed: May 2025)

[8] For clarity, for each figure and listing, we indicate the originator of the code.

Listing 3: Generated constraint model for the TSP problem. LLM created the code and the iterator prints the answer of the verification method. Part 2/2.

````python
arc_costs = [model.intvar(0, max_cost, f"arc_cost_{i}") for i in range(C)]
for i in range(C):
    get_arc_cost_expr(model, tour[i], tour[(i+1)%C], arc_costs[i])

cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()

# First, find the minimal cost
solver = model.get_solver()
min_cost = None
for s in solver.find_all_solutions():
    c = s.get_int_val(cost)
    if min_cost is None or c < min_cost:
        min_cost = c

# Now, collect all minimal cost tours
model = Model()
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arith(tour[0], "=", 0).post()
arc_costs = [model.intvar(0, max_cost, f"arc_cost_{i}") for i in range(C)]
for i in range(C):
    get_arc_cost_expr(model, tour[i], tour[(i+1)%C], arc_costs[i])
cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()
model.arith(cost, "=", min_cost).post()

solver = model.get_solver()
solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    solutions.append(t)

verify_solutions(solutions, "tsp")
```
````

---

user:

```
All solutions are correct
```

We do so because we would not rely on a specific keyword such as $Python\_REPL$ which shall trigger LangCHAIN (see Section 5.1).

A specific task is given through the definition of a concrete constraint satisfaction problem (see Fig. 1 "user:")

3.3. Architecture

Fig. 2 shows our iterative architecture. Only the initial prompt is given by a human, then our iterator program calls the LLM, which generates eventually the code (i.e., a constraint model written with the constraint solver PyChoco). If code was generated by the LLM, it is given to the Python executer, which
```

Listing 4: First, wrong version of the LLM-generated cost handling for the TSP model. Line 6 is wrong. The PyChoco library reports the error (Line 6 to 8).

```
1 # Add cost variable
2 cost = model.intvar(0, 1000, "cost")
3 # Express the cost of the tour (cycle)
4 model.sum(
5     [
6         model.element(D, tour[i], tour[(i+1)%C])
7         for i in range(C)
8     ],
9     "=", cost
10 ).post()
11
12 # Find the minimal cost
13 solver = model.get_solver()
14 min_cost = None
15 for s in solver.find_all_solutions():
16     c = s.get_int_val(cost)
17     if min_cost is None or c < min_cost:
18         min_cost = c
19
20 #### Lines skipped ####
21
22 verify_solutions(solutions, "tsp")
23 ‘‘‘
```

---

user:

```
1 Traceback (most recent call last):
2   File "/tmp/tmpar_b5jgq.py", line 22, in <module>
3     [
4   File "/tmp/tmpar_b5jgq.py", line 23, in <listcomp>
5     model.element(D, tour[i], tour[(i+1)%C])
6   File "/home/chbaeh/miniconda3/envs/llmconst/lib/python3.10/site-packages/
        pychoco/constraints/int_constraint_factory.py", line 245, in element
7     if len(table) == 0:
8 TypeError: object of type 'IntVar' has no len()
```

simply executes the code as usual with a Python interpreter. The constraint solver has to be imported through an import statement through the code which the LLM generates. The result of the interpreter, let it by a syntactic error indication (exceptions) (e.g., because the API of PyChoco is not used properly) or a semantic error, i.e., the verification method computes an error, is given back in the prompt to the LLM. Thus, the prompt, or better the context of the prompt, is increased with each iteration. If the LLM provides no code, the iteration stops.

Similar frameworks (sometime also called "reasoning frameworks") are LangChain [34], OpenAI Reasoning [35], Chain of Thought [36], ReAct [37], or those as in [38] based on the Model Context Protocol (MCP) [9]. However, we implemented our own loop (see Section 5.1).

4. Verifying Constraint Models

For verifying generated constraint models, we define tests or verification methods for the problem at hand, i.e., a problem-specific verification test that checks a solution for consistency or/and optimality.

```
user:
2 solutions are correct,  4 are not correct.
Expected 2 correct solutions:
2 solutions have minimal cost
6 solutions are hamiltonian
6 solutions start with city 0.
```

Figure 3: Verification responds that provides concrete hints for the LLM.

Listing 5: Central loop for iteratively (Line 2, `for`) calling the LLM (Line 5, `ask_openai`) and the tool (Line 10, `python_executer`), the context is increased (Lines 7 and 12, `append`). Line 13 relates to Line 20 in Listing 1, hence, checks if the solution was verified. (Human created.)

```
1 run_loop(messages: list[MessageType], n_iters: int = MAX_ITERS):
2     for i in range(n_iters):
3         print_message(messages[-1])
4         print(f"{i}: call llm")
5         ret_message = ask_openai(messages)
6         print_message(ret_message)
7         messages.append(ret_message)
8         code = extract_python_code(ret_message.get("content", ""))
9         if code is not None:
10            tool_output = python_executer.run(code)
11            tool_message = user_message(tool_output)
12            messages.append(tool_message)
13            if "All solutions are correct" in tool_output:
14                print_message(messages[-1])
15                return messages
16        else:
17            return messages
18    return messages
```

To ensure the accuracy of these tests, they are manually created by a human software engineer, as is usual for software implementation. However, perhaps with the support of source code generators such as Copilot[9], however, with the explicit confirmation of the human engineer[10]. Hence, here, we explicitly exclude a pure generation approach to ensure correctness, avoiding confabulation of the LLM. As pointed out in Section 3.2 the call to the verification method is parameterized with the problem name and has to be included in the generated code by the LLM, which really was the case in all of our tests (see Section 5.2). The method `verify_solutions` (see Fig. 1) verifies each problem separately to verify the solutions created by the program generated by the LLM.

In any case, of course, the concrete task instance, as it has to be verified, has to be coded into the verification method. However, the solution, since it is also a known representation specified in the prompt (see Section 3.2), is checked for consistency with the restrictions that hold for each problem. For example, the positions of the created queens solutions must not threaten each other. Additionally to consistency, in the TSP model, not only the Hamiltonian path (each city is in the solution) is of interest but the solution should also be minimal. Here, in one chat (see the Addendum), the LLM generated first a not correct solution which was not minimal. Our verification method verbosely provides an explanation of the not correct results and includes the responses shown in Fig. 3 in the context, which leads to an improved version that adds the notion of cost to the chat for the first time (see Listing 4) -

[9]https://copilot.microsoft.com/ (accessed: May 2025)

[10]We strictly follow our regime P.b.H.W.b.IT.E.b.H, i.e., Prompted by Human, Written by IT, Evaluated by Human.

the responses triggered the LLM to consider cost when solving the TSP. This means that for the TSP, the name "TSP" given in the initial prompt was not sufficient to ensure that an optimality criterion (the cost of the paths) is needed for solving the TSP, not only a consistency one. Hence, this is an example for changing or refining requirements during the chat and an example for a reasoning process. However, the first occurrence of cost handling in Listing 4 was wrong and later corrected in Listing 2.

The role of the user in this process is to provide problem-specific correctness criteria in the form of verification routines. These routines do not encode the complete CSP model, but rather provide a minimal set of semantic checks that any valid solution must satisfy (e.g., non-attacking queens in N-Queens, or route optimality in TSP). In practice, these checks are often simpler to formulate than an entire constraint model, since they can be expressed as direct computational tests over candidate solutions. The completeness of the verification depends on how precisely the user captures the essential constraints of the domain: if all relevant conditions are checked, then a generated model that passes verification can be regarded as correct. If only partial conditions are provided, the generated model will be correct only with respect to those conditions. Thus, the user's effort is shifted from fully modeling the problem to defining targeted, testable requirements that guide the LLM toward producing consistent CSPs.

Hence, in total, the verification method needs:

- concrete constraints of a constraint problem,
- a test method for checking these restrictions of the CSP in general,
- explanations as output if errors occur for guiding the LLM to a solution.

5. Implementation and Results

5.1. Implementation

The implementation utilizes Python, leverages ChatGPT-4.1 and is available in a repository at Zenodo[11]. As a constraint solver, we used PyChoco as a Python API for the constraint solver Choco [32, 33].

We do not include tool calls in the prompt, but only the demand to create Python code. Hence, we do not depend on the need that the LLM generates strings such as "Python_REPL". Furthermore, we could simpler manipulate the output of the Python interpreter and, hence, compress the context for saving context tokens (not further discussed here). The Listing 5 shows the implementation of the core loop with prompt increase, call to LLM, extracting the code, and executing it. The verification is called implicitly in the LLM-generated code.

5.2. Result

As an example, Listing 2 and Listing 3 show the LLM-generated code for the TSP problem and the final verification responds. One can see how the method `verify_solutions` is included in the code to start verification. The code is a constraint model, because constraint variables with appropriate variable domains are defined (lines 24, 32), constraints are defined (here by *post* in lines 21, 25, 26, 33), and finally the constraint solver is called (line 36) to generate solutions (line 36) which are consistent with the posted constraints.

For each of the selected concrete problems N-queen, Magic Square, Map Coloring, and TSP, we could generate a constraint model and verify its correctness. For Map Coloring and Magic Square an only one-step iteration was needed, i.e., the LLM could directly generate the solution code which was syntactically correct, the API was used correctly, and the semantic could correctly being verified through our verification methods. The N-Queen solution ($N = 8$) was sometimes found directly and, in other cases, some iterations were needed due to a misuse of the API (error messages from Python execution were included in the chat). Due to the indeterminism of the LLM, the iteration numbers

[11] https://doi.org/10.5281/zenodo.17132331

Success	Iterations	Exceptions	Validation Errors
True	6	4	1
True	5	3	1
True	4	1	2
True	4	2	1
True	8	5	2
True	8	6	1
True	3	1	1
True	6	4	1
True	4	2	1
True	4	2	1
True	8	4	3
True	9	6	2
True	5	1	3
True	7	5	1
True	5	3	1
True	6	4	1
True	6	4	1
True	8	6	1
True	7	5	1
True	7	5	1

Table 1
TSP iterations of 20 runs with the same concrete task instance (see Fig. 1).

Success	Iterations	Exceptions	Validation Errors
True	1	0	0
True	2	1	0
True	2	1	0
True	2	1	0
True	2	1	0
True	1	0	0
True	11	1	9
True	8	7	0
True	19	1	17
True	9	8	0
True	2	1	0
True	1	0	0
True	4	3	0
True	1	0	0
True	3	2	0
True	2	1	0
True	8	7	0
True	1	0	0
True	7	6	0
True	1	0	0

Table 2
8-Queens iterations of 20 runs with the same concrete task instance (not shown).

vary. One example for the TSP needed five iterations with three API errors and one semantic error (see Listing 2).

Table 1 shows the result of 20 runs for generating a solution of the TSP. Exceptions are API errors or syntactic Python errors. The numbers of iterations and error differ, however, in any case a verifiable solution could be found. Similarly, Table 2 shows the runs for the N-Queens example. All chats are listed in our repository.

In a further test, we used the final generated TSP constraint model to apply it manually to larger problem instances. For small constraint sets, for example, up to 10 cities, the approach still works. However, beyond that, timeouts occur (after 60 seconds). There are three solutions, but these are found quickly because the LLM did not use PyChoco; instead, it relied on a purely combinatorial approach with Python. Importantly, in these three solutions the LLM never used the PyChoco API (such as `setObjective(Model.MINIMIZE, cost)`) to find optimal solutions; it only generated all the solutions and then filtered them afterwards. This approach is feasible only for very small problem instances.

6. Discussion

The iterative approach that we introduce in this paper has the following characteristics:

- A system prompt defining a general task and a user prompt for defining a concrete instance of the task.
- Demanding to use Python as a programming language and to use a specific constraint solver (here PyChoco).
- One-shot learning by giving one example for a code which shall be created.
- Defining that the code shall produce a certain representation (here a list of solutions) and, to call with this list as parameter a specific function for verifying the generated code.
- Defining that an iteration will take place.
- Manually coding a verification method for a specific problem.
- Implementing a simple loop that calls the LLM to generate code, calls Python to execute it, and successively appends the Python output to the context (prompt).
- Help the LLM through specific verification hints that refine the requirements (here, the problem definition) given in the initial prompt.
- For all problem instances a verifiable code could be generated.
- For this generation of constraint models, the iterative approach is important, only for simple tasks one iteration step was sufficient.
- No use of reasoning frameworks because of the simplicity of looping through tools (here constraint solver) that leads to full control over input and output.
- Using our current approach, we aim to facilitate the process of knowledge acquisition to achieve consensus on the accuracy of the knowledge base. Our automation not only supports the initial development of the knowledge base but also accommodates updates and modifications as the knowledge evolves. Additionally, knowledge engineers and domain experts can focus on designing and performing verification tests.

As new LLMs are constantly being released and their non-deterministic nature means that the numbers presented in the tables represent only a snapshot, future advancements in even more powerful models are likely to yield different results.

The verification method we present in this paper ensures complete (global) consistency and, in the case of the TSP, optimality of the resulting constraint model. This is feasible because the verification method has full access to the entire set of constraints that must be satisfied. In other cases, typically known from test-driven development of software artifacts, only partial aspects of the software can be formalized, tested, and verified. In CSPs, full access to the entire set of constraints is due to the formal character of the given constraints which can easily be implemented as verification tests. However, the formalization of the constraint model used to compute solutions is generated by the LLM and is often more challenging than simply writing verification tests.

A crucial point with solvers is that they implement specific efficient algorithms that are general and work on descriptive models. However, those algorithms have to be used by the API of the solver library. In our experiments, the LLM does not use these algorithms, especially $find\_all\_optimal\_solutions$

but generates an algorithm for the brute-force method. Hence, specific prompts for leveraging these APIs should be developed to make use of these efficient algorithms. A related aspect is the generation or better exploration of new efficient algorithms for constraint solving similar to exploring protein structures.

7. Conclusion

The paper shows that correct constraint models can be generated with the help of an LLM if one combines their verification with constraint solvers in an iterative loop. Thus, it demonstrates that the fundamental approach of combining the generative capabilities of LLMs with the precision of constraint solvers and programming language interpreters is both feasible and poised to transform the future development of constraint models. Our next steps will be to use textual descriptions of constraint problems instead of known names.

Acknowledgments

This project was funded through the Federal Ministry for Economic Affairs and Climate Action (Bundesministerium für Wirtschaft und Klimaschutz), Germany, within the "Joint project: EnOB: IntelMOD - Intelligent modernization platform based on functional cost splitting; sub-project: Infrastructure of the modernization platform." ("Verbundvorhaben: EnOB: IntelMOD - Intelligente Modernisierungsplattform auf Basis des Funktionalen Kostensplittings; Teilprojekt: Infrastruktur der Modernisierungsplattform.")

Declaration on Generative AI

During the preparation of this work, the authors used the LLM models ChatGPT, DeepSeek, and Claude exclusively for translation (German to English) and for literature and research searches. The LLM models were not employed to generate core content of the paper, i.e., they were not involved in analysis, methodology, results, or conclusions. Additionally, the AI-generated code discussed in the paper constitutes research data or experimental results and is presented as evidence supporting the research findings; it was not used as a tool to write the paper itself.

References

[1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration – From Research to Business Cases, Morgan Kaufmann, 2014.

[2] Y. Huang, H. Liu, W. K. Ng, W. Lu, B. Song, X. Li, Automating knowledge acquisition for constraint-based product configuration, Journal of Manufacturing Technology Management 19 (2008) 744–754. doi:10.1108/17410380810888120.

[3] T. E. Nordlander, E. C. Freuder, R. J. Wallace, Maintaining constraint-based applications, in: 4th International Conference on Knowledge Capture, K-CAP '07, ACM, 2007, pp. 79–86. doi:10.1145/1298406.1298422.

[4] U. Junker, Configuration, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006, pp. 837–873.

[5] R. Barták, M. A. Salido, Constraint satisfaction for planning and scheduling problems, Constraints 16 (2011) 223–227. URL: https://doi.org/10.1007/s10601-011-9109-4. doi:10.1007/s10601-011-9109-4.

[6] I. Ozkaya, A. Carleton, J. Robert, D. Schmidt, Application of Large Language Models (LLMs) in Software Engineering: Overblown Hype or Disruptive Change?, Carnegie Mellon University, Software Engineering Institute's Insights (blog), 2023. URL: https://doi.org/10.58012/6n1p-pw64, accessed: 2024-May-16.

[7] K. Michailidis, D. Tsouros, T. Guns, Constraint Modelling with LLMs Using In-Context Learning, in: P. Shaw (Ed.), 30th International Conference on Principles and Practice of Constraint Programming (CP 2024), volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2024, pp. 20:1–20:27. doi:10.4230/LIPIcs.CP.2024.20.
[8] F. Régin, E. D. Maria, A. Bonlarron, Combining constraint programming reasoning with large language model predictions, in: Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP 2024), volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024, pp. 25:1–25:16. URL: https://arxiv.org/abs/2407.13490. doi:10.4230/LIPIcs.CP.2024.25.
[9] P. Anthropic, Model Context Protocol (MCP), https://modelcontextprotocol.io, 2025. Accessed: 2025-04-20.
[10] L. Hotz, C. Bähnisch, S. Lubos, A. Felfernig, A. Haag, J. Twiefel, Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems, volume 3812 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 91–100.
[11] B. O'Sullivan, Automated modelling and solving in constraint programming, in: 24th AAAI Conference on Artificial Intelligence, AAAI'10, AAAI Press, 2010, pp. 1493–1497.
[12] B. P. Allen, L. Stork, P. Groth, Knowledge Engineering Using Large Language Models, Transactions on Graph Data and Knowledge (TGDK), Special Issue on Trends in Graph Data and Knowledge 1 (2023) 3:1–3:19. doi:10.4230/TGDK.1.1.3.
[13] D. Benavides, S. Segura, A. Ruiz-Cortes, Automated analysis of feature models 20 years later: A literature review, Information Systems 35 (2010) 615–636.
[14] A. Felfernig, G. Friedrich, D. Jannach, UML as domain specific language for the construction of knowledge-based configuration systems, International Journal on Software Engineering and Knowledge Engineering 10 (2000) 449–469.
[15] C. Bessiere, F. Koriche, N. Lazaar, B. O'Sullivan, Constraint acquisition, Artificial Intelligence 244 (2017) 315–342. doi:10.1016/j.artint.2015.08.001.
[16] L. Xu, F. Hutter, H. H. Hoos, K. Leyton-Brown, Satzilla: Portfolio-based algorithm selection for sat, Journal of Artificial Intelligence Research 32 (2008) 565–606.
[17] K. Leyton-Brown, E. Nudelman, Y. Shoham, Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions, in: 8th International Conference on Principles and Practice of Constraint Programming, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 556–572.
[18] E. C. Freuder, Conversational modeling for constraint satisfaction, AAAI Conference on Artificial Intelligence 38 (2024) 22592–22597. URL: https://ojs.aaai.org/index.php/AAAI/article/view/30268. doi:10.1609/aaai.v38i20.30268.
[19] F. Petroni, T. Rocktäschel, P. P. Lewis, A. Bakhtin, Y. Wu, A. Miller, S. Riedel, Language models as knowledge bases?, 2019. URL: https://arxiv.org/pdf/1909.01066.
[20] Y. Ding, A. Poudel, Q. Zeng, T. Weninger, B. Veeramani, S. Bhattacharya, EntGPT: Linking Generative Large Language Models with Knowledge Bases, 2024. URL: https://arxiv.org/abs/2402.06738.
[21] S. Razniewski, A. Yates, N. Kassner, G. Weikum, Language models as or for knowledge bases, 2021. URL: https://arxiv.org/abs/2110.04888.
[22] F. Suchanek, Luu, Knowledge bases and language models: Complementing forces, in: 7th International Joint Conference on Rules and Reasoning, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 3–15.
[23] A. Nayak, H. P. Timmapathini, Llm2kb: Constructing knowledge bases using instruction tuned context aware large language models, 2023. URL: https://arxiv.org/abs/2308.13207.
[24] T. Ahmed, S. Choudhury, LM4OPT: Unveiling the Potential of Large Language Models in Formulating Mathematical Optimization Problems, 2024. URL: https://arxiv.org/abs/2403.01342.
[25] Y. Zhang, S. Mao, T. Ge, X. Wang, A. de Wynter, LLM as a mastermind: A survey of strategic reasoning with large language models, 2023. URL: https://arxiv.org/abs/2404.01230v1. arXiv:2404.01230.
[26] W. Chen, X. Ma, X. Wang, W. W. Cohen, Program of thoughts prompting: Disentangling computa-

tion from reasoning for numerical reasoning tasks, Transactions on Machine Learning Research (2023). URL: https://openreview.net/forum?id=YfZ4ZPt8zd.
[27] P. Clark, O. Tafjord, K. Richardson, Transformers as soft reasoners over language, in: 29th International Joint Conference on Artificial Intelligence, IJCAI'20, 2021, pp. 3882–3890.
[28] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. H. hsin Chi, D. Zhou, Self-consistency improves chain of thought reasoning in language models, ArXiv (2022). URL: https://arxiv.org/abs/2203.11171.
[29] L. Pan, A. Albalak, X. Wang, W. Y. Wang, Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning, 2023. arXiv:2305.12295.
[30] M. Acher, J. G. Duarte, J.-M. Jézéquel, On programming variability with large language model-based assistant, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 8–14. URL: https://doi.org/10.1145/3579027.3608972.
[31] M. Funk, S. Hosemann, J. C. Jung, C. Lutz, Towards ontology construction with language models, 2023. URL: https://arxiv.org/abs/2309.09898.
[32] C. Prud'homme, J.-G. Fages, Choco-solver: A Java library for constraint programming, Journal of Open Source Software 7 (2022) 4708. URL: https://doi.org/10.21105/joss.04708. doi:10.21105/joss.04708, https://github.com/chocoteam/pychoco (PyChoco, Python binding).
[33] Choco Team, PyChoco: Python binding for Choco-solver, https://github.com/chocoteam/pychoco, 2025.
[34] V. Mavroudis, LangChain v0.3, Technical Report, Preprints, 2024. URL: https://doi.org/10.20944/preprints202411.0566.v1. doi:10.20944/preprints202411.0566.v1, preprint, not peer-reviewed.
[35] OpenAI, Learning to Reason with LLMs, https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2025-04-20.
[36] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 24824–24837. URL: https://arxiv.org/abs/2201.11903.
[37] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, Y. Cao, React: Synergizing reasoning and acting in language models, 2023. URL: https://arxiv.org/abs/2210.03629. arXiv:2210.03629.
[38] S. Szeider, MCP-Solver: Integrating language models with constraint programming systems, https://arxiv.org/abs/2501.00539, 2025. ArXiv preprint arXiv:2501.00539, accessed: 2025-05-20.

Addendum

The figures 4 to 12 show one complete chat.

---

system:
You are an agent designed to solve constraint problems, you write python code to solve and verify constraint problems. Your code is run and verified by the user. If you get an error, debug your code and try again. You only use the constraint solver pychoco. You conclude your python code with a variable "solutions" and verify it with the method call "verify_solutions(solutions, <problem_name>", where <problem_name> is given to you as the first word of the instructions. "solutions" should be a list of all solutions to the constraint problem. If verification fails, you correct your solution.
Use the following example as a template for solving constraint problems with pychoco:
User: ints10. find numbers one to 10.
Assistant:

```
```python
from pychoco.model import Model
from test_tool import verify_solutions

model = Model()
x = model.intvar(1, 10, "x")

solver = model.get_solver()
solutions = [s.get_int_val(x) for s in solver.find_all_solutions()]
verify_solutions(solutions, "ints10")
```
```

---

user:
tsp. solve the traveling salesman problem, use the following problem instance:

```
# Number of cities
C = 4

# Distance matrix
D = [[0, 10, 15, 20], [10, 0, 35, 25], [15, 35, 0, 30], [20, 25, 30, 0]]
```

the solution should be a list of valid solutions, each solution being a list of integers representing the cities to be vistited.

---

assistant:

```
```python
from pychoco.model import Model
from test_tool import verify_solutions
```

Figure 4: Complete chat for generating a solution for the TSP problem. Part I.
```

```
C = 4
D = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

model = Model()

# The tour: a permutation of cities, starting at city 0 (fix first city)
tour = model.intvars([0] + [1, C-1], C, "tour")  # tour[0] = 0

# All cities must be visited exactly once
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()

# To break symmetries, fix the first city to 0
# (already done above)

solver = model.get_solver()

solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    # Make the tour a cycle by returning to 0
    if t[0] == 0 and len(set(t)) == C and all(0 <= city < C for city in t):
        solutions.append(t)

verify_solutions(solutions, "tsp")
‘‘‘
```

Figure 5: Complete chat for generating a solution for the TSP problem. Part II.

user:

```
Traceback (most recent call last):
  File "/tmp/tmps5uh3m8t.py", line 15, in <module>
    tour = model.intvars([0] + [1, C-1], C, "tour")  # tour[0] = 0
  File "/home/chbaeh/miniconda3/envs/llmconst/lib/python3.10/site-packages/
      pychoco/variables/variable_factory.py", line 71, in intvars
    return [self.intvar(lb, ub, name) for i in range(0, size)]
TypeError: 'list' object cannot be interpreted as an integer
```

assistant:

```
```python
from pychoco.model import Model
from test_tool import verify_solutions

C = 4
D = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

model = Model()

# The tour: a permutation of cities, starting at city 0 (fix first city)
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]

# All cities must be visited exactly once
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()

solver = model.get_solver()

solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    # Ensure it is a valid tour starting at 0
    if t[0] == 0 and len(set(t)) == C and all(0 <= city < C for city in t):
        solutions.append(t)

verify_solutions(solutions, "tsp")
```
```

Figure 6: Complete chat for generating a solution for the TSP problem. Part III.

user:

2 solutions are correct, 4 are not correct. Expected 2 correct solutions:
2 solutions have minimal cost
6 solutions are hamiltonian
6 solutions start with city 0.

assistant:

```
```python
from pychoco.model import Model
from test_tool import verify_solutions

C = 4
D = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

model = Model()

tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()

# Add cost variable
cost = model.intvar(0, 1000, "cost")
# Express the cost of the tour (cycle)
model.sum(
    [
        model.element(D, tour[i], tour[(i+1)%C])
        for i in range(C)
    ],
    "=", cost
).post()
```

Figure 7: Complete chat for generating a solution for the TSP problem. Part IV.
```

```
# Find the minimal cost
solver = model.get_solver()
min_cost = None
for s in solver.find_all_solutions():
    c = s.get_int_val(cost)
    if min_cost is None or c < min_cost:
        min_cost = c

# Search again, collect only minimal cost tours
model = Model()

tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()
cost = model.intvar(0, 1000, "cost")
model.sum(
    [
        model.element(D, tour[i], tour[(i+1)%C])
        for i in range(C)
    ],
    "=", cost
).post()
model.arithm(cost, "=", min_cost).post()

solver = model.get_solver()
solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    solutions.append(t)

verify_solutions(solutions, "tsp")
‘‘‘
```

Figure 8: Complete chat for generating a solution for the TSP problem. Part V.

user:

```
Traceback (most recent call last):
  File "/tmp/tmpar_b5jgq.py", line 22, in <module>
    [
  File "/tmp/tmpar_b5jgq.py", line 23, in <listcomp>
    model.element(D, tour[i], tour[(i+1)%C])
  File "/home/chbaeh/miniconda3/envs/llmconst/lib/python3.10/site-packages/
      pychoco/constraints/int_constraint_factory.py", line 245, in element
    if len(table) == 0:
TypeError: object of type 'IntVar' has no len()
```

assistant:

```
'''python
from pychoco.model import Model
from test_tool import verify_solutions

C = 4
D = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

model = Model()
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()

# Create variables for each arc's cost
arc_costs = [model.intvar(0, max(max(row) for row in D), f"arc_cost_{i}") for i
    in range(C)]

for i in range(C):
    # arc from tour[i] to tour[(i+1)%C]
    # Use element1d to get D[from][to]
    from_var = tour[i]
    to_var = tour[(i+1)%C]
    for from_city in range(C):
        for to_city in range(C):
            # If from_var==from_city and to_var==to_city then arc_costs[i] == D[
                from_city][to_city]
            model.if_then(
                model.and_(
                    model.arithm(from_var, "=", from_city),
                    model.arithm(to_var, "=", to_city)
                ),
                model.arithm(arc_costs[i], "=", D[from_city][to_city])
            )

cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()
```

Figure 9: Complete chat for generating a solution for the TSP problem. Part VI.

```python
# First, find the minimal cost
solver = model.get_solver()
min_cost = None
for s in solver.find_all_solutions():
    c = s.get_int_val(cost)
    if min_cost is None or c < min_cost:
        min_cost = c

# Now, collect all minimal cost tours
model = Model()
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()
arc_costs = [model.intvar(0, max(max(row) for row in D), f"arc_cost_{i}") for i
    in range(C)]
for i in range(C):
    from_var = tour[i]
    to_var = tour[(i+1)%C]
    for from_city in range(C):
        for to_city in range(C):
            model.if_then(
                model.and_(
                    model.arithm(from_var, "=", from_city),
                    model.arithm(to_var, "=", to_city)
                ),
                model.arithm(arc_costs[i], "=", D[from_city][to_city])
            )
cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()
model.arithm(cost, "=", min_cost).post()
```

Figure 10: Complete chat for generating a solution for the TSP problem. Part VII.

```
solver = model.get_solver()
solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    solutions.append(t)

verify_solutions(solutions, "tsp")
```

---

user:

```
Traceback (most recent call last):
  File "/tmp/tmp8jxdh1xk.py", line 29, in <module>
    model.if_then(
AttributeError: 'Model' object has no attribute 'if_then'
```

---

assistant:

```python
from pychoco.model import Model
from test_tool import verify_solutions

C = 4
D = [
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
]

max_cost = max(max(row) for row in D)

def get_arc_cost_expr(model, from_var, to_var, arc_cost_var):
    # add table constraint for (from_var, to_var, arc_cost_var)
    tuples = []
    for from_city in range(C):
        for to_city in range(C):
            tuples.append([from_city, to_city, D[from_city][to_city]])
    model.table([from_var, to_var, arc_cost_var], tuples).post()

model = Model()
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()

arc_costs = [model.intvar(0, max_cost, f"arc_cost_{i}") for i in range(C)]
for i in range(C):
    get_arc_cost_expr(model, tour[i], tour[(i+1)%C], arc_costs[i])

cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()
```

Figure 11: Complete chat for generating a solution for the TSP problem. Part VIII.

```
# First, find the minimal cost
solver = model.get_solver()
min_cost = None
for s in solver.find_all_solutions():
    c = s.get_int_val(cost)
    if min_cost is None or c < min_cost:
        min_cost = c

# Now, collect all minimal cost tours
model = Model()
tour = [model.intvar(0, C-1, f"tour_{i}") for i in range(C)]
model.all_different(tour).post()
model.arithm(tour[0], "=", 0).post()
arc_costs = [model.intvar(0, max_cost, f"arc_cost_{i}") for i in range(C)]
for i in range(C):
    get_arc_cost_expr(model, tour[i], tour[(i+1)%C], arc_costs[i])
cost = model.intvar(0, 1000, "cost")
model.sum(arc_costs, "=", cost).post()
model.arithm(cost, "=", min_cost).post()

solver = model.get_solver()
solutions = []
for s in solver.find_all_solutions():
    t = [s.get_int_val(tour[i]) for i in range(C)]
    solutions.append(t)

verify_solutions(solutions, "tsp")
‘‘‘
```

---

user:
All solutions are correct

Figure 12: Complete chat for generating a solution for the TSP problem. Part IX.

Towards LLM-enhanced Compiler Optimization

Damian Garber[1,*], Tamim Burgstaller[1], Sebastian Lubos[1], Patrick Ratschiller[1] and Alexander Felfernig[1]

[1]Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

Optimization has always been a central focus in computer science. There are various approaches to achieving this, from finding better algorithms to optimizing compiled code. One such approach is compiler optimization, where we can customize the compiler's configuration to optimize for runtime, energy consumption, or binary size, among other factors. However, these optimizations must be carefully selected for each program and typically require expert knowledge. We utilize compiler autotuning to address this, which automatically selects a program's optimization options. Most current solutions for this task require a significant amount of time. Driven by the growing popularity of AI-assisted coding, we have investigated the potential of Large Language Models (LLMs) as a tool for solving the task of compiler autotuning. We show that LLMs can produce well-performing optimization configurations within a reasonable timeframe acceptable for interactive settings.

Keywords

Compiler Autotuning, Optimization, Large Language Models

1. Introduction

Optimizing code can be achieved through various methods, with compiler optimization being one of the most straightforward approaches. Compilers like the GNU Compiler Collection (GCC)[1] offer over 200 optimization options that can be enabled or disabled. The selection of appropriate optimization options often requires an expert-level understanding. In order to allow non-expert users to use compiler optimization, GCC provides sets of recommended default optimizations depending on the optimization goal. For example, the `-Os` flag contains the recommended set of optimization options to minimize the binary size of the compiled executable. In the following, the most important of the default optimization sets is the `-O3` flag, which optimizes the runtime of the compiled executable. However, these default options may lead to suboptimal results [1]. Compiler autotuning solves this problem by selecting optimization options individually for a given program. The state-of-the-art modern compiler autotuning consists primarily of iterative approaches that consume a significant amount of time due to the need for repeated compilations to generate compiler optimizations, making them not scalable for larger projects. This paper investigates the applicability of using Large Language Models (LLMs) for compiler autotuning. To this end, we use ChatGPT-4o[2] to generate optimized GCC commands and compare their performance with state-of-the-art compiler autotuning approaches.

The remainder of this paper is organized as follows. Section 2 discusses related works on compiler autotuning and LLMs. Section 3 outlines the experimental setup, and Section 4 presents the findings. We address potential threats to validity in Section 5 and explore potential extensions of this work in Section 6. Finally, we present our conclusions in Section 7.

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

*Corresponding author.

damian.garber@tugraz.at (D. Garber); tamim.burgstaller@ist.tugraz.at (T. Burgstaller); sebastian.lubos@tugraz.at (S. Lubos); patrick.ratschiller@ist.tugraz.at (P. Ratschiller); alexander.felfernig@tugraz.at (A. Felfernig)

0009-0005-0993-0911 (D. Garber); 0009-0007-4522-8497 (T. Burgstaller); 0000-0002-5024-3786 (S. Lubos); 0009-0005-5065-5149 (P. Ratschiller); 0000-0003-0108-3146 (A. Felfernig)

[1]https://gcc.gnu.org/

[2]https://openai.com/index/gpt-4o-and-more-tools-to-chatgpt-free/

2. Related Work

The field of compiler autotuning addresses two key challenges: the phase selection problem and the phase ordering problem, both aimed at optimizing program performance [2]. The phase selection problem identifies which optimizations to apply, while the phase ordering problem determines the sequence of these optimizations. This work focuses solely on phase selection. In the modern state-of-the-art, iterative solutions have become the standard approach [3, 4, 5, 6, 7]. Bodin et al. [8] propose one of the earliest iterative approaches. Their approach starts with an initial set of optimization options activated, compiles the program, evaluates its performance, and refines the configuration in a loop until satisfactory results are achieved. Newer approaches focus primarily on increasing the efficiency of iterative approaches. For example, *COBAYN* [9] uses Bayesian Networks to narrow the search space to the most promising configurations. The current state-of-the-art method, *BOCA* [10], employs Bayesian Optimization to identify key optimizations and streamline the search process. *CompTuner* [11] builds a prediction model for the runtime of different optimization options and uses a particle swarm optimization algorithm [12] to improve the search performance. *Cole* [5] can perform multi-target optimization (for example, runtime and energy consumption) by iteratively creating a Pareto front.

However, performance is the central problem for the computationally intensive iterative state-of-the-art approaches, requiring several compilations, which, with increasing project size, becomes a substantial problem. *Cole*, for example, needs to create a Pareto front, which takes 50 days on a single machine [5]. New lightweight approaches such as *Optimization Space Learning (OSL)* [13] try different strategies to achieve a responsive tool that provides optimization options faster, with the trade-off of lower prediction quality. *OSL* combines configuration space learning and collaborative filtering to achieve this. First, *OSL* generates a set of synthesized optimization configurations using a t-wise feature coverage heuristic and measures their performance for multiple benchmarks. *OSL* then recommends optimization configurations for new programs using collaborative filtering [14].

In this work, we explore the applicability of LLMs in the context of compiler autotuning. LLMs have already been used successfully in similar situations. For example, [15] uses a purpose-trained model to minimize the size of the compiled binary, achieving a 3% improvement over the default optimizations and outperforming several state-of-the-art iterative approaches. Another example is [16], which uses LLMs to generate hardware-optimized code, or [17], which proposes the Meta Large Language Model Compiler based on the CodeLLama model.

3. Experimental Setup

We used the following setup to evaluate the applicability of using LLMs in the context of compiler autotuning. We conducted all experiments on a machine running GCC version 11.4.0 on a Xubuntu-22.04 machine with an Intel i7 processor. No multithreading or multiprocessing was applied. We used the most recent release of OpenAI's ChatGPT-4o to generate the GCC command that would minimize the execution time of the resulting binary. To this end, we used the prompt visualized in Figure 1. We considered prompting techniques other than the zero-shot approach, such as few-shot or chain of thought, but they were ultimately disregarded. The few-shot approach is disregarded due to the lack of a dataset containing code and its optimal compiler optimization settings. At the same time, the chain of thought goes directly against the idea of automatization, without expert input, inherent to the concept of compiler autotuning.

We evaluate our results using the PolyBench[3] benchmarks, commonly employed in compiler autotuning evaluation. We run the prompt shown in Figure 1 for each of the 30 benchmarks, exchanging the "{Code}" with the full content of the respective C file for each benchmark. We use the framework

[3]https://github.com/MatthiasJReisinger/PolyBenchC-4.2.1/tree/master

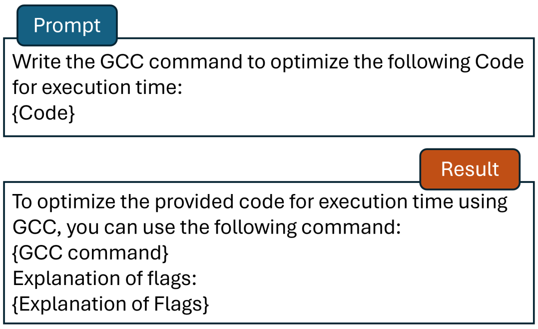


Figure 1: The prompt used to generate GCC commands and its result.

used by *OSL*[4], another compiler autotuning approach, to evaluate the performance of the generated GCC command [13]. The conversion to the *OSL* framework means that some optimization options, for example, hardware architecture-specific optimizations such as `-march=native`, are intentionally discarded. Discarding these options minimizes the influence of system-specific behavior and thus leads to more general results [13]. These results are then compared to the performance of the GCC command using `-O3` for the same program similarly converted to the *OSL* framework. The execution time of the binaries generated by both commands is measured using perf stat[5] and the speedup of the LLM generated GCC command (t_{LLM}) against the `-O3` GCC command (t_{O3}) is calculated using (1).

$$speedup = \frac{t_{O3}}{t_{LLM}} \tag{1}$$

4. Results

First, we investigate the LLM-generated optimization results on its own, and in the second step, we compare the results with other state-of-the-art alternatives.
We measured an average speedup of 1.020 when using the LLM-generated GCC command compared to the default optimization settings of the `-O3` GCC command over the 30 benchmarks tested. The median is marginally higher, with a speedup of 1.021 and a standard deviation of 0.046. We provide a histogram in Figure 2 to further visualize these results.

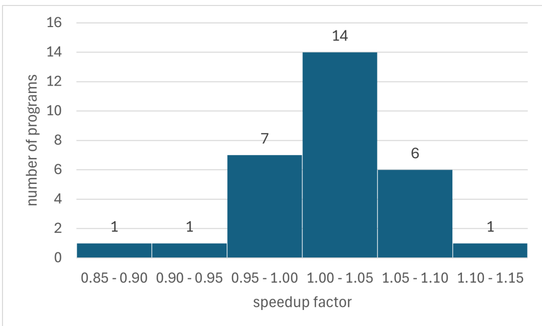


Figure 2: Histogram showing the performance speedups achieved by the LLM optimizations compared to `-O3`.

The time needed to generate the GCC commands is, on average, 8.96s. These results show the potential of an LLM-supported compiler autotuning approach, as it outperforms the default GCC optimization in 21 out of 30 tested benchmarks while needing a reasonable time.

[4]https://github.com/AIG-ist-tugraz/OptimizationSpaceLearning
[5]https://perf.wiki.kernel.org/index.php/Main_Page

We now compare our LLM-based approach with other state-of-the-art compiler autotuning approaches, more precisely eight other approaches, which are shown in Table 1.

Table 1
List of approaches used for comparison and their references

Approach	Reference
OSL	[13]
CompTuner	[11]
BOCA	[10]
TPE	[10]
Random Iterative Compilation (RIO)	[4]
Genetic Algorithms (GA)	[6]
OpenTuner	[18]
COBAYN	[9]

To allow for a direct comparison with the other approaches, we only visualize 10 of the 30 benchmarks provided by Polybench, as was done by [13, 10, 11]. The ten programs selected are listed in Table 2. We adapted Table 2 from a table provided by [13]. We compare the speedup of our results in Table 3 and the time to generate these results in Table 4 with the alternatives. The other results were taken from a table provided by [13] and extended with our results. We discuss the use of external data in Section 5.

We will first discuss the time needed to generate the results shown in Table 4. We can split the results into three categories. *OSL* provides the first and fastest in the single-digit millisecond range. Our approach provides the second fastest results in the single-digit second range. The remaining approaches operate in a range of several thousand seconds. Thus, we can conclude that *OSL* outperforms all other approaches in speed by an order of magnitude. However, while outperformed by *OSL*, our approach is still an order of magnitude faster than the other state-of-the-art approaches. It allows for a reasonably fast response for direct user interaction.

Regarding the speedup of the compiled code, we outperform the state of the art for the programs P4 and P8. We can only compare individual results for most alternatives since they usually calculate overall results using additional programs on top of the benchmark set used here or only use parts of it. *BOCA* [10], for example, calculates its overall performance using only 10 of the 30 programs from PolyBench, in addition to 10 programs from another benchmark, claiming that no significant speedup can be achieved for the remaining 20 programs. In our case, the average speedup increases from 1.020 to 1.026 when using only the 10 programs compared to the entire benchmark. The only directly comparable approach is *OSL*, which reports an average speedup of 0.994 over the entire benchmark. Our results outperform these results significantly, averaging a speedup of 1.020.

5. Threats to Validity

This work represents a proof of concept, exploring the potential use of LLMs in compiler autotuning. We demonstrated that the optimizations generated by LLMs could outperform default optimizations on average.

Several factors could have influenced the results of this work, but they were not within the scope of this study. Firstly, we utilized an externally hosted LLM, which could have affected result generation speed. We anticipate that using a locally hosted model would yield faster results. Secondly, we

Table 2
The list of programs from PolyBench used for the comparison with other approaches

ID	Program	#SLOC	Description
P1	correlation	248	Correlation computation
P2	covariance	218	Covariance computation
P3	symm	231	Symmetric matrix-multiply
P4	2mm	252	2 matrix multiplications
P5	3mm	267	3 matrix multiplications
P6	cholesky	212	Cholesky decomposition
P7	lu	210	LU decomposition
P8	nussinov	569	DP for sequence alignment
P9	heat-3d	211	Heat equation (3D data dom.)
P10	jacobi-2d	200	2-D Jacobi stencil comp.

Table 3
The speedup of the programs in Table 2 compared to -O3 as defined in (1). The best speedup is marked in bold font, while "-" denotes no speedup (...) denotes external data.

Technique	ID	Speedup	ID	Speedup	ID	Speedup	ID	Speedup	ID	Speedup
LLM	P1	1.019	P2	-	P3	-	P4	**1.109**	P5	1.021
OSL		1.000		1.043		-		-		-
CompTuner		**1.077** (...)		**1.080** (...)		1.042 (...)		1.071 (...)		1.041 (...)
BOCA		-		-		**1.075** (...)		1.071 (...)		**1.046** (...)
TPE		-		-		1.046 (...)		1.072 (...)		-
RIO		-		-		1.042 (...)		-		-
GA		-		-		-		-		1.041 (...)
OpenTuner		-		-		-		1.075 (...)		-
COBAYN		-		**1.080** (...)		1.068 (...)		1.079 (...)		-
LLM	P6	1.025	P7	1.046	P8	**1.057**	P9	-	P10	1.050
OSL		1.010		1.016		-		**1.109**		-
CompTuner		1.013 (...)		1.073 (...)		1.029 (...)		1.025 (...)		**1.055** (...)
BOCA		1.014 (...)		-		1.030 (...)		1.028 (...)		**1.055** (...)
TPE		-		-		-		1.027 (...)		-
RIO		1.016 (...)		-		1.029 (...)		-		-
GA		1.013 (...)		-		-		1.025 (...)		-
OpenTuner		-		**1.075** (...)		1.033 (...)		-		-
COBAYN		**1.064** (...)		-		-		1.028 (...)		-

employed ChatGPT-4o, a general model. We expect a model trained explicitly for this purpose to yield superior results.

Furthermore, we only calculated the non-iterative approaches and sourced the results for the iterative approaches externally, recognizing that this may introduce distortions. This step was necessary because only around half of the approaches made their code publicly available, and the calculation of results would have taken several days per program per approach. The distortion is mitigated by comparing the relative speedup of two optimizations tested on the same machine rather than directly comparing the runtime of the selected benchmarks. Although comparing the time to calculate an optimization directly can lead to issues, in our case, the time differences are so significant that we consider any distortions negligible for the comparisons.

6. Future Work

We see future extensions of this work go in three principal directions. The first is increasing the prediction performance of the used LLM by creating a purpose-trained model dedicated to compiler

Table 4
The time needed to generate the GCC command of the programs in Table 2. The best speedup is marked in bold font, while "-" denotes no speedup and is thus disregarded (...) denotes external data.

Technique	ID	Time [s]	ID	Time [s]	ID	Time [s]	ID	Time [s]	ID	Time [s]
LLM		7.35		-		-		**7.78**		**7.96**
OSL		**0.0043**		**0.0039**		-		-		-
CompTuner		3107.00 (...)		4067.00 (...)		2573.00 (...)		3720.00 (...)		2976.00 (...)
BOCA		-		-		1923.00 (...)		3726.00 (...)		3639.00 (...)
TPE	P1	-	P2	-	P3	3775.00 (...)	P4	3112.00 (...)	P5	-
RIO		-		-		4172.00 (...)		-		-
GA		-		-		-		-		3160.00 (...)
OpenTuner		-		-		-		4691.00 (...)		-
COBAYN		-		4727.00 (...)		**1092.00** (...)		3102.00 (...)		-
LLM		11.76		5.64		**10.04**		-		**10.61**
OSL		**0.0039**		**0.0048**		-		**0.0040**		-
CompTuner		4726.00 (...)		5549.00 (...)		3661.00 (...)		2976.00 (...)		2192.00 (...)
BOCA		4971.00 (...)		-		4082.00 (...)		3420.00 (...)		3026.00 (...)
TPE	P6	-	P7	-	P8	-	P9	2637.00 (...)	P10	-
RIO		3018.00 (...)		-		3264.00 (...)		-		-
GA		3862.00 (...)		-		-		3684.00 (...)		-
OpenTuner		-		6792.00 (...)		4970.00 (...)		-		-
COBAYN		3109.00 (...)		-		-		4116.00 (...)		-

optimization. While cost-intensive in data and processing power, we expect such an endeavor to show significantly improved results, allowing a fast solution while still providing high-quality results. However, extending this approach to other models such as Gemini 2.5 Pro[6], Claude 4.0 Opus[7], or Codestral[8] is likely more cost-effective than training a completely new model and is very likely to yield improvements.

Another research direction would be to integrate this with the fast-emerging AI coding tools like GitHub's Copilot[9], JetBrains' AI Assistant[10], or CodeCompanion[11]. These tools are directly embedded into the Integrated Development Environment (IDE) and are already fully aware of the complete code base. Thus, they would be in a perfect environment to predict compiler optimizations. Additionally, this leads to the possible applicability of our approach to more extensive projects, for which most of the state-of-the-art is not suited.

Lastly, this work could be extended by including compiler optimization experts, both for creating datasets and prompts that could be used to enhance the approach directly, or to compare their recommended optimization options with the results produced by this and other compiler autotuning approaches.

7. Conclusion

This paper shows the applicability of using LLMs in compiler autotuning. The compiler optimizations generated using ChatGPT-4o for the GCC compiler improved the tested benchmark's runtime on average by a factor of 1.020 while taking an average of 8.96s to generate the optimizations. We outperform the state-of-the-art approaches in 2 out of 10 benchmarks while performing an order of magnitude faster. These results suggest that this approach is scalable also for large projects, a significant shortcoming of

[6]https://ai.google.dev/gemini-api/docs/models#gemini-2.5-pro
[7]https://www.anthropic.com/news/claude-4
[8]https://mistral.ai/news/codestral
[9]https://github.com/features/copilot
[10]https://www.jetbrains.com/ai/
[11]https://codecompanion.ai/

the existing iterative state-of-the-art approaches.

Acknowledgments

This study was funded by GENRE, Austrian Research Promotion Agency (Grant No. 915086).

Declaration on Generative AI

While preparing this work, the author(s) used ChatGPT-4 (GPT-4-turbo) and Grammarly to check grammar and spelling and improve formulations. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] J. Gong, T. Chen, Deep configuration performance learning: A systematic survey and taxonomy, arXiv preprint arXiv:2403.03322 (2024).

[2] A. H. Ashouri, W. Killian, J. Cavazos, G. Palermo, C. Silvano, A survey on compiler autotuning using machine learning, ACM Computing Surveys (CSUR) 51 (2018) 1–42.

[3] S. Triantafyllis, M. Vachharajani, N. Vachharajani, D. I. August, Compiler optimization-space exploration, in: International Symposium on Code Generation and Optimization, 2003. CGO 2003., IEEE, 2003, pp. 204–215.

[4] Y. Chen, S. Fang, Y. Huang, L. Eeckhout, G. Fursin, O. Temam, C. Wu, Deconstructing iterative optimization, ACM Transactions on Architecture and Code Optimization (TACO) 9 (2012) 1–30.

[5] K. Hoste, L. Eeckhout, Cole: compiler optimization level exploration, in: Proceedings of the 6th annual IEEE/ACM International Symposium on Code Generation and Optimization, 2008, pp. 165–174.

[6] U. Garciarena, R. Santana, Evolutionary optimization of compiler flag selection by learning and exploiting flags interactions, in: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, 2016, pp. 1159–1166.

[7] L. Pérez Cáceres, F. Pagnozzi, A. Franzin, T. Stützle, Automatic configuration of gcc using irace, in: Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13, Springer, 2018, pp. 202–216.

[8] F. Bodin, T. Kisuki, P. Knijnenburg, M. O'Boyle, E. Rohou, Iterative compilation in a non-linear optimisation space, in: Workshop on profile and feedback-directed compilation, 1998.

[9] A. H. Ashouri, G. Mariani, G. Palermo, E. Park, J. Cavazos, C. Silvano, Cobayn: Compiler autotuning framework using bayesian networks, ACM Transactions on Architecture and Code Optimization (TACO) 13 (2016) 1–25.

[10] J. Chen, N. Xu, P. Chen, H. Zhang, Efficient compiler autotuning via bayesian optimization, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 1198–1209.

[11] M. Zhu, D. Hao, J. Chen, Compiler autotuning through multiple phase learning, ACM Trans. Softw. Eng. Methodol. (2024). URL: https://doi.org/10.1145/3640330. doi:10.1145/3640330, just Accepted.

[12] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, IEEE, 1995, pp. 1942–1948.

[13] T. Burgstaller, D. Garber, V.-M. Le, A. Felfernig, Optimization space learning: A lightweight, noniterative technique for compiler autotuning, in: Proceedings of the 28th ACM International Systems and Software Product Line Conference, 2024, pp. 36–46.

[14] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, et al., Collaborative filtering recommender systems, Foundations and Trends® in Human–Computer Interaction 4 (2011) 81–173.

[15] C. Cummins, V. Seeker, D. Grubisic, M. Elhoushi, Y. Liang, B. Roziere, J. Gehring, F. Gloeckle, K. Hazelwood, G. Synnaeve, et al., Large language models for compiler optimization, arXiv preprint arXiv:2309.07062 (2023).
[16] C. Hong, S. Bhatia, A. Haan, S. K. Dong, D. Nikiforov, A. Cheung, Y. S. Shao, Llm-aided compilation for tensor accelerators, arXiv preprint arXiv:2408.03408 (2024).
[17] C. Cummins, V. Seeker, D. Grubisic, B. Roziere, J. Gehring, G. Synnaeve, H. Leather, Meta large language model compiler: Foundation models of compiler optimization, arXiv preprint arXiv:2407.02524 (2024).
[18] J. Ansel, S. Kamil, K. Veeramachaneni, J. Ragan-Kelley, J. Bosboom, U.-M. O'Reilly, S. Amarasinghe, Opentuner: An extensible framework for program autotuning, in: Proceedings of the 23rd international conference on Parallel architectures and compilation, 2014, pp. 303–316.

Towards Compiler Parameter Recommendation Using Code Embeddings

Damian Garber[1,*], Sebastian Lubos[1] and Alexander Felfernig[1]

[1]*Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria*

Abstract

We present a lightweight compiler autotuning approach that combines concepts from configuration space learning with recommender techniques. Our approach uses code embeddings generated by different large language models for data representation and calculation of similarity scores. The best-performing code embedding approach shows, on average, 4.11% faster binaries than the best-performing code metric-based alternative.

Keywords

Compiler Autotuning, Code Embeddings, Collaborative Filtering, Code Metrics

1. Introduction

Compilers are powerful and highly configurable tools. The C compiler GCC[1] has about 200 optimization options that can be activated or deactivated independently. Each option may positively or negatively impact different properties, such as the generated binary's runtime, size, or energy consumption. If these options are correctly utilized, the generated program binaries can be faster, smaller, or more energy-efficient without further investing resources into code refinement. However, choosing the correct options requires expertise in compiler optimization and the program to be optimized. Compiler autotuning addresses this issue by recommending optimization options for a program without any expert involvement. Most approaches for compiler autotuning are computationally expensive and take days to continuously refine the recommended options [1, 2, 3, 4, 5, 6]. Alternative lightweight approaches for compiler autotuning proposed by Burgstaller et al. [7] and Garber et al. [8], can reduce the time needed for recommendation to milliseconds allowing an interactive user experience. This lightweight approach is called Optimisation Space Learning (OSL) [7] and relies on training data collected in advance that is then used for recommendation utilizing nearest-neighbor-based collaborative filtering [9] based on extracted code metrics. The major contributions of this paper are as follows: (1) We extend OSL by incorporating and comparing different code embeddings. (2) We show that the new embeddings significantly outperform the standard compiler optimization options in terms of the runtime performance of the generated program.

The remainder of this paper is structured as follows. Related work is presented in Section 2. In Section 3, our recommendation approach is discussed in detail. We discuss our experimental setup and the evaluation in Section 4, while discussing possible future extensions in Section 5. The paper is concluded with Section 6.

2. Related Work

Compiler autotuning is the automated selection of advantageous compiler optimization options for a program. It can be divided into the *phase selection* problem and the *phase ordering* problem [10]. Phase ordering tries to find an optimal sequence to apply the options, while phase selection, the focus of this

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
damian.garber@tugraz.at (D. Garber); sebastian.lubos@tugraz.at (S. Lubos); alexander.felfernig@tugraz.at (A. Felfernig)
0009-0005-0993-0911 (D. Garber); 0000-0002-5024-3786 (S. Lubos); 0000-0003-0108-3146 (A. Felfernig)

[1]https://gcc.gnu.org/

work, tries to identify which optimizations should be applied. The optimality of options can be defined with different properties, the most common of which is runtime. However, space, energy, or similar measurable properties could also be employed.

The state-of-the-art in compiler autotuning is primarily dominated by iterative approaches [1, 2, 3, 4, 5, 6]. Bodin et al. [11] propose one of the first compiler autotuning approaches. They generate an initial set of optimization options to be activated, compile the program using these options, measure its performance, and refine the configuration in a loop until achieving satisfactory results. Most newer approaches build on this concept, like COBAYN [12], which uses Bayesian Networks to narrow the search space. The current state-of-the-art method, BOCA [13], employs Bayesian Optimization to identify key optimizations and streamline the search process. CompTuner [14] builds a prediction model for the runtime of different optimization options and uses a particle swarm optimization algorithm [15] to improve the search performance. Cole [4] can perform multi-target optimization by iteratively creating a Pareto front.

Performance is a key challenge for the computationally intensive, iterative, state-of-the-art approaches, as they require numerous compilations. As project sizes grow, this becomes a significant issue. For instance, Cole must construct a Pareto front, which takes 50 days on a single machine [4]. To address these limitations, newer lightweight approaches like Optimization Space Learning (OSL) [7] adopt alternative strategies to provide faster optimization recommendations, trading off a small degree of recommendation quality for improved responsiveness.

OSL achieves this by combining configuration space learning [16, 17] techniques like the t-wise feature coverage heuristic [16, 17, 18] with collaborative filtering [9]. In this context, collaborative filtering relies on code metrics (e.g., McCabe, Halstead, or counts of keywords) extracted from the optimized programs. This paper presents an alternative collaborative filtering approach based on code embeddings [19].

3. Recommendation Approach

Optimization Space Learning (OSL) is a compiler autotuning approach introduced initially by Burgstaller et al. [7]. The approach combines concepts from configuration space learning [16, 17] for data generation and collaborative filtering [9] for configuration recommendation. The key contribution of OSL is its recommendation speed, which is achieved after a one-time collection of training data within tens of milliseconds. Meanwhile, the iterative state-of-the-art compiler autotuning approaches [1, 2, 3, 4, 5, 6], report computation times of several days. These differences are due to the iterative approaches requiring a continuous refinement process of recommendation result testing, adaptation, and restarting.

3.1. Data Collection

OSL needs to collect initial training data to provide recommendations for a new hardware environment. Two decisions have to be made to generate the training data. The first is which programs to use for training. The second is the heuristic used for generating the sample configurations.

The training approach is based on configuration space learning [16, 20, 17], motivated by the infeasibility of exhaustively exploring configuration spaces due to their exponential size [21]. For example, GCC includes around 200 options, yielding a configuration space of roughly 2^{200} configurations. Even assuming $1ms$ per compilation and measurement, full exploration would take $5 * 10^{49}$ years. Therefore, following Pereira et al. [16], collecting a small, representative configuration subset is necessary.

In order to collect such a small representative set of configurations, we use sampling approaches discussed by Pereira et al. [16] and Garber et al. [17]. Burgstaller et al. [7] considered initially two sampling approaches: Uniform Random Sampling (URS) and t-wise Feature Coverage Heuristics (FCH). URS is well-established [16, 17, 22, 23, 24], but has drawbacks with scalability. The main drawback of FCH, on the other hand, is its expensive computation, which is mitigated by the unconstrained nature of the problem (options are independent of each other) and the fact that this needs to be performed only once. Therefore, OSL ultimately relies on the t-wise FCH [16, 17] to generate the samples for the

Table 1
Example user-based collaborative filtering recommendation setting in compiler autotunning. The values show programs' runtime [s] when compiled with the referenced compiler configuration c_i.

Programs	c_1	c_2	c_3	c_4	c_5	c_6
p_1	1.21	1.70	1.27	1.19	1.76	1.32
p_2	25.01	18.69	16.32	17.06	16.45	16.47
p_3	0.50	0.56	0.48	0.54	0.74	0.73
p_4	?	?	?	?	?	?

training data. The samples generated by t-wise FCH are guaranteed to contain all possible tuples of size t that can be present in the system at least once in the generated configurations. Burgstaller et al. [7] report FCH with $t = 3$ to perform best for this task, which is confirmed by Garber et al. [8] and our work presented in this paper. Next, we need a set of programs to synthesize the needed data. We use the same benchmark used in the original work by Burgstaller et al. [7] and in the improved OSL by Garber et al. [8]. The PolyBench benchmarks [25] provide 30 programs written in C and are widely used in related literature [7, 13, 14, 8].

We construct the training data by compiling an executable for each configuration provided by the sampling approach and each program in the benchmark. The performance properties of these executables are then measured using *perf-stat*.[2]

OSL extracts at this point a vector of 111 source code metrics, such as McCabe's Cyclomatic Complexity [26], Halstead Complexity [27], or simple counts like the number of times a particular keyword occurs, using the CQMetrics tool by [28]. OSL uses the first 66 of those source code metrics to calculate program similarities during the recommendation process since the latter metrics primarily are related to coding style, i.e., indentation space counts. A complete list of the metrics extracted is provided in the CQMetrics documentation.[3] We compare the performance of this code metric-based similarity with our approach of using code embeddings-based similarity. A description of the code embeddings used is provided in Section 4.

3.2. Recommendation

Essentially, we apply nearest neighbor-based collaborative filtering [9] on synthesized data [29, 22], which has been obtained using heuristics known from configuration space learning [16, 17].

Our variant of user-based collaborative filtering differs slightly from the standard setting (see Table 1). Here, programs act as users, configurations as items, and runtime serves as the rating (a lower runtime is analogous to a higher rating). Unlike typical scenarios, we have complete performance data for all program-configuration pairs generated by the data collection process, except for the target program. Thus, we require an external metric to estimate program similarity. The version of OSL used by Burgstaller et al. [7] and Garber et al. [8] computes similarity using source code metrics and the Euclidean distance [30, 7] (see Formula 1 and Formula 2). In this context, x and y are n-dimensional vectors with components x_1 to x_n and y_1 to y_n, representing programs x and y. In OSL, these vectors consist of $n = 66$ code metrics, while in our approach, they are the extracted fixed-size (n) embedding vectors.

$$dis(x, y) = \sum_{i=1}^{n} |y_i - x_i|^2 \tag{1}$$

$$sim(x, y) = \frac{1}{1 + dis(x, y)} \tag{2}$$

Table 2 shows a simplified example of the code metric vectors used, and Table 3 shows an example of how the distances and similarities, as defined in Formula 1 and Formula 2 respectively, would look like

[2] https://perf.wiki.kernel.org/index.php/Main_Page
[3] https://github.com/dspinellis/cqmetrics/blob/master/metrics.md

Table 2
Simplified example of code metric vector, limited to McCabe [26], Halstead [27], and occurrences of *const*.

Program	Halstead	McCabe	Keywordcount: *const*
p_1	110	10	4
p_2	111	7	6
p_3	108	9	4
p_4	104	13	2

Table 3
Example distance and similarity calculations based on metrics shown in Table 2.

x	y	$dis(x, y)$	$sim(x, y)$	$sim(x, y)$ [%]
p_1	p_2	14	0.067	6.7 %
p_1	p_3	5	0.167	16.7 %
p_1	p_4	49	0.020	2.0 %

Table 4
Code embeddings tested for recommendation.

Name	Description
BGE	A BAAI general embedding model that transforms any given English text into a compact vector
RoBERTa	A DistilRoBERTa-base model trained for code search

Table 5
Example aggregation of three well-performing compiler parameter configurations using a parameter-wise majority voting for final recommendation.

	opt_1	opt_2	opt_3	opt_4	opt_5
$conf_1$	1	1	0	1	0
$conf_2$	0	1	0	0	1
$conf_3$	1	0	1	0	0
recommendation	1	1	0	0	0

for these values. In the example, the highest similarity is 16.7 % between p_1 and p_3. Thus, p_3 is the most similar program to p_1. We propose the use of code embeddings extracted from the programs instead. To this end, we test the performance of two embeddings, shown in Table 4. After testing several common ways of calculating the similarities of two embedding vectors, such as the cosine similarity, we use the same Euclidian distance-based approach described earlier.

The remaining process is identical to the typical user-based collaborative filtering procedure. The best-rated (fastest runtime) configuration of the most similar program recommends a configuration for p_4.

The final recommendation step aggregates the results. Since the FCH-collected configurations cover only a small subset of all compiler settings, we generate multiple recommendations from the nearest neighbors and combine them via majority vote (Table 5). Following Burgstaller et al. [7], we set the number of top configurations and nearest neighbors to 5, a choice we confirmed and applied to all experiments. Thus, the final recommendation aggregates the 5 best configurations from the 5 nearest neighbors.

Table 6
The runtime performance (RT) of recommended optimization options and their speedup (SU) compared to O3. The runtime is given in seconds, and the best performers for each program are bold.

Program	O3	OSL		OSL N&E		BGE		RoBERTa	
	RT	RT	SU	RT	SU	RT	SU	RT	SU
correlation	1.841	1.781	**1.034**	1.868	0.985	1.804	1.021	1.840	1.001
covariance	1.817	2.471	0.735	2.502	0.726	1.903	0.955	1.857	**0.979**
2mm	2.163	2.181	0.992	3.149	0.687	2.187	0.989	2.099	**1.031**
3mm	3.795	3.684	1.030	3.885	0.977	3.664	**1.036**	3.783	1.003
atax	0.016	0.014	1.137	0.013	**1.177**	0.015	1.074	0.014	1.161
bicg	0.017	0.017	**0.968**	0.020	0.847	0.019	0.894	0.018	0.929
doitgen	0.521	0.503	1.034	0.505	1.031	0.498	**1.045**	0.499	1.044
mvt	0.018	0.018	**1.027**	0.020	0.928	0.019	0.972	0.019	0.965
gemm	1.153	1.149	1.003	0.644	1.790	1.158	0.995	0.554	**2.080**
gemver	0.026	0.024	1.087	0.026	0.988	0.024	1.063	0.023	**1.129**
gesummv	0.012	0.012	**0.985**	0.013	0.899	0.014	0.837	0.017	0.722
symm	1.625	1.619	1.003	1.615	1.006	1.614	**1.007**	1.646	0.987
syr2k	1.789	1.776	1.007	1.742	1.027	1.864	0.960	1.739	**1.029**
syrk	0.613	0.531	1.154	0.453	1.354	0.489	1.253	0.418	**1.466**
trmm	1.336	0.721	1.853	0.724	1.844	0.727	1.839	0.709	**1.884**
cholesky	12.856	12.015	1.070	12.727	1.010	12.495	1.029	10.741	**1.197**
durbin	0.004	0.003	1.288	0.003	1.260	0.002	1.898	0.002	**1.952**
gramschmidt	1.950	1.958	0.996	2.022	0.965	1.893	**1.030**	1.923	1.014
lu	16.486	13.676	**1.205**	14.898	1.107	14.915	1.105	14.963	1.102
ludcmp	13.993	12.799	**1.093**	16.469	0.850	14.546	0.962	14.384	0.973
trisolv	0.007	0.009	0.784	0.006	1.134	0.006	**1.204**	0.007	1.120
deriche	0.149	0.140	1.066	0.146	1.023	0.139	**1.073**	0.143	1.042
floyd-warshall	17.773	17.656	1.007	17.790	0.999	13.671	**1.300**	14.022	1.267
nussinov	3.418	2.751	1.242	3.129	1.093	2.665	**1.283**	2.721	1.256
adi	9.659	9.347	**1.033**	9.876	0.978	9.709	0.995	9.783	0.987
fdtd-2d	1.908	1.702	1.121	1.671	1.142	1.697	1.125	1.667	**1.145**
heat-3d	3.479	2.172	1.602	1.949	**1.784**	2.096	1.660	2.064	1.685
jacobi-1d	0.002	0.001	1.424	0.001	**1.515**	0.002	1.169	0.002	1.138
jacobi-2d	2.143	1.445	**1.483**	1.487	1.441	1.479	1.449	1.487	1.441
seidel-2d	20.167	15.467	1.304	11.566	**1.744**	20.075	1.005	20.084	1.004

Table 7
We present different aggregations (mean, median, TOP 1, and TOP 2) of the results shown in Table 6.

Function	OSL	OSL N&E	BGE	RoBERTa
MEAN	1.126	1.144	1.141	**1.191**
MEDIAN	1.050	1.025	1.040	**1.073**
TOP 1	8/30	4/30	8/30	**10/30**
TOP 2	16/30	11/30	14/30	**19/30**

4. Evaluation

In this section, we evaluate the use of code embeddings to recommend compiler optimization options and whether they outperform code metric-based approaches like OSL [7] or its enhanced version of OSL Normalized and Equalized (OSL N&E) [8]. Code embeddings represent code as fixed-sized numerical vectors containing semantic and structural information [19]. They are usually employed by machine learning or large language models when working with code. We test two embeddings BGE [4], a general

[4]https://huggingface.co/BAAI/bge-base-en-v1.5

text embedding, and RoBERTa [5], a specialized code embedding (described in Table 4).

4.1. Experimental Setup

Our evaluation uses GCC version 14.2.1 on a Lenovo ThinkPad P53s machine with an Intel i7-8665U processor and 32GB memory running Linux 6.1.119-1-MANJARO. We use the PolyBench/C Benchmark [25] for training and testing, which contains 30 programs written in C and is commonly used in compiler autotuning evaluation settings [7, 13, 14, 8]. Due to the relatively small sample size, we apply leave-one-out cross-validation [31]. Thus, each benchmark program in PolyBench was tested using a model trained with the remaining 29. In order to visualize the performance more efficiently, we define a speedup factor compared to GCC's set of default optimizations O3 in Equation 3.

$$speedup = \frac{t_{O3}}{t_{REC}} \tag{3}$$

t_{O3} and t_{REC} represent the program's runtime compiled using O3 and the recommended parameter settings respectively. A speedup of 1.1 indicates a 1.1 times faster runtime.

4.2. Results

Table 6 presents the performance of the tested approaches, with aggregated results in Table 7. Both code embeddings outperformed the baseline OSL method, which achieved an average speedup of 1.126. BGE reached 1.141, slightly below the enhanced OSL N&E at 1.144. RoBERTa achieved the highest average speedup of 1.191. Regarding frequency as a top performer, RoBERTa leads (Top 1 in 10/30 cases, Top 2 in 19/30), followed by BGE narrowly outperforming OSL, while OSL N&E comes last. These results indicate that embeddings are effective for recommending compiler optimizations, especially when using models like RoBERTa, which are specifically trained on code.

5. Future Work

The first results of using code embeddings in the context of lightweight compiler autotuning show promise. However, in future work, we would like to expand the number of evaluated code embeddings, especially further towards models specialized in coding or code manipulations, such as CodeBERT or GraphCodeBERT, potentially improving our results further.

6. Conclusion

In summary, we evaluated using code embeddings to recommend compiler optimizations. Our results show that embeddings perform comparably to code metric-based approaches and surpass them in the case of embeddings from models trained on code. The best-performing method leverages embeddings from a RoBERTa model trained for code search, achieving an average runtime speedup factor of 1.191, 4.11% faster than the enhanced code metrics baseline. Major tasks of future work include the extension of the dataset as well as the testing of additional embeddings.

Acknowledgments

This study was funded by GENRE, Austrian Research Promotion Agency (Grant No. 915086).

[5]https://huggingface.co/flax-sentence-embeddings/st-codesearch-distilroberta-base

Declaration on Generative AI

While preparing this work, the author(s) used ChatGPT-4 (GPT-4-turbo) and Grammarly to check grammar and spelling and improve formulations. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] Y. Chen, S. Fang, Y. Huang, L. Eeckhout, G. Fursin, O. Temam, C. Wu, Deconstructing iterative optimization, ACM Transactions on Architecture and Code Optimization (TACO) 9 (2012) 1–30.

[2] U. Garciarena, R. Santana, Evolutionary optimization of compiler flag selection by learning and exploiting flags interactions, in: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, 2016, pp. 1159–1166.

[3] S. V. Gheorghita, H. Corporaal, T. Basten, Iterative compilation for energy reduction, Journal of Embedded Computing 1 (2005) 509–520.

[4] K. Hoste, L. Eeckhout, Cole: compiler optimization level exploration, in: Proceedings of the 6th annual IEEE/ACM international symposium on Code generation and optimization, 2008, pp. 165–174.

[5] L. Pérez Cáceres, F. Pagnozzi, A. Franzin, T. Stützle, Automatic configuration of gcc using irace, in: Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13, Springer, 2018, pp. 202–216.

[6] S. Triantafyllis, M. Vachharajani, N. Vachharajani, D. I. August, Compiler optimization-space exploration, in: International Symposium on Code Generation and Optimization, 2003. CGO 2003., IEEE, 2003, pp. 204–215.

[7] T. Burgstaller, D. Garber, V.-M. Le, A. Felfernig, Optimization space learning: A lightweight, noniterative technique for compiler autotuning, in: Proceedings of the 28th ACM International Systems and Software Product Line Conference, SPLC '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 36–46. URL: https://doi.org/10.1145/3646548.3672588. doi:10.1145/3646548.3672588.

[8] D. Garber, S. Lubos, V.-M. Le, A. Felfernig, Enhanced optimization space learning: Towards real-time compiler optimization, in: 38th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2025. Accepted.

[9] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, et al., Collaborative filtering recommender systems, Foundations and Trends® in Human–Computer Interaction 4 (2011) 81–173.

[10] A. H. Ashouri, W. Killian, J. Cavazos, G. Palermo, C. Silvano, A survey on compiler autotuning using machine learning, ACM Computing Surveys (CSUR) 51 (2018) 1–42.

[11] F. Bodin, T. Kisuki, P. Knijnenburg, M. O'Boyle, E. Rohou, Iterative compilation in a non-linear optimisation space, in: Workshop on profile and feedback-directed compilation, 1998.

[12] A. H. Ashouri, G. Mariani, G. Palermo, E. Park, J. Cavazos, C. Silvano, Cobayn: Compiler autotuning framework using bayesian networks, ACM Transactions on Architecture and Code Optimization (TACO) 13 (2016) 1–25.

[13] J. Chen, N. Xu, P. Chen, H. Zhang, Efficient compiler autotuning via bayesian optimization, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 1198–1209.

[14] M. Zhu, D. Hao, J. Chen, Compiler autotuning through multiple phase learning, ACM Trans. Softw. Eng. Methodol. (2024). URL: https://doi.org/10.1145/3640330. doi:10.1145/3640330, just Accepted.

[15] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-international conference on neural networks, volume 4, IEEE, 1995, pp. 1942–1948.

[16] J. Alves Pereira, M. Acher, H. Martin, J.-M. Jézéquel, G. Botterweck, A. Ventresque, Learning

software configuration spaces: A systematic literature review, Journal of Systems and Software 182 (2021) 111044.
[17] D. Garber, T. Burgstaller, A. Felfernig, V.-M. Le, S. Lubos, T. Tran, S. Polat-Erdeniz, Collaborative recommendation of search heuristics for constraint solvers, in: ConfWS'23: 25th International Workshop on Configuration, Sep 6–7, 2023, Málaga, Spain, 2023.
[18] J. Oh, P. Gazzillo, D. Batory, T-wise coverage by uniform sampling, in: Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A, SPLC '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 84–87. URL: https://doi.org/10.1145/3336294.3342359. doi:10.1145/3336294.3342359.
[19] Z. Chen, M. Monperrus, A literature study of embeddings on source code, arXiv preprint arXiv:1904.03061 (2019).
[20] D. Benavides, P. Trinidad, A. Ruiz-Cortés, Automated reasoning on feature models, in: International Conference on Advanced Information Systems Engineering, Springer, 2005, pp. 491–503.
[21] M. Acher, H. Martin, J. A. Pereira, A. Blouin, J.-M. Jézéquel, D. E. Khelladi, L. Lesoil, O. Barais, Learning very large configuration spaces: What matters for Linux kernel sizes, Ph.D. thesis, Inria Rennes-Bretagne Atlantique, 2019.
[22] K. S. Meel, Counting, sampling, and synthesis: The quest for scalability., in: IJCAI, 2022, pp. 5816–5820.
[23] J. Oh, D. Batory, R. Heradio, Finding near-optimal configurations in colossal spaces with statistical guarantees, ACM Transactions on Software Engineering and Methodology 33 (2023) 1–36.
[24] Q. Plazar, M. Acher, G. Perrouin, X. Devroey, M. Cordy, Uniform sampling of sat solutions for configurable systems: Are we there yet?, in: 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), IEEE, 2019, pp. 240–251.
[25] L.-N. Pouchet, Polybench: The polyhedral benchmark suite, http://www.cs.ucla.edu/~pouchet/software/polybench/, 2012. Accessed: 2024.
[26] T. J. McCabe, A complexity measure, IEEE Transactions on software Engineering (1976) 308–320.
[27] M. H. Halstead, Elements of Software Science (Operating and programming systems series), Elsevier Science Inc., 1977.
[28] D. Spinellis, P. Louridas, M. Kechagia, The evolution of c programming practices: a study of the unix operating system 1973–2015, in: Proceedings of the 38th International Conference on Software Engineering, ICSE '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 748–759. URL: https://doi.org/10.1145/2884781.2884799. doi:10.1145/2884781.2884799.
[29] J. Alves Pereira, M. Acher, H. Martin, J.-M. Jézéquel, Sampling effect on performance prediction of configurable systems: A case study, in: Proceedings of the ACM/SPEC International Conference on Performance Engineering, 2020, pp. 277–288.
[30] G. Jain, T. Mahara, K. N. Tripathi, A survey of similarity measures for collaborative filtering-based recommender system, in: Soft Computing: Theories and Applications: Proceedings of SoCTA 2018, Springer, 2020, pp. 343–352.
[31] T.-T. Wong, Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation, Pattern recognition 48 (2015) 2839–2846.

Complexity Indicators and Their Impact on Algorithm Performance in Automotive Part Selection

Daniel Bischoff[1,*], Tobias Nerz[1] and Kaan Ekiz[1,2]

[1]Mercedes-Benz AG, Leibnizstraße 6/1, 71032 Böblingen, Germany
[2]Aalen University – Engineering, Business and Health, Beethovenstraße 1, 73430 Aalen, Germany

Abstract

This paper presents a comprehensive study on complexity indicators and relative scaling behavior of an algorithm designed for visualizing and changing part selection data in the automotive industry. The algorithm transforms Boolean formulas into multi-way decision diagrams within the context of a tool called POSEIDON. The case study is based on real industrial configuration data from a leading German automotive manufacturer, highlighting the challenges posed by high variant diversity and logical interdependencies.

To characterize the structural and distributional properties of the dataset, various complexity indicators are introduced and computed, including Shannon entropy and average path lengths in idealized decision diagrams. These metrics are then used to evaluate the behavior and effectiveness of the POSEIDON algorithm in generating compact, interpretable, and scalable decision diagrams. The results show that the average path length in the POSEIDON-generated diagrams scales linearly with entropy-based measures, confirming the adequacy of the algorithm in an environment where we've observed rising complexity year over year.

Keywords

Complexity Indicators, Algorithm Performance, Automotive Part Selection, Variant Configuration, Configuration Rules, Industrial Data

1. Introduction

Product configuration, particularly part selection, is a critical task in the automotive industry, based on vast and intricate datasets derived from manufacturing and engineering processes. The inherent complexity of such industrial data poses significant challenges for configuration systems. Understanding and quantifying this complexity is essential not only for benchmarking existing tools but also for improving their performance. Furthermore, evaluating how algorithms behave under realistic levels of complexity is crucial for ensuring their practical applicability.

Building on prior published work by the author(s) [1], this paper evaluates an algorithm that transforms Boolean formulas, used to encode feature-to-part relationships in automotive product lines, into multi-way decision diagrams. Unlike traditional binary decision diagrams (BDDs) [2], these structures consist of decision nodes representing feature choices with multiple discrete options (plus an explicit 'otherwise' branch encompassing remaining choices) and utilize the actual selected parts as terminal nodes, directly reflecting the configuration outcome. A key aspect distinguishing this approach is its application context: the decision diagrams, generated within the *POSEIDON* tool, serve as the primary interactive visualisation. Data engineers use them to create, analyse, and modify configuration logic, with a strong focus on comprehensibility and correctness by construction. This paper presents an in-depth case study utilizing authentic industrial data from this automotive context. Our primary contributions are twofold: first, we propose and measure complexity indicators designed to capture characteristics of industrial Bill of Materials, focusing on the combinatorial solution space described by their variation points. Second, we evaluate the performance of the decision diagram generation algorithm on our data, primarily focusing on the compactness of the resulting diagrammatic representation. For brevity, we will henceforth refer to this simply as *the algorithm*.

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
Daniel.Bischoff@Mercedes-Benz.com (D. Bischoff); Tobias.Nerz@Mercedes-Benz.com (T. Nerz); Kaan.Ekiz@Mercedes-Benz.com (K. Ekiz)

While various techniques exist for representing and visualising product configuration logic—including tabular formats, feature models, Karnaugh maps, and tree-based structures [3]—this study discusses multi-way decision diagrams.

Previous research has addressed the measurement of complexity in product configuration systems from different perspectives. Ghosh et al. [4] analyse the cognitive complexity of configurators using a UML-based metric, but their approach does not operate on the level of parts or Bills of Materials and does not compare visualisation techniques with objective complexity indicators. Similarly, Schmidt et al. [5] develop key performance indicators for managing variant diversity in complex product families, focusing on economic portfolio management rather than analysing objective complexity metrics or variant logic visualisation. Other approaches, such as Modrak and Bednar [6] and Herrera-Vidal et al. [7], apply entropy-based measures to quantify uncertainty and complexity but primarily address assembly or process-level dynamics rather than configuration rules at the Bill-of-Materials level. Additionally, Modrak and Soltysova [8] demonstrate that the mere number of possible variants is insufficient to capture true system complexity and propose information-theoretic measures to better reflect underlying dependencies.

Despite these contributions, no study has systematically connected quantitative complexity metrics with visual representations of configuration logic. This paper combines measures such as Shannon entropy and Huffman tree path lengths with the size of decision diagrams to investigate how configuration complexity affects the compactness of decision diagrams. To the best of our knowledge, no prior study has empirically investigated this relationship between data complexity and the visualisation of configuration logic.

1.1. Case Study Context and Data Source

The study is based on industrial data sourced from the product data management systems of a major automotive manufacturer, specifically focusing on part selection rules. The nature of this data reflects the challenges encountered in large-scale industrial configuration environments.

The data set, its scope and source, as well as the data processing, are described in Sect. 3.

2. Foundations

This section details the case study setup, i.e. the complexity indicators chosen to characterise the data, and the performance metrics used for the assessment of Poseidon.

2.1. Variation Points in the Bill of Materials

In product line engineering, a variation point is a bundle of alternative parts with corresponding selection criteria, often modelled simply as a list of formulas. The variation points analysed by the algorithm are documented in the so-called 150% bill of materials and are influenced by the product overview—also commonly referred to as the high-level configuration or feature model [1]. Each variation point in the bill of materials consists of at least one variant (i.e. physical part) and while there might also be several variants in one variation point, for each configured car, exactly one variant should be chosen out of each variation point. To ensure this, for every variant there is an item selection rule — based on the car configuration codes — and dedicated tools, that check, if documentation rules are satisfied. Detailed information on these tools is published in [9]. In Fig. 1 an example for a variation point containing three variants is visualised in POSEIDON.

2.2. Configuration rules

Configuration rules are documented in the product overview. There, properties (or features) of the cars — called codes — and consistency rules are defined. Following those rules, only certain combinations of codes are allowed to be configured. The resulting set of all possible car configurations is called *variation* *space*. In [1] we call this *context* and model it as a Boolean formula γ encoding the conjunction of all configuration constraints. In the example in Fig. 1 we assume that the context consists of only one rule, namely $\neg(A \wedge B \wedge C)$, leading to the $-C$ label generated on one edge.

PV	Code Rule
010	$\neg A \wedge \neg B$
020	$(\neg A \wedge B) \vee (A \wedge \neg B \wedge \neg C)$
030	$A \wedge ((\neg B \wedge C) \vee (B \wedge \neg C))$

Table 1
Example of part selection rules for a variation point containing three Variants.

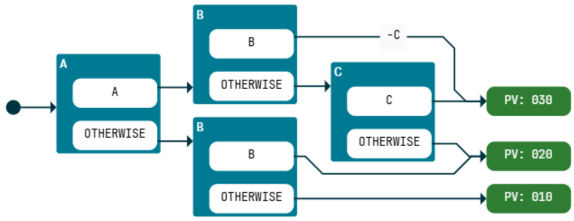


Figure 1: Variation point from Table 1 — visualised as decision diagram in Poseidon.

2.3. Algorithm Description

The core algorithm evaluated in this study transforms the input Boolean formulas into a multi-way decision diagram representation, as previously detailed in [1]. The key characteristics relevant to this evaluation are:

- Transformation: It systematically converts the propositional logic of the Boolean formulas into a directed acyclic graph (DAG).
- Multi-way Decision Nodes: Internal nodes represent decision points (corresponding to product feature alternatives). Each node can have multiple outgoing edges, representing the selection of specific alternatives for a given feature, plus a dedicated "otherwise" edge capturing the case where none of the explicitly listed options apply.
- Part Terminals: Terminal nodes (leaves) of the diagram directly represent the specific automotive part(s) selected when the conditions along the path from the root node are met.
- Application: The resulting diagram is the central data structure used in a visualiser/editor tool designed for configuration data engineers.
- Completeness: The diagram guarantees that every possible configuration path leads to a terminal node, ensuring that the representation is exhaustive and no valid variant is omitted.

2.4. Mathematical preliminaries

To make an evaluation of how efficient the graphical illustrations of POSEIDON are, we started with entropy, which is a well known concept in information theory when it comes to measuring the uncertainty of information [10]. We continued to calculate other complexity metrics; the three metrics used in this study are defined below.

For all calculations in this study consider a finite set

$$U = \{u_1, ..., u_n\} \quad \text{with} \quad |U| = n, \tag{1}$$

which in this study represents the set of all valid vehicle configurations under consideration. The assignment of vehicle configurations to a specific variant is modelled by a disjointed partition π of U, i.e. a set of sets (bundles). These bundles are technically component equivalence classes, as each bundle B_i contains all configurations that lead to the same installed part.

$$\pi = \{B_1, ..., B_m\} \quad \text{with} \quad |\pi| = m \tag{2}$$

such that the following properties hold:

$$\textstyle\bigcup_{i=1}^{m} B_i = U \tag{3}$$

$$B_i \cap B_j = \emptyset, \quad \text{if } i \neq j \tag{4}$$

Visualisations of partitions can be found in Fig. 2 and Fig. 3.

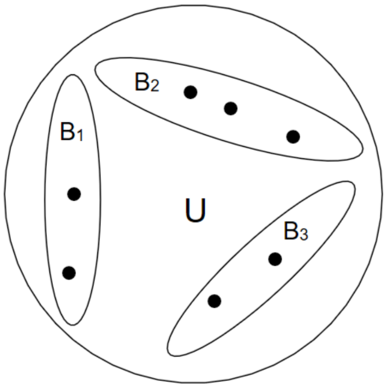


Figure 2: Visualisation of the set U associated to the example in Fig. 1 — containing seven Elements with a partition $\pi_1 = \{B_1, B_2, B_3\}$. The sets $B1$, B_2 and B_3 correspond to variants 010, 020 and 030 in Fig. 1 respectively.

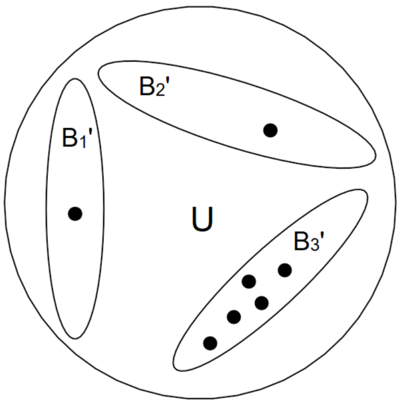


Figure 3: Visualisation of an alternative partition $\pi_2 = \{B_1', B_2', B_3'\}$ over the same universe U as in Fig. 2. The sets B_1', B_2' and B_3' represent a different grouping of the same elements and are not to be confused with B_1, B_2, and B_3 in Fig. 2.

2.5. Complexity Indicators

2.5.1. Model Counting

An important factor for our calculations is the number of satisfying assignments (called *models*) for the Boolean formula, or relevant sub-formulas corresponding to alternative selection bundles. It reflects the total number of configurations which are valid as defined by the context. In our setting, this corresponds to the number of valid vehicle configurations represented by the set U. The variables (codes) used to describe the configurations are only those present in the item selection rules. Thus we project the context to them. This greatly reduces the effective model count by discarding all irrelevant dimensions for telling configurations apart.

In the context of our modelling, we denote:

$$|U| = \text{Number of valid configurations} \tag{5}$$

Remark 2.1. *Note that for the context of this work the variant space is projected to the variables occurring in a single variation point. For instance, in the example in Fig. 1, only the variables A, B and C are taken into consideration. Other variables might exist, but are ignored in the context of this variation point. Thus, in this example we have $|U| = 2^3 - 1$, as one possible combination of variables is excluded by context.*

Remark 2.2. *This metric provides a direct measure of the solution space size and indicates how many valid feature combinations exist. It serves as the base set for calculating derived measures such as the relative frequencies of variants in entropy-based evaluations.*

2.5.2. Shannon Entropy

We use Shannon entropy [10], a well-established metric from information theory, to quantitatively describe the complexity of a given variation point. It captures the uncertainty in the distribution of part variants across all valid vehicle configurations. A high entropy value indicates a balanced usage of many variants, while low entropy suggests the presence of dominant, frequently used options. This allows for direct comparisons between positions or even across different model series.

For some partition π, Shannon Entropy $H_S(\pi)$ is defined as:

$$H_S(\pi) = \sum_{B_i \in \pi} p_i \cdot \log_2\left(\frac{1}{p_i}\right) \quad \text{where } p_i = \frac{|B_i|}{|U|} \tag{6}$$

Definition 2.3. *Let π_1 and π_2 be two partitions of the same set U. We say that π_1 is* more complex in terms of variant distribution *than π_2 if*

$$H_S(\pi_1) > H_S(\pi_2),$$

i.e., if the Shannon entropy of π_1 is strictly greater than that of π_2.

Remark 2.4. *The Shannon entropy has the following property:*

$$0 \leq H_S(\pi) \leq \log_2(m) \tag{7}$$

Example 2.5. *To get an intuition for this complexity measure, take a look at the examples defined in Fig. 2 and Fig. 3:*

$$H_S(\pi_1) = \sum_{i=1}^{3} p_i \cdot log_2\left(\frac{1}{p_i}\right)$$

$$
\begin{aligned}
&= \frac{2}{7} \cdot log_2\left(\frac{7}{2}\right) + \frac{3}{7} \cdot log_2\left(\frac{7}{3}\right) + \frac{2}{7} \cdot log_2\left(\frac{7}{2}\right) \\
&\approx 1.557 \\
H_S(\pi_2) &= \sum_{i=1}^{3} p_i \cdot log_2\left(\frac{1}{p_i}\right) \\
&= \frac{1}{7} \cdot log_2(7) + \frac{1}{7} \cdot log_2(7) + \frac{5}{7} \cdot log_2\left(\frac{7}{5}\right) \\
&\approx 1.149
\end{aligned}
$$

We observe that π_1 exhibits a higher entropy than π_2. This results from a more uniform distribution of elements across the subsets in π_1, whereas in π_2 (cf. Fig. 3), the majority of elements are concentrated in B'_3.

2.5.3. Huffman Trees

In addition, we consider the average path length of a Huffman tree constructed using the relative frequencies of the part variants [11]. This length reflects the average number of binary decisions required to uniquely identify a variant. Thus, the Huffman tree simulates an optimal binary decision tree and serves as a theoretical benchmark for evaluating actual representations, such as those used in tools like POSEIDON. Constructing a Huffman tree can be done by implementing the following algorithm for some partition $\pi = \{B_1, ..., B_n\}$:

1. Create a working list of nodes for each B_i with value $|B_i|$.
2. Look for two nodes i and j with smallest value, i.e. $|B_k| \geq |B_i|$ and $|B_k| \geq |B_j|$ for all $k \neq i, j$.
3. Remove the nodes i and j from the list and add a node which is their (new) parent with the sum of the children's values as its value.
4. Repeat until one node is left, this is the root of the resulting Huffman tree.

Looking at this algorithm in terms of building a decision diagram, we combine the two least common endpoints in a sub tree and add up their relative frequencies. This is repeated, until the whole decision diagram is built. By evaluating this Huffman tree, we get a binary encoding — consisting of all yes-no-decisions that have to be taken to get to the regarding endpoint — for every subset B_i of the partition π. We denote the length of this binary code as l_i.

With this, we can analyse the average path length of the Huffman trees

$$\frac{1}{n}\sum_{i=1}^{n} l_i, \tag{8}$$

as well as the weighted average path lengths

$$\sum_{i=1}^{n} l_i \cdot p_i, \text{ where } p_i \text{ is the relative frequency at which a path is chosen.} \tag{9}$$

This indicators reflect how many binary decisions are needed on average to uniquely identify a variant, treating the different parts equally or taking into account their relative frequencies respectively.

2.6. Performance of POSEIDON

To evaluate the effectiveness of the algorithm and the comprehensibility of the resulting graph, several performance metrics are applied. These refer both to the structure of the generated decision diagram—particularly with regard to its suitability as a compact and interpretable representation—and to characteristics of the underlying binary structure used during its construction. In addition, the Pearson correlation coefficient is used to analyse relationships between selected complexity metrics.

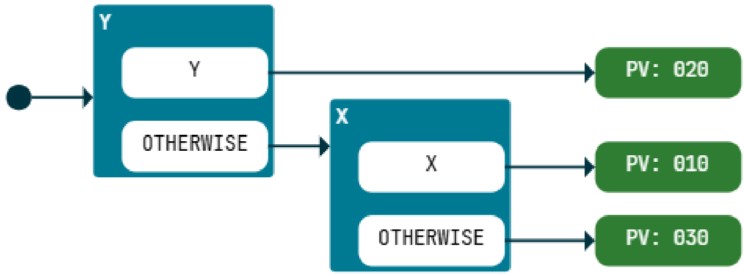


Figure 4: Huffman tree for the example mentioned in Table 1. X and Y are idealised configuration values that subdivide vehicle configuration as needed for the tree construction.

2.6.1. Average Path Length in Multi-valued Decision Diagrams (MDD)

The *average path length* in a multi-valued decision diagram (MDD) quantifies how many decision steps are required on average to reach a terminal node from the root node. A terminal node corresponds to a specific part variant.

Let n denote the number of valid paths in the MDD that lead to a concrete output. For each such path i (with $i = 1, \dots, n$), let d_{m_i} be the number of decisions made along that path, where each decision may involve selecting one of multiple alternatives.

Then, the average path length L_{MDD} is computed as:

$$L_{\text{MDD}} = \frac{1}{n} \sum_{i=1}^{n} d_{m_i} \tag{10}$$

This metric reflects the compactness and interpretability of the diagram structure: shorter paths imply fewer decision steps to reach a result. Accordingly, a lower L_{MDD} indicates a more concise and potentially more comprehensible configuration logic.

2.6.2. Binary Average Path Length in Binary Decision Diagrams (BDD)

The *binary average path length* quantifies the expected number of binary decisions required to reach a terminal node from the root node in a binary decision diagram (BDD). In contrast to the average path length in an MDD—which may count multi-valued decisions as single steps—this metric assumes that all variation points are represented through binary decisions. That is, each possible alternative is encoded as a separate yes/no condition that must be evaluated along the path.

Let n be the number of valid paths in the BDD that lead to product variants. For each path i (with $i = 1, \dots, n$), let d_{b_i} denote the number of binary decisions made along that path.

Then, the binary average path length L_{BDD} is computed as:

$$L_{\text{BDD}} = \frac{1}{n} \sum_{i=1}^{n} d_{b_i} \tag{11}$$

Remark 2.6. *Comparing the binary average path length to the average path length of the multi-way decision diagram generated by POSEIDON allows us to assess how efficiently the algorithm reduces decisions required.*

2.6.3. Pearson Correlation

To quantitatively assess the linear relationship between two complexity metrics, we apply the Pearson correlation coefficient r [12]. This coefficient indicates whether and to what extent two metric variables are linearly correlated. A positive value suggests a direct relationship, a negative value an inverse relationship, and a value close to 0 indicates no linear correlation.

For two variables x and y with N observations, r is defined as follows:

$$r = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{(N-1) \cdot s_x \cdot s_y} \tag{12}$$

Where:

- x_i, y_i: individual observations of the variables x and y
- $\bar{x}, \bar{y}$: arithmetic means of x and y
- s_x, s_y: empirical standard deviations of x and y

Remark 2.7. *The Pearson coefficient is a dimensionless value ranging from -1 to 1. Values close to $|1|$ indicate a strong linear relationship, while values near 0 suggest a weak or no linear correlation.*

Remark 2.8. *In this study, the coefficient is used to examine the extent to which structural metrics, such as average path length, align with information-based metrics. Each observation x_i and y_i represents the value of a specific complexity metric (e.g., entropy, path length, model count) measured at an individual BOM position. The results of these analyses are visualized in the correlation matrices shown in Fig. 5.*

While alternative complexity measures such as Kolmogorov complexity or spectral entropy were also considered, they were found to be impractical for this specific use case due to either their computational intractability or their limited interpretability in the context of real-world configuration logic.

3. Data Set — Engineering Bill of Materials

3.1. Data Source and Scope

This analysis is based on real-world data from the product configuration system of a major automotive manufacturer. The central data source is the so-called 150% Bill of Materials (BOM), a maximal BOM containing all potentially installable components and assemblies across all available product variants. In this work, we focus on the engineering stage, where the 150% BOM is represented as the Engineering Bill of Materials (E-BOM), which serves as the foundation for the systematic analysis of variant logic at the position level. Following Eigner et al. [13, p. 272], the E-BOM denotes the bill of materials during the engineering stage and is often referred to as a generic BOM [14].

In addition, we incorporate configuration rules from the context. These define which combinations of feature characteristics lead to valid — i.e., buildable — vehicle configurations. Based on this data, it is possible to determine for each variant how often each part is used, denoted as valid configurations. This distribution forms the basis for relative frequency calculations used in the complexity metrics.

The data under investigation pertains to specific model series. A model series describes a technical base model of a vehicle that is being produced in various variants—for example, with different engine types, equipment levels, or drivetrains. The selected model series have been produced since 2023 and 2024 respectively. As such, the underlying dataset is derived from real-world production data and is well-suited for in-depth analysis. Note, however, that real-world data is always evolving, e.g. our dataset might include currently unused future part variants etc. We address this further in Sect. 3.4.

For the subsequent analysis, we distinguish between two datasets, each representing a separate model series. These will be referred to as *Model Series A* and *Model Series B* throughout the remainder of this study.

3.2. Structure and Representation of Variant Logic

Each BOM position is linked to a set of code rules, expressed in the form of Boolean formulas, which define the conditions under which a particular component is installed. These rules form the central logical structure for variant management.

The internal tool POSEIDON is used for visualising these code rules. It transforms the Boolean formulas into decision diagrams represented as directed graphs. This form of representation enables a structured understanding of rule complexity and also supports the validation of rule sets with regard to uniqueness, completeness, and satisfiability [15].

3.3. Analytical Objective

The objective of this analysis is to quantitatively assess and evaluate the complexity of variant logic at selected BOM positions. For this purpose, established metrics such as Shannon entropy and the average path length in the corresponding Huffman tree are employed. These indicators were discussed in detail in Sect. 2.5 and enable a standardized assessment of variant diversity across different positions.

Based on those general measures of complexity, we aim to evaluate the performance of the POSEIDON algorithm by correlating the results obtained using POSEIDON with the underlying data complexity.

3.4. Data Cleaning

Analysis shows that some decision diagrams contain paths that end in leaf nodes without an associated component. These incomplete endpoints may result from various causes—for instance, a component may not be required for certain feature combinations, a part may not yet be maintained in the system, or inconsistencies may exist in the rule set.

To accurately capture the true structure of the variant logic, these paths are explicitly included in the calculation of the complexity metrics. In other words, whenever the data implies no part selection, we treat this as intentional and make that option explicit in the data before doing our calculations.

Furthermore, there are part selection rules that are logically contradictory or unsatisfiable within the context. These parts were completely ignored in the analysis, as they have a model count of zero and thus no influence on the output data. The item selection rule in such a case is semantically equivalent to false with regard to the valid vehicle configurations.

In addition, all outdated positions for which no feasible part remained were removed from the dataset.

This approach allows for a comprehensive assessment that considers not only the defined variants but also potential rule gaps or modelling issues.

4. Encoding and Visualisation of Variation Points

This section discusses encoding options for variation points such as truth tables, KV diagrams, Venn diagrams, and decision diagrams, while highlighting the advantages and disadvantages of these respective options, especially with regard to integrability of context information.

While various representations for Boolean configuration logic exist—such as tabular formats, Karnaugh maps, and Venn diagrams—this study focuses on multi-way decision diagrams for encoding variation points.

All these methods can be used to determine which of the variant(s) in a given variation point can be chosen for a given Boolean configuration. However, in the use case of the variation point visualisation, a considerable part of the theoretical variant space is not relevant a priori, because it is not buildable under consideration of the product overview. A key consideration for the applicability of an encoding is how effectively it conceals the non-feasible variant space.

In the following paragraphs, the methods for presenting Boolean logic, the necessary modifications for displaying mappings, and the relevant literature are discussed.

4.1. The Role of Visualisation in Data Quality and Related Work

Data visualisation can play a key role in data quality assurance by providing an intuitive and interactive way to represent and analyse data. By visualising the data, it becomes easier for data workers to identify patterns, trends, and anomalies in the data, which can help identify errors, inconsistencies, or other issues that can affect the quality of the data. Visualisation can also help to highlight the relationships and dependencies between different data elements, making it easier to understand the context in which the data is used and how it may be affected by changes.

4.2. Visualisation Option: Formulas

Encoding variation points using Boolean formulas is a common industrial practice [16]. For instance, the BOM uses Boolean formulas as item selection rules. Some companies require formulas to match certain normalizing criteria, whereas others allow any syntactically correct formula.

Formulas have the advantage of acceptable scaling, both in the case of many involved variables, as well as in the case of complex inner relations. Though in the worst-case-scenario a Boolean formula will double in size with the introduction of a new variable, in the average case it will grow significantly less. However Formulas have the disadvantage of being hard to evaluate mentally. In addition, the known normal forms for Boolean formulas are either not canonical, hard to comprehend for humans, or too large.

4.3. Visualisation Option: Truth- & Decision Tables

A common approach to visualise variability data are various forms of tables. The naive idea is to list all possible feature combinations, i.e. a truth table.

Each row of the table represents a possible combination of values for a set of Boolean inputs, and the corresponding column indicates whether the combination satisfies a given Boolean expression (output). Truth tables can be a useful tool for analysing and manipulating Boolean data, as they provide a clear representation of the possible combinations of values and the relationships between them.

However, truth tables can become unwieldy as the number of Boolean variables increases. As the number of variables of the expressions increases, the number of rows in the truth table grows exponentially, making it difficult to understand and manipulate the data.

To limit the size of the table, combinations that do not satisfy a validity constraint (in our case, the product overview) can be hidden.

A similar table standard is used by Tidstam et al. [17] for assigning part variants to feature combinations in a configuration context. In Table 2, taken from [17], the red cells represent invalid feature selections.

One way of improving the scaling of the table is that after successful assignment, several vehicle properties/variables can often be bundled into mutually exclusive families instead of as a separate column, as in a truth value table, which enables the consolidation of many Boolean columns to multi-value columns. For instance, Table 3 doesn't show dedicated Boolean columns for *Gasoline* and *Diesel* as in the case of *Turbo, Sport* and *City*, but instead the multi-value column *Fuel* with the exclusive variants *gasoline* and *diesel.*

A further reduction is possible if only slightly different vehicle variants are assigned to the same component. For instance, the last two lines in Table 3, that only differ in their value in the city column, show the same component. For these kind of constellations, some table representations use *don't-care* symbols. In this case, a line *Volume=1.2, Turbo=no, Sport=no, City=*, Fuel=gasoline, Item(s)=E12* could have subsumed the last two lines and thus replace them.

However, this approach doesn't terminate quickly and yields its own optimisation issue, addressed by the Quine-McCluskey algorithm [18][19].

A table, that has been optimised that way, eventually contains only prime implicants for each item and in general there are multiple table rows required. Due to the potential overlap of the prime implicants, the use of all prime implicants is only necessary in the worst case.

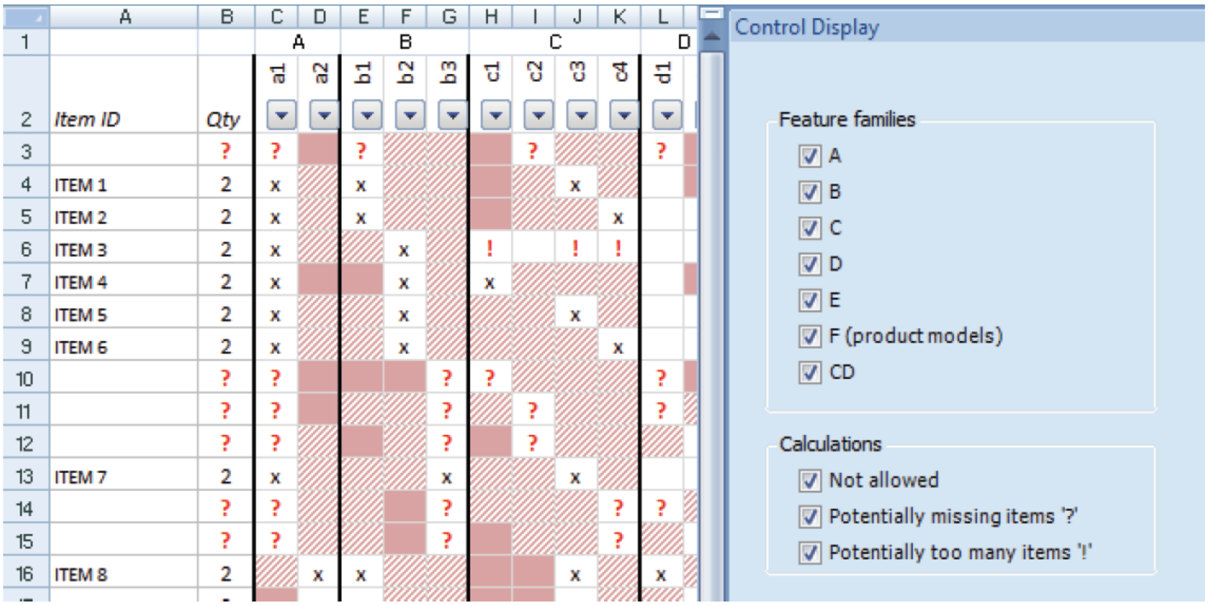


Table 2
Improved visualisaion of item selection rules as truth table, taken from Tidstam et al. [17].

Families and variants					
Volume	Turbo	Sport	City	Fuel	*Item(s)*
1.6	yes	yes	no	gasoline	*E16T*
1.6	no	no	no	diesel	*E16D*
1.2	no	no	yes	gasoline	*E12*
1.2	no	no	no	gasoline	*E12*

Table 3
Tabular assignment of vehicle variants to component variants as displayed in Voronov et al. [16].

In terms of human readability, the table has the considerable advantage that it is understood intuitively and that there are good tools for editing tables. In contrast, the scaling properties of tables are no longer sufficient for many practical data constellations.

For these reasons, truth tables may not be well-suited for visualising complex Boolean data for data workers. The scaling properties make them undesirable or outright impossible to apply in some real-world scenarios. In such cases, other visualisation methods, such as decision diagrams or graphical models, may be more effective for helping data workers to understand and analyse the data.

4.4. Visualisation Option: Decision Diagrams

Decision Diagrams have been selected as the visualisation method for POSEIDON. Our algorithm utilizes Binary Decision Diagrams (BDDs) to transform data into multiple decision diagrams (MDDs).

By leveraging BDDs, we obtain a data structure that meets numerous requirements. The exclusion of unbuildable spaces is straightforward, achieved by removing the corresponding subgraphs. Configurable variable ordering, along with ordering heuristics to minimize graph size, provides customizable perspectives on the data. Reduced decision diagrams enhance efficiency by reusing equivalent subtrees, resulting in more concise representations

In [20] the authors encode vehicle configuration at Renault into a Constraint Satisfaction Problem (CSP). Whereas Amilhastre et al. [21] encode context validity in a CSP, compile it into an automaton and represent the automaton graphically. The automaton accepts valid feature selections, but, in contrast to our approach, does not compute which part selection follows.

4.5. Other Visualization Options

Karnaugh Maps [22], also known as KV diagrams, are a graphical representation of Boolean expressions used to simplify and minimise the expressions. They are based on the idea of grouping adjacent terms in a truth table that represent the same logical function. KV diagrams appear to be unsuitable for the visualisation of variation points, since they cannot simply hide a cell because the relative positions of cells to each other are of vital importance, and their meaning is defined via their geometric position.

Similar issues can be observed for Venn Diagrams. They also become notoriously difficult to work with, even for a small number of variables.

5. Experimental Results and Evaluation

5.1. Quantifying BOM Complexity

In addition to the number of valid configurations (Model Count), the evaluation incorporates indicators of distributional characteristics of the solution space. These indicators include entropy-based measures and average path lengths derived from Huffman Trees, Multi-valued Decision Diagrams (MDDs), and Binary Decision Diagrams (BDDs).

For each metric, key descriptive statistics were calculated, including minimum, maximum, arithmetic mean, median, and standard deviation. The results are detailed in Table 4 and Table 5.

Table 4
Statistical Key Figures – Model Series A

Metric	Min	Max	Mean	Med.	Std. Dev.
Model Count	2.00	2520.00	18.34	5.00	74.66
Shannon Entropy	0.00	4.82	0.96	0.97	0.66
Huffman Weighted Avg. Path	1.00	5.90	2.03	2.00	0.67
Huffman Avg. Path	1.00	7.82	2.19	2.00	0.95
MDD Avg. Path	2.00	9.93	2.93	2.67	1.10
BDD Avg. Path	2.00	70.98	5.39	4.00	3.60

Table 5
Statistical Key Figures – Model Series B

Metric	Min	Max	Mean	Med.	Std. Dev.
Model Count	2.00	1760.00	17.39	6.00	49.64
Shannon Entropy	0.00	5.29	0.96	0.97	0.67
Huffman Weighted Avg. Path	1.00	6.34	2.04	2.00	0.68
Huffman Avg. Path	1.00	7.74	2.23	2.00	0.97
MDD Avg. Path	2.00	8.80	3.06	2.75	1.09
BDD Avg. Path	2.00	68.98	6.00	5.00	3.65

Model Count. The mean is 18.34 (A) and 17.39 (B), while the median is considerably lower at 5.00 and 6.00, respectively. This indicates a right-skewed distribution—most items have few valid variants, while a few show high combinatorial complexity, as also reflected by the high standard deviations (74.66 and 49.64).

Shannon Entropy. With a mean of 0.96 in both datasets, this suggests low combinatorial uncertainty—i.e., in most cases, only a few equally probable variants exist.

Huffman Tree Weighted Avg. Path length. Mean values of 2.03 (A) and 2.04 (B) indicate a more differentiated distribution, where rare but valid configurations contribute significantly to the total information content.

Huffman Tree Avg. Path length. Values of 2.19 (A) and 2.23 (B) suggest that, on average, about two binary decisions are required to uniquely identify a specific variant. Note: This is under the assumtion that such binary dimensions already exist and their interrelationship with other variables is modelled elsewhere. This is why the Huffman Tree represents a lower bound for real world BDDs.

MDD Avg. Path Length. The mean is 2.93 (A) and 3.06 (B). This metric reflects the average number of decision steps required to reach a terminal node in the multi-valued decision diagram generated by Poseidon and thus characterizes the structural depth of the rule logic.

BDD Avg. Path Length. The mean is 5.39 (A) and 6.00 (B). A purely binary encoding requires roughly twice as many steps as the MDD, highlighting the structural compactness of the multi-valued representation.

5.2. Evaluating MDD generation with regards to BOM Complexity

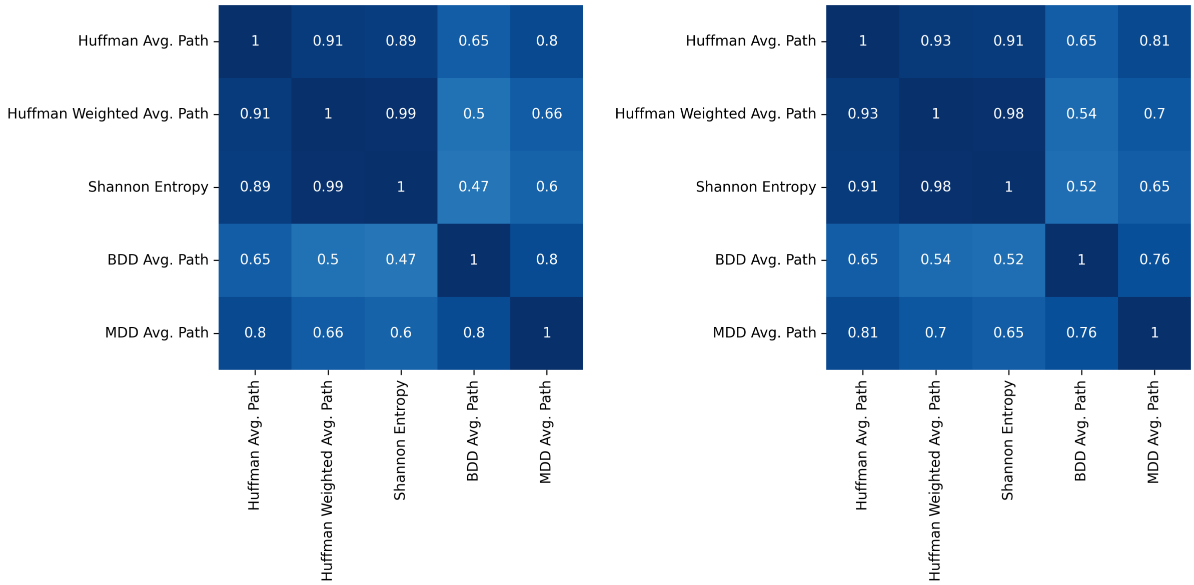


Figure 5: Comparison of the correlation matrices of the analysed metrics for variant complexity. Model Series A shown on the left, Model Series B on the right.

This section aims to investigate to what extent the decision diagrams generated by POSEIDON reflect the underlying variant structure. As reference measures, both information-theoretic and structural metrics are considered, including Shannon entropy and (weighted) Huffman Tree average path lengths.

5.2.1. Correlation Analysis of Complexity Metrics

To evaluate the relationships between the selected complexity metrics, the Pearson correlation coefficient r was calculated (cf.Sect. 2.5). The correlation matrices in Fig. 5 display the dependencies among entropy-based measures, the (weighted) average path length in the Huffman Tree, and the path lengths in decision diagrams based on MDDs and BDDs.

MDD Structure The average path lengths in the MDDs also show a positive correlation with entropy-based measures, albeit to a lesser extent. For Model Series A, correlation values range between $r = 0.65$ and 0.70, and for Model Series B, between $r = 0.60$ and 0.66. These correlations indicate that the structure of the decision diagrams reflects key aspects of the information-theoretic complexity of the configuration logic. This suggests a non-exponential scaling behaviour of the MDD algorithm with respect to increasing variant diversity — even though the underlying algorithm does not explicitly optimize for entropy.

Comparison with BDDs In contrast, the BDDs (Binary Decision Diagrams) exhibit significantly lower correlation with entropy-based metrics. In particular, the value for Shannon entropy in Model Series B drops to $r = 0.47$, indicating that BDDs are less sensitive to the actual distribution of configuration options. This reflects the binary nature of BDDs, where path lengths are primarily determined by the number of Boolean decisions rather than actual occurrence probabilities.

Model Comparison The comparison of the two model series reveals overall consistent correlation patterns. In Model Series A, the correlations are slightly stronger throughout, which may point to a more homogeneous variant structure or a lower degree of extreme configuration cases.

Additional observations Unsurprisingly, in both model series, a very high correlation is observed between Huffman Tree weighted Avg. Path length and Huffman Tree path lengths, as well as BDD and MDD path lengths. This is due to the fact that both Huffman Tree Path lengths are obtained from the same tree, while the MDDs are derived from the BDDs.

Conclusion Overall, the findings suggest that the structure of the MDD scales linearly with increasing variant complexity. This supports the conclusion that the logical complexity embedded in the configuration data is represented in a comprehensible and traceable manner.

5.2.2. MDD compared to Complexity Metrics

In Fig. 6, we investigate how structural and information-theoretic complexity metrics relate to the average path length in the MDD representation, based on all configuration positions.

Across all three metrics, a clear and approximately linear correlation is observed: Positions with greater entropy or more uneven variant distributions tend to produce deeper MDD structures. While the correlation strength differs slightly between metrics, the overall pattern indicates that the MDD depth increases in line with the underlying distributional complexity.

These results suggest that the MDD structure captures distributional complexity linearly, indicating that the MDD generation will continue to scale with increasing complexity.

5.2.3. Comparison betweeen MDD and BDD Average Path Lengths

In Fig. 7 we present the quotient between the Avg. Path Lengths of the BDD and the MDD for each variation point in Model Series A (top) and B (bottom) are visualised in relation to the Shannon Entropy. It appears that, independent of the underlying entropy, the MDD generation reduces the path length of the BDD by a factor of about 1.7.

5.3. Implications for Practical Applicability

The results presented in this chapter demonstrate that the algorithm used to transform configuration logic into decision diagrams performs robustly even as variant complexity increases. Despite increasing entropy and growing structural depth, the generated diagrams remain compact and interpretable.

This is particularly important for integration into the POSEIDON tool: since the generated structures are directly edited and reviewed by data engineers, a clear and concise representation is essential for

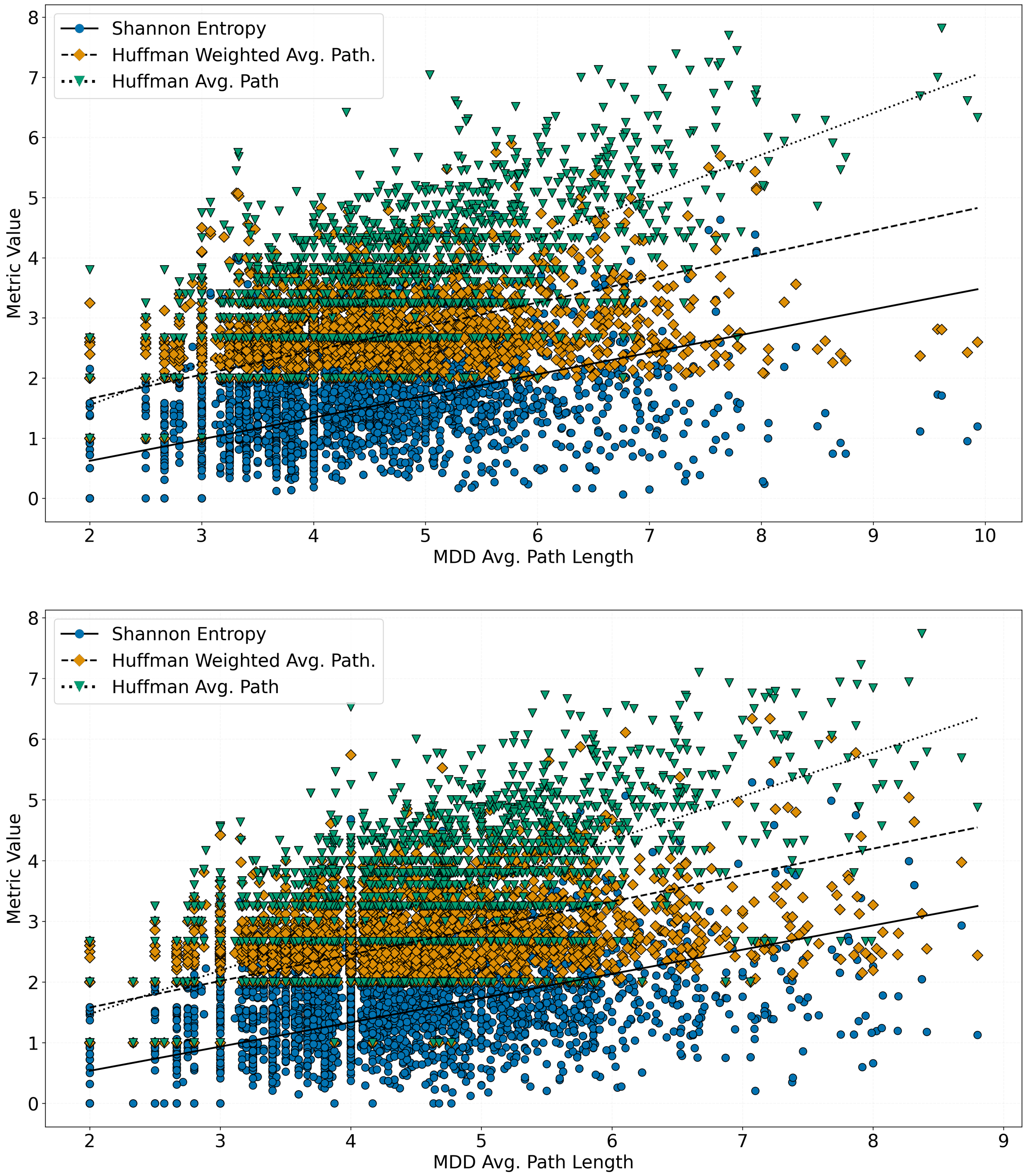


Figure 6: Comparison of the MDD Avg. Path Length and the complexity metrics per configuration position for Model Series A (top) and B (bottom). The average path length in the MDD is plotted against Shannon entropy (circle), Huffman Tree Weighted Avg. Path Length (diamond), and Huffman Tree Avg. Path Length (triangle), including corresponding trend lines.

maintainability, error prevention, and efficiency in handling complex product logic. The algorithm's ability to suppress non-configurable regions of the solution space while clearly structuring relevant branches offers a distinct advantage over alternative visualisation techniques.

Overall, the analysis shows that the measured complexity indicators are not merely theoretical constructs but have tangible implications for the usability and scalability of the tool in real-world application scenarios.

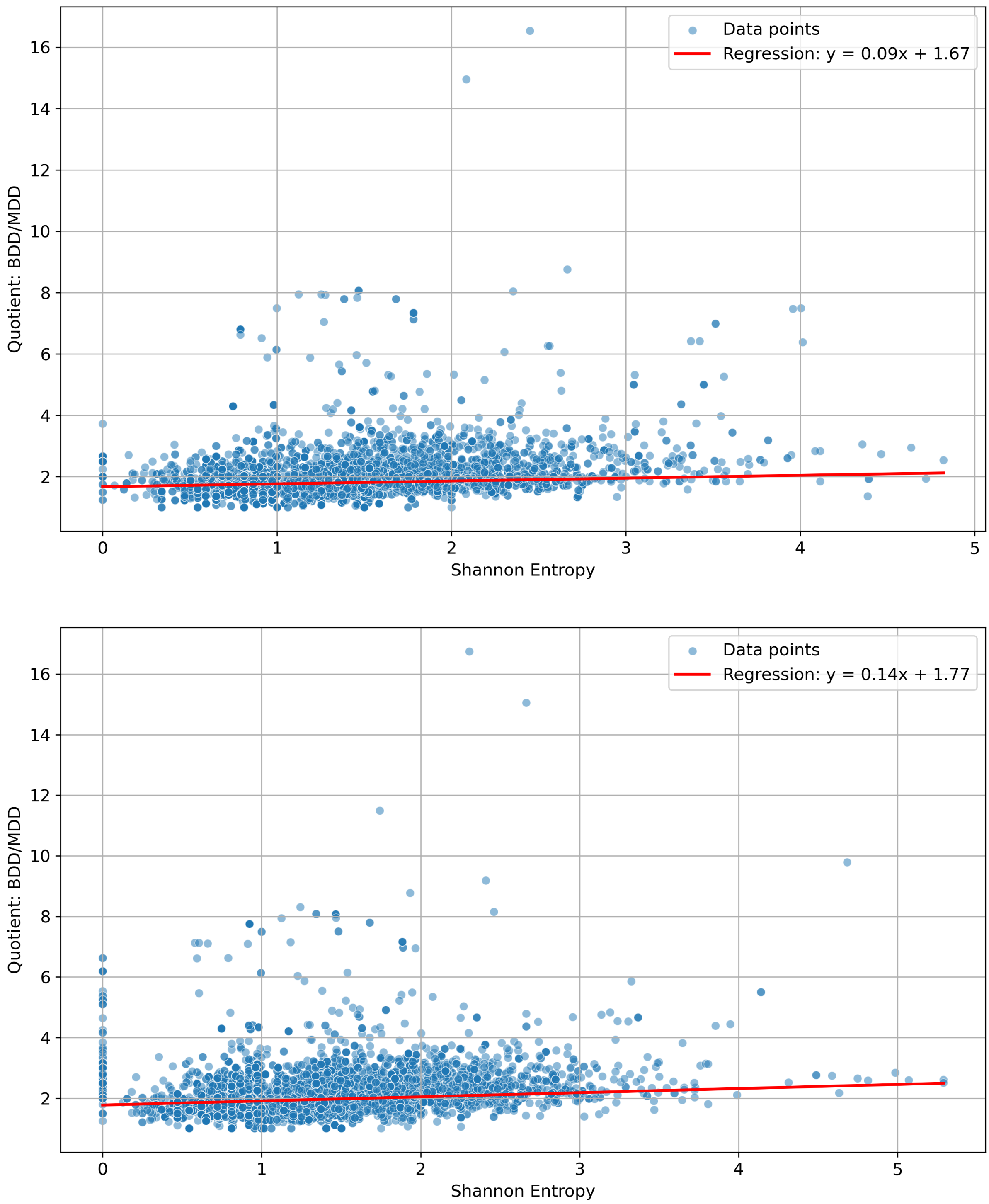


Figure 7: Relationship between Shannon Entropy and the quotient of average path lengths in BDDs and MDDs for Model Series A (top) and B (bottom). Each point represents a variation point.

6. Conclusion

This paper investigated the relationship between complexity indicators and the scalability of the POSEIDON algorithm for visualizing and manipulating automotive configuration data. This study demonstrated that the generation of POSEIDON's multi-way decision diagrams (MDDs) exhibit a linear scaling behavior with increasing variant complexity, as measured by Shannon entropy and Huffman tree path lengths. Compared to binary decision diagrams (BDDs), POSEIDON offers a distinct advantage by suppressing non-configurable regions of the solution space and providing a more compact and interpretable representation of the configuration logic, as demonstrated by the lower average path lengths observed in Tables 4 and 5.

The findings, derived from real-world configuration data, have significant implications for the practical applicability of POSEIDON. The linear scaling of MDD complexity ensures that the generated diagrams remain compact and interpretable even as variant diversity and logical interdependencies increase. This is crucial for data engineers who directly edit and review these structures, as it supports maintainability, error prevention, and efficient handling of complex product logic. The study confirms that the measured complexity indicators are not merely theoretical constructs but have tangible implications for the usability and scalability of the tool in real-world application scenarios.

This work contributes a comprehensive evaluation, combining theoretical indicators with empirical results, to demonstrate POSEIDON's effectiveness in managing the rising complexity observed in automotive product configuration.

While the evaluation focused on automotive data, future research should investigate POSEIDON's applicability to non-automotive domains and more heterogeneous data sources, for instance by analysing codebases and rule structures from other industries.

Additionally, further research could explore optimizing the MDD generation algorithm, especially for product lines with extremely high variant diversity. Another promising direction is to investigate alternative visualisation algorithms that maintain linear scalability while achieving a lower growth rate of the visualisation, enabling even more compact and efficient diagram representations.

Declaration on Generative AI

During the preparation of this work, the authors used an internal GenAI tool (based on GPT-4o) and Thesify for grammar, spelling checks, and literature proposals. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Bischoff, W. Küchlin, O. Kopp, Poseidon: A graphical editor for item selection rules within feature combination rule contexts, in: F. Noël, F. Nyffenegger, L. Rivest, A. Bouras (Eds.), Product Lifecycle Management. PLM in Transition Times: The Place of Humans and Transformative Technologies, volume 667 of *IFIP Advances in Information and Communication Technology*, Springer Nature Switzerland, Cham, 2023, pp. 3–14. doi:10.1007/978-3-031-25182-5_1.

[2] Akers, Binary decision diagrams, IEEE Transactions on computers 100 (1978) 509–516.

[3] S. Shafiee, K. Kristjansdottir, L. Hvam, A. Felfernig, A. Myrodia, Analysis of visual representation techniques for product configuration systems in industrial companies, in: 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE, 2016, pp. 793–797. doi:10.1109/ieem.2016.7797985.

[4] A. Ghosh, K. Kristjandottir, L. Hvam, E. Vareilles, Measuring the complexity of product configuration systems, in: Proceedings of the 20th International Configuration Workshop, volume 2220, CEUR Workshop Proceedings, 2018, pp. 61–68.

[5] M. Schmidt, J. Schwöbel, M. Lienkamp, Developing key performance indicators for variant management of complex product families, in: Proceedings of NordDesign 2018, Technical University of Munich, Linköping, Sweden, 2018, pp. 1–12.

[6] V. Modrak, S. Bednar, Entropy based versus combinatorial product configuration complexity in mass customized manufacturing, Procedia CIRP 41 (2016) 183–188. doi:10.1016/j.procir.2015.12.100.

[7] G. Herrera-Vidal, J. R. Coronado-Hernandez, I. Derpich, B. Paredes, G. Gatica, Measuring com-

plexity in manufacturing: Integrating entropic methods, programming and simulation, Entropy 27 (2025) 50. doi:10.3390/e27010050.
[8] V. Modrak, Z. Soltysova, Assessment of product variety complexity, Entropy 25 (2023) 119. doi:10.3390/e25010119.
[9] C. Sinz, A. Kaiser, W. Küchlin, Formal methods for the validation of automotive product configuration data, Ai Edam 17 (2003) 75–97.
[10] C. E. Shannon, A mathematical theory of communication, The Bell System Technical Journal 27 (1948) 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.
[11] D. A. Huffman, A method for the construction of minimum-redundancy codes, Proceedings of the IRE 40 (1952) 1098–1101.
[12] K. Pearson, Mathematical contributions to the theory of evolution. iii. regression, heredity, and panmixia, Philosophical Transactions of the Royal Society of London. Series A 187 (1896) 253–318. doi:10.1098/rsta.1896.0007.
[13] M. Eigner, D. Roubanov, R. Zafirov (Eds.), Modellbasierte virtuelle Produktentwicklung, 1 ed., Springer Vieweg Berlin, Heidelberg, 2014. doi:10.1007/978-3-662-43816-9.
[14] H. Hegge, J. Wortmann, Generic bill-of-material: a new product model, International Journal of Production Economics 23 (1991) 117–128. doi:https://doi.org/10.1016/0925-5273(91)90055-X.
[15] D. Bischoff, W. Küchlin, Adapting binary decision diagrams for visualizing product configuration data, in: INFORMATIK 2017, Gesellschaft für Informatik, Bonn, 2017, pp. 1499–1509. doi:10.18420/in2017_149.
[16] A. Voronov, A. Tidstam, K. Åkesson, J. Malmqvist, Verification of item usage rules in product configuration, in: IFIP International Conference on Product Lifecycle Management, Springer, 2012, pp. 182–191.
[17] A. Tidstam, L.-O. Bligård, F. Ekstedt, A. Voronov, K. Åkesson, J. Malmqvist, Development of industrial visualization tools for validation of vehicle configuration rules, in: Proceedings of the Tools and Methods of Competitive Engineering (TMCE), Organizing Committee of TMCE 2012, Karlsruhe, Germany, 2012, pp. 305–318.
[18] W. V. Quine, The problem of simplifying truth functions, The American mathematical monthly 59 (1952) 521–531.
[19] E. J. McCluskey, Minimization of boolean functions, The Bell System Technical Journal 35 (1956) 1417–1444.
[20] J. Astesana, L. Cosserat, H. Fargier, Constraint-based modeling and exploitation of a vehicle range at renault's: new requests for the csp formalism, in: International Conference on Tools with Artificial Intelligence (ICTAI), IEEE Computer Society, Arras, France, 2010, pp. 68–75. doi:10.1109/ICTAI.2010.19.
[21] J. Amilhastre, H. Fargier, P. Marquis, Consistency restoration and explanations in dynamic csps—application to configuration, Artificial Intelligence 135 (2002) 199–234.
[22] M. Karnaugh, The map method for synthesis of combinational logic circuits, Transactions of the American Institute of Electrical Engineers, Part I: Communication and Electronics 72 (1953) 593–599.

Heterogeneity: A Challenge in Automotive Product Configuration

Daniel Bischoff[1], Carsten Sinz[2,*]

[1]Mercedes-Benz AG, Leibnizstraße 6/1, 71032 Böblingen, Germany
[2]Karlsruhe University of Applied Sciences, Moltkestraße 30, 76133 Karlsruhe, Germany

Abstract
Automotive configuration systems have been in productive use for many decades. Although historically mainly dealing with mechanical components, there has been a tremendous shift towards electronic, software- and cloud-based components and companion services over the past years. Systems have been extended accordingly, leading to a divergence in methods used for product description, such that today one has to deal with a variety of heterogeneous techniques and systems. In this paper we describe the current situation in the automotive industry and possible ways towards more uniform formalisms for the future taking, in particular, concepts from programming languages into account.

1. Introduction

Cars are complex products assembled from a large number of components, including e.g. motors, tires, seats, and entertainment units. Depending on customer wishes and production sites, the number of variants, in which a particular car model can be produced, often exceeds the number of atoms in the universe.

Over the last years, more and more software features are incorporated into cars leading to an even larger space of configurability, with fast-changing software features and versions, cloud components, and updates.

Configuration data is relevant in many business units and departments of an automotive company, including sales, engineering, production, and after-sales, among others. The systems and formalisms employed are, however, not unified and homogeneous. Instead, different departments use different systems with (often only slightly) differing syntax and semantics. Electronic equipment is handled separately from mechanical components in engineering, the sales department uses simplified models (hiding, e.g. features relevant only to control the production process), factories need additional information about part availability and timing.

Thus, a large number of heterogeneous systems are nowadays present in automotive companies. Differing syntax and semantics in these systems result in complex interfaces and overly complex processes. Thus, working towards a unified product modeling language seems to be a profitable venture.

2. Diversity in Automotive Product Configuration

Current systems for automotive variability modeling at many (European) car manufacturers have core characteristics like:

- A large number of (Boolean) configuration options (called *codes* at Mercedes-Benz or *features* in the SPL community), typically around or above a few thousand.
- A large number of constraints (Boolean formulas) between these options, typically in the tens of thousands. Each constraint may include only a few or up to several dozens of codes.

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
daniel.bischoff@mercedes-benz.com (D. Bischoff); carsten.sinz@h-ka.de (C. Sinz)
0000-0001-7116-9338 (C. Sinz)

- A two-level configuration formalism, consisting of a high-level specification language based on the codes (called *product documentation* at Mercedes-Benz), and a low-level language dealing with parts and their association with the high-level codes. Constraints can only be formulated in the high-level language. Mapping from combinations of high-level codes to parts is accomplished using Boolean formulas over the codes, describing under which conditions a part is included in the car.
- Conceptually, the presence of many different views on the product documentation, e.g., restricted to a particular country a car is shipped to, or to certain product (sub-)lines, e.g. convertibles. Practically, tool support for generating and dealing with views is still limited, though.
- A sprawling and heterogeneous landscape of software systems and tools, often evolving over many years or even decades, leading to inconsistencies and a lack of uniformity. Additional systems are required to configure and maintain parameters of electronic components (e.g., the number of motor steps needed to open a window). Software versions must be managed throughout a vehicle's lifetime, including support for over-the-air updates.

As an example, let's consider the research and development departments of any OEM. They integrate a wide range of data formats, systems, domains, and processes that collectively determine the current "data state" for all products and the production environment.

One system deals with production planning and bill-of-materials (BOM), while another system extends this data with plant-specific views, including, e.g., deployment dates or parts availability.

The types of computations performed on this data includes, among others:

1. *Buildability checks* validate whether complete orders can be built and evaluated.
2. *Completion* of partial vehicle configurations to form valid, buildable vehicles.
3. *Parts requirement analysis*, where "part" can refer to a physical component, a color, or even a software parameter.

The product data is continuously transferred between different departments, extended and aggregated, e.g., for certification or to comply with standards (e.g. WLTP) including measuring emissions, fuel and energy consumption.

3. Towards a Unified Language

To integrate different automotive configuration systems we propose a unified language with special features tailored in particular for the automotive domain. Core properties we envisage:

1. *Type system:* The main types are Boolean (reflecting *codes*) and enumeration type (groups of mutually exclusive components). Numerical attributes should also be expressible (e.g. for power consumption) via integer and floating point types.
2. *Constraints:* Complex interactions between Boolean or enumeration variables, as well as numeric properties, should be expressible as rules in a suitable language.
3. *Separation between features and parts level:* There is a separation between the feature space and the parts space with a mapping from (high-level) features to (low-level) parts. Currently, we assume that constraints are solely expressed on the high/feature level.
4. *Modularization:* On the parts level, it should be possible to express nested sub-structures reflecting the cars' assembly process.
5. *Versioning:* Sophisticated versioning should be available (in particular needed for software components), e.g. using semantic versioning.
6. *Timed:* Temporal restrictions should be possible on most entities (rules, parts, features) indicating time intervals in which they are available or enabled.

Our goal is to develop a specification language – or even a programming language – tailored towards product configuration and development, with a particular focus on the automotive industry.

In software development, key challenges such as managing complexity and supporting evolution are central. Programming languages and software management tools (e.g. git) have a long history. We thus aim to explore whether concepts from this domain can be effectively transferred to the context of product development.

4. Transferring Concepts from Programming Languages

In this section we want to recapitulate features from programming languages that might also be suitable for describing products and their development.

Type Systems. Type systems serve as a means to specify the set of values a variable can take and the structure of this set. So, e.g. *product types* (such as structs or classes) have the effect of an AND-operation (an object needs attribute 1 and attribute 2, etc.), wheres *sum types* (e.g. enumerations) have the effect of an OR-operation. Using both types, in principle, complex Boolean type structures can be built, however, in a cumbersome way.

So extending a type system with sum and product types by special constructs, e.g. extensions or restrictions of enumerations, adding rule-based type restrictions seems appropriate.

Object-Orientation. Features and parts can have attributes, often addressed with a dot-syntax, in object-oriented languages. Variables for, e.g., different motors, `M1`, `M2`, modeled as features, could have properties such as displacement, or number of cylinders, which can be described as attributes:

```
M1.displacement = 2.0
M2.nrCylinders = 6
```

An attribute a can be regarded as a functional dependency of an object o (feature or part), which can be addressed as $a(o)$. Typically, there is no ambiguity with which object an attribute is associated.

This is different for rules (a.k.a. constraints), which connect several objects, and where it is often not clear, which is the "main object" the rule should be associated with. Even for simple exclusions such as $\neg(A \wedge B)$ this problem shows up: Should this constraint be associated with A or B? For implications $A \implies B$ the constraint is often stored together with the antecedent (A). But this has the effect, that either the constraint has to be shown a second time for B, or the link of the constraint to B is not visible directly. No simple way seems to be available to solve this association problem. In practice, data engineers develop semi-strict patterns and standards, which, at least for the editing use cases, seem unavoidable. For consumers of the same data, different approaches are available.

Additionally, to alleviate the problem of associating constraints, we propose to group rules based on their *purpose*. There could be groups for legal restrictions in a particular country, for geometrical restrictions based on part geometry, etc.

Such grouping, if nested, enforces an order on which purpose is considered the most important (the outermost group) and which is less important or "smaller" (the innermost group).

Such an enforcement could be avoided by associating rule groups with variables for restrictions, of which multiple could be applied to a group, or *context*, as we will call it now.

```
context Country=USA, Motor=M2 {
  // rules concerned with use of Motor M2 in USA
}

context Country=Austria {
  // rules concerned with Austria
}
```

Versioning. Almost every entity in automotive product configuration is subject to changes over time. This includes features, their attributes, constraints between them, parts, and more. Thus versioning should be included in the language, e.g., in the form of semantic versioning, as often used in software development. Moreover, parts can be equipped with further constraints that are added at each plant, e.g., reflecting parts availability.

For complete vehicles, Mercedes and many other OEMs use specific codes to identify the so called model year of vehicles.

Integration in the Business Process. A configuration language is always part of a larger business process in practice. We therefore propose an integration approach as follows (cf. Fig. 1): Users generate configuration models using visual tools, specialized for different purposes. These tools translate visual constraints into the unified product configuration language that serves as an integrating formalism. Then, additional tools can be employed to run algorithms on a logical (Boolean) model derived from the configuration language, e.g. to detect errors or inconsistencies in the model, to determine parts requirements, etc.

Figure 1: Integrating a unified product configuration language in a business setting.

Business Perspectives. Visualization and restricted views are an important aspect to make large knowledge bases comprehensible. To that end, in the Poseidon approach presented in [1], we showed how to use a) partial assignments (for staged configuration in the sense of [2]) on a Boolean formula level, b) projection, and c) configurable variable orderings (ROBDD-likes) to produce concise and customizable data visualizations of such configuration data.

Using Poseidon's techniques, a sales department can, for example, project constraint sets into the subset of variables they care about. Thereby the technical details of *why* some option excludes another might be hidden, only the fact itself remains as a constraint between the sales-level variables. The engineering departments, in turn, can project to a more technical level.

In general, state-of-the-art PLM/PDM systems allow modeling of variability, nowadays often using object-like or predicate-logic notation (e.g. feature models, domain specific configuration languages). These notations typically need to be translated to a suitable logic, e.g. finite-domain logic or propositional logic, for algorithmic processing (e.g. completeness checks, performance in evaluation, etc.) anyway, which is why we propose a common language on the propositional level.

Modularization and Subsystems. Modularization in programming languages has several purposes: separation of concerns, improved readability and maintenance, code reuse and simplified collaboration, among others. It is achieved by multiple concepts, varying from one language to another, e.g., by breaking large programs in multiple files, folders or packages; by introducing namespaces, functions or classes; or by adding access modifiers (private variables not visible outside a certain scope).

All these concepts could be transferred to configuration. With textual variability modeling languages, splitting into multiple files and folders seems a simple first approach.

Some Examples. We shortly want to give some first ideas, how the syntax of such transferred contexts could look like:

```
group Motor {   // an enumeration type
  M1, M2, M3    // for different motors
}

group SalesTypes {
  S1: M1, T1  // group of three sales
  S2: M2, T1  // types, each with a set of
  S3: M3, T2  // defining component options
}
```

```
group Transmission {
  T1, T2
}

context USA {
  not M1 and not T2
}
```

Following these definitions, an overall high-level constraint, hlc, could be derived, e.g. for a tool used in the sales department. We use pseudo-boolean exactly-one-constraints (EXO) to express necessity and pairwise exclusion for the groups:

$$\begin{aligned}\text{hlc:}\quad & \text{EXO}(\texttt{M1},\texttt{M2},\texttt{M3}) \wedge \text{EXO}(\texttt{T1},\texttt{T2}) \wedge \\ & \text{EXO}(\texttt{S1},\texttt{S2},\texttt{S3}) \wedge (\texttt{S1} \longleftrightarrow \texttt{M1} \wedge \texttt{T1}) \wedge \\ & (\texttt{S2} \longleftrightarrow \texttt{M2} \wedge \texttt{T1})\ \wedge (\texttt{S3} \longleftrightarrow \texttt{M3} \wedge \texttt{T2}) \wedge \\ & (\texttt{USA} \longrightarrow \neg\texttt{M1} \wedge \neg\texttt{T2})\end{aligned}$$

Now, a business perspective for sales could be calculated by projecting the high-level constraint to the set of sales types ($\texttt{S1}$, $\texttt{S2}$, $\texttt{S3}$) and markets ($\texttt{USA}$ and others). In other words, by existential quantifying out all non-relevant variables ($\texttt{M1}$, $\texttt{M2}$, $\texttt{M3}$, $\texttt{T1}$, $\texttt{T2}$) in hlc, we obtain:

$$\begin{aligned}\text{projection}(\text{hlc}, \{\texttt{S1},\texttt{S2},\texttt{S3},\texttt{USA}\}) &= \exists\, \texttt{M1},\texttt{M2},\texttt{M3},\texttt{T1},\texttt{T2} : \text{hlc} \\ &= (\texttt{S1} \wedge \neg\texttt{S2} \wedge \neg\texttt{S3} \wedge \neg\texttt{USA}) \\ &\quad \vee (\neg\texttt{S1} \wedge \texttt{S2} \wedge \neg\texttt{S3}) \\ &\quad \vee (\neg\texttt{S1} \wedge \neg\texttt{S2} \wedge \texttt{S3} \wedge \neg\texttt{USA})\end{aligned}$$

This result shows that both $\texttt{S1}$ and $\texttt{S3}$ are not valid in the USA, while $\texttt{S2}$ has no such restriction. The result is visualized in Fig. 2 using the Poseidon tool.

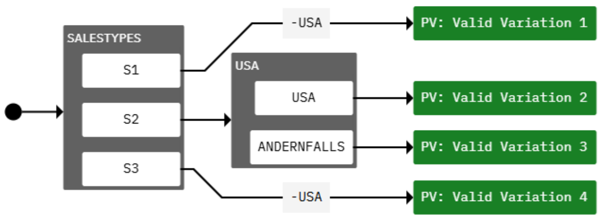


Figure 2: A visualization of the validity constraint from a sales perspective derived by projecting the high-level constraint to the set of sales types and markets, sorting those options and compiling them into an (RO)BDD-like structure.

5. Related Work

There has been extensive work on formalisms for product configuration, both in textual and graphical form. Due to the vast number of relevant contributions in this area, it is difficult to adequately acknowledge all authors. We thus want to focus on a few. Felfernig et al. provide the foundational material for modeling and composing complex products [3]. Schmid and Eichelberger maintain a list and classification of textual variabiltiy modeling languages, which is publicly available under https://sse.uni-hildesheim.de/en/textual-variability-overview. This list has last been updated in June 2017, but still gives a good reference to the work until then. On this website they also cover, e.g., scalability support (D.4). Another nice overview is given by Beek *et al.* [4]. The Universal Variability Language (UVL) is a community effort towards a unified language for variability models [5, 6]. These languages are intended to be "general-purpose", in that they do not focus on a particular industry.

There have also been proposals tailored towards the automotive industry, e.g. by Zellmer *et al.* [7], Visser *et al.* [8] or Jost and Sinz [9].

6. Conclusion

This short paper is intended to reflect the current state of product configuration in automotive industry practice, which is characterized by many non-uniform specification mechanisms and accompanying systems. This heterogeneity leads to challenges in maintaining both systems and product data, as well as for integrating new requirements – particularly those arising from the increasing shift toward software-based components.

We assembled a set of properties that we consider important in a formalism for integrated, large-scale automotive product configuration; however, no project has yet been launched to realize the approach, and its scalability still needs to be assessed in practice.

Moreover, while the extent to which our approach will be adopted by industrial practitioners remains to be demonstrated, we regard the prospects as promising. With the integration approach outlined above (see Fig. 1) and the support of intuitive visual tools, we anticipate that acceptance in industry can be greatly strengthened.

Acknowledgments. We would like to thank Christian Seiler and Klaus Anwender from Mercedes-Benz AG for fruitful discussions on type-based product line engineering (TPLE) for Mercedes' cars.

Declaration on Generative AI. During the preparation of this work, the authors used a Mercedes-internal GenAI tool (based on GPT-4o) and Thesify for grammar, spelling checks, and literature proposals. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] D. Bischoff, W. Küchlin, O. Kopp, Poseidon: A graphical editor for item selection rules within feature combination rule contexts, in: IFIP International Conference on Product Lifecycle Management, Springer, 2022, pp. 3–14.

[2] K. Czarnecki, S. Helsen, U. W. Eisenecker, Staged configuration using feature models, in: R. L. Nord (Ed.), Software Product Lines, Third International Conference, SPLC 2004, Boston, MA, USA, August 30-September 2, 2004, Proceedings, volume 3154 of *Lecture Notes in Computer Science*, Springer, 2004, pp. 266–283. URL: https://doi.org/10.1007/978-3-540-28630-1_17. doi:10.1007/978-3-540-28630-1_17.

[3] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based configuration: From research to business cases, Newnes, 2014.

[4] M. H. t. Beek, K. Schmid, H. Eichelberger, Textual variability modeling languages: An overview and considerations, in: Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B, SPLC '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 151–157. doi:10.1145/3307630.3342398.

[5] C. Sundermann, K. Feichtinger, D. Engelhardt, R. Rabiser, T. Thüm, Yet another textual variability language? a community effort towards a unified language, in: Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A, SPLC '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 136–147. URL: https://doi.org/10.1145/3461001.3471145. doi:10.1145/3461001.3471145.

[6] C. Sundermann, S. Vill, T. Thüm, K. Feichtinger, P. Agarwal, R. Rabiser, J. A. Galindo, D. Benavides, UVLParser: Extending UVL with language levels and conversion strategies, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume B, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 39–42. doi:10.1145/3579028.3609013.

[7] P. Zellmer, L. Holsten, T. Leich, J. Krüger, Product-structuring concepts for automotive platforms: A systematic mapping study, in: Proceedings of the 27th ACM International Systems and Software Product Line Conference - Volume A, SPLC '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 170–181. doi:10.1145/3579027.3608988.

[8] R. Visser, A. Basson, K. Kruger, An architecture for the integration of product and production digital twins in the automotive industry, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS Companion '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 431–441. doi:10.1145/3652620.3688257.

[9] F. Jost, C. Sinz, Handling automotive hardware/software co-configurations with integer difference logic, in: Proceedings of the 18th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 103–111. doi:10.1145/3634713.3634728.

[10] P. Ochs, T. Pett, I. Schaefer, Consistency is key: Can your product line realise what it models?, in: Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS Companion '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 690–699. doi:10.1145/3652620.3687812.

[11] M. Eggert, K. Günther, J. Maletschek, A. Maxiniuc, A. Mann-Wahrenberg, In three steps to software product lines: a practical example from the automotive industry, in: Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A, SPLC '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 170–177. doi:10.1145/3546932.3547003.

Toward a Contingent-Configurational Perspective on Configuration Systems in the AEC Industry

Julius Olukayode Oluwole[1,*], Enrico Sandrin[2,†] and Cipriano Forza[3,†]

[1,2,3] *University of Padua, Vicenza, Italy*

Abstract

The Architecture, Engineering, and Construction (AEC) industry faces increasing pressure to deliver customized solutions at scale, yet research and practice remain fragmented around configuration systems. This configuration-centric systematic literature review synthesizes 137 publications, mapping customization strategies, enabling mechanisms, and performance outcomes. Results highlight configuration systems as essential for advanced customization but reveal significant gaps in theory, terminology, and empirical validation. To address this, we propose an integrative analytical framework—structured around customization strategies, enablers, and outcomes—interpreted through the Technology–Organization–Environment (TOE) lens. We outline a research agenda to bridge theory and practice and support scalable and adaptive customization in digitalized AEC industry. This review provides a foundation for more context-sensitive, theory-driven approaches to configuration in the sector.

Keywords

Product configuration systems, Mass customization, Architecture, Engineering, and Construction (AEC), Systematic literature review, Technology–Organization–Environment (TOE) Framework

1. Introduction

The Architecture, Engineering, and Construction (AEC) industry is undergoing rapid transformation in response to increasing demand for flexibility, efficiency, and end-user customization [1, 2, 3]. Driven by the twin forces of digitalization and industrialization [4], construction stakeholders are seeking new strategies and tools to deliver bespoke solutions at scale, moving beyond traditional approaches toward better performing modes of production [5, 6]. However, despite significant technological advancements and a proliferation of customization practices, the systematic integration of configuration systems in AEC remains under considered in both research and in practice [7, 8]. This is further complicated by the socio-technical complexity, fragmentation and the need for integrated systems approaches in digitalized AEC and modular construction, as shown by recent work on the complementarity of systems integration and Building Information Modeling (BIM) [29], and the foundational challenges of complexity in modular construction [30].

Product configuration systems, long established in sectors such as manufacturing, automotive, and Information and Communication Technology (ICT) [9, 10], offer the potential to manage complexity, enable mass customization, and bridge the gap between client requirements and industrialized delivery in building construction. Yet, in the AEC domain, research on customization strategies is fragmented, with limited adoption of theoretical background and terminology which is established for configuration systems and configuration-based customization approaches. The AEC sector therefore faces a critical need for structured frameworks that can guide the design, implementation, and

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

[1*] Corresponding author.
[†] These authors contributed equally.
juliusolukayode.oluwole@studenti.unipd.it (J. O. Oluwole); enrico.sandrin@unipd.it (E. Sandrin); cipriano.forza@unipd.it (C. Forza)
0009-0009-2578-1839 (J. O. Oluwole); 0000-0001-9170-0683 (E. Sandrin); 0000-0003-4583-2962 (C. Forza)

evaluation of configuration-based customization strategies, especially as project delivery grows increasingly complex and multi-actor in nature [1, 11, 24].

This paper addresses these gaps by presenting a configuration-centric systematic literature review (SLR) that classifies and synthesizes the current body of literature on customization in AEC. Using a novel analytical framework that integrates both established customization strategies and core mass customization (MC) enablers [10, 13] alongside inductively identified enablers and performance outcomes, the review maps sector-specific patterns, trade-offs, and theoretical limitations in existing studies. In particular, the findings highlight the limited uptake of configuration concepts (such as the operationalization of established models and terminology from configuration body of knowledge) and the absence of context-sensitive, theory-driven frameworks that address the contingent nature of configuration system integration in AEC.

Based on this comprehensive synthesis, the paper develops an integrative analytical framework that structures the field around customization strategies, enabling mechanisms, and performance outcomes, employing the Technology–Organization–Environment (TOE) theoretical lens to interpret how technological, organizational, and environmental contingencies influence mass customization strategies in the AEC sector [14]. Building on these insights, we outline a future research agenda, to ground further theory development and contribute to bridge the gap between academic research and practical application on this topic.

This paper systematically synthesizes evidence from 137 publications on configuration systems, customization strategies, and enabling mechanisms in the AEC sector. By combining this evidence with a forward-looking research agenda, this work provides a structured evaluation of the current configuration body of knowledge. This approach lays a robust foundation for advancing both research and practice at the intersection of configuration knowledge, digital transformation, and innovation in building construction. In summary, the current literature is characterized by persistent fragmentation, socio-technical complexity, and a lack of context-sensitive, theory-driven frameworks for configuration system integration in AEC.

To address these challenges, we propose a contingent-configurational perspective of configuration system integration in the AEC sector. The “configurational perspective” emphasizes the importance of internal consistency among multiple interdependent elements within an organization or system to achieve effectiveness. In our context, successful outcomes depend on achieving a good fit among various enablers, so that they work coherently together. The “contingent perspective” highlights that the effectiveness of enabler configurations depends on contingency factors—such as technological, organizational, and environmental conditions, as interpreted through the TOE framework. This theoretical perspective argues that optimal outcomes are not achieved by rigidly applying the same enablers in every situation, but by adapting them to the specific strategy and context. By explicitly articulating this contingent-configurational perspective, the paper offers a new way to interpret the diverse patterns, trade-offs, and gaps identified in the literature, and establishes a foundation for both research and practice to move toward more adaptive, scalable, and effective customization in the digitalized AEC industry.

2. Background & related work

Product configuration systems have long been established as essential enablers of mass customization in industries such as manufacturing and automotive, where rule-based logic, modular product platforms, and digital tools allow organizations to deliver individualized solutions efficiently at scale [5, 10]. Over the past two decades, configuration research has produced robust models for the design and management of customizable product families, supporting both academic inquiry and practical implementation [15, 10].

In the AEC sector, however, the adoption and theoretical integration of configuration systems remains limited. Although interest in mass customization, modularization, and digitalization has grown, reflected in studies on off-site construction, prefabrication, and BIM-enabled processes—most AEC research continues to focus on isolated technologies or project-level innovations [16, 11, 8, 12].

Explicit application of configuration logic, and systematic frameworks for linking customization strategies to enabling mechanisms and performance outcomes, are still rarely found in the AEC literature.

Foundational theories of mass customization [6, 10] offer important conceptual tools for understanding the design and implementation of customized solutions. Yet, their translation into the AEC context remains patchy and inconsistent. Recent systematic reviews have highlighted the fragmented nature of AEC customization research, the absence of performance-based classification schemes, and the lack of theory-driven approaches that address organizational, technological, and project-level contingencies [17, 18, 19, 20]. Similar integration challenges, arising from the interplay of technical systems and organizational processes, are widely recognized in recent studies of modular construction and digital integration in AEC, where the socio-technical complexities of managing systems, technologies, and collaborations have been explicitly highlighted [29; 30].

To address these limitations, this paper presents a configuration-centric SLR that classifies and synthesizes 137 publications at the intersection of customization and configuration in the AEC industry. By building on an integrative analytical framework, this review provides a structured synthesis of current knowledge, identifies sector-specific patterns and gaps, and establishes a foundation for future research directions, including the potential development of more context-sensitive and theory-driven frameworks tailored to the complexities of the AEC industry.

3. Method

This study employs a configuration-centric SLR to synthesize and classify research on configuration systems and customization strategies in the AEC industry. The SLR approach was selected for its capacity to rigorously map a fragmented field, identify theoretical and empirical gaps, and establish an evidence-based foundation for future research. The review protocol was developed and implemented in accordance with established SLR guidelines [21, 22].

The review focused exclusively on literature that addresses the integration, implementation, or evaluation of configuration systems, configuration logic, or related strategies within the AEC context. A comprehensive search was performed in the Scopus database, using a set of keywords and Boolean operators targeting configuration, customization, modularization, and AEC-specific terms. The final search string was: (configurat* OR customi* OR personali* OR individuali* OR "made to measure" OR "engineer* to order" OR "custom made" OR variet*) AND (aec OR architect* OR construction OR building OR hous* OR dwelling OR "infrastructure project") AND ("mass customi*" OR "mass personali*" OR "industrial construct*" OR "off-site construction" OR modular* OR platform OR "additive manufacturing" OR "3d print*" OR bim OR "build* information system*" OR "prefabricated" OR "precast" OR "volumetric" OR "paneli*" OR "industriali*").

Scopus is used as the sole indexing source due to its broad, cross-disciplinary coverage of engineering, construction, and information systems; unified metadata (e.g., DOIs, affiliations) enabling consistent coding and de-duplication; and export functions that support transparent replication of the search. This choice entails potential database bias and the omission of niche or regional outlets not indexed by Scopus. To mitigate this limitation in future replications, the search may be triangulated with complementary sources (e.g., Web of Science).

Scopus was searched on October 2024 for records from database inception–October 2024, querying title–abstract–keywords using the Boolean string reported above. At import, English-language and document-type limits were applied (articles, reviews, conference papers, books/chapters). The search retrieved 138,603 records. A quality-filtering step was then applied to manage volume while preserving influence: books/chapters/conference papers published before 2021 were retained only if cited at least once, whereas all journal articles were retained regardless of year. After these automated filters, 123,188 records proceeded to screening. A summary of the selection process is shown in Figure 1 (PRISMA), and stage counts by source type are listed in Table 1.

Studies were included in the review if they described, analyzed, or deployed a configuration system, or a functionally equivalent mechanism (such as a rules-based process, platform logic, or systematized modularization that enables user-driven product configuration), as part of their customization approach in the AEC sector. The inclusion criteria were also extended to studies providing empirical, theoretical, or conceptual insights into these mechanisms, even if not labeled explicitly as configuration systems. Conversely, studies focused solely on isolated digital or manufacturing technologies, or on general customization practices without explicit or implicit links to configuration logic, were excluded to ensure a targeted, configuration-centric dataset.

The screening process followed a multi-stage approach, beginning with title and abstract screening and followed by a full-text review and snowballing. Title screening identified 717 publications, abstract screening narrowed these to 215 publications and full-text review yielded 132 publications. Snowballing identified five additional sources (three journal articles and two conference papers), resulting in a final dataset of 137 publications: 74 journal articles or reviews and 63 books and conference papers. The selection process is summarized in Figure 1, with stage counts by source type in Table 1.

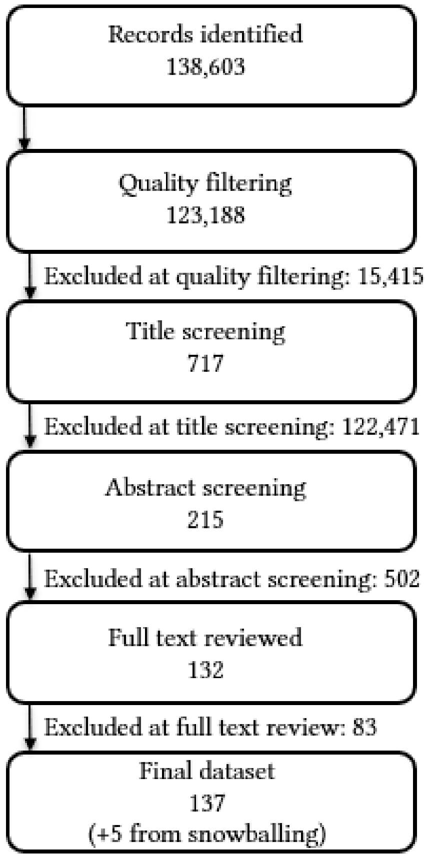


Figure 1: PRISMA flow diagram summarizing the review process from identification to inclusion, with counts at each stage; final included studies = 137.

Each retained study was systematically coded using an analytical framework adapted from [6, 10], tailored for application in the AEC context. The customization strategy component of the framework used in coding comprised five categories (pure customization, customized fabrication, customized assembly, customized distribution, and variety without customization) deductively derived from established literature [6, 10].

Table 1
Overview of screening phases and publication counts

Stage	Journals	Books & Conferences	Total
Initial search results	74,139	64,464	138,603
Phase 1: Quality Filtering	74,139	49,049	123,188
Phase 2: Title Screening	378	339	717
Phase 3: Abstract Screening	123	92	215
Phase 4: Full-Text Review	71	61	132
Phase 5: Snowballing	3	2	5
Final dataset	74	63	137

Enabling mechanisms were coded into core and other classes using operational criteria. An enabler was classified as core when it directly instantiated configuration by generating or validating options and/or enforcing product–process rules; practically, removing it would break configuration because choices could no longer be translated into a feasible, manufacturable or constructible solution. An enabler was classified as other when it supported, integrated, extended or scaled configuration (e.g., via data environments, automation, or delivery methods) without itself encoding option-generation or rule logic. Consistent with prior implementation-guideline reviews, the core set comprises IT-based product configuration (PC), product platform development (PP), product modularization (M), process modularity (PM), part standardization (S), group technology (GT), form postponement (P), and concurrent product–process–supply-chain engineering (CE). Suzić et al. [13] explicitly identify these eight as foundational mass-customization enablers and discuss their typical interdependencies and sequencing in implementation guidelines, reinforcing their classification as "core" [13]. Each enabler is classified as core because it directly instantiates configuration: PC encodes options and constraints and emits validated solutions; PP provides common architectures and parameters that generate families of variants; M enables variety through re-combinable modules; PM decouples subprocesses so configured variants can be executed or substituted without global disruption; S constrains part variety to keep the rules and option space tractable; GT structures similarity families that discipline variant rules; P defers differentiation so configuration rules drive late-stage options; and CE integrates design, manufacture and logistics early to maintain feasibility of configured options [13].

By contrast, Digital Integration (e.g., BIM, CAD, digital twins), Emerging Technologies (e.g., 3D printing, AI, IoT, AR/VR), and Off-site construction methods (panelised, volumetric, hybrid) were identified inductively from recurrent patterns in the AEC literature and are classified as other (supportive) mechanisms: they connect actors and systems, extend capability, or industrialize delivery, but do not themselves instantiate configuration.

Each publication was further classified according to the performance dimensions it addressed (cost, time, quality, flexibility, scalability, and sustainability) and the type of evidence reported (quantitative, qualitative, conceptual, or not reported). The performance dimensions of cost, time, quality and sustainability were deductively derived from established literature on mass customization and configuration in AEC [16, 18, 26]. Here, flexibility encompasses both design flexibility (the ability to accommodate a variety of customer and project requirements through modularization and kit-of-parts) and process flexibility (the ability to adapt production and assembly processes across project phases). The additional performance dimension of scalability was included inductively as it emerged as a significant theme during the review process. Evidence types were defined deductively, following established SLR guidelines [21, 22].

The analysis and synthesis combined descriptive statistics, heatmaps, and cross-tabulation to analyze the distribution and co-occurrence of enabling mechanisms and customization strategies, and to map performance outcomes across the literature. Each publication was first coded for its primary customization strategy, forming the basis for further analysis. All discussed enablers (core MC and others), were identified and recorded, allowing for a detailed mapping of enabler presence by customization strategy. The analysis distinguished between studies examining single versus bundled enablers, with "bundled" referring to cases where two or more enablers were present, regardless of whether they were explicitly integrated. In a further step, the review sought to identify cases of genuine synergy—where two or more enablers were not just present, but functionally integrated or operationally combined, resulting in demonstrable mutual benefit or new capabilities. Synergy types were classified as core MC to core MC, core MC to other, and other to other enabler integrations.

Finally, thematic coding was applied to extract insights across the six performance dimensions. This multi-step synthesis enabled the identification of sector-specific patterns, trade-offs, and context-sensitive high-performing configurations. Studies were systematically grouped by customization strategy and by the presence, bundling, and synergy of enablers, supporting systematic comparisons that highlight both theoretical and practical implications for the integration of configuration systems in the AEC industry. This structured and transparent approach provides a rigorous basis for mapping the current state of research and identifying critical gaps in the literature on configuration systems within the AEC sector. Figure 2 summarizes the four analysis domains; results follow in section 4.

The objective of the review is to explain how customization strategies interact with enabler types to influence cost, time, quality, flexibility, scalability and sustainability. Guided by prior theory and patterns observed in the reviewed literature, three propositions are examined: first, fit—that configurations exhibiting stronger internal alignment between the chosen strategy and core enablers are associated with superior operational outcomes; second, complementarity—that bundles of mutually reinforcing enablers (for example, PP with PC, anchored in robust digital integration) yield super-additive performance relative to piecemeal adoption; and third, contingency—that Technology–Organization–Environment (TOE) conditions moderate these relationships, such that ostensibly similar bundles can perform differently across contexts. These propositions structure the synthesis and motivate the cross-tabulations and thematic analyses reported in Section 4.

4. Results

This section presents the findings of the SLR according to the analytical framework developed for this study (see Figure 2). The framework structures the analysis and the synthesis around four interdependent domains: customization strategies, core enablers, other enablers, and performance outcomes. The arrows show how each domain influences the others. Specifically, the choice of customization strategy (top of the framework) shapes which performance outcomes are prioritized and achieved. For example, adopting a pure customization strategy may maximize design flexibility and user satisfaction, but can increase cost and reduce scalability. In contrast, a variety without customization strategy (standardized products) might enhance efficiency, reduce cost, and speed up delivery, but may offer less flexibility or personalization. Customized fabrication and customized assembly offer trade-offs between flexibility, scalability, and efficiency, depending on how enablers are integrated. This direct link is represented by the arrow from “Customization strategy” to “Performance outcomes. This framework guided both the coding of studies and the thematic analysis, enabling a systematic mapping of research patterns, gaps, and actionable implications for the AEC sector. Each study was coded customisation strategy, enablers, outcomes, evidence type, the aggregated distribution are reported in the figures and tables in section 4 (Results).

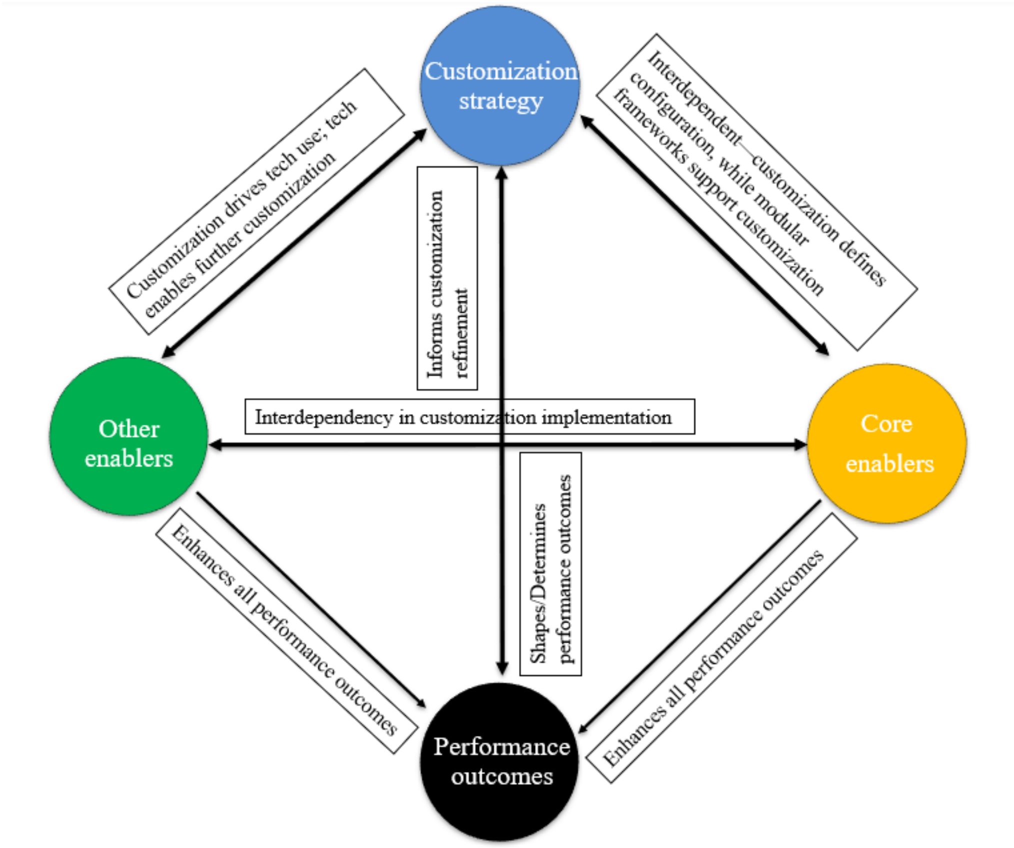


Figure 2: Analytical framework for coding and synthesis—block diagram showing four domains (customization strategy, core enablers, other enablers, and performance outcomes) with their interdependence (see Section 3).

4.1. Study and publications' set characteristics

The final SLR dataset comprises 137 publications spanning journal articles (74) and conference papers (63) published between 2005 and 2024 (conference papers published before 2021 have been retained only if they received at least one citation). The sample covers a broad spectrum of AEC contexts, including building construction, modular housing, and off-site manufacturing. Most studies appeared in the last ten years (72%), reflecting growing academic and industry attention to configuration and mass customization in AEC (Figure 3).

In terms of research methods, there is a predominance of conceptual and qualitative studies, with relatively few papers employing robust quantitative studies. This limited methodological rigor, particularly in assessing performance outcomes, highlights the need for more empirical validation in future research.

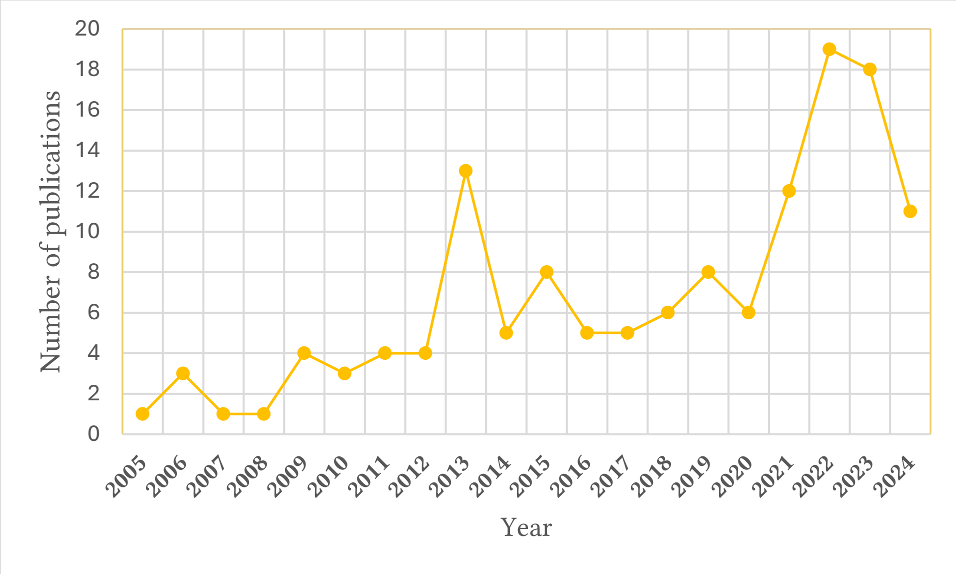


Figure 3: Annual number of included publications (2005–2024)—line chart showing a clear upward trend.

4.2. Distribution of customization strategies

Applying the analytical framework, analysis reveals that customized fabrication (45 publications, 33%) and pure customization (42, 31%) are the most prevalent strategies, together accounting for about two-thirds of the sample (see Table 2). Customized assembly is represented in 31 studies (22%), while variety without customization is least frequent (19, 14%). No publications were classified under customized distribution.

Table 2
Distribution of publications by customization strategy

Customization strategy	Number of publications	Percentage (%)
Pure customization	42	31
Customized fabrication	45	33
Customized assembly	31	22
Customized distribution	0	0
Variety without customization	19	14

In this review, pure customization is coded whenever end-user or project requirements influence the design, within a bounded solution space. This includes parameterized variants and engineer-to-order practices implemented via configurator platforms, parametric/BIM workflows, or equivalent rules-based processes. Under this operational definition, pure customization represents a large share of the sample (42/137; 31%), second only to customized fabrication (45/137; 33%). This explains why many studies fall into pure customization even when a configurator is not explicitly referenced, because rules-based parametric/BIM workflows or engineer-to-order processes meet the operational definition. This absence may reflect the nature of the AEC industry, where products are typically large, immobile, and project-specific, thus limiting opportunities for customer-driven distribution customization. These findings indicate a strong research focus on strategies that maximize design flexibility and user input, while digital integration tools (e.g., BIM/CAD) co-occur across all strategies, with the highest counts in pure customization (37 studies; 30.8%) and customized fabrication (36; 30.0%), and fewer in variety without customization (18; 15.0%).

The breakdown by execution type and project scope is shown in Table 5.

4.3. Adoption and roles of configuration systems

A total of 81 studies explicitly deploy or analyze configuration systems as core elements of customization. Among these, 50 incorporate modularization as a configuration mechanism (i.e., process or tool that enables the systematic definition, selection, or assembly of customizable building elements), while the remaining 31 utilize approaches such as BIM-based platforms, parametric modeling, and rule-based systems. Additionally, 56 studies employ digital tools or methods that enable systematic configuration or customization, even though they are not formally labeled or explicitly referred to as "configuration systems" in the studies. Of these, 23 incorporate modularization as a mechanism for customization, while the remaining 33 utilize tools such as BIM-based platforms, parametric modeling, and rule-based systems and AI-assisted decision support—used for configuration-like purposes but described using different terminology. Collectively, these findings indicate that both formally identified configuration systems and a broad range of digital tools and platforms (even when described with different terminology) contribute to customization in the AEC sector, highlighting the centrality of digitalization in contemporary AEC-related configuration research.

4.4. Enabler combinations and patterns

Figure 4 shows clear patterns in enabler use across customization strategies. Pure customization relies most on IT-based product configuration and digital integration. Customized fabrication exhibits the most diverse enabler mix, with frequent use of product modularization, process modularity, digital integration, and off-site methods. Customized assembly also combines modularization, process modularity, and digital tools, while variety without customization depends almost entirely on digital integration. Strategies like customized fabrication and assembly are more likely to integrate multiple enablers in combination, supporting higher customization and improved performance.

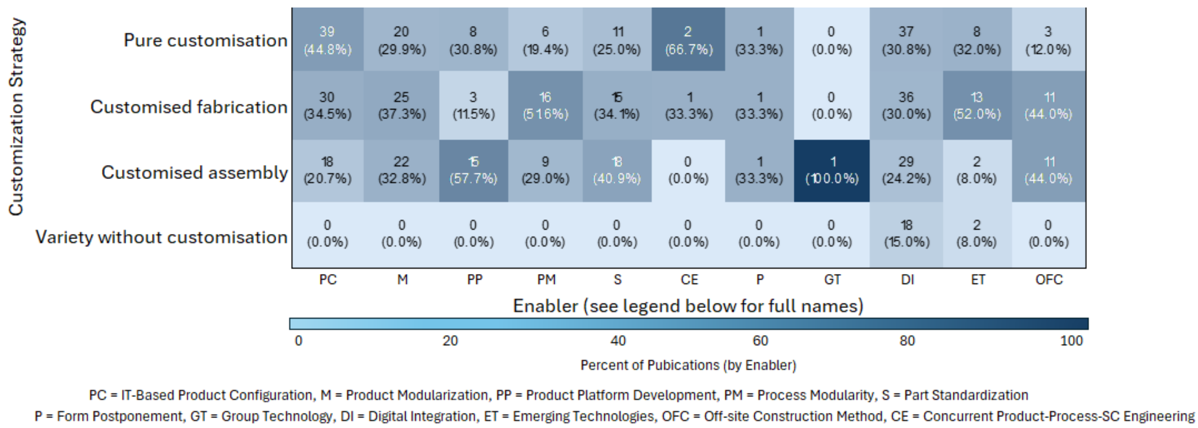


Figure 4: Co-occurrence of enablers per customization strategy. Darker cells indicate more frequent co-occurrence (e.g., IT-based configuration with modularization). Each cell reports the count and the column percentage (denominator = number of publications citing that enabler; column totals = 100%). Because publications can cite multiple enablers, row totals may exceed 100%. See legend for enabler abbreviations.

4.5. Enabler synergies

As seen in Figure 4, IT-based product configuration is the most considered enabler in pure customization, is the second most considered in customized fabrication and, though to a lesser extent, appear in customized assembly. Notably, all these three strategies involve the use of multiple enablers in combination.

To systematically identify patterns of enabler synergy, all 137 reviewed publications were coded not only for individual enablers, but also for the co-occurrence and integration of multiple enablers within each study. During data extraction, we specifically recorded instances where two or more enablers were functionally integrated (i.e., working together to enable or enhance customization outcomes), rather than merely present in the same project or case. Each instance of enabler co-occurrence was analyzed to determine whether it constituted a true synergy (i.e., an intentional and functional integration of two or more enablers resulting in enhanced customization, efficiency, or new capabilities, as reported by the study). This process enabled us to classify the observed synergies according to the nature of the enablers involved (Core MC ↔ Core MC, Core MC ↔ Other enabler, Other enabler ↔ Other enabler).

The most innovative and impactful approaches, as summarized in Table 3 were those in which studies provided empirical or conceptual evidence that such integration delivered substantial benefits (e.g., accelerated project delivery, improved information flow, increased client involvement, or operational efficiency).

Table 3
Detailed synergy examples

Study reference	Synergy type	Enablers involved	Context/Project type	Justification for synergy
Jensen et al. (2012), Automation in Construction	Core MC ↔ Core MC	Product Modularization, IT-based configuration	Prefabricated multi-storey timber building, floor slab modules	Modularization supplies standardized, parametric modules; the configurator operationalizes these by embedding rules/constraints, enabling automatic generation of buildable design variants.
Wang & Chen (2024), Buildings	Core MC ↔ Other enabler	Product configurator, BIM	Modular single-family housing (Canada)	BIM stores all parametric rules/data; the configurator uses this to auto-match user preferences with buildable, code-compliant solutions, enforcing constraints in real time.
Zhou et al. (2021), Automation in Construction	Other ↔ Other enabler	IoT, BIM	Modular public housing (Hong Kong, Modular Integrated Construction (MiC)	Functional integration of BIM and IoT (SBIM) supports the systematic configuration and real-time management of modular assembly, enabling real-time data-driven customization of on-site assembly processes.

The three principal types of synergy, derived from repeated patterns across the literature, are described below:

1. Core MC enabler ↔ Core MC enabler: This involves two or more core MC enablers (e.g., configuration systems, modular product/process/platform) are functionally integrated to enable customization
2. Core MC enabler ↔ Other enabler: This is when core MC enabler and other enabler are interconnected to enable data flow or operational feedback in support of customization
3. Other enabler ↔ Other enabler: This is when two or more other enablers (e.g., BIM, 4D, 3D Printing, AI) are used together in an integrated workflow to enhance customization outcomes

Table 3 provides detailed examples of these synergy types, the enablers involved, the context or project type, and the justification from recent studies. For instance, Jensen et al. [23] demonstrate how product modularization and IT-based configuration (core MC ↔ core MC) reduce design effort and accelerate time-to-market in prefabricated timber construction. Wang and Chen [24] exemplify core MC ↔ other enabler synergy by combining a product configurator with BIM to streamline planning and procurement in modular housing. Zhou et al. [25] showcase other enabler ↔ other enabler synergies, integrating IoT and BIM to automate workflows, enhance productivity, and reduce errors in both offsite and onsite construction contexts. It is worth noting that, consistent with our analytical approach, we included studies where digital platforms perform configuration-like functions, even if not explicitly referred to as “configuration systems” by the original authors.

These cases illustrate that intentional, well-designed enabler synergies, and not mere co-occurrence, are central to achieving advanced customization, operational efficiency, and new capabilities across the AEC sector. The evidence from the literature confirms that such integrations can drive substantial improvements in productivity, information flow, client involvement, and overall project outcomes.

4.6. Performance Outcomes and Evidence Quality

Performance outcomes are most often reported for cost and time, particularly in pure customization and customized fabrication. However, quantitative evidence is limited (12–32% for cost, 14–29% for time), with most studies relying on qualitative or conceptual arguments. For flexibility, scalability, and sustainability, empirical evidence is especially scarce; over 50% of studies offer only conceptual or no evidence for these dimensions. Overall, positive claims for customization are widespread, but supporting evidence is dominated by conceptual and qualitative findings, underlining the need for research with more quantitative empirical evidence. Table 4 summarizes the evidence distribution across six key performance dimensions by customization strategy. An overall summary of which outcomes are reported appears in Figure 5, while detailed breakdowns by strategy and evidence type are provided in Table 4.

Table 4
Performance outcomes across customization levels

Customization strategy	#	Cost (%)				Time (%)				Quality (%)				Flexibility (%)				Scalability (%)				Sustainability (%)			
		Q	D	C	N	Q	D	C	N	Q	D	C	N	Q	D	C	N	Q	D	C	N	Q	D	C	N
Pure Customization	42	12	5	71	12	14	17	67	2	2	24	74	0	2	73	25	0	1	4	82	13	3	13	39	45
Customized Fabrication	45	18	4	76	2	22	11	65	2	9	20	69	2	1	19	79	1	1	1	97	1	8	6	44	42
Customized Assembly	31	32	3	65	0	29	20	48	3	0	45	48	7	0	35	60	5	3	8	82	7	4	6	30	60
Variety Without Customization	19	26	5	58	11	16	11	63	10	10	26	53	11	0	6	44	50	0	6	44	50	14	5	41	40

Grading Key: Q = empirical quantitative, D = empirical qualitative/descriptive, C = conceptual/speculative, N = no evidence

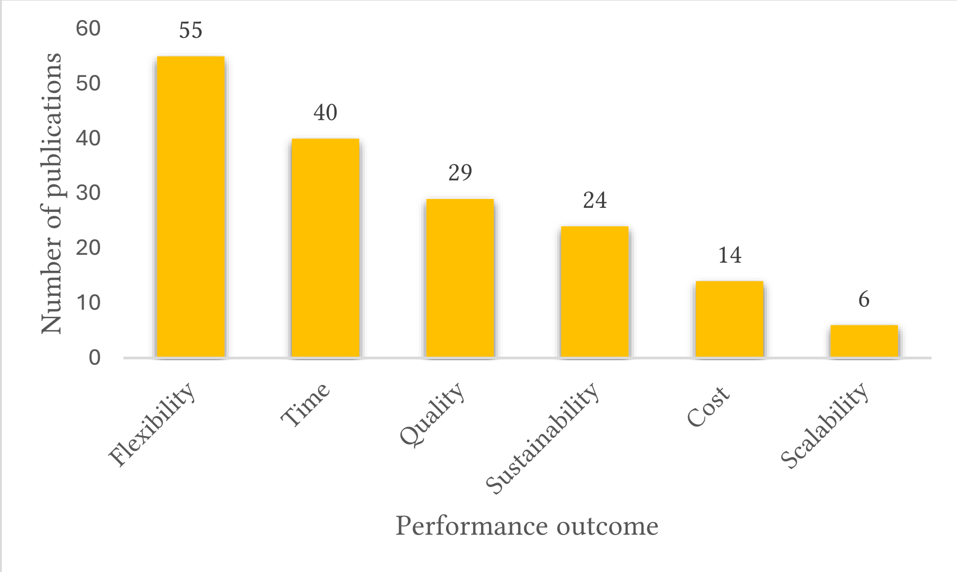


Figure 5: Share of publications reporting each performance outcome (n = 137). Totals exceed 100% because studies can report multiple outcomes.

4.7. Sector-specific patterns and trade-offs

Off-site and hybrid execution modes are most frequently reported in research on customization strategies. In this context, “research on customization strategies” refers to studies identified and classified in the review according to the primary customization strategy addressed—such as pure customization, customized fabrication, customized assembly, and so on—as described in Section 3. Research on customized fabrication is heavily concentrated in off-site contexts, whereas research on pure customization spans off-site, hybrid, and on-site implementations. Research on customized assembly is also closely associated with hybrid and off-site execution. In terms of application scope, research on pure customization often targets whole-building solutions, while research on customized fabrication and assembly is oriented toward component-level interventions. Most reviewed projects are new-builds, but some evidence of retrofit applications exists, particularly in research on customized fabrication and assembly.

Actor involvement differs across strategies: architects are central to pure customization, engineers to customized fabrication and assembly, and manufacturers are more visible in customized assembly. Client involvement is highest in pure customization, aligning with its user-driven nature.

These sector specific patterns highlight the contingent nature of customization strategies in the AEC industry. Execution mode, project scope, actor roles, and client involvement each condition the choice and effectiveness of a given customization strategy—demonstrating that configuration solutions must be tailored to specific technological, organizational, and environmental contexts. This reinforces the value of adopting a contingent-configurational perspective in analyzing and implementing customization in the sector.

Table 5
Distribution of customization strategies by execution type, project scope, project type, and actor involvement

Customization strategy	Execution Type				Scope			Project Type			Architect		Engineer		Client		Developer		Manufacturer	
	On-site	Off-site	Hybrid	Unstated/ unclear	Whole building	Component level	Unstated /unclear	New	Retrofi	Both	#	%	#	%	#	%	#	%	#	%
Pure Customization	10	16	12	4	22	20	0	39	0	3	18	25%	17	23%	16	22%	11	15%	11	15%
Customized Fabrication	4	30	9	2	15	30	0	38	4	3	19	32%	30	48%	3	5%	5	8%	5	8%
Customized Assembly	3	14	11	3	6	24	1	24	2	5	12	26%	15	32%	2	4%	9	19%	9	19%
Variety Without Customization	9	0	3	7	7	11	1	13	0	6	6	29%	12	57%	1	5%	1	5%	1	5%

As summarized in Table 5, these patterns reveal important trade-offs: strategies that maximize flexibility and whole-building customization increase complexity and demand strong digital infrastructure and collaboration, while component-level, engineer-driven strategies are more scalable but may offer less deep personalization. The diversity of execution modes, project scopes, and actor roles emphasizes the context-dependent nature of successful configuration implementation—a relationship captured by the framework and interpreted through the TOE lens.

4.8. Theoretical and practical gaps

Despite substantial progress, several limitations persist in the literature:

1. Enablers are often considered in isolation by researchers, rather than being studied considering their interactions, which limits understanding of their combined effectiveness and scalability in real world applications.
2. Empirical evidence for key performance outcomes, especially flexibility, scalability, and sustainability is limited.
3. Scalability challenges affect all customization strategies, with little empirical evidence showing that any approach can be effectively scaled for broader deployment.
4. Implementation frameworks need for robust empirical validation, and emerging technologies remain under-researched in actual contexts.
5. Sustainability research is often limited to environmental aspects, with economic and social dimensions underexplored.

Addressing these gaps will require:

1. Future research systematically exploring and empirically validating enabler synergies, using, for example, expert knowledge as a primary data source, given their efficiency and suitability for rapid theory-building.
2. Increased methodological rigor, including integrating qualitative insights (e.g. from experts) with quantitative findings (e.g. drawn from existing studies or from company reports) where feasible.
3. Broader research attention to flexibility, scalability, sustainability (across all dimensions), and sectoral diversity is needed, as these areas remain underexplored in the current literature.
4. Empirical validation of implementation frameworks, particularly in less-studied project types and contexts.

In summary, from this review it emerges that the most impactful and innovative configuration strategies in the AEC sector arise from the intentional, synergistic integration of enablers—as captured by the analytical framework. Closing the identified gaps will require coordinated efforts to develop, implement, and empirically validate context-sensitive, scalable, and sustainable configuration approaches for the digitalized AEC sector.

5. Discussion: Advancing a contingent-configurational perspective for configuration in AEC

This review shows that scalable and adaptive customization in the AEC sector depends on systematic, integrated use of core enablers and other enablers across all project stages, rather than fragmented tools adoption [8, 12]. The most successful cases integrate digital platforms, modularization, and configuration systems, effectively bridging mass production efficiency and user-specific outcomes [16, 25]. In contrast, fragmented or isolated efforts tend to deliver only limited and often costly gains [4, 18].

The analysis adopts a contingent-configurational perspective: the effectiveness of specific combinations of enablers is contingent upon the customization strategy employed. Distinct strategies (e.g., pure customization, customized fabrication, customized assembly, variety without customization) require different configurations of enablers to achieve desired performance outcomes. For example, pure customization and customized fabrication support high flexibility and user involvement but often struggle with scalability—gaps that can be addressed through the targeted integration of core enablers and other enablers. Customized assembly balances efficiency and personalization through enabler synergy, while strategies focusing on variety without customization primarily expand standardized offerings through digital tools (other enablers), limiting deep client-driven design. These differences underscore that the specific alignment or "fit" between strategy and enabler configuration must be tailored to the context and maturity of each case—consistent with the contingent-configurational perspective advanced in the literature [27].

However, strategy is not the only important contingency factor in the AEC context. In addition to the adopted customization strategy, other contextual factors, such as sector maturity, project complexity, delivery models, and stakeholder engagement, critically shape which configurations are most effective [8, 11]. Our findings show that the performance impact of configuration systems is not universal, but depends on their suitability with chosen strategy, project context and enabler synergy. Robust, context-sensitive integration can deliver substantial cost and time benefits, while mismatched or isolated enablers yield only marginal gains [18, 19]. This highlights that optimal outcomes cannot be achieved through a one-size-fits-all approach but require that configurations of strategies and enablers be tailored to specific technological, organizational, and environmental conditions.

These contextual factors align closely with the TOE framework, originally proposed by Tornatzky and Fleischer (1990) and widely adopted for studying technology adoption and integration in organizational settings [14, 28]. The TOE framework serves as a guiding lens for interpreting the findings. Specifically:

- Technological factors include the availability and maturity of digital platforms, BIM integration, IT infrastructure, and modular construction technologies. These determine the feasibility and performance of advanced configuration systems, influencing how easily customization strategies can be implemented and scaled.
- Organizational factors encompass delivery models, process maturity, stakeholder engagement, project governance, and organizational readiness for change. These shape the selection, integration, and synergy of enablers, as well as the ability to move from isolated to systematized approaches.
- Environmental factors comprise market dynamics, regulatory requirements, sectoral maturity, and client expectations. These set the external conditions for customization, impacting adoption rates and the prioritization of scalable versus flexible solutions.

Interpreting the results through the TOE lens clarifies how each dimension—technology, organization, and environment—uniquely contributes to the success or limitation of configuration system integration. This systematic consideration of context further substantiates the contingent-configurational perspective advanced in this paper.

Thus, the contingent-configurational perspective advanced in this paper explains and predicts how different combinations of customization strategies and enablers, tailored to organizational, technological, and environmental contexts, shape outcomes in the AEC sector. This theoretical perspective accounts for the dynamic interplay between configuration systems and contextual variables, providing practical guidance for selecting, integrating, and aligning enablers to achieve scalable, client-centric solutions.

For researchers, these findings highlight the need to systematically investigate both the mechanisms of enabler integration (configurational) and the contextual contingencies (contingent) that underpin successful outcomes, by moving beyond typologies to empirically grounded models that can inform theory and practice. For practitioners, the results offer actionable guidance: successful implementation requires not just investment in digital or modular tools, but also a strategic approach to synergy and adaptation to project-specific demands and organizational readiness.

Advancing contingent-configurational perspective will require continued empirical study to capture real-world complexities, overcome implementation barriers, and develop robust, context-sensitive approaches to scalable customization in the digitalized AEC sector.

6. Conclusion & Implications

This review advances understanding of scalable and adaptive customization in the AEC sector by systematically analyzing how configuration systems, enabler integration, and performance outcomes intersect across 137 publications. By positioning configuration systems at the core, the study clarifies where these approaches add the most value [10], identifies sector-specific patterns and trade-offs [8,11], and highlights persistent gaps—most notably the fragmented use of enablers, limited empirical validation, and the prevalence of isolated rather than synergistic adoption of digital and modular tools [8,12].

The findings make clear that current AEC customization efforts often fall short when configuration systems are implemented in isolation, without deliberate integration or alignment with project context. Such approaches typically lead to suboptimal outcomes, limited scalability, and missed opportunities for genuine client-centric solutions. Simply investing in digital tools or modularization, without ensuring synergy and contextual suitability, is unlikely to deliver the promised benefits of mass customization.

To address these limitations, this paper advances a contingent-configurational perspective for configuration system integration in the AEC sector. This theoretical contribution emphasizes that optimal outcomes are not achieved by universally applying the same strategies and enablers across all contexts. Instead, success depends on carefully selecting, integrating, and adapting customization strategies and enabling mechanisms to fit the specific technological, organizational, and environmental conditions of each project or organization. In other words, scalable and effective customization requires context-sensitive configuration, rather than a one-size-fits-all approach.

The integrative framework developed here connects customization strategies, enablers, and performance dimensions, providing both theoretical clarity and practical guidance for researchers and industry professionals at the intersection of digitalization, modularization, and user-driven design [10, 13]. For scholars, this work establishes a stronger theoretical basis for context-sensitive and empirically grounded research. For practitioners, it highlights actionable opportunities to leverage configuration logic and enabler synergies for scalable, client-centric solutions that are attuned to project and organizational realities.

Looking ahead, the AEC sector has significant potential to close the gap with leading industries like manufacturing—provided it adopts more context-sensitive, synergistic, and empirically validated approaches to configuration. Achieving this will depend on stronger alignment of technological, organizational, and environmental factors, as emphasized by the TOE framework and encapsulated in the contingent-configurational perspective advanced in this study.

Declaration on Generative AI

During the preparation of this work, the authors created all figures using Microsoft Excel. ChatGPT was consulted only for suggestions on color schemes, layout improvements, and label clarity. All data visualization, chart design, and content decisions were made entirely by human authors. No generative AI was used to create visual content.

References

[1] P. A. E. Piroozfar, F. T. Piller (Eds.), Mass Customization and Personalization in Architecture and Construction, Routledge, Abingdon, UK, 2013. ISBN: 978-0-415-62283-7. doi:10.4324/9780203437735.

[2] Harvard Business Review Analytic Services, The Future of Building Design: Fostering Partnerships Between AEC Firms and Building Product Manufacturers, Harvard Business School Publishing, Boston, MA, 2020. URL: https://hbr.org/hbr-analytic-services.
[3] J. L. Blanco, D. Rockhill, A. Sanghvi, A. Torres, From start-up to scale-up: Accelerating growth in construction technology, McKinsey & Company, May 2023. URL: https://www.mckinsey.com/industries/private-capital/our-insights/from-start-up-to-scale-up-accelerating-growth-in-construction-technology.
[4] T. Bock, T. Linner, Robotic Industrialization: Automation and Robotic Technologies for Customized Component, Module, and Building Prefabrication, Cambridge University Press, Cambridge, 2015.
[5] B. J. Pine II, B. Victor, A. C. Boynton, Making mass customization work, Harvard Business Review, 71 (1993) 108–119.
[6] J. Lampel, H. Mintzberg, Customizing customization, Sloan Management Review, 38 (1996) 21–30.
[7] E. R. P. Farr, P. A. E. Piroozfar, D. Robinson, BIM as a generic configurator for facilitation of customization in the AEC industry, Automation in Construction, 45 (2014) 119–125. doi:10.1016/j.autcon.2014.05.012.
[8] S. Yadav, A. Prakash, M. Arora, A. Mittal, Digital transformation: exploring cornerstones for construction industry, Kybernetes, 53 (2024) 5378–5401. doi:10.1108/K-05-2023-0895.
[9] F. Salvador, C. Forza, M. Rungtusanatham, How to mass customize: Product architectures, sourcing configurations, Business Horizons, 45 (2002) 61–69.
[10] C. Forza, F. Salvador, Product Information Management for Mass Customization, Palgrave Macmillan, 2006.
[11] B. E. Mohamed, C. Carbone, Mass customization of housing: A framework for harmonizing individual needs with factory produced housing, Buildings, 12 (2022) 955. doi:10.3390/buildings12070955.
[12] X. Chen, A. Chang-Richards, F. Y. Y. Ling, T. W. Yiu, A. Pelosi, N. Yang, Digital technologies in the AEC sector: a comparative study of digital competence among industry practitioners, International Journal of Construction Management, 2024. doi:10.1080/15623599.2024.2304453.
[13] N. Suzić, C. Forza, A. Trentin, Z. Anišić, Implementation guidelines for mass customization: current characteristics and suggestions for improvement, Production Planning and Control, 29 (2018) 856–871. doi:10.1080/09537287.2018.1485983.
[14] L. G. Tornatzky, M. Fleischer, The Processes of Technological Innovation, Lexington Books, Lexington, MA, 1990.
[15] T. Blecker, N. Abdelkafi, G. Kreutler, G. Friedrich, Product configuration systems: state of the art, conceptualization and extensions, in: A. B. Hamadou, F. Gargouri, M. Jmaiel (Eds.), Génie logiciel & Intelligence artificielle. Eighth Maghrebian Conference on Software Engineering and Artificial Intelligence (MCSEAI 2004), Sousse, Tunisia, Centre de Publication Universitaire, Tunis, 2004, pp. 25–36.
[16] J. Cao, D. F. Bucher, D. M. Hall, J. Lessing, Cross-phase product configurator for modular buildings using kit-of-parts, Automation in Construction, 123 (2021) 103507. doi:10.1016/j.autcon.2020.103437.
[17] P. Wu, J. Wang, X. Wang, A critical review of the use of 3-D printing in the construction industry, Automation in Construction, 68 (2016) 21–31. doi:10.1016/j.autcon.2016.04.005.
[18] N. M. Larsen, J. A. Guajardo, M. Hviid, Mass customization in the house building industry: literature review and research directions, Journal of Manufacturing Technology Management, 30 (2019) 176–199. doi:10.3389/fbuil.2019.00115.
[19] M. Salzano, A. Ferrante, I. Ferretti, F. Zerbini, Systematic literature review of open infrastructure BIM: toward a reference framework for data and information sharing, Journal of Information Technology in Construction, 28 (2023) 66–84. doi:10.3390/buildings13071593.

[20] H. Alawag, A. M. Sharif, M. M. Butt, Artificial intelligence in the smart construction industry: a review of the existing state, challenges, and future threats, Technological Forecasting and Social Change, 197 (2024) 122921. doi:10.1063/5.0194653.
[21] D. Tranfield, D. Denyer, P. Smart, Towards a methodology for developing evidence-informed management knowledge by means of systematic review, British Journal of Management, 14 (2003) 207–222. doi:10.1111/1467-8551.00375.
[22] D. Moher, A. Liberati, J. Tetzlaff, D. G. Altman, P. Group, Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement, Annals of Internal Medicine, 151 (2009) 264–269. doi:10.1136/bmj.b2535.
[23] P. Jensen, T. Olofsson, H. Johnsson, Configuration through the parameterization of building components, Automation in Construction, 23 (2012) 1–8. doi:10.1016/j.autcon.2011.11.016.
[24] S. Wang, Q. Chen, Development of a cloud-based Building Information Modeling design configurator to auto-link material catalogs with code-compliant designs of residential buildings, Buildings, 14 (2024) 2084. doi:10.3390/buildings14072084.
[25] J. X. Zhou, G. Q. Shen, S. H. Yoon, X. Jin, Customization of on-site assembly services by integrating the internet of things and BIM technologies in modular integrated construction, Automation in Construction, 126 (2021) 103677. doi:10.1016/j.autcon.2021.103663.
[26] I. Ibrahim, F. Eltarabishi, H. Abdalla, M. Abdallah, 3D printing in sustainable buildings: Systematic review and applications in the United Arab Emirates, Buildings, 12 (2022) 1703. doi:10.3390/buildings12101703.
[27] E. Sandrin, A. Trentin, C. Forza, Leveraging high-involvement practices to develop mass customization capability: A contingent-configurational perspective, International Journal of Production Economics, 196 (2018) 335–345. doi:10.1016/j.ijpe.2017.12.005.
[28] J. Baker, The technology–organization–environment framework, in: Y. K. Dwivedi, M. R. Wade, S. L. Schneberger (Eds.), Information Systems Theory: Explaining and Predicting Our Digital Society, Vol. 1, Integrated Series in Information Systems 28, Springer, New York, NY, 2011, pp. 231–245. doi:10.1007/978-1-4419-6108-2_12.
[29] M. Chatzimichailidou, T. Whitcher, N. Suzic, Complementarity and compatibility of systems integration and building information management, IEEE Systems Journal (2024). doi:10.1109/JSYST.2024.3387064.
[30] M. M. Chatzimichailidou, J. Whyte, Dealing with complexity in modular construction, in: Proceedings of the 8th International Conference on Mass Customization and Personalization – Community of Europe (MCP-CE 2018), Novi Sad, Serbia, September 19–21, 2018, pp. 47–50.

Towards LLM-based Configuration and Generation of Books

Jovan Mihajlovic[1,*], Alexander Felfernig[1]

[1]Graz University of Technology, Inffeldgasse 16b, Graz, 8010, Austria

Abstract

Large Language Models (LLMs) can support a wide range of content generation tasks. Interaction with LLMs can occur either through user-friendly web interfaces or via provided APIs. Our work focuses on content generation for a specific use case: creating lecture books from recorded lectures. To support this goal, a web application with a simple configuration interface has been developed. Users can include transcripts of recordings, configure options, and generate books through the application. It allows for flexibility by offering different prompts and the ability to select among various LLMs. Initial results demonstrate that LLMs can generate books from the recorded lectures, however, evaluation results show varying output quality depending on the selected configuration.

Keywords

Large Language Models, Configuration, Generation of Books

1. Introduction

Configuration can be regarded as a specialized form of design activity in which the final product is assembled from a predefined set of component types, all of which must comply with a corresponding set of domain-specific constraints [1, 2, 3, 4]. In this paper, we demonstrate how Large Language Models (LLMs) can be leveraged for the automated generation of university lecture books. This generation is based on a collection of video recordings (of lecture units) [5]. Our application enables the configuration of key properties relevant to book generation and automatically produces a draft book proposal from the transcripts of these lectures. The major motivation for our work is to exploit simple ways to provide students with additional learning contents that help to improve the overall learning experience and to accelerate learning. The focus of our work is to apply Large Language Models (LLMs) [6] to generate book contents based on LLM prompts which are themselves generated on the basis of configured book and generation process properties.

There are multiple ways to interact with LLMs. First, queries can be defined in user interfaces of LLM providers. Second, models can be accessed via APIs offered by these providers, allowing the development of customized applications tailored to specific tasks such as generating book content from lecture transcripts. Our book generator application serves as an intermediary between the LLM and the user – it facilitates the process of transforming lecture transcripts into book content. It simplifies the interaction with LLMs by abstracting the underlying complexity of prompt engineering. Generating structured book content involves first establishing context and defining the generation goal, then supplying the transcript text along with relevant options and constraints. While crafting a prompt manually might be effective for single-use scenarios, repeating this process for multiple transcripts is labor-intensive. Our system addresses this by using pre-defined prompt templates.

Due to their generative nature, large language models can be applied in various generation-related tasks. Related examples in the configuration context are the generation of explanations [7] (e.g., sustainability-aware explanations nudging configurator users towards more sustainable consumption patterns), the generation of configuration knowledge bases [8] (which helps to reduce efforts in the

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
jovan.mihajlovic@student.tugraz.at (J. Mihajlovic); alexander.felfernig@ist.tugraz.at (A. Felfernig)
https://ase.sai.tugraz.at/ (A. Felfernig)
0000-0003-0108-3146 (A. Felfernig)

context of configuration knowledge base development and maintenance), and interactive configuration [9] where LLMs can be applied to support interactive chat-based interfaces that support product and service configuration by allowing users to describe their requirements/preferences in natural language (and to generate a set of solver-understandable preferences thereof). In contrast to existing work in the context of combining LLMs with configuration technologies, the work presented in this paper focuses on the generation of artifacts (in our case, books) on the basis of pre-configured parameters representing intended book properties and also technical properties relevant for the book generation phase.

The major contribution of this paper is to present an initial idea of a book generation interface based on configured parameter settings. Furthermore, we summarize initial insights from the analysis of a first prototype implementation which is currently not available for productive use.

The remainder of this paper is organized as follows. Section 2 explains how the implemented web application automates the book generation workflow. Section 3 discusses the outcomes obtained using real-world lecture recordings. Finally, Sections 4 and 5 outline directions for future development and summarize our findings.

2. LLM-based Book Generator

The book generator application is structured into a backend, implemented using the NestJS framework[1] and a frontend which uses the React[2] library. It generates LLM prompts from input data (e.g., transcripts from lecture videos) and selected (configured) options. The generated prompts are forwarded to the LLM via API. The input data consists of the following elements:

- one or more lecture video transcript files (textual)
- *(optional)* a LaTeX template file as basis for guiding the LLM-based book generation (in LaTeX format).
- *(optional)* parameters (which prompt to use, book title, chapter size and other parameters)

The user gets a result either as a single LaTeX file, or a ZIP file containing multiple output files. There are three "pipelines" available for the user to choose, these are described in the following. Each pipeline presents one or more prompts used to generate a book from the input data (see Table 1).

Table 1
LLM pipelines available in the current book generator prototype where *name* denotes the name of the pipeline, *input* denotes the number of transcript files processes at a time (n represents the number of input transcripts), *parameters* indicates whether configuration parameters can be specified within this selected pipeline, *concepts* indicates whether LLM-identified key concepts should be used for organizing individual sections, and *#chap* indicates whether one or more chapters are generated based on the selected pipeline and the provided further input.

name	input	parameters	concepts	#chap
T2SC	n	no	no	n
C2C	n	no	yes	#concepts
C2CO	n	yes	yes	#concepts

2.1. Transcript to Single Chapter (T2SC)

This pipeline focuses on generating a single LaTeX book chapter for each provided transcript file, which is derived from the video of an individual lecture unit. The LLM is instructed to create one cohesive chapter. Structuring elements to be used (e.g., sections and subsections) are enumerated explicitly. To ensure consistency, each chapter is generated based on the same prompt template.

[1] https://nestjs.com, accessed 16-June-2025
[2] https://react.dev, accessed 16-June-2025

The primary focus of each chapter is to explain the core concepts of the corresponding lecture unit in detail. *Examples* can be included if they help in clarifying the discussed concepts. The style in which examples are presented is not strictly specified. At the end of each chapter, self-evaluation questions are included to encourage further learning. In the current version of the book generator, the LLM is explicitly instructed to generate all book contents in English. To assure consistency in formatting, each generated chapter is generated on the basis of a basic LaTeX template.

The following is the full prompt (transcript contents are formatted/represented as "[...]"):

> An audio transcript of a lecture is provided within the quotes at the end of this message. Generate a LaTeX chapter for this lecture. The chapter should start with \chapter (you can use sections, subsections and everything else provided in standard LaTeX). Focus on explaining the main concepts of the lecture. Use examples if they can help in explaining. Add questions for self preparation at the end of the chapter. Include some concepts which are related, but not mentioned in the transcript. The output language should be English. The transcript is: [...]

2.2. Concepts to Chapters (C2C)

Compared to T2SC, the C2C pipeline includes an additional step for generating chapters. Rather than creating individual chapters directly from each transcript derived from a lecture unit video, the LLM is first asked to identify the key concepts within the transcripts. Once the key concepts are identified, the LLM is instructed to enhance the presentation of the content using lists, text styling, structural elements, and similar formatting techniques. Each concept should be clearly explained to the reader, with related concepts further elaborated upon in corresponding subsections. As in T2SC, the LLM is also directed to include relevant examples to aid in the explanation.

The following is the prompt, where CONCEPT_NAME corresponds to the target concept the prompt is applied to:

> Within the context of a Computer Science lecture called "Software development processes", a concept called "CONCEPT_NAME" is present. Write a LaTeX chapter about that concept. The output language should be English.
>
> [list of detailed instructions]

2.3. C2C with Options (C2CO)

The main difference between C2CO and the previous pipelines is the provision of options (parameters) that help to further configure the pipeline – see a corresponding example user interface in Figure 1. These options can be used to further tailor the generated contents.

The idea of the C2CO pipeline is the same as C2C, however, the used prompts are different (as a direct consequence of the provided options). The probably most influential option is to allow the usage of a customized LaTeX template (see Section 3). Selected options are represented as a list of rules part of the prompt. For example, the first rule to follow when generating a chapter is the following:

> "the chapter starts with introductory paragraph which explains the concept concisely"

Irrelevant elements in provided custom templates (e.g., texts and bibliographic entries from other articles written on the basis of this template) are first removed by the LLM. The purpose of this step is to ease further use of the template by the LLM (e.g., texts from other papers have the risk to confuse the LLM and let the LLM integrate related contents into the generated book). The remaining steps are quite similar to the C2C pipeline – the major difference are the inserted rules which are defined by the user via book generator user interface.

The used prompt is the following:

Your task is to generate a LaTeX chapter for the following concept (within the context of a Computer Science lecture called "Software development processes"): "CONCEPT_NAME"
You should follow these rules: [a list of rules like the one stated previously]

The returned content should be a LaTeX \chapter which can be included in a template.

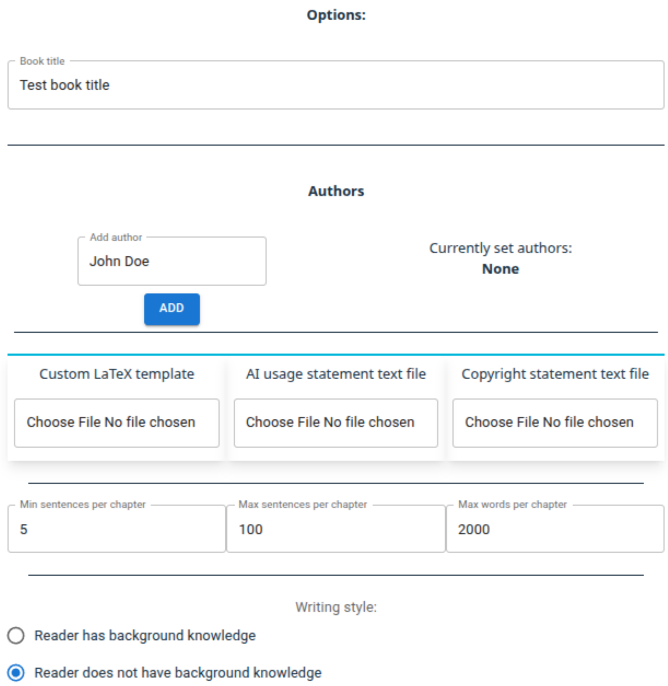


Figure 1: Options provided by the book generator UI (C2CO). The book title and a list of authors can be defined. Different files like a custom LaTeX template, AI usage statement and copyright statement can be uploaded. Lastly, constraints for the length of each chapter can be set, as well as a choice between two writing styles.

3. Preliminary Evaluation

The following subsections discuss first results of applying our LLM-based approach to video transcripts generated from a lecture on the topic of *software development processes*. In this context, each individual subsection discusses the observed results of applying an individual pipeline. In its current version, our book generator application lacks automated quality assurance mechanisms. The generation of the transcripts was performed on the basis of the OpenAI Whisper model.[3]

Our initial evaluation of the LLM-generated outputs (books) has been performed manually and the corresponding results (feedback of two persons) are presented in an aggregated fashion in the following paragraphs. For evaluation purposes, the generation focused on an individual lecture unit related to different techniques in the context of the topic of *software requirements prioritization* including subtopics such as *release planning* and *minimum viable products*.

An overview of the different applied LLMs is provided in Table 2.

Table 2
Large Language Models (LLMs) used by the book generator application for content generation purposes.

Name	link
gemini-2.0-flash	https://cloud.google.com/vertex-ai/generative-ai
lama-3.3-70b-versatile	https://www.llama.com

[3]https://huggingface.co/openai/whisper-large-v3

In the following, we summarize the initial evaluation results of the LLM-generated outputs on the basis of following four evaluation dimensions:

- **Understandability** of the content
- **Completeness**: degree to which transcript contents are covered
- **Example quality**: quality of the included (generated) examples
- **Additional content**: presence of additional information related to transcript contents, but not contained in the transcript

3.1. T2SC Pipeline

gemini-2.0-flash The content is clear and easy to understand, and it is provided in the requested language. The coverage of the transcript content is comprehensive, including a thorough introduction and an in-depth classification of *requirement prioritization approaches* (this is the lecture topic). The examples provided are generally of high quality, with a few that could benefit from improvements in readability and formatting. Compared to Llama, the quality and quantity of examples are significantly better. The subsection on additional content lists seven related concepts, each accompanied by a concise description. However, there is a need for better structure, as the concepts are currently presented in a single paragraph without clear separation.

llama-3.3-70b-versatile The content is generated in German, despite the instruction specifying English. Additionally, some sections are a bit unclear—specifically, the subsections on "Basic Release Planning" and "Integrated Release Planning" both contain identical sentences. Only the latter mentions additional factors, which suggests there is some intended difference between the two but fails to clarify it adequately. The overall completeness is in the lower-medium range. Key concepts like "Minimum Viable Product," which are included in the other model, are missing. The chapter is concise, fitting onto just two pages, including the self-preparation questions, but lacks depth in certain areas. The examples provided are of low quality, with only one example given in two short sentences, which fails to adequately illustrate the concept. This model performed poorly in terms of additional content, as it did not provide any supplementary material in this area.

3.2. C2C Pipeline

gemini-2.0-flash The C2C pipeline generates multiple chapters along with corresponding sections and subsections, for which a table of contents can be generated. The content is generally clear and well-presented. There are some formatting issues with the questions that could hinder readability, but the questions themselves are understandable.. The transcript is largely covered, but some concepts, such as basic/integrated release planning (despite the presence of a "Release Planning" chapter) and utility-based prioritization, are notably missing. The examples, primarily presented in table form, are easy to understand. A snippet of a subsection with an example is shown in Figure 2. However, many of other example tables overflow the page, which can make them difficult to read. The additional content is excellent, with well-organized per-chapter sections and separate chapters that enrich the material.

llama-3.3-70b-versatile The content is clear and easy to understand, with no notable issues. The transcript is well-covered, though the "Release Planning" chapter lacks references to basic/integrated release planning, similar to the Gemini model. On the positive side, it does include a chapter on utility-based prioritization, which the other model does not. The quality of examples is mixed: the textual example in Chapter 2 is strong, while the one in Chapter 3 is quite brief and doesn't fully explore the concept. The remaining examples are decent, though it's difficult to assess them fully due to tables spilling out of the page, making the examples incomplete or hard to view. The additional content has a satisfactory level of depth and organization.

1.3.2 Ranking

Ranking involves assigning a numerical value or rank to each requirement based on its importance. The requirements are then sorted based on their rank, with the highest-ranked requirements receiving the highest priority. Simple ranking schemes might use a scale of 1 to 5 (with 5 being the highest priority), while more complex schemes may incorporate multiple factors and weighted scores.

Example:

Requirement	Rank (1-5)
User Authentication	5
Product Search	5
Shopping Cart	4
Order Tracking	3
Customer Support Chat	2

Figure 2: A subsection containing text which explains how requirements can be ranked.

3.3. C2CO Pipeline

gemini-2.0-flash For the chosen template used in evaluation, the title page clearly presents the title of the book in bold text, with authors directly below. The header on the page following the title page suffers from broken formatting, though it remains legible with some effort. Otherwise, the content is generally easy to understand, though there are a few areas where improved formatting could enhance clarity. The completeness is similar to the C2C case, as the approach is quite comparable, with only minor differences in the prompts used for generating the concept chapter. There are numerous examples presented in table format, all of good quality. Some tables extend beyond the page boundaries so significant parts of examples are lost. Each chapter references related concepts for additional content, but the formatting is inconsistent. In some instances, these concepts are listed clearly, while in others, they are presented continuously in a single paragraph. The prompt could benefit from a more consistent structure for how additional content should be organized.

llama-3.3-70b-versatile The content is generally clear and easy to follow, though there are some issues with tables extending beyond the page, similar to the issues found in the Gemini case. The level of completeness is comparable to the C2C case for the same reasons mentioned in the previous section. The quality of examples is similar to that of Gemini: some tables are well-structured and present clear values, while others overflow the page. One example is supposed to demonstrate how to calculate opportunity costs using the Kano model (as indicated by its title), but it only provides the final values, without showing the calculations involved. Additional content is provided, but it appears somewhat brief and/or poorly formatted, which could benefit from further refinement.

Table 3
Evaluation of the generated book chapters on the basis of the evaluation dimensions *understandability*, *completeness*, *example quality*, and *additional content* with a rating scale (1 (poor) .. 5 (excellent). Evaluated LLMs: gemini-2.0-flash (GF) and lama-3.3-70b-versatile (LV).

LLM	understandability			completeness			example quality			additional content		
	T2SC	C2C	C2CO	T2SC	C2C	C2CO	T2SC	C2C	C2CO	T2SC	C2C	C2CO
GF	4.5	4.5	4	4.5	3.5	4	4	4	4	4.5	4.5	4
LV	3	4.5	4	3	3.5	4	2	3	3	1	4.5	3

Initial Summary The Gemini model shows most consistent performance across different dimensions and pipelines. The approach used by C2C/C2CO pipelines seems to be the right direction, as it allows more control over structure and length of generated content. Based on our evaluation provided in Table 3, the two analyzed LLMs show major differences in terms of the evaluation dimensions *example quality* and *additional content*.

4. Future Work

The current version of this application provides a solid foundation with consistent results, but there is considerable room for improvement. The next logical step is to evolve the content generation process into a more structured, configurable approach that allows for step-by-step refinement.

By adding more customizable options, we can address a broader range of users. For instance, default settings would meet the needs of those seeking quick, approximate results, while users who require greater control can directly adjust specific parameters. Introducing more interactivity between the user and the model would provide even finer control over the generated content.

One approach could involve having the model analyze the input data without producing results immediately. Instead, it could prompt the user for additional options or guidance before proceeding, in contrast to the current method where the user interacts only once, at the data input stage.

Another crucial feature currently missing is quality assurance. At present, results are generated by the model and returned directly to the user without any validation. To address this, we will introduce a quality-check mechanism either at the end of the process, between various steps, or ideally, both. These quality checks will provide valuable feedback to users, allowing them to review suggested improvements or request more detailed revisions.

An important open issue in this context is also to include mechanisms that help to include information about information sources used by the LLM – primarily for the purpose of preventing violations of copyrights and similar issues. Also this aspect has not been taken into account in the current version of the book generator which is also the reason why the system currently is not applied in productive use.

Finally, for future versions, we also plan to include the possibility of defining seed knowledge, i.e., already proposing a basic structure of the book which is then enriched by the LLM.

By incorporating these changes, we hope to not only enhance the content generation process but also to foster greater user involvement and satisfaction.

5. Conclusion

The presented application demonstrates the potentials of LLMs in supporting the automated generation of books where different LLMs and prompt configurations produce varying outcomes. This sets the stage for further development to explore the extent to which these results can be refined. The current approach leaves room for enhancements, particularly with new features added, opportunities to improve both, functionality and quality, are given.

Declaration on Generative AI

The authors used ChatGPT-4o[4] for grammar checking, spellchecking, and improving the formulation of the text. All AI-generated suggestions were carefully reviewed and edited by the authors, who take full responsibility for the content of this publication.

References

[1] D. Sabin, R. Weigel, Product configuration frameworks-a survey, IEEE Intelligent Systems 13 (1998) 42–49. doi:10.1109/5254.708432.
[2] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1 ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.
[3] A. Felfernig, A. Falkner, D. Benavides, Feature Models – AI Driven Design, Analysis, and Applications, Springer, Cham, 2024.

[4]https://chatgpt.com

[4] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V.-M. Le, K. Pilsl, M. Enzelsberger, T. N. T. Tran, An overview of machine learning techniques in constraint solving, J Intell Inf Syst 58 (2022) 91–118. doi:10.1007/s10844-021-00666-5.
[5] S. Lubos, A. Felfernig, D. Garber, V.-M. Le, M. Henrich, R. Willfort, J. Fuchs, Towards Group Decision Support with LLM-based Meeting Analysis, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '25, ACM, New York, NY, USA, 2025, pp. 331–335. URL: https://doi.org/10.1145/3708319.3733646. doi:10.1145/3708319.3733646.
[6] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, S. Zhong, B. Yin, X. Hu, Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, ACM Trans. Knowl. Discov. Data 18 (2024). doi:10.1145/3649506.
[7] S. Lubos, A. Felfernig, L. Hotz, T. N. T. Tran, S. P. Erdeniz, V. Le, D. Garber, M. E. Mansi, Responsible Configuration Using LLM-based Sustainability-Aware Explanations, in: É. Vareilles, C. Grosso, J. M. Horcas, A. Felfernig (Eds.), 26th International Workshop on Configuration (ConfWS 2024), CEUR-WS.org, 2024, pp. 68–73.
[8] L. Hotz, C. Bähnisch, S. Lubos, A. Felfernig, A. Haag, J. Twiefel, Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems, in: ConfWS'24, volume 3812, CEUR, 2024, pp. 91–100.
[9] P. Kogler, W. Chen, A. Falkner, A. Haselboeck, S. Wallner, Configuration Copilot: Towards Integrating Large Language Models and Constraints, in: ConfWS'24, volume 3812, CEUR, 2024, pp. 101–110.

Towards LLM-Enhanced Product Line Scoping

Alexander Felfernig[†], Damian Garber[†], Viet-Man Le[†], Sebastian Lubos[†] and Thi Ngoc Trang Tran[†]

Institute of Software Engineering and AI, Graz University of Technology, Graz, Austria

Abstract

The idea of product line scoping is to identify the set of features and configurations that a product line should include, i.e., offer for configuration purposes. In this context, a major scoping task is to find a balance between commercial relevance and technical feasibility. Traditional product line scoping approaches rely on formal feature models and require a manual analysis which can be quite time-consuming. In this paper, we sketch how Large Language Models (LLMs) can be applied to support product line scoping tasks with a natural language interaction based scoping process. Using a working example from the smarthome domain, we sketch how LLMs can be applied to evaluate different feature model alternatives. We discuss open research challenges regarding the integration of LLMs with product line scoping.

Keywords

Product Line Scoping, Feature Models, Configuration

1. Introduction

Configurable products and services such as smarthomes, cars, and software systems have a high variability in terms of which components can be combined with each other [1, 2, 3]. To be able to handle variability in an efficient fashion, product line (PL) approaches have been widely adopted [4]. The idea of product lines is to allow a systematic reuse of shared assets which enables the reduction of development costs, reduced time to market, and higher product quality. Product line scoping is at the heart of PL engineering [5, 6, 7, 8, 9] – it is the process of defining which features and constraints should be included in a product line, i.e., which features and corresponding constraints should be part of a feature model. High-quality scoping decisions are crucial since they directly have an influence on the feasibility and commercial success of a product line.

Determining an optimal scope for a PL is a challenging task. This includes the evaluation of market trends, the balancing of potentially contradicting stakeholder requirements, and also ensuring the technical feasibility of the offered feature model configurations. A typical PL scoping process is based on workshops with experts. Related scoping decisions can be suboptimal due to a limited market and domain knowledge and – as a consequence – product lines have the risk of being under- or over-restricted. The underlying group decision task makes product line scoping a task directly related to requirements prioritization [10, 11] and group recommender systems [12, 13, 14].

Developments in the context of large language models (LLMs) [15] have created the potential to improve a variety of PL related tasks [16, 17, 18]. For example, in the context of software development, LLMs can be applied for re-engineering purposes allowing an LLM-based creation of PLs on the basis of artifacts such as UML diagrams, state charts, and Java programs [19]. Furthermore, LLMs have shown to be applicable in the context of new feature identification from different textual sources such as appstore evaluations [20]. Finally, LLMs have also shown to be applicable for model generation tasks, more precisely, the generation of feature models out of textual domain descriptions [21]. In the context of

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

[†]These authors contributed equally.

alexander.felfernig@tugraz.at (A. Felfernig); damian.garber@tugraz.at (D. Garber); v.m.le@tugraz.at (V. Le); sebastian.lubos@tugraz.at (S. Lubos); trang.tran@tugraz.at (T. N. T. Tran)

0000-0003-0108-3146 (A. Felfernig); 0009-0005-0993-0911 (D. Garber); 0000-0001-5778-975X (V. Le); 0000-0002-5024-3786 (S. Lubos); 0000-0002-3550-8352 (T. N. T. Tran)

PL scoping, LLMs have the potential to support engineers in their tasks of analyzing trade-offs and identifying commercially promising variability concepts.

In this paper, we propose the idea of applying LLMs to pro-actively support different tasks in product line scoping. This includes the aspects of estimating model optimality (in terms of market potential, alignment with customer preferences, and cost efficiency) and technical feasibility of the offered features by taking into account the existing product development resources.

We want to emphasize that these tasks are also relevant beyond software product lines, for example, in the context of designing and configuring complex products such as cars and smarthome systems. As a basis for our discussions, we introduce a simplified working example from the domain of smarthome systems. With the introduced feature model, we sketch scenarios where LLM-supported product line scoping can provide help in estimating commercial relevance and technical feasibility.

The basic idea sketched in this paper is to exploit LLMs for pro-actively supporting group decision processes in product line scoping, i.e., we envision a scenario based on human-AI collaboration where LLMs provide additional insights (not covered by experts), indicate new alternatives, and explain the consequences of specific decisions [22].

The major contributions of this paper are: first, we introduce the idea of exploiting LLMs for supporting product line scoping processes. Second, we sketch our ideas on the basis of a working example from the domain of smarthomes. Finally, we discuss related open research issues.

The remainder of this paper is organized as follows. In Section 2, we present a feature model from the smarthome domain. In Section 3, we analyze different scenarios in which LLMs can be employed to support product line scoping. Section 4 discusses open research issues. With Section 5, we conclude the paper.

2. Working Example: SmartHome Feature Model

In the following, we introduce a simplified feature model from the smarthome domain which will be used as a working example throughout the paper. Smarthome systems include a diverse set of features/functionalities including security, lighting, and climate control. Figure 1 depicts a feature model of a simplified smarthome product line. The root feature `SmartHomeSystem` includes three basic subfeatures which are `Security`, `Lighting`, and `ClimateControl`. Each of those features has further subfeatures (either optional or mandatory ones).

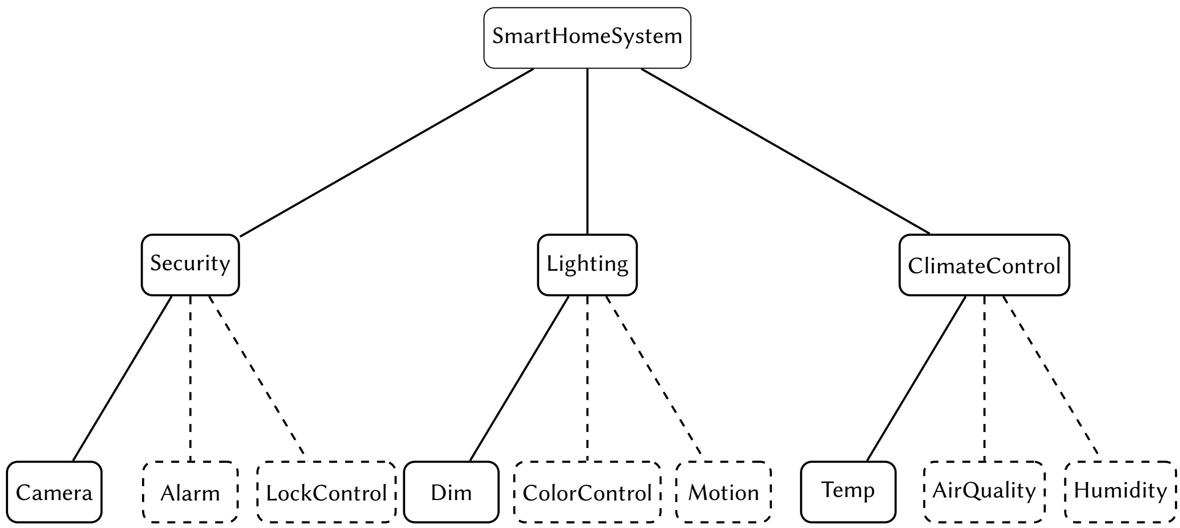


Figure 1: Feature model of a smarthome product line (dashed lines represent optional features, the other mandatory ones.)

In the feature model of Figure 1, the feature `SmartHomeSystem` is regarded as mandatory (due to the fact that we are not interested in empty configurations). The first-level child features `Security`, `Lighting`, and `ClimateControl` are represented as *mandatory* which means that each smarthome configuration must include (in one way or another) each of those subfeatures (core features). Importantly, within the scope of a product line scoping process, this model can be regarded as flexible, i.e., features can be deleted or adapted and additional features (and also constraints) can be included. Such adaptations can be triggered by new insights from market analyses as well as insights directly related to the technical feasibility of allowed configurations.

Features at the second level can either be mandatory of optional – the features `Camera`, `Dimming(Dim)`, and `Temperature(Temp)` are regarded as mandatory, since they represent basic equipment to be included in every smarthome configuration. The remaining features of the model are regarded as optional, for example, the feature `ColorControl` can be offered to a customer but does not have to be included in every configuration. Note that further modeling concepts can be used for representing feature model properties. For a detailed discussion of feature model knowledge representations, we refer to [1, 23].

Beyond hierarchical relationships such as mandatory and optional features, feature models often include cross-tree constraints that express dependencies between features. Such constraints further restrict the configuration space. Related example constraints in the smarthome domain could be `Alarm` requires `MotionSensor` (i.e., $Alarm \rightarrow MotionSensor$) and `Alarm` excludes `ColorControl` (i.e., $\neg(Alarm \wedge ColorControl)$). On the basis of such a variability model, users can perform different analysis operations (representing individual queries on the feature model). Examples of such queries are: *What are the minimum features required for a basic smarthome system with climate control*? or *Which feature combinations are most relevant for urban apartment customers*? A related LLM-based assistant has the potential to provide explanations why specific features should be included in the feature model.

3. Leveraging Large Language Models for Product Line Scoping

Large language models (LLMs) have a deep contextual understanding and vast commonsense knowledge which makes them applicable in assisting complex decision-making. In the following, we analyze in which way LLMs can be used to analyse the optimality of a product line (in terms of market relevance and technical feasibility). Product line scoping includes the task of identifying which features or feature combinations are commercially relevant and technically feasible. In contrast to often manual scoping operations on the basis of feature models, LLMs can augment and partly automate scoping operations by supporting analysis operations, feasibility checks, and related commercial insights.

In such scenarios, LLMs can be applied to answer scoping questions such as *Does a smarthome system including `Security` with `Alarm` and `LockControl` but excluding `Lighting` make commercial sense?* In this example, an LLM can infer potential consequences of omitting the `Lighting` feature. Furthermore, the related market acceptance could be estimated on the basis of knowledge about typical customer preferences, market trends in the smarthome domain, and technical background knowledge about the feasibility of such configurations. To some extent, LLMs can also take over reasoning/inference tasks such as assuring that constraints integrated in the feature model do not induce an inconsistency.

Since LLMs are capable of processing natural language, they can be applied for developing conversational interfaces that support, for example, product line scoping processes. On the basis of such interfaces, users (members of the product line scoping team) can express complex queries without necessarily being able to understand the formal semantics of feature models. Furthermore, product line scoping is not necessarily based on feature models but can also be based on a textual definition of a product line (a blueprint-based representation).

Examples of complex user queries are the following: a product manager might ask *What is the most commercially attractive combination of security features for urban apartments*? or *Suggest configurations that maximize energy efficiency while keeping costs low.*, or *Create a feature model that supports the previously mentioned configurations.*

Such a query-based interaction in product line scoping is based on the following LLM-related capabilities. On the basis of available product domain knowledge, LLMs can identify when specific feature model parts are potentially triggering technical infeasibility. On the basis of information from product reviews, market reports, and other (potentially external) information sources in the training data, LLMs can estimate market potentials and the market relevance of specific features.

LLMs are able to generate human-readable explanations of the provided feedback/explanations which can also be tailored to the users' background knowledge [24]. For example, technical argumentations can be provided to users with the corresponding technical background. This also helps to create transparency and decision confidence for users part of the product line scoping team. Furthermore, product line scoping can be supported in an interactive fashion, i.e., alternative feature model implementations can be explored and users receive immediate feedback on the implications of their scoping decisions.

4. Open Research Issues

In the context of applying LLMs for supporting product line scoping processes, there exist a couple of open research issues which will be discussed in the following.

Reliability of LLM Feedback. LLM feedback/assessment quality regarding product line optimality and feasibility needs to be assured. Since LLMs do not have formal reasoning capabilities, hallucinations and inconsistencies can occur (e.g., an LLM could generate feature models or parts thereof which are inconsistent, i.e., do not allow the generation of a configuration). In this context, hybrid approaches need to be developed which allow a combination of LLMs with formal consistency checking (e.g., on the basis of constraint solvers or SAT solvers) [18].

LLM Updates. LLMs are based on domain-specific knowledge which experiences frequent updates. Since product line scoping has to depend on up-to-date market trend information and information about technological advances, and regulatory changes, efficient methods are needed that are able to continuously update the used LLMs and include information from Web search in result presentations. Furthermore, methods need to be developed that help to explain LLM feedback in terms of explaining the knowledge sources responsible for the given LLM feedback. This will help to support the aspects of transparency and trust which are crucial in the context of making high-involvement decisions for complex products and services.

Scalability of Inference Services. Since variability models can become quite large, the corresponding analysis and inference tasks require significant computational resources. Consequently, there is a need for inference services within reasonable runtime performance.

Dialog Management. Product line scoping is a complex (often group-based) decision task. This requires guidance in terms of proposing appropriate decision strategies (and decision processes) to be used for completing a decision task and also in terms of informing the user in an understandable fashion about the next steps to be completed to achieve the overall goal of identifying an optimal variability model of a product line. In this context, natural language interaction can be quite intuitive for users. However, communication has to be personalized, i.e., each user should receive system feedback and explanations in an understandable fashion.

Sustainability Aspects. Technical feasibility (T) and market relevance (M) are regarded as important decision criteria in the context of product line scoping. However, an important additional aspect to be taken into account are sustainability criteria (S) as defined by the United Nations Sustainability Development Goals (SDGs).[1] In this context, T, M, and C goals can be regarded as basic input of an optimization problem with the goal to identify optimal solutions.

Evaluation Metrics. Evaluation metrics need to be developed that help to evaluate the outcomes of LLM-enhanced product line scoping processes. Specifically, the inclusion of outdated knowledge and LLM hallucinations needs to be avoided. Related results need to be compared with the outcome of baseline processes without the support of LLM features. Example metrics include aspects such as

[1] https://sdgs.un.org/goals

commercial impact, technical feasibility, satisfaction of the customer community, and longterm positive sustainability effects. Finally, the LLM output also needs to be evaluated with regard to potential biases, for example, manipulating a group decision into a specific direction.

Acceptance of Group Decision Support. For different reasons, group decision support tools often suffer from limited user acceptance [25]. On the one hand, such tools often require user feedback in terms of specifying explicit preferences which is not appreciated in complex scenarios such as product line scoping. On the other hand, there are issues related to aspects such as decision manipulation and limited preparedness to share his/her preferences. An important open issue in the context is find better ways of providing user support leading to more tool support acceptance as it is the case now.

5. Conclusions

In this paper, we have introduced the basic idea of exploiting large language models (LLMs) to the support decision processes in product line scoping for complex products and services. On the basis of a working example from the domain of smarthomes, we have sketched how variability modeling can be combined with LLMs with the goal to increase the quality of product line scoping. This way, stakeholders can be supported and guided in complex decision tasks in a more efficient fashion.

However, there are a couple of open research issues including for example, the aspects of LLM feedback reliability and explainability of the LLM output. Our next step will be a more detailed analysis of the commercial needs of LLM-supported product line scoping. The corresponding results will be the major features of our envisioned tool for supporting LLM-enhanced product line scoping.

Declaration on Generative AI

The authors used ChatGPT for language refinement and improving readability. All AI-generated suggestions were carefully reviewed and edited by the authors, who take full responsibility for the content of this publication.

Acknowledgments

The work presented in this paper has been developed within the research project GenRE (Generative AI for Requirements Engineering) funded by the Austrian Research Promotion Agency under the project number 915086.

References

[1] A. Felfernig, A. Falkner, D. Benavides, Feature Models: AI-Driven Design, Analysis and Applications, Springer, 2024. doi:10.1007/978-3-031-61874-1.

[2] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1 ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[3] A. Popescu, S. Polat-Erdeniz, A. Felfernig, M. Uta, M. Atas, V.-M. Le, K. Pilsl, M. Enzelsberger, T. N. T. Tran, An overview of machine learning techniques in constraint solving, J Intell Inf Syst 58 (2022) 91–118. doi:10.1007/s10844-021-00666-5.

[4] A. Metzger, K. Pohl, Software product line engineering and variability management: achievements and challenges, in: Future of Software Engineering Proceedings, FOSE 2014, ACM, New York, NY, USA, 2014, pp. 70–84. doi:10.1145/2593882.2593888.

[5] J.-M. deBaud, K. Schmid, A systematic approach to derive the scope of software product lines, in: International Conference on Software Engineering, 1999, pp. 34–43. doi:10.1145/302405.302409.

[6] I. John, J. Knodel, T. Lehner, D. Muthig, A Practical Guide to Product Line Scoping, in: 10th International on Software Product Line Conference, SPLC '06, IEEE Computer Society, USA, 2006, pp. 3–12.
[7] L. Marchezan, E. Rodrigues, W. K. G. Assunção, M. Bernardino, F. P. Basso, J. a. Carbonell, Software product line scoping: a systematic literature review, in: 26th ACM International Systems and Software Product Line Conference, SPLC '22, ACM, New York, NY, USA, 2022, p. 256. doi:10.1145/3546932.3547012.
[8] M. C. C. Ojeda, J. A. H. Alegría, F. J. A. Rodriguez, An exploratory study for scoping software product lines in a collaborative way, in: 11th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE '18, ACM, New York, NY, USA, 2018, p. 17–20. doi:10.1145/3195836.3195852.
[9] K. Schmid, A comprehensive product line scoping approach and its validation, in: 24th International Conference on Software Engineering, ICSE '02, ACM, New York, NY, USA, 2002, pp. 593–603. doi:10.1145/581339.581415.
[10] S. Lubos, A. Felfernig, D. Garber, V.-M. Le, M. Henrich, R. Willfort, J. Fuchs, Towards Group Decision Support with LLM-based Meeting Analysis, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '25, ACM, New York, NY, USA, 2025, pp. 331–335. doi:10.1145/3708319.3733646.
[11] R. Samer, M. Stettinger, A. Felfernig, Group Recommender User Interfaces for Improving Requirements Prioritization, in: 28th ACM Conference on User Modeling, Adaptation and Personalization, UMAP '20, ACM, New York, NY, USA, 2020, pp. 221–229. doi:10.1145/3340631.3394851.
[12] A. Felfernig, L. Boratto, M. Stettinger, M. Tkalcic, Group Recommender Systems, Springer, 2024.
[13] V.-M. Le, T. N. T. Tran, A. Felfernig, Consistency-based integration of multi-stakeholder recommender systems with feature model configuration, in: 26th ACM International Systems and Software Product Line Conference, SPLC '22, ACM, New York, NY, USA, 2022, pp. 178–182. doi:10.1145/3503229.3547050.
[14] J. Masthoff, A. Delić, Group recommender systems: Beyond preference aggregation, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer US, New York, NY, 2022, pp. 381–420. doi:10.1007/978-1-0716-2197-4_10.
[15] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, A. Mian, A Comprehensive Overview of Large Language Models, ACM Trans. Intell. Syst. Technol. (2025). doi:10.1145/3744746.
[16] M. Acher, J. G. Duarte, J.-M. Jézéquel, On programming variability with large language model-based assistant, in: 27th ACM International Systems and Software Product Line Conference, SPLC '23, ACM, New York, NY, USA, 2023, pp. 8–14. doi:10.1145/3579027.3608972.
[17] S. Greiner, K. Schmid, T. Berger, S. Krieter, K. Meixner, Generative AI And Software Variability - A Research Vision, in: 18th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS '24, ACM, New York, NY, USA, 2024, pp. 71–76. doi:10.1145/3634713.3634722.
[18] L. Hotz, C. Bähnisch, S. Lubos, A. Felfernig, A. Haag, J. Twiefel, Exploiting Large Language Models for the Automated Generation of Constraint Satisfaction Problems, in: É. Vareilles, C. Grosso, J. M. Horcas, A. Felfernig (Eds.), ConfWS 2024, volume 3812 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 91–100.
[19] M. Acher, J. Martinez, Generative AI for Reengineering Variants into Software Product Lines: An Experience Report, in: 27th ACM International Systems and Software Product Line Conference, SPLC '23, ACM, New York, NY, USA, 2023, pp. 57–66. doi:10.1145/3579028.3609016.
[20] J. Wei, A.-L. Courbis, T. Lambolais, B. Xu, P. L. Bernard, G. Dray, W. Maalej, Getting Inspiration for Feature Elicitation: App Store- vs. LLM-based Approach, in: ASE 2024, ASE '24, ACM, New York, NY, USA, 2024, p. 857–869. doi:10.1145/3691620.3695591.
[21] J. A. Galindo, A. J. Dominguez, J. White, D. Benavides, Large Language Models to generate meaningful feature model instances, in: 27th ACM International Systems and Software Product Line Conference, SPLC '23, ACM, New York, NY, USA, 2023, pp. 15–26. doi:10.1145/3579027.

3608973.

[22] S. Lubos, M. Gartner, A. Felfernig, R. Willfort, Leveraging LLMs to Explain the Consequences of Recommendations, in: 33rd ACM Conference on User Modeling, Adaptation and Personalization, UMAP '25, ACM, New York, NY, USA, 2025, pp. 318–322. doi:10.1145/3699682.3728328.

[23] D. Benavides, S. Segura, A. Ruiz-Cortes, Automated analysis of feature models 20 years later: A literature review, Information Systems 35 (2010) 615–636.

[24] S. Lubos, T. N. T. Tran, A. Felfernig, S. Polat Erdeniz, V.-M. Le, LLM-generated Explanations for Recommender Systems, in: 32nd ACM Conference on User Modeling, Adaptation and Personalization, UMAP Adjunct '24, ACM, New York, NY, USA, 2024, pp. 276–285. doi:10.1145/3631700.3665185.

[25] M. C. C. Ojeda, F. A. Rodriguez, C. A. Collazos, Identifying Collaborative Aspects During Software Product Lines Scoping, in: 23rd International Systems and Software Product Line Conference, SPLC '19, ACM, New York, NY, USA, 2019, pp. 98–105. doi:10.1145/3307630.3342420.

Generative Design as a Configuration Problem

Jonathan Raines[1,*], David Barton[1] and Ben Hicks[1]

[1]University of Bristol, UK

Abstract

Generative design techniques such as topology optimisation can produce lightweight structures that significantly reduce emissions in aerospace and automotive applications. However, a gap exists between computationally generated designs and manufacturable parts: while topology optimisation produces optimal shapes for 3D printing or single-piece machining, industrial manufacturing relies on assemblies of standard components using processes like welding, stamping, and cutting. This paper formalises the problem of approximating topology-optimised designs using off-the-shelf parts and conventional manufacturing processes as a configuration problem. We define this problem as finding high-performing configurations of parts from industrial catalogues, modified by available processes, that minimise cost and weight while maximising geometric similarity to the target design. The key challenges include managing discrete part catalogues, representing complex 3D geometries, navigating solution spaces that grow exponentially, and handling mixed discrete-continuous optimisation variables. By framing generative design approximation as a configuration problem, we aim to bridge the gap between computational design tools and the reality of industrial manufacturing.

Keywords

Generative design, Topology Optimization, Configuration Problem, Manufacturing, Discrete Optimization, Standard Parts, Weight Optimization, Lightweighting, Design for Manufacturing

1. Introduction

The aerospace and automotive industries are both significant contributors to climate change. Aerospace contributes $2.5\,\%$ [1] of global carbon-dioxide emissions, and road passenger transport $10.8\,\%$ [2]. In both industries, lightweighting is a key means for reducing emissions. A study by the International Transport Forum concluded that if the mass of cars could be reduced back to 1970s levels (a $40\,\%$ reduction), then CO_2 emissions could be reduced by an additional $90\,\mathrm{Mt}$ ($18\,\%$) [3]. Topology optimisation techniques such as Solid Isotropic Material with Penalization (SIMP) [4] can search for the lightest part that meets a loading condition. Topology optimisation (and more broadly, generative design) tools are available in commercial software such as Autodesk Fusion [5] and COMSOL [6]. However, parts designed using these methods are not commonly used in these industries or other commercial projects due to the following limitations. Firstly, the geometry of the parts generally requires 3D printing, casting or machining the part in a single piece. Aerospace and automotive are safety-critical applications, inhibiting the adoption of 3D printing for structural parts. The 3D printing process can introduce microscopic cracks, leading to unacceptable part strength variations. Secondly, both industries need to manufacture parts at scale, and 3D printing costs do not scale with production volume. Machining and casting are practical at high volumes, but are not practical for every part. The generatively-designed bracket shown in Figure 1 would be economically infeasible to machine due to its complexity and the proportion of the blank that would be scrap.

The following industry cases illustrate the gap between the objectives of topology optimisation and industrial use. In 2016, Airbus unveiled a prototype "bionic partition". Created using generative design, the 3D printed design was $50\,\%$ lighter [7]. The prototype exceeded the capacity of 3D printers at the time, so the prototype was made in 122 parts and fastened together [8]. However, a later news report revealed that the approach was abandoned (due to manufacturing cost [9]), and the first installed

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

*Corresponding author.

jonathan.raines@bristol.ac.uk (J. Raines); david.barton@bristol.ac.uk (D. Barton); ben.hicks@bristol.ac.uk (B. Hicks)

https://JonathanRaines.github.io (J. Raines); https://cityinthesky.co.uk (D. Barton)

0009-0001-8325-2005 (J. Raines); 0000-0002-0595-4239 (D. Barton); 0000-0001-8294-4694 (B. Hicks)

version was "a sandwich panel with a honeycomb core and carbon fibres (CFRP)" [10]. In 2017, Autodesk collaborated with the *Bandito Brothers* to create a hotrod car chassis [11]. The team used Autodesk's project DreamCatcher [12] to design the chassis using telemetry data from a prototype. They then manually approximated the design in such a way that it could be constructed from welded tubing [13, 14]. The team aimed to fully 3D print the chassis but, at the time of writing, we could not find a record of them succeeding.

The benefits of generative design cannot be realised when it is limited to a small subset of manufacturing processes available. Automating the manual approximation of a generated design to use off-the-shelf parts and processes would bridge the gap between available tools and industry use. This would properly integrate generative design into engineers' toolboxes as one way to lightweight parts and reduce emissions. To this end, we present the approximation process as a configuration problem.

2. Problem Definition

In this section, we present a generalised form of the problem and provide illustrative examples.

The Manufacturing Problem

Instance:

- a target shape generated using topology optimisation
- a library of parts
- a set of processes that can modify instances of parts in the library
- one or more optimisation criteria
- a target production volume

Task:
Find the configuration of part instances, each possibly modified by a sequence of processes, that minimises the optimisation criteria.

2.1. Configuration Definition

A configuration is a directed tree where:

- **Nodes** are manufacturing processes with parameters
- **Leaf nodes** are processes adding parts from a part library
- **Edges** are the flow of parts
- **Root** is the final assembly process, which outputs the completed product

An example configuration is shown in Figure 2.

2.2. Validity Constraints

A configuration is valid if:

- The tree is connected (single root),
- there are no intersecting parts,
- all joints/connections are physically realisable,
- each process node has compatible incoming edges,
- and the parameters of each process node are feasible.

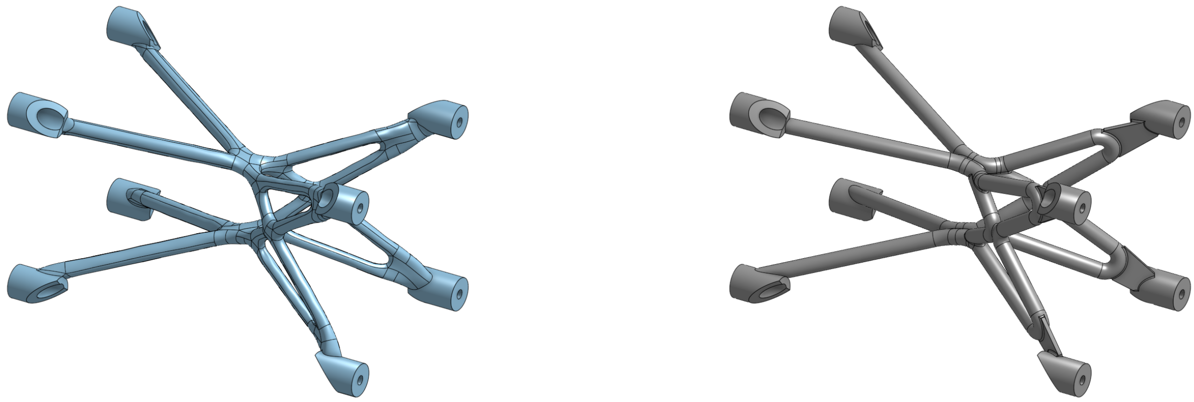


Figure 1: A generatively designed bracket (left) and a version redesigned to be manufactured at volume using CNC bending, sheet metal cutting and turning (right).

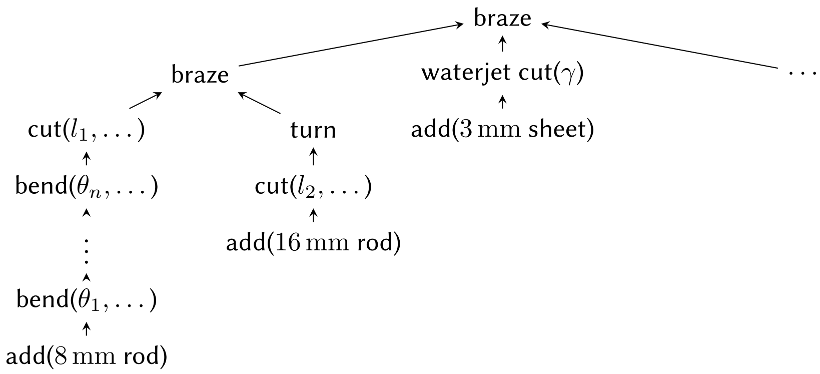


Figure 2: An abridged example configuration tree for the bracket shown in Figure 1 (right). The left-hand branch represents the main body of the bracket, the second branch the bosses at the corners, and the third branch the plates used to connect the bosses on the right-hand side.

2.3. Evaluation Criteria

Finding valid configurations is not sufficient. Many will be manufacturable but perform poorly against a given objective. Functional evaluations, such as Finite Element Analysis (FEA) and real-world testing are required in safety-critical industries such as aerospace and automotive. However, they are expensive (in terms of compute, time and resources). For searching through valid configurations, or training a system to generate them in a data-driven approach, a proxy is required. A configuration can be evaluated based on its geometric similarity to a target form generated using Topology Optimization. This can be done by instantiating 3D models of the stock material and applying the modifications of the processes in the tree. The resulting shape can be compared to the target using the Hausdorff distance (the maximum distance between any point on one shape and its nearest point on the other shape). Generally, this approach can be thought of as using a continuous representation and gradient descent to find a design, then approxmating that design with discrete operations. Using a shape-based metric also offers flexibility. A user of a configuration generator could provide the output of available generative design and topology optimisation tools, or model a freeform shape by hand. A drawback to this approach is that small changes in geometry can lead to large changes in deflection or peak stress. As such, if such a system were being used in the aerospace or automotive industries, configurations suggested by the tool of interest to a designer would be evaluated using functional evaluations such as FEA.

Many valid configurations will approximate the target shape, but may not be optimal in terms of cost or production volume. As such, a cost objective must also be applied. This can be achieved by assigning a cost to each part in the library and summing the costs of the parts used in the configuration.

Each process can be costed by assigning a set-up cost and an operation cost. The set-up cost is incurred once if the process is used in a configuration, and the operation cost is incurred for every instance of a process node in a configuration.

2.4. Data

The SELTO dataset [15] contains 9848 example parts generated using SIMP [4]. Each example is comprised of a voxel representation of the generated part, as well as the forces and boundary conditions used to generate it.

3. Challenges

The manufacturing problem presents three key challenges.

First, the number of possible configurations grows exponentially with the size of the component library, the process library, the number of nodes added to a configuration, and the number of design variables. Industrial part catalogues contain thousands of components. This is often referred to as the curse of dimensionality.

Second, the challenge of representing a configuration. The parts and processes can be represented as trees as described in Section 2.1. However, the configuration also represents a 3D shape. It is necessary to generate and check the 3D shape for self-intersection. A configuration that appears valid based on the tree structure may still be invalid due to a self-intersection. Consider a bar that has been bent 270 degrees. Potential 3D representations include Signed Distance Functions (SDFs) or Boundary Representations (B-reps).

Finally, the problem combines discrete and continuous variables, for example, tubing comes in fixed diameters but can be cut to any length, and sheet materials have standard thicknesses but arbitrary cut shapes. The number of variables also varies depending on the configuration. A bend will have a different number of parameters than a cut.

4. Related Work

In this section, we describe related problems and the ongoing research into them. We aim to differentiate this problem, as well as explain the inspiration for the research avenues detailed in the next section.

4.1. Configuration Design

Mittal and Frayman [16] defined a general framework for configuration design. The problem presented in this article adds the complication that components can be modified using a library of operations before being combined. As highlighted in [17], representing engineering components in a reusable manner has proven challenging. The problem presented in this article limits the component library to stock materials that can be represented as a set of parametric shapes.

4.2. Manufacturing Constraints for Topology Optimisation

Researchers have modified density-based methods (such as SIMP [4]) to respect minimum feature size and overhang constraints of 3D printing [18, 19, 20] and to impose constraints for 2.5 and 5 axis machining, using projections to penalize areas inaccessible to the tool during optimisation [21]. Greminger [22] adopted a data-driven approach, training a Generative Adversarial Network (GAN) on examples of machinable parts. These processes make progress towards manufacturability. However, the assumption of a solid isotropic material is intrinsic, so separate parts that may have been pre-processed cannot be represented.

4.3. Analog Circuit Synthesis

Circuit topology synthesis shares the goal of configuring a library of parts. Typically, the circuit is represented as a graph, with parts as nodes and connections as edges (e.g. [23]). Gao et al. [24] argued that this representation is ambiguous, as parts have pins with different functionalities. They proposed adding pins as an additional node type to the graph, allowing for explicit pin-to-pin connections. They also demonstrated the effectiveness of converting the graph to a set of sequences using Eulerian walks and applying a Transformer [25] in a system they dubbed AnalogGenie.

4.4. Program Synthesis

Program synthesis is the search for a program that generates a desired output. In the context of 3D modelling, the output is commonly a target shape. Before the popularisation of Large Language Models (LLMs), Domain Specific Languages (DSLs) were used to constrain the search space to a tractable size. Jones et al. [26] proposed a DSL called ShapeAssembly. They trained a hierarchical Variational Autoencoder (VAE) on ShapeAssembly programs reverse engineered from assemblies in PartNet [27]. They could then generate new programs, and thus assemblies, by sampling from the latent space of programs. Ellis et al. [28] proposed DreamCoder, that could expand its own DSL through a process of self-improvement.

A limitation of DSLs is that they constrain what can be expressed. We note that this may actually be a useful property in the context of ensuring manufacturability. However, to overcome this, researchers have recently favoured using LLMs to generate programs in Turing-complete languages, for example, Python. Notable work related to geometry generation includes CAD-CODER, which takes an image of a part and produces a parametric Computer-Aided Design (CAD) model [29]. The approach makes use of a Vision Language Model (VLM) to generate Python code that generates the model. While such approaches demonstrate the potential to convert generatively designed parts into parametric CAD models, they do not fully address the manufacturing problem presented in this paper, as CAD models are not necessarily manufacturable.

5. Research Avenues

We identify two broad categories of approaches for addressing the configuration problem presented in this paper. The first involves searching for a configuration directly, which we term an output-centric approach. The second focuses on searching for a program that generates a configuration, which we refer to as a program-centric approach.

For output-centric approaches, several promising directions emerge. Graph generation techniques offer a way to generate configurations directly. Transformer-based models show promise, as demonstrated in AnalogGenie [24].

The configuration described in Section 2.1 can be viewed as the Abstract Syntax Tree (AST) of a program. One program-centric approach is to use an LLM to write the program using a supplied library of functions. Another approach is to generate a program that searches for a configuration. Both approaches can be further enhanced by employing evolutionary algorithms on the output programs, feeding high-performing programs back into the model for iteration.

Acknowledgments

Jonathan Raines acknowledges funding from the UKRI for a Centre for Doctoral Training studentship in Interactive Artificial Intelligence at the University of Bristol. (EP/S022937/1)

Declaration on Generative AI

During the preparation of this work, the author(s) used Claude Sonnet 4 (Anthropic) and Grammarly for: drafting content, paraphrasing and rewording, improving writing style, abstract drafting, grammar and spell check, peer review simulation, and content enhancement. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] H. Ritchie, What share of global CO_2 emissions come from aviation?, Our World in Data (2024). URL: https://ourworldindata.org/global-aviation-emissions, accessed: 2025-06-17.

[2] H. Ritchie, Cars, planes, trains: Where do CO_2 emissions from transport come from?, Our World in Data (2020). URL: https://ourworldindata.org/co2-emissions-from-transport, accessed: 2025-06-17.

[3] E. Windisch, V. Benezech, G. Chen, J. Kauppila, Lightening Up: How Less Heavy Vehicles Can Help Cut CO2 Emissions, Case-Specific Policy Analysis Report OECD/ITF 2017, The International Transport Forum, 2017. URL: https://www.itf-oecd.org/sites/default/files/docs/less-heavy-vehicles-cut-co2-emissions.pdf, accessed: 2025-06-17.

[4] M. P. Bendsøe, N. Kikuchi, Generating optimal topologies in structural design using a homogenization method, Computer Methods in Applied Mechanics and Engineering 71 (1988) 197–224. doi:10.1016/0045-7825(88)90086-2.

[5] Autodesk, Autodesk Fusion, 2024. URL: https://www.autodesk.com/uk/products/fusion-360/, accessed: 2025-07-11.

[6] COMSOL, COMSOL Multiphysics 6.0, 2021. URL: https://www.comsol.com/comsol-multiphysics, accessed: 2025-07-11.

[7] D. Nagy, D. Zhao, D. Benjamin, Nature-Based Hybrid Computational Geometry System for Optimizing Component Structure, in: K. De Rycke, C. Gengnagel, O. Baverel, J. Burry, C. Mueller, M. M. Nguyen, P. Rahm, M. R. Thomsen (Eds.), Humanizing Digital Reality: Design Modelling Symposium Paris 2017, Springer, Singapore, 2018, pp. 167–176. doi:10.1007/978-981-10-6611-5_15.

[8] K. Walmsley, Autodesk and Airbus generatively design a 3D-printed bionic partition for the A320, 2016. URL: https://www.keanw.com/2016/02/autodesk-and-airbus-generatively-design-a-3d-printed-bionic-partition-for-the-a320.html, accessed: 2025-06-18.

[9] B. Koenig, 3D Printing Cleared for Takeoff, 2020. URL: https://www.sme.org/technologies/articles/2020/january/3d-printing-cleared-for-takeoff/, accessed: 2025-06-18.

[10] M. Kirby, Press Release: Bucher's Bionic Partition installed successfully on A320 -, 2021. URL: https://runwaygirlnetwork.com/2021/04/buchers-bionic-partition-installed-successfully-a320/, accessed: 2025-06-17.

[11] TED, The incredible inventions of intuitive AI | Maurice Conti, 2017. URL: https://youtu.be/aR5N2Jl8k14?t=782, accessed: 2025-09-25.

[12] G. Fitzmaurice, Project Dreamcatcher: Generative Design Solutions in CAD, 2017. URL: https://www.research.autodesk.com/projects/project-dreamcatcher/, accessed: 2025-07-02.

[13] DEVELOP3D, Garage bandits, 2017. URL: https://develop3d.com/profiles/garage-bandits-industrial-design-hackrod-automotive/, accessed: 2025-06-18.

[14] D. Terdiman, Inside The Hack Rod, The World's First AI-Designed Car, 2015. URL: https://www.fastcompany.com/3054028/inside-the-hack-rod-the-worlds-first-ai-designed-car/, accessed: 2025-06-18.

[15] S. Dittmer, D. Erzmann, H. Harms, R. Falck, M. Gosch, SELTO Dataset, 2023. URL: https://zenodo.org/records/7781392.

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

[17] R. Malik, M. Ahmad, J. Vepsäläinen, Modular representation of components to enable generative engineering, in: 2024 IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC), 2024, pp. 1–9. doi:10.1109/ESARS-ITEC60450.2024.10819828.

[18] B. Barroqueiro, A. Andrade-Campos, R. a. F. Valente, Designing Self Supported SLM Structures via Topology Optimization, Journal of Manufacturing and Materials Processing 3 (2019) 68. doi:10.3390/jmmp3030068.

[19] E. Fernández, K.-k. Yang, S. Koppen, P. Alarcón, S. Bauduin, P. Duysinx, Imposing minimum and maximum member size, minimum cavity size, and minimum separation distance between solid members in topology optimization, Computer Methods in Applied Mechanics and Engineering 368 (2020) 113157. doi:10.1016/j.cma.2020.113157.

[20] B. M. Weiss, J. M. Hamel, M. A. Ganter, D. W. Storti, Data-Driven Additive Manufacturing Constraints for Topology Optimization, in: ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection, 2018. doi:10.1115/DETC2018-85391.

[21] H. Y. Lee, M. Zhu, J. K. Guest, Topology optimization considering multi-axis machining constraints using projection methods, Computer Methods in Applied Mechanics and Engineering 390 (2022) 114464. doi:10.1016/j.cma.2021.114464.

[22] M. Greminger, Generative Adversarial Networks With Synthetic Training Data for Enforcing Manufacturing Constraints on Topology Optimization, in: ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers Digital Collection, 2020. doi:10.1115/DETC2020-22399.

[23] Z. Dong, W. Cao, M. Zhang, D. Tao, Y. Chen, X. Zhang, CktGNN: Circuit Graph Neural Network for Electronic Design Automation, 2024. doi:10.48550/arXiv.2308.16406. arXiv:2308.16406.

[24] J. Gao, W. Cao, J. Yang, X. Zhang, AnalogGenie: A Generative Engine for Automatic Discovery of Analog Circuit Topologies, 2025. URL: https://arxiv.org/abs/2503.00205v1.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention Is All You Need, 2017. doi:10.48550/arXiv.1706.03762. arXiv:1706.03762v1.

[26] R. K. Jones, T. Barton, X. Xu, K. Wang, E. Jiang, P. Guerrero, N. J. Mitra, D. Ritchie, ShapeAssembly: Learning to generate programs for 3D shape structure synthesis, ACM Trans. Graph. 39 (2020) 234:1–234:20. doi:10.1145/3414685.3417812.

[27] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, H. Su, PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[28] K. Ellis, C. Wong, M. Nye, M. Sablé-Meyer, L. Morales, L. Hewitt, L. Cary, A. Solar-Lezama, J. B. Tenenbaum, DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning, in: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2021, Association for Computing Machinery, New York, NY, USA, 2021, pp. 835–850. doi:10.1145/3453483.3454080.

[29] A. C. Doris, M. F. Alam, A. H. Nobari, F. Ahmed, CAD-Coder: An Open-Source Vision-Language Model for Computer-Aided Design Code Generation, 2025. doi:10.48550/arXiv.2505.14646. arXiv:2505.14646.

Visualizing Customization: The Impact of Product Visualization Modalities on User-Friendly Description in Online Configurators

Achraf Arouya[1,*,†], Enrico Sandrin[1,†] , Cipriano Forza[1,†] and Alessio Trentin[1,†]

[1] *University of Padova, Department of Management and Engineering, Stradella San Nicola 3, 36100 Vicenza, Italy*

Abstract

This study investigates the influence of various product visualization modalities on the user-friendly product space description capability of online sales configurators (OSCs). As the possibility for customers to self-customize a product becomes more and more common in e-commerce, understanding how different visualization techniques affect OSC users' comprehension and decision-making is important. We examine ten visualization modalities, including 2D and 3D visualizations, augmented reality, and virtual try-on, across four aspects of user-friendly product space description capability: comprehensive presentation, balanced description, adaptability to user expertise, and versatility in highlighting product capabilities and structure. Using data from 516 evaluations of different OSCs, we employ regression analysis to examine the effectiveness of each modality. Our findings reveal that static and semi-interactive modalities, such as 2D visualization and virtual images, consistently enhance user-friendly product space description capability across all its considered aspects. In contrast, more complex modalities, such as 3D walkthroughs, show mixed results. We also explore the impact of visualization timing and gender differences, finding that end-of-configuration visualizations generally outperform real-time updates. These insights contribute to the optimization of OSC design, potentially improving user experience and decision-making in digital customization environments.

Keywords

Online Sales Configurator, Product Visualization Modalities, User-Friendly Description Capability

1. Introduction

On the current market, more and more online sales configurators are being introduced by major consumer companies [1]. These configurators enable customers to personalize products based on their preferences. Functionally, they are knowledge-based systems that support potential customers in completely and correctly specifying a product solution within a company's product space [2, 3]. From a technical viewpoint, sales configurators are rule-based systems that guide users through the configuration process, based on predefined product options and combinability constraints modeled into the configurator [4, 5].

While the technical aspects of configuration systems are well-developed, research on user perceptions and consumer behavior within the configuration process continues to evolve. Recent studies have highlighted the critical role of visualization in shaping user experience and decision-making within configurators. In particular, Sandrin and Forza [6] emphasize the importance of re-examining visualization strategies in light of modern technologies. They argue that advancements in augmented reality (AR), virtual reality (VR), and other immersive technologies have opened new avenues for enhancing product representation and user interaction in configurators.

Building on these insights, Petterle et al. [7] introduce an evaluation framework to better understand how visualization tools function in practice and to assess the effectiveness of these tools.

* *ConfWS'25: 27th International Workshop on Configuration, October 25–26, 2025, Bologna, Italy*

* Corresponding author.

† These authors contributed equally.

achraf.arouya@studenti.unipd.it (A. Arouya); enrico.sandrin@unipd.it (E. Sandrin); cipriano.forza@unipd.it (C. Forza); alessio.trentin@unipd.it (A. Trentin)

0009-0003-1331-0956 (A. Arouya); 0000-0001-9170-0683 (E. Sandrin); 0000-0003-4583-2962 (C. Forza); 0000-0002-7853-4104 (A. Trentin)

This framework categorizes visualization modalities according to eleven descriptors, such as presence, embodiment, realism, vividness, and interactivity. This framework enables researchers and practitioners to compare different visualization modalities within configurators in a structured and meaningful way.

Indeed, these technological advancements, including virtual try-on, 3D visualization, and 360-degree views, have transformed how consumers engage with customizable products. However, despite the availability of multiple visualization modalities, their effectiveness in improving the user-friendly product space description capability of configurators remains unclear. According to Trentin et al. [2], this is one of five key capabilities that OSCs should deploy to reduce cognitive load and anticipated regret, alongside focused navigation, flexible navigation, benefit–cost communication, and easy comparison. As Blazek [8] points out, the evolution of product configurators must focus not just on technological capabilities but on creating customization experiences that truly support user understanding and engagement. Accordingly, in order to augment our knowledge on how technological advancements in the area of product visualization enhance customization experiences, this study aims to investigate how different product visualization modalities influence the user-friendly product space description capability of online sales configurators. Specifically, we examine four key aspects of this capability:

- Comprehensive Presentation Across Time Constraints
- Balanced Description for General and Detailed Understanding
- Adaptability to User Expertise Levels
- Versatility in Highlighting Product Capabilities and Structure

Additionally, we explore the impact of visualization timing (real-time updates vs. end-of-configuration visualization) and gender differences in shaping user evaluations of this capability. In doing so, we respond to recent calls in the field to consider both the technological and human factors in configurator design [9].

This study is exploratory in nature: our goal is to uncover meaningful patterns and generate insights into what makes the experience clearer, easier, or more engaging for users. By exploring this still underdeveloped area, we hope to contribute to a deeper understanding of how visualization supports user decision-making in digital configuration settings. In the long run, these findings can help e-commerce platforms, digital marketers, and UX designers seeking to optimize online configurators and enhance product comprehension, user engagement, and overall satisfaction in digital product customization.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on product visualization in online sales configurators. Section 3 describes the research method, including data collection, measures, and data analysis. Section 4 presents the empirical results of the regression analyses, organized around regression model performance, visualization modality effectiveness, and the effects of visualization timing, gender, and product type. Section 5 discusses the findings in relation to prior research, while Section 6 concludes with the main contributions of this research to theory and practice as well its limitations and related opportunities for further research.

2. Literature Review

Online sales configurators (OSCs) have become indispensable tools in implementing mass-customization strategies within e-commerce environments. These systems are designed to support customers in specifying a valid product configuration that matches their needs and preferences, within the constraints of a firm’s product offering [2, 3]. As highlighted by Walcher and Piller [10], OSCs empower customers to define their desired product configurations within a company's predefined solution space, thus fostering co-creation, as customers actively participate in defining their individualized product solutions [11, 12].

The effectiveness of OSCs largely depends on their ability to present product information clearly and intuitively. As emphasized by Forza and Salvador [13], the commercial dialogue in OSCs should mirror the natural way customers describe their product preferences. Recent research by Grosso and Forza [14] suggests that integrating social interaction features in OSCs can further enrich the customization experience by enabling users to seek advice from friends, online communities, or company representatives.

In this context, Trentin et al. [2] propose five key capabilities that effective configurators should deploy: user-friendly product-space description, focused navigation, flexible navigation, benefit–cost communication, and easy comparison. Each of these capabilities contributes to reducing the user's cognitive and emotive costs [2] while increasing perceived customization value [15, 16]. Among these capabilities, the ability to present the product space in a user-friendly manner is particularly important, as it allows the configurator to adjust its presentation according to different usage contexts, such as the user's level of expertise or available decision time [2]. This principle resonates with Forza and Salvador's call for configurators to reflect natural customer language and behavior [13].

A growing body of research underscores the central role of product visualization in enhancing user experience and facilitating informed decision-making in OSCs. For instance, Di et al. [17] highlight the important role played by product images in capturing consumer attention, building trust, and increasing conversion rates. The evolution of visualization technologies has led to a shift from static 2D images to more interactive and immersive formats. Recent work by Petterle et al. [7] offers a structured framework for evaluating these advancements, identifying eleven key variables, including presence, realism, vividness, and interactivity, that differentiate traditional and advanced visualization modalities.

Among traditional tools, 3D visualization has gained significant attention in recent years. For instance, Ozok and Komlodi [18] found that users perceived 3D product representations as more detailed, engaging, and informative than 2D images, resulting in higher consumer satisfaction. Similarly, Moritz [19] established that interactive 3D visualizations were particularly beneficial for customizable products.

At the frontier of product visualization technologies are augmented reality (AR) and virtual reality (VR), which offer immersive product interaction and represent the cutting edge of product visualization in e-commerce. A study by Jessen et al. [20] suggests that AR enhances customer engagement, influences purchase decisions, and fosters positive brand perception. In parallel, Liu et al. [21] find that VR shopping environments can simulate physical stores, offering consumers a sense of presence and engagement comparable to in-person shopping. However, the effectiveness of these advanced visualization modalities is not universally established. For instance, Befort [22] compares the effectiveness of 3D product visualization via AR and VR in e-commerce and finds that, while traditional 2D images remain effective, AR outperforms VR in user engagement, particularly among older generations.

Another key factor influencing the effectiveness of product visualization is the timing of visualization. Whether visual information is delivered in real time during the configuration process or at the end plays an important role in shaping the user experience. Sandrin and Forza [6] highlight the need for further research on how these different timings of visualization impact user comprehension and decision-making in OSCs.

Additionally, user diversity is emerging as a relevant consideration in configurator design. Gender-based differences in how users interpret and respond to visual content have been noted in various studies, but their specific impact on OSC visualization effectiveness remains underexplored. However, Yi et al. [9] have stressed the importance of considering user diversity in configurator design to create more inclusive and effective customization experiences. Similarly, recent work by Blazek [8] underscores the importance of balancing technological advancements with user-centric design principles in creating effective customization experiences.

In summary, while extensive research exists on various aspects of OSCs and product visualization, there remains a gap in understanding how specific visualization modalities impact the user-friendly product space description capability of sales configurators, particularly when considering factors such as visualization timing and gender differences. This study aims to address this gap and contribute to the ongoing evolution of OSC design and effectiveness.

3. Method

This study adopts a quantitative approach to explore the impact of different product visualization modalities on the user-friendly product space description capability of OSCs. In this exploration, the research also considers the potential moderating effects of visualization timing and gender.

In line with established methodological guidelines for such research, this section begins by outlining the data collection procedure and describing the sample characteristics. It then presents the measures of the focal constructs and concludes by illustrating the statistical techniques employed to analyze the data.

3.1. Data collection procedure

The data were collected from students enrolled in a digital customization course at the University of Padova during the 2022-2023 academic year. Each participant was randomly assigned a list of OSCs by the course professor to ensure a diverse range of visualization modalities. From the assigned list, each student selected four configurators and, with each of them, configured a product from start to finish and then modified his/her configuration to explore different options. Each participant completed individual evaluation forms, including pre- and post-surveys, which were specifically designed to collect data on various dimensions used to evaluate the online configurators. The final dataset includes 516 evaluations, with 348 (67.4%) from male participants and 168 (32.6%) from female participants (Figure 1). All respondents were native Italian speakers enrolled in the same course with similar academic backgrounds.

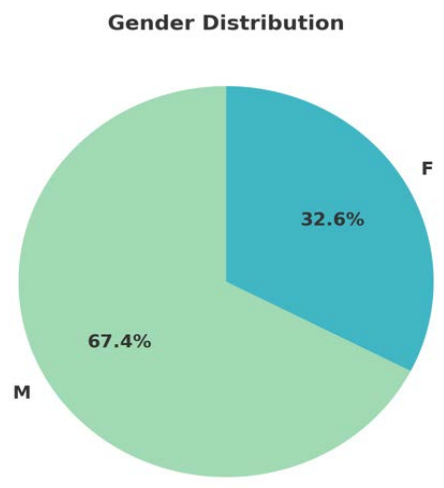


Figure 1: Gender Distribution of Respondents.

3.2. Measures

3.2.1. Independent Variables

Ten visualization modalities were assessed for their availability within each configurator (see Table 1). These included virtual try-on, augmented reality, 3D walkthrough, 3D visualization, 360 view, 2D visualization, product video, photo of the real product, virtual image, and the product in motion. The availability of each modality was coded: present (1) or absent (0).

Table 1: Independent Variables - Product Visualization Modalities

CODE	NAME	DEFINITION	RANGE
Q367	Virtual Try-On	Allows users to see how a product would look on their face or body, typically used for makeup, accessories, clothing, shoes, etc.	0 – NO 1 – YES
Q368	Augmented Reality (AR)	Displays products within the user's real-world environment, such as furniture in their living room or other items in their space.	0 – NO 1 – YES
Q369	3D Walkthrough	Enables users to virtually explore an environment, such as walking through an apartment or other spaces.	0 – NO 1 – YES
Q370	3D Visualization	Products are presented in three-dimensional models, allowing a more detailed and interactive experience.	0 – NO 1 – YES
Q371	360 View	Provides a full, interactive view of a product from all angles.	0 – NO 1 – YES
Q372	2D Visualization	Traditional flat images of a product, such as pictures from various perspectives, typically used in online shopping.	0 – NO 1 – YES
Q373	Video	A moving image of the product in action, often used to demonstrate its features or functionality.	0 – NO 1 – YES
Q374	Photo of the Real Product	A photograph of the actual product, offering a realistic view of what the consumer would receive.	0 – NO 1 – YES
Q375	Virtual Image	A digitally created image or model of the product, generated by software to visualize the product in a simulated environment.	0 – NO 1 – YES
Q376	The Product in Motion	The product is shown in action or in motion, allowing users to see how it operates or behaves during use.	0 – NO 1 – YES

3.2.2. Dependent Variables

Each aspect of the user-friendly product space description capability was measured by means of one item taken from validated instruments used in previous OSC-related studies [e.g., 2, 3] (see Table 2). Each item was measured on a 7-point Likert scale (1 = strongly disagree, 7 = strongly agree).

Table 2: Dependent Variables - Item-level of User-Friendly Description Capability

CODE	NAME	DEFINITION	RANGE
Q58	Comprehensive Presentation Across Time Constraints	The system gives an adequate presentation of the choice options for when you are in a hurry, as well as when you have enough time to go into the details.	1=Low 7=High
Q60	Balanced Description for General and Detailed Understanding	The product features are adequately presented for the user who just wants to find out about them, as well as for the user who wants to go into specific details.	1=Low 7=High
Q86	Adaptability to User Expertise Levels	The choice options are adequately presented for both the expert and inexpert user of the product.	1=Low 7=High
Q90	Versatility in Highlighting Product Capabilities and Structure	The site gives an adequate representation of the products for when one wants to know what the product is used for, as well as what it consists of.	1=Low 7=High

3.2.3. Moderating Variables

Visualization timing and gender were included in the analysis as possible moderating variables. The two alternative solutions concerning visualization timing were measured with the binary variables reported in Table 3.

Table 3: Visualization Timings Variables

CODE	NAME	DEFINITION	RANGE
Q377	Real-Time Visualization	Product visualization updates simultaneously or immediately after a modification of selected options.	0 – NO 1 – YES
Q377	End-Point Visualization	Product visualization is displayed only at the end of the configuration process, requiring users to complete all selections before viewing the final product.	0 – NO 1 – YES

3.3. Data Analysis

The dataset was analyzed using Generalized Least Squares (GLS) regression with a hybrid weighting approach. This approach combines the frequency of visualization modality availability and the distribution of product types to assign weights that account for both common and rare cases, ensuring each is fairly represented in the results. This type of regression was chosen to address issues of heteroscedasticity and autocorrelation in the residuals identified during preliminary Ordinary Least Squares (OLS) regression analyses. To enhance interpretability and address potential non-linearity, we applied a log transformation to the dependent variables.

Separate GLS models were estimated for each of the four dependent variables to assess the impact of the visualization modalities. In addition, subgroup analyses were conducted to explore the role of visualization timing and gender as moderating variables.

4. Results

This section presents the results from the GLS regression analyses. The findings are organized according to five thematic dimensions: (1) overall model performance, (2) effectiveness of visualization modalities, (3) influence of visualization timing, (4) influence of gender, and (5) influence of product type.

4.1. Overall Model Explanatory Power

The GLS regression models demonstrate modest explanatory power across the four dimensions of user-friendly product space description capability. *R*-squared values range from 0.033 to 0.070 (Figure 2), indicating that visualization modalities account for a small but meaningful portion of the variance in user evaluations. Among the four dimensions, Balanced Description for General and Detailed Understanding shows the highest *R*-squared value at 0.070, suggesting that visualization choices explain this dimension better than the others. Comprehensive Presentation and Versatility in Structural Description follow with *R*-squared values of 0.044 and 0.048, respectively. In contrast, Adaptability to Expertise shows the lowest explanatory power ($R^2 \approx 0.033$).

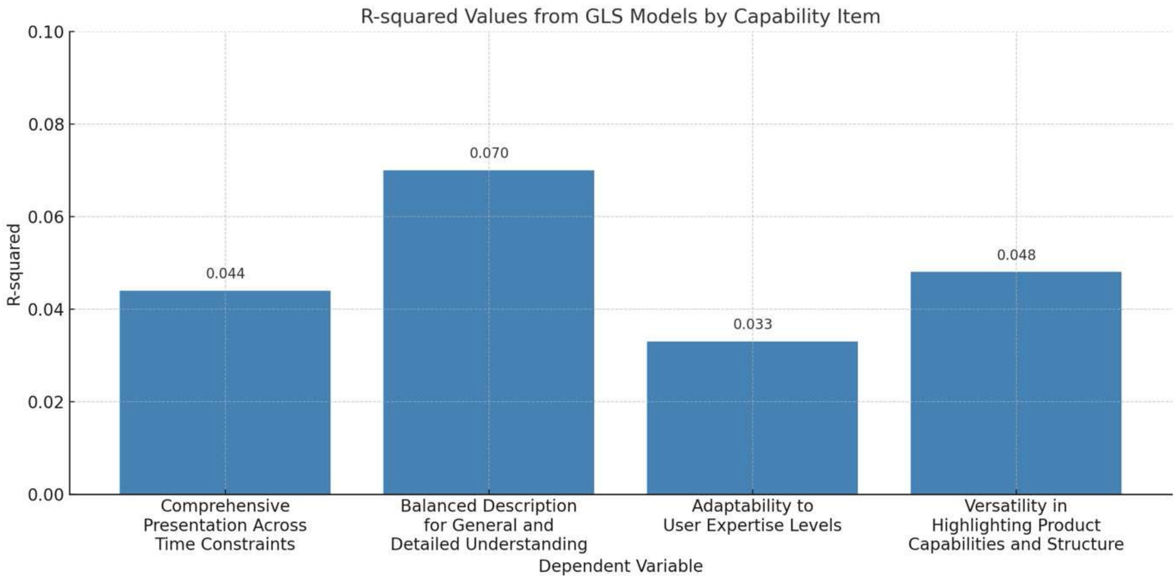


Figure 2: Bar Chart of *R*-Squared

4.2. Effectiveness of Visualization Modalities

The heatmap presented in Figure 3 illustrates the standardized regression coefficients derived from the GLS models across the four dimensions of the user-friendly product space description capability. A few patterns emerge across visualization categories:

Static and semi-interactive formats, including 2D visualization, virtual image, photo of the real product, and 3D visualization, tend to be positively associated with all four dimensions. For instance, photo of the real product has significant positive effects on presentation comprehensiveness across time constraints ($\beta = 0.07$, $p = 0.023$) and balanced description for general and detailed understanding ($\beta = 0.06$, $p = 0.018$). Similarly, interactive formats, such as 360 view, have a significant positive effect on balanced description ($\beta = 0.05$, p = 0.075).

Conversely, immersive technologies, particularly virtual try-on, exhibit significant negative effects across multiple dimensions, particularly balanced description ($\beta = -0.12$, $p = 0.005$) and versatility in highlighting product capabilities and structure ($\beta = -0.13$, $p = 0.007$), suggesting potential challenges in terms of user-friendly product space description capability.

Finally, dynamic formats, such as video, show positive effects, particularly for comprehensive presentation across time constraints ($\beta = 0.08$, $p = 0.091$) and balanced description ($\beta = 0.07$, $p = 0.054$), indicating their value in supporting user-friendly product space description capability.

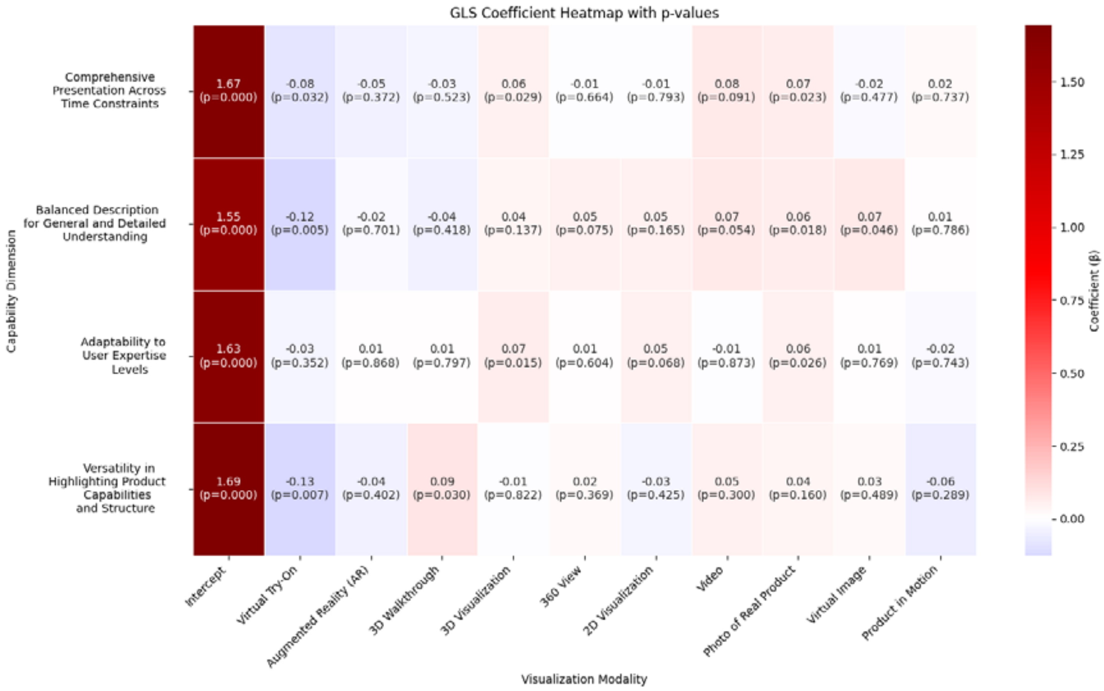


Figure 3: Heatmap of Regression Coefficients.

These results are complemented by the regression coefficient plot with confidence intervals presented in Figure 4, which visually represent the statistical significance and direction of effects across modalities.

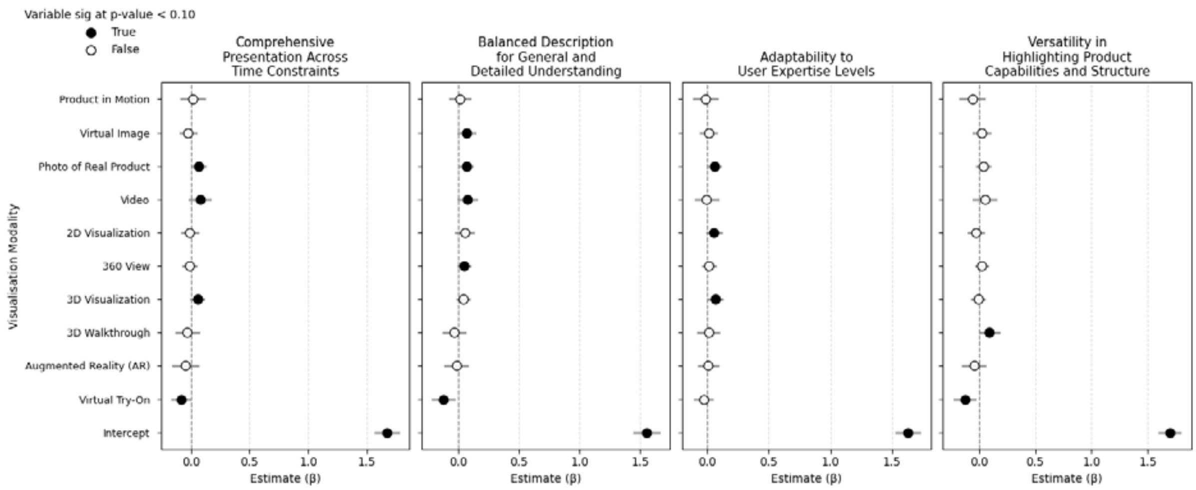


Figure 4: Regression Coefficient Plots.

4.3. Visualization Timing Influence

The timing of visualization (i.e., product visualization during the configuration process or at the end of the configuration process) emerges as a critical determinant of configurator effectiveness. Its role is clearly illustrated in the heatmap (Figure 5), which compares immediate updates and end-of-configuration visualization.

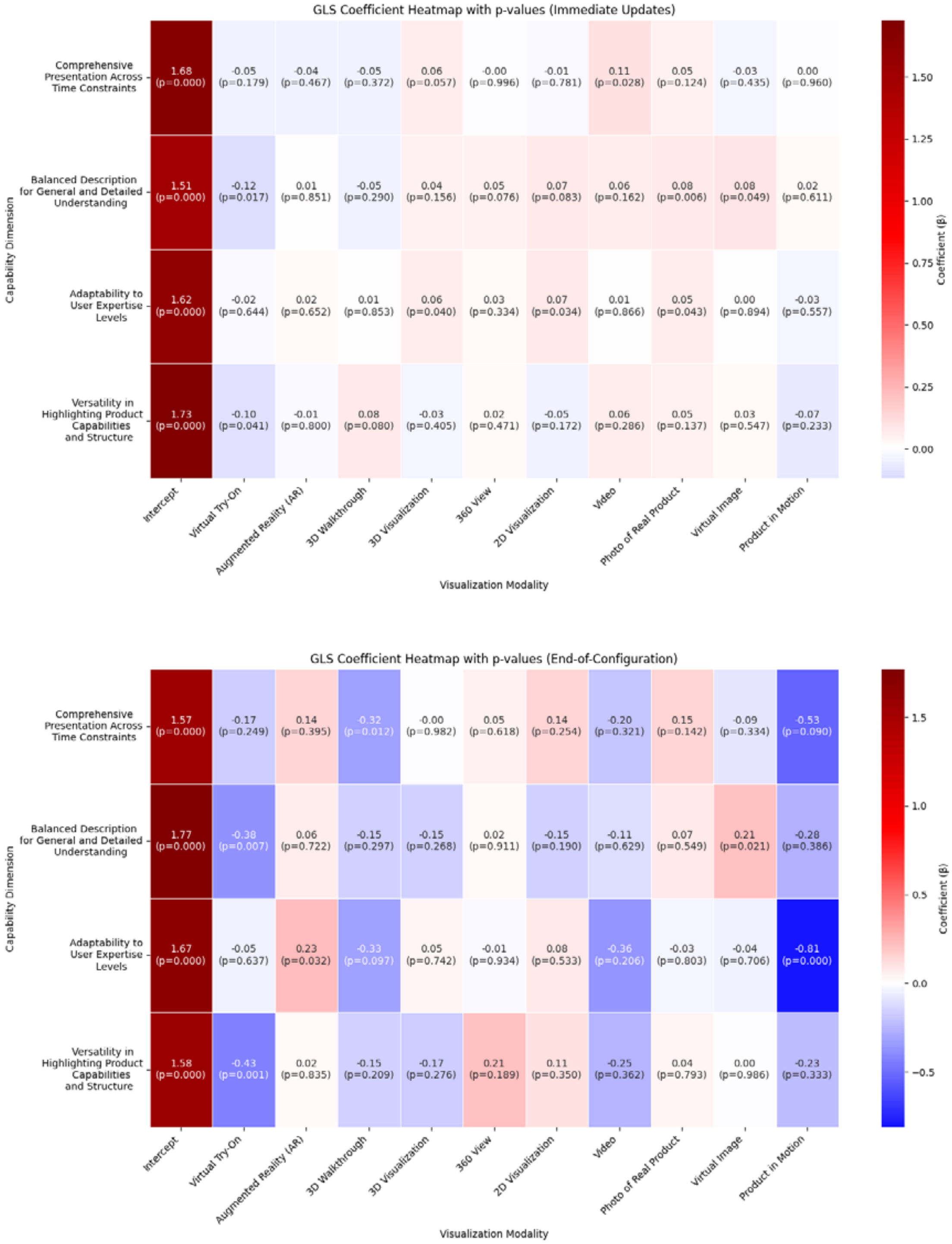


Figure 5: Heatmap of Regression Coefficients For the Two Values of Visualization Timing.

These results are complemented by the regression coefficient plots (Figure 6), which visually represent the statistical significance and direction of effects across modalities.

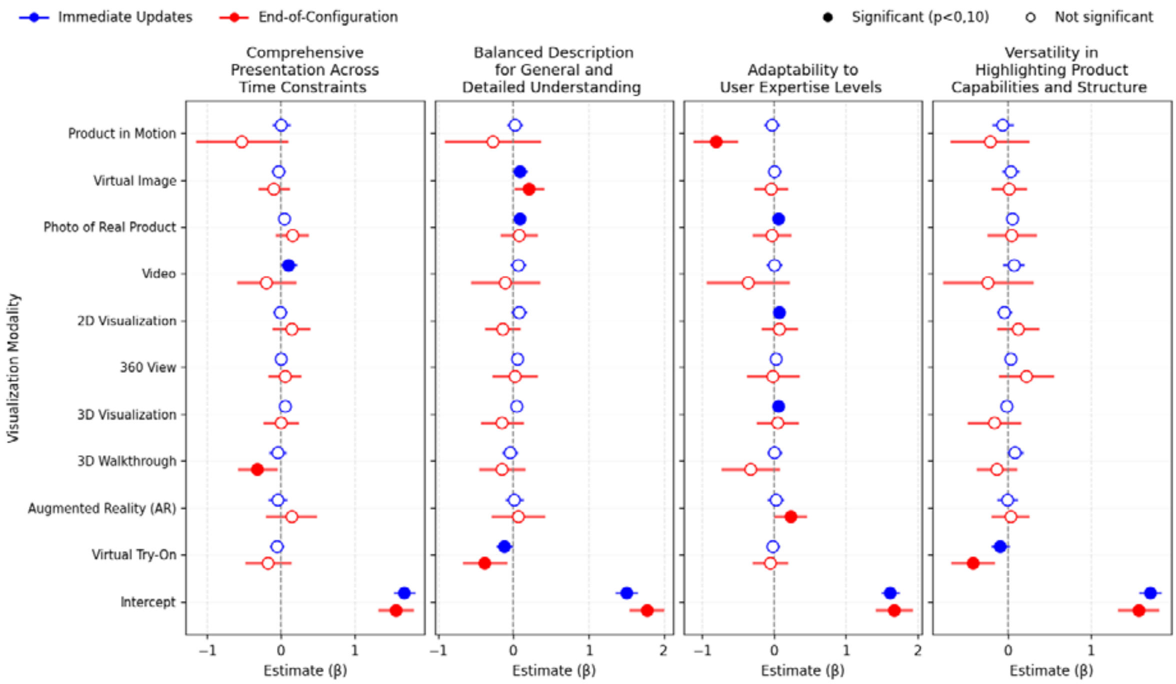


Figure 6: Regression Coefficient Plots For the Two Values of Visualization Timing.

End-of-configuration visualization consistently outperforms real-time updates in explanatory power across all four dimensions. For instance, the *R*-squared value for presentation comprehensiveness across time constraints increases from 0.045 for immediate updates to 0.281 for end-of-configuration visualization (Figure 7).

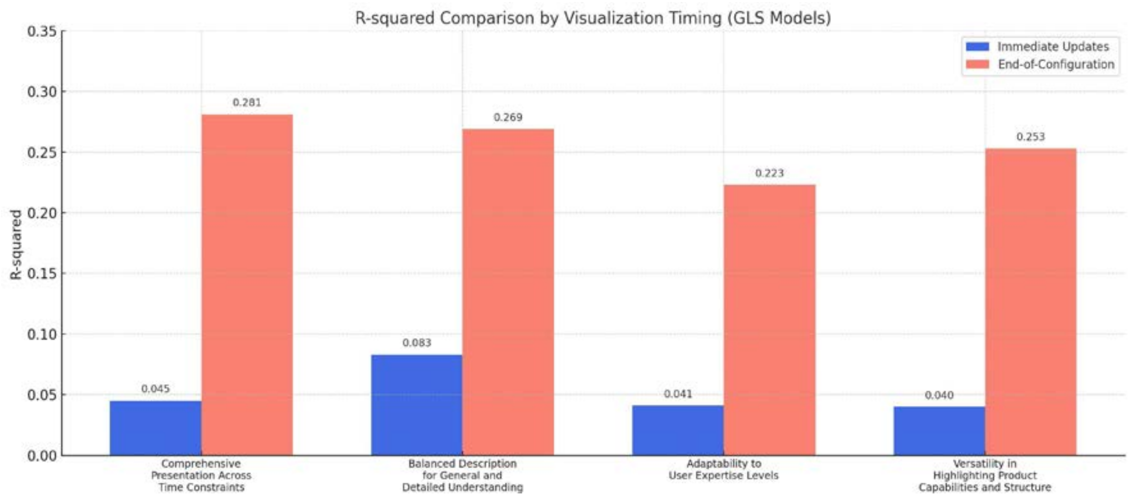


Figure 7: Bar Chart of R-squared for Visualization Timings.

4.4. Gender Influence

Gender-based models reveal notable differences in how visualization modalities impact the configurator capability. These patterns are clearly visualized in the gender-based heatmap (Figure 8), which illustrates the variation in modality effectiveness across gender groups.

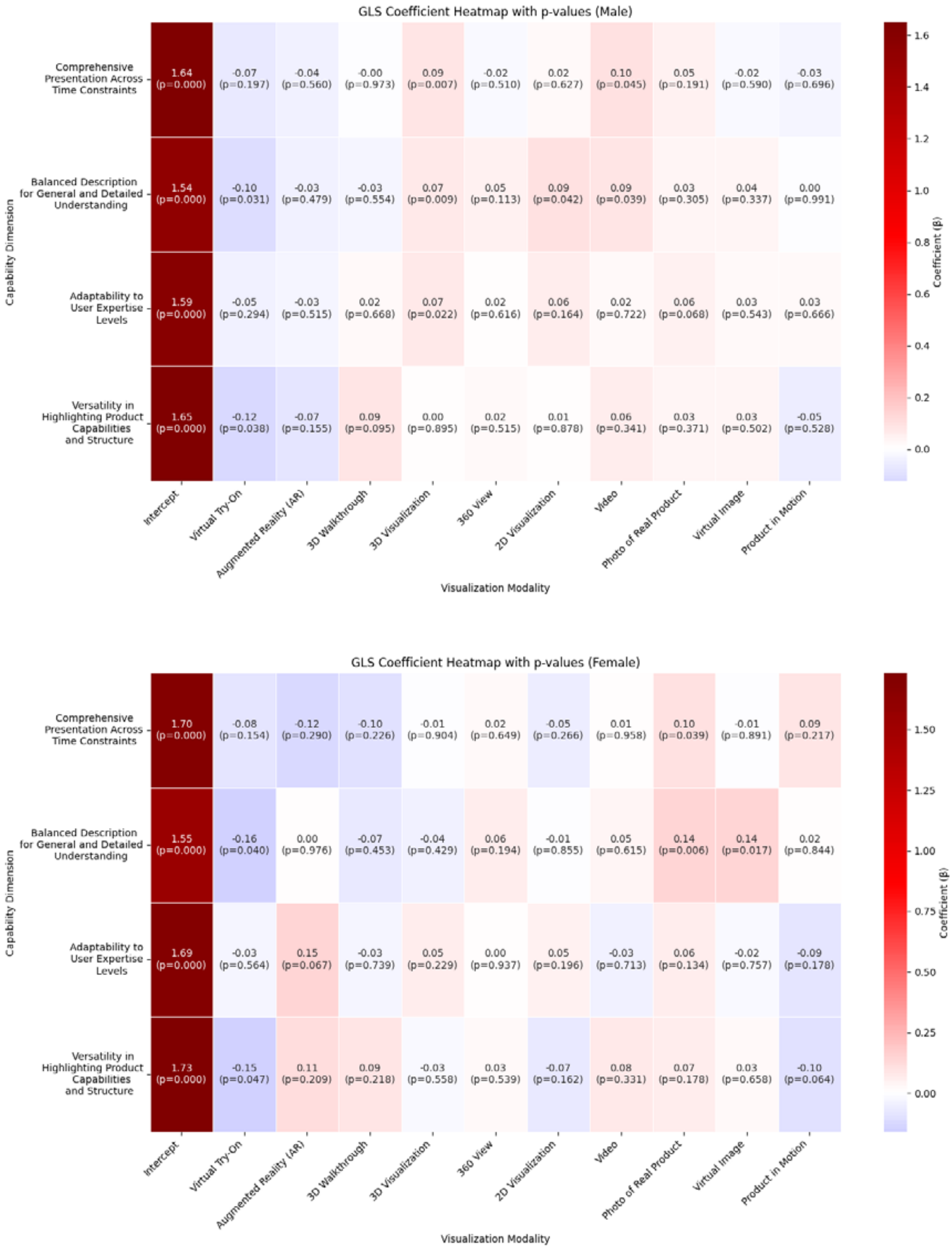


Figure 8: Heatmap of Regression Coefficients for the Two Genders.

These results are complemented by the regression coefficient plot with confidence intervals presented in Figure 9, which visually represent the statistical significance and direction of effects across modalities.

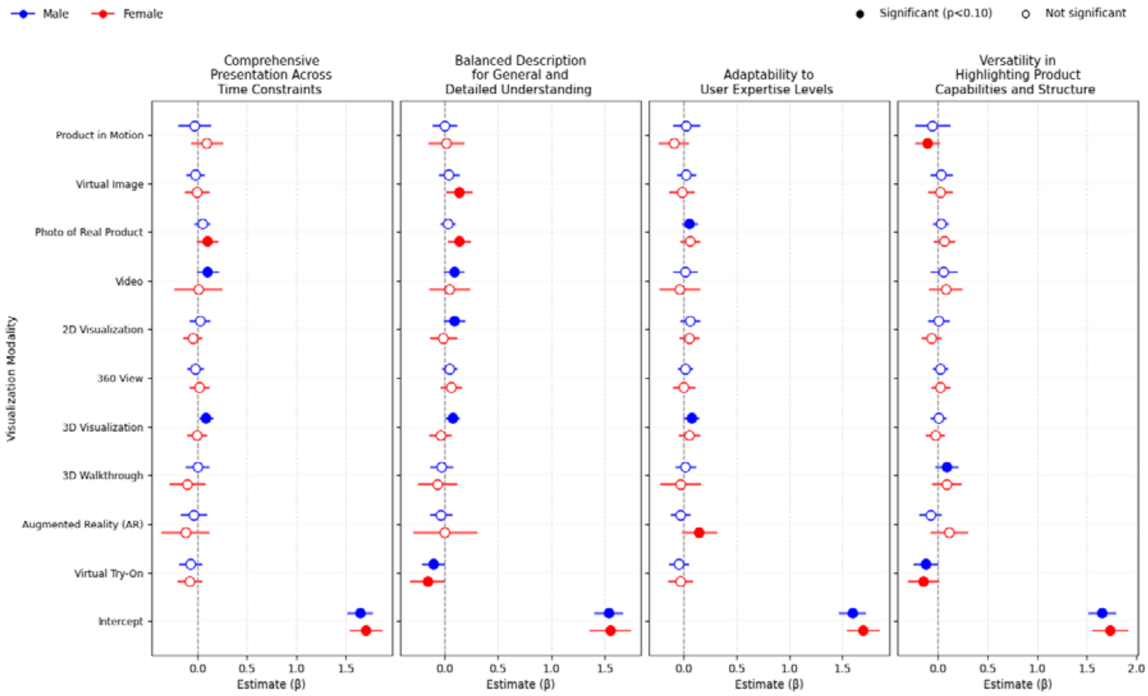


Figure 9: Regression Coefficient Plots for the Two Genders.

Female-specific models yield higher *R*-squared values across all four capability items compared to male-specific models (Figure 10). The largest gap appears in presentation comprehensiveness across time constraints, where the female model achieves an R^2 of 0.091 versus 0.043 for males.

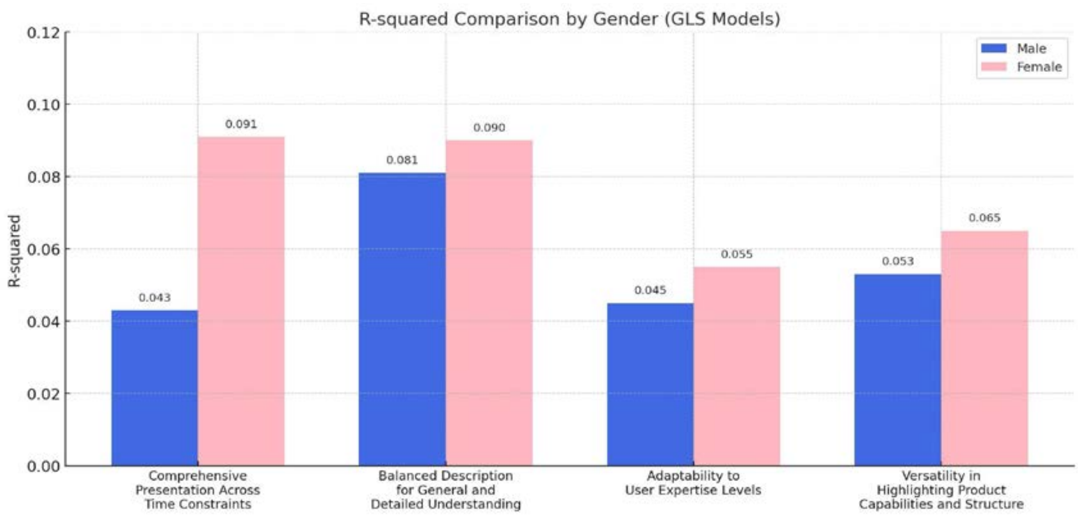


Figure 10: Bar Chart of *R*-Squared Comparison by Gender-Based

Among male users, 3D visualization and video show consistent positive effects across all dimensions. In contrast, female users respond more favorably to photo of the real product and virtual image, particularly as regards comprehensiveness and balanced understanding.

4.5. Product Type Influence

Although not a primary focus of the study, the descriptive analysis indicates that the adoption of visualization modalities varies across product types. The presence of the ten visualization modalities, in percentage of the total number of configurators analyzed in the study, is shown in Figure 11, suggesting that many OSCs adopt multiple modalities.

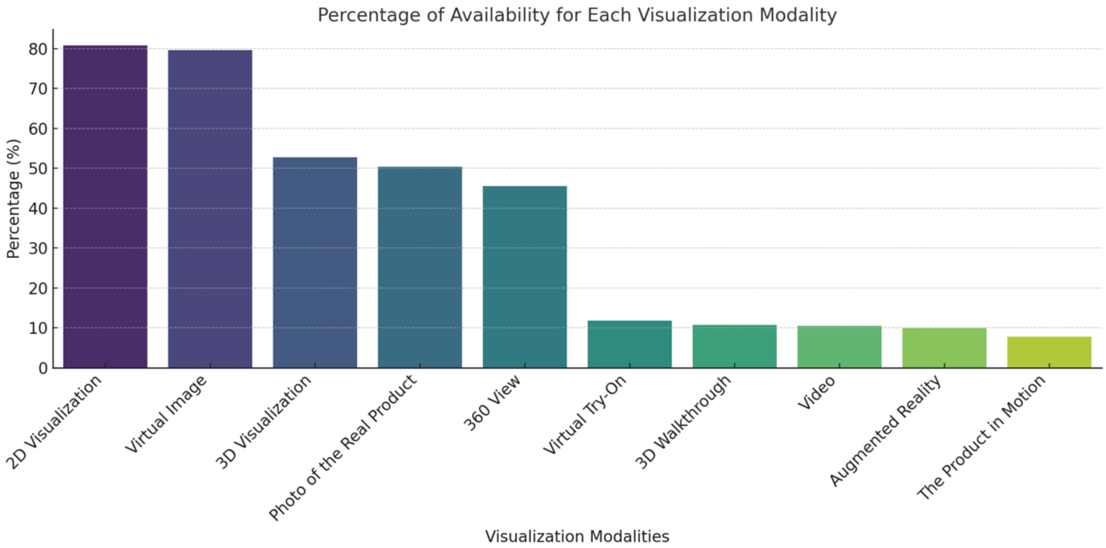


Figure 11: Distribution of Visualization Modalities in Online Sales Configurators.

Building on this, the counts of products across visualization modalities are reported in Figure 12, where products are classified into 14 product categories.

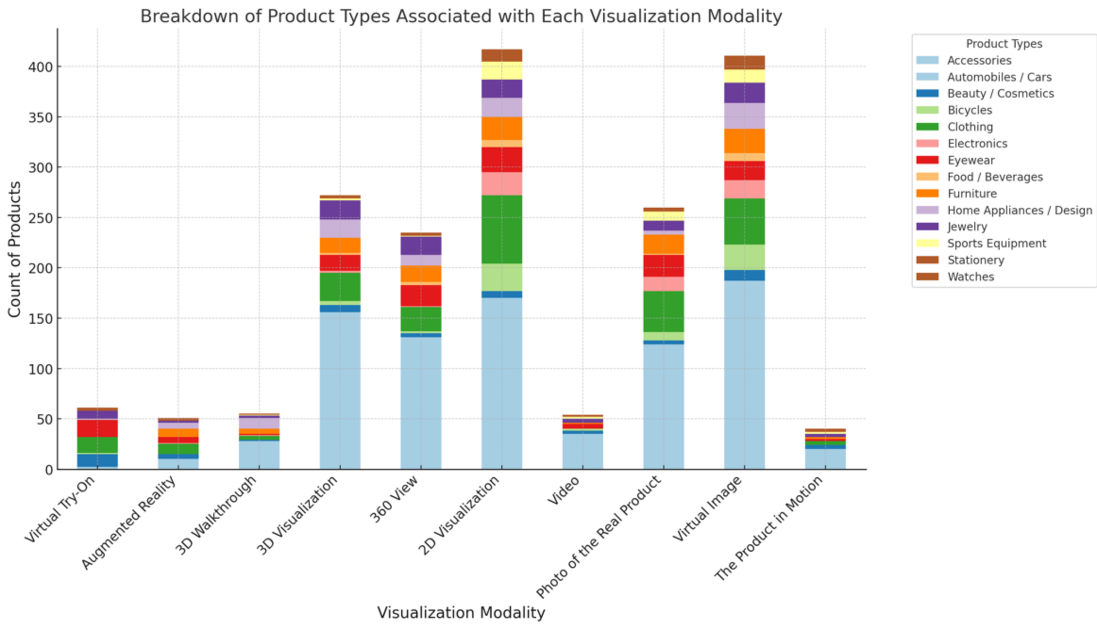


Figure 12: Count of Visualization Modalities Across Product Types.

Finally, Figure 13 emphasizes differences in modality adoption within each product category. The data are normalized per category to highlight the proportion of configurators adopting each modality, allowing comparison of preferred visualization strategies across domains such as beauty, eyewear, automobiles, furniture, and electronics. For instance, virtual try-on shows higher adoption rates in beauty/cosmetics (22.0%) and eyewear (12.7%) configurators, compared to other product types.

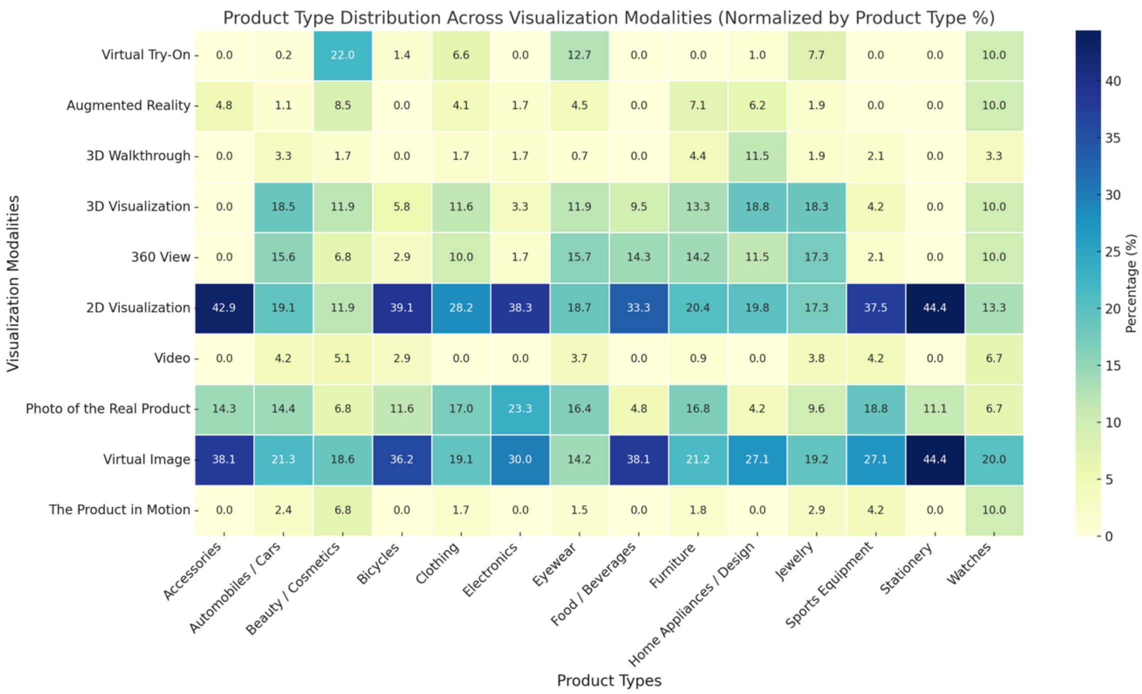


Figure 13: Distribution of Visualization Modalities by Product Type.

5. Discussion

The findings of this study offer nuanced insights into the role of visualization modalities in shaping the user-friendly product space description capability of online sales configurators (OSCs). Rather than pointing to a single optimal solution, the results suggest a complex interaction between visualization modality, visualization timing, user profile, and product context. These dynamics influence how users comprehend, evaluate, and interact with configurable product spaces.

5.1. Effectiveness of Static and Semi-Interactive Formats

The consistent positive associations of 2D visualization, virtual image, photo of the real product, and 3D visualization across all four dimensions of the focal capability underscore the enduring value of clear, easily interpretable visual information. These formats appear to strike a balance between providing sufficient detail and maintaining cognitive ease, aligning with findings from earlier studies [e.g., 17, 18]. Their effectiveness may be attributed to their ability to reduce cognitive load while still offering essential product information, thereby supporting efficient decision-making processes.

5.2. Challenges with Immersive Technologies

The negative effects associated with virtual try-on, particularly for balanced description and versatility in highlighting product capabilities and structure, present an interesting contrast to some previous research that has highlighted the potential benefits of immersive technologies in e-commerce [e.g., 20, 22]. While immersive formats such as AR and VR can enhance user engagement and brand perception, our results suggest that these benefits may not always translate into improved clarity or usability within the context of product configuration. This finding aligns with Blazek's [8] observation that the complexity and novelty of immersive technologies may sometimes disrupt rather than support user decision-making.

5.3. Importance of Visualization Timing

End-of-configuration visualizations significantly outperform real-time updates across all four capability dimensions. The strongest effect is observed for presentation comprehensiveness across time constraints, where explanatory power more than triples when visualization is presented after the user has finalized their selections.

This result aligns with Sandrin and Forza's [6] assertion that visualization timing should be treated as a deliberate design variable rather than a default setting. Their work emphasizes that end-of-configuration visualization allows users to process the configured product as a whole, reducing decision fatigue and supporting clearer interpretation. In contrast, real-time updates, while more dynamic, can fragment the user's cognitive focus, particularly in complex configurations.

5.4. Gender-Based Differences

The observed gender-based differences in visualization effectiveness add to a growing body of literature advocating for user-personalized interfaces. Female users responded more positively to realistic and static formats, such as photos and virtual images. Male users, by contrast, showed stronger alignment with dynamic formats like 3D visualization and video. These results echo the arguments of Yi et al. [9], who emphasize the importance of demographic-sensitive design in digital tools. Moving forward, adaptive configurators that tailor visualization types to individual user profiles may offer significant gains in usability and satisfaction.

5.5. Product Context and Modality Effectiveness

Product-type data point toward important contextual effects. For instance, while virtual try-on showed generally negative associations, its use was concentrated in product categories where physical fit and aesthetics are paramount, such as beauty and eyewear. While this modality demonstrated lower effectiveness overall in supporting user-friendly product space description, its frequent adoption in these contexts may point to a niche functional value that is not adequately captured by generic usability measures. A visualization format that enhances configurator usability in one category may fail to do so in another if it lacks alignment with the product's core interaction features.

6. Conclusion

This study makes several contributions to the literature on mass customization and e-commerce. First, it extends the understanding of user-friendly product-space description in online configurators by empirically examining the impact of various visualization modalities. This addresses a gap in the literature identified by Sandrin and Forza [6], who called for more research on the effectiveness of modern visualization technologies in configurators. Second, our findings on the superiority of end-of-configuration visualization over real-time updates challenge existing assumptions about immediate feedback in digital interfaces. Finally, the observed gender-based differences in visualization preferences contribute to the growing literature on user diversity in digital interfaces. These results highlight the need for more nuanced theoretical models that account for demographic variation in interaction patterns with customization tools.

From a practical standpoint, this study offers valuable suggestions for OSC designers and e-commerce platforms. They suggest that, while advanced visualization technologies offer exciting possibilities, their implementation should be carefully considered in the context of user cognitive processes and preferences. Prioritizing clear, easily interpretable visual formats and providing comprehensive visualization at the end of the configuration process may enhance the overall user experience and decision-making efficiency. Additionally, our results on gender differences in visualization preferences suggest that adaptive interfaces catering to different user groups could significantly enhance the overall user experience and potentially increase conversion rates.

However, it's important to note the limitations of this study. The participants were university students, which may limit generalizability to broader consumer populations. Future research could explore these relationships in more diverse consumer groups and investigate how user preferences and the effectiveness of different visualization modalities evolve over time.

Acknowledgements

The authors thank the University of Padova for funding projects DTG SID 2024 and DOR.

Declaration on Generative AI

During the preparation of this work, the corresponding author used ChatGPT (GPT-4) and Grammarly to check grammar and spelling and to improve writing style. After the use of these tools/services, all the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Krause, F., & Franke, N. (2024). Understanding Consumer Self-Design Abandonment: A Dynamic Perspective. Journal of Marketing, 88 (2), 79-98. doi:10.1177/00222429231183977.

[2] Trentin, A., Perin, E., & Forza, C. (2013). Sales configurator capabilities to avoid the product variety paradox: construct development and validation. Computers in Industry, 64, 436-447. doi: 10.1016/j.compind.2013.02.006.

[3] Sandrin, E., Trentin, A., Grosso, C., & Forza, C. (2017). Enhancing the consumer-perceived benefits of a mass-customized product through its online sales configurator: an empirical examination. Industrial Management & Data Systems, 117, 1295-1315. doi:10.1108/IMDS-05-2016-0185.

[4] Blecker, T., Abdelkafi, N., Kaluza, B., & Friedrich, G. (2004). Product configuration systems: State-of-the-art, conceptualization and extensions. In Proc. 8th Int. Conf. on System Configuration and Integration (pp. 1-10).

[5] Soininen, T., Tiihonen, J., Männistö, T., & Sulonen, R. (1998). Towards a general ontology of configuration. AIEDAM, 12(4), 357–372. doi:10.1017/S0890060400001795.

[6] Sandrin, E., & Forza, C. (2023). Visualization in configurators: Reflections for future research. In Proc. 25th Int. Workshop on Configuration (ConfWS 2023), Málaga, Spain, pp. 8–11.

[7] Petterle, S., Sandrin, E., & Forza, C. (2024). Product visualization in configurators: Laying the foundations for a comparative description. In Proc. 26th Int. Workshop on Configuration (ConfWS 2024), Girona, Spain, pp. 54–63.

[8] Blazek, P. (2023). Creating customization experiences: The evolution of product configurators. In T. Aichner & F. Salvador (Eds.), Mass customization and customer centricity (pp. 179–209). Palgrave Macmillan. doi:10.1007/978-3-031-21756-9_10.

[9] Yi, M., Huang, Z., & Yu, Y. (2022). Creating a sustainable e-commerce environment: The impact of product configurator interaction design on consumer personalized customization experience. Sustainability, 14(23), 15903. doi:10.3390/su142315903.

[10] Walcher, D., & Piller, F. T. (2012). The customization 500: An international benchmark study on mass customization and personalization in consumer e-commerce. Lulu Press.

[11] Piller, F. T. (2004). Mass customization: Reflections on the state of the concept. International Journal of Flexible Manufacturing Systems, 16(4), 313-334. doi:10.1007/s10696-005-7092-7.

[12] Da Silveira, G. J. C. (2011). Our own translation box: Exploring proximity antecedents and performance implications of customer co-design in manufacturing. International Journal of Production Research, 49(13), 3833-3854. doi:10.1080/00207543.2010.492412.

[13] Forza, C., & Salvador, F. (2006). Product information management for mass customization: Connecting customer, front-office and back-office for fast and efficient customization. Palgrave Macmillan.

[14] Grosso, C., & Forza, C. (2018). Users' preferences for social interaction while shopping via online configurators. In Proc. 8th Int. Conf. on Mass Customization and Personalization in Central Europe (MCP-CE 2018).

[15] Trentin, A., Perin, E., & Forza, C. (2014). Increasing the consumer-perceived benefits of a mass-customization experience through sales-configurator capabilities. Computers in Industry, 65, 693-705. doi:10.1016/j.compind.2014.02.004.
[16] Sandrin, E., Trentin, A., Grosso, C., & Forza, C. (2017). Enhancing the consumer-perceived benefits of a mass-customized product through its online sales configurator: An empirical examination. Industrial Management & Data Systems, 117(7), 1295–1315. doi:10.1108/IMDS-05-2016-0185.
[17] Di, W., Sundaresan, N., Piramuthu, R., & Bhardwaj, A. (2014). Is a picture really worth a thousand words? On the role of images in e-commerce. In Proc. 7th ACM Int. Conf. on Web Search and Data Mining (WSDM '14) (pp. 633–642). ACM. doi:10.1145/2556195.2556231.
[18] Ozok, A. A., & Komlodi, A. (2009). Better in 3D? An empirical investigation of user satisfaction and preferences concerning 2D and 3D product representations in B2C e-commerce. International Journal of Human–Computer Interaction, 25(4), 243–281. doi:10.1080/10447310802546724.
[19] Moritz, F. (2010). Potentials of 3D-web-applications in e-commerce: Study about the impact of 3D-product-presentations. In 2010 IEEE/ACIS 9th Int. Conf. on Computer and Information Science (pp. 307–314). IEEE. doi:10.1109/ICIS.2010.82.
[20] Jessen, A., Hilken, T., Chylinski, M., Mahr, D., Heller, J., Keeling, D. I., & De Ruyter, K. (2020). The playground effect: How augmented reality drives creative customer engagement. Journal of Business Research, 116, 85–98. doi:10.1016/j.jbusres.2020.05.005.
[21] Liu, Y., Liu, Y., Xu, S., Cheng, K., Masuko, S., & Tanaka, J. (2020). Comparing VR- and AR-based try-on systems using personalized avatars. Electronics, 9(11), 1814. doi:10.3390/electronics9111814.
[22] Befort, A. (2021). Augmented & Virtual Reality in E-Commerce (Master's thesis, University of Twente). https://essay.utwente.nl/87808/.

Designing for circularity: exploring configurator-based decision support for Eco-design in food packaging*

Chiara Grosso[1,*], Marco Scatto[2] and Karen Venturini[2]

[1] Sapienza University of Rome, Rome 00185, Italy
[2] University of San Marino, San Marino 47890, Republic of San Marino

Abstract
The packaging industry occupies a central position in the sustainability transition, particularly as regulatory frameworks increasingly mandate alignment with circular economy (CE) principles. In the European Union, the upcoming Packaging and Packaging Waste Regulation (PPWR), effective from January 2025, requires all packaging to be either reusable or recyclable in a technically and economically feasible manner. Since the majority of a product's environmental burden is determined during its early design phase, digital tools must evolve beyond conventional parametric modeling to incorporate environmental metrics, material recovery pathways, and lifecycle intelligence. While sustainable packaging design has received growing academic attention, the deployment of AI-based configurators to support eco-design and end-of-life strategies remains underdeveloped. This study investigates the potential of product configurators as intelligent, rule- based decision-support systems capable of embedding CE-aligned design logic in the food packaging sector. Adopting a multi-method empirical approach, combining Multi-Criteria Decision Analysis, Analytical Hierarchy Process, and expert evaluation, the research assesses the relative suitability of reuse, mechanical recycling, chemical recycling, and organic recycling against criteria defined by industry specialists. Furthermore, the study develops a conceptual framework for a next-generation configurator, designed to integrate eco-design principles, modular product architecture, and traceability data within packaging systems. Findings indicate that configurators can be re-engineered to function as intelligent interfaces for operationalizing CE principles in product development workflows. The study highlights modularity, material knowledge, and traceability as critical enablers, providing a roadmap for engineers and practitioners developing CE-compliant packaging configurators.

Keywords
eco-design, configuration systems, sustainable packaging, circular economy, end-of-life strategies, lifecycle-based design [1]

1. Introduction

Packaging has become a critical target for systemic redesign in the transition toward resource-efficient and sustainable production systems. Its ubiquity in consumer markets and its significant contribution to post-consumer waste make it a key intervention point for circular economy (CE) strategies [1]. As environmental concerns intensify and natural resource constraints become more pronounced, both regulatory institutions and industrial actors are seeking to replace linear, single-use models with systems oriented toward reuse, material recovery, and lifecycle closure. In this context, the forthcoming European Packaging and Packaging Waste Regulation (PPWR), entering into force in 2025, introduces mandatory requirements for all packaging placed on the market to be either reusable or recyclable in a technically and economically viable manner [2].

The implementation of CE principles in packaging design necessitates a shift not only in material selection and manufacturing methods but also in the digital systems that support design decision-making. While the academic literature on sustainable packaging has matured considerably, current industrial design tools remain largely focused on performance, branding, and

*ConfWS'25: 27th International Workshop on Configuration, Oct 25--26, 2025, Bologna, Italy
1* Corresponding author.
chiara.grosso@uniroma1.it (C. Grosso); m.scatto@studio.unirsm.sm (M. Scatto); kventurini@unirsm.sm (K. Venturini)
0009-0008-2691-3947 (C. Grosso); 0000-0001-7707-6669 (M. Scatto); 0000-0002-0842-4006 (K. Venturini)

cost metrics, often omitting environmental or end-of-life considerations. As a result, the opportunity to embed lifecycle intelligence and regulatory compliance directly into the design phase remains underexploited.

Digital product configurators, originally developed for managing product variability and enabling mass customization, offer architectural features that could be adapted to meet these new sustainability requirements. These systems are structured around constraint propagation, rule-based validation, and user-guided decision paths, enabling them to integrate environmental data, simulate product usage scenarios, and support design-for-recyclability or reuse [3-4]. However, the vast majority of current configurator applications in the packaging domain are limited to superficial customization of graphics, size, or labeling, and do not address eco-design functions such as modularity, material traceability, or End-of-life (EoL) strategy optimization.

The present study explores the extent to which configurators can be programmed to act as intelligent design systems for CE-aligned packaging. Specifically, the research aims are to (a) identify the most critical sustainability criteria as perceived by packaging professionals, (b) assess the relative performance of key EoL strategies across those criteria, and (c) define the structural and functional requirements for configurators capable of supporting CE-compliant product development. To address these objectives, a two-phase empirical methodology is employed. The first phase applies Multi-Criteria Decision Analysis (MCDA) and the Analytical Hierarchy Process (AHP) to evaluate expert preferences for four EoL pathways: reuse, mechanical recycling, chemical recycling, and organic recycling. The second phase proposes a conceptual framework for configurator functionality based on empirical findings and the evaluation of an existing digital configurator platform used in the food packaging industry.

The implication of the present study aims at contributing both at theoretical and managerial levels. On a theoretical level, it aims to advance the understanding of how design configurators can be aligned with circular economy imperatives. Managerially, it aims to identify the technical enablers, modularity, traceability, and material intelligence that could inform the development of next- generation configurators capable of meeting circular design principles and sustainability demands.

2. Theoretical background

2.1. Circular packaging: opportunities and challenges

Packaging plays an important role in the circular economy. To reduce its environmental impact, it is necessary to review its production processes, materials, and especially its End-of-life [5]. In the circular economy model, the revision to a sustainable perspective should be done at the initial design stage in which the decisions about materials and their reuse or End-of-life can be made.

In the product design phase, products are developed to be reused or returned to the environment in a way that benefits ecosystems and eliminates waste [6]. An eco-design perspective assesses the environmental impact of a product throughout its life cycle. According to the principles of circular and waste management [7], there are five possible actions to decrease the pollution that packaging creates: I) reject the use of packaging, II) rethink the concept of packaging to achieve a more sustainable option, III) reduce the use of material and energy used to produce packaging, IV) reuse packaging, and V) recycle packaging materials.

Circular economy (CE) models focus on reducing waste and extending the life of products through recycling, reuse, and remanufacturing practices [2]. A business model describes business operations, and the values connected and how the organization achieves them. It is important to define precisely how values are achieved in closed material chains [8]. The application of circular business models requires interventions in design, production, optimal product utilization, waste reduction, product life extension, and increased resource efficiency [9]. On the other hand, in traditional supply chains, for example in food, production processes typically follow a linear model, where products are created, consumed, and ultimately discarded at the end of their life cycle. In this scenario, packaging is often viewed merely as a means to protect and transport the product, representing the endpoint of its lifecycle. However, in the context of CE, packaging is recognized as a valuable product rather than viewing packaging solely as the end of processing or waste. In other words, within the CE framework, packaging consists as an integral component of the product life cycle. Moreover, the end- of-life phase includes the design of packaging, promoting practices such as reuse, recycling, and recovery, thereby extending its value and reducing overall waste. Consequently, the transition from a linear to a circular economy not only redefines the role of packaging but also enhances sustainability. For instance, the application of CE principles within food supply chains is significantly improving sustainability and resource efficiency, especially toward packaging. Many companies are adopting bio-degradable and compostable packaging, reducing environmental impact and reliance on fossil fuels. This fosters a more sustainable approach aligned with CE, contributing to environmental preservation and sustainable consumption. The deposit return system (DRS) is an example of reuse in packaging where a selective collection system mainly used for single-use beverage packaging. It works by charging the consumer a small refundable deposit, in addition to the purchase price of the product. This deposit is fully refunded when the consumer returns the empty packaging—such as bottles or cans—to a designated collection point, often available at retailers. This encourages consumers to return the packaging, helping to increase recycling rates substantially. This model is primarily used for materials like glass, plastic (notably PET), and aluminum containers for beverages.

Brands like Coca-Cola and Unilever use recycled materials, closing the loop on waste. Innovative solutions like edible packaging and reusable packaging systems are also emerging, further minimizing waste and optimizing material life cycles in food packaging [10]. In this context, biodegradable materials are often highlighted for their ability to decompose naturally, thus reducing waste and environmental impact. Biodegradable packaging refers to materials that can break down into natural substances, such as carbon dioxide, water, and biomass, through the action of microorganisms within a specified period of time. Examples of biodegradable packaging include plant-based plastics made from renewable resources like starch and polylactic acid (PLA), as well as traditional materials such as paper and organic fibers. These materials decompose through natural processes, making them more environmentally friendly compared to conventional petroleum-based plastics. The use of biodegradable packaging contributes to a closed- loop system by redirecting waste away from landfills, where it would otherwise contribute to methane emissions under anaerobic conditions. Instead, when disposed of in composting facilities or through home composting, biodegradable packaging can be converted into valuable compost, enriching soil and supporting plant growth while minimizing the overall carbon footprint associated with packaging waste [11]. This means that while biodegradable options are valuable, they should be part of a strategy that includes innovative recycling processes and efficient resource management to achieve true sustainability in packaging solutions [12]. Whether biodegradable packaging can play a crucial role in reducing landfill waste, it is essential to ensure that these materials are integrated into a broader system that prioritizes circularity. For instance, some biodegradable plastics may not fully degrade in home composting conditions, which typically operate at lower temperatures. For that reason, CE maximizes packaging material lifespan and minimizes virgin resource extraction through reuse, recycling, and recovery, going beyond simple biodegradability.

A specific role within the discussion of renewable materials and their lifecycle impacts is played by compostable materials. In packaging, these materials, which can be produced from biomass such as starch, cellulose, or renewable polymers, are intended to substitute for non-biodegradable plastics. They are specifically designed to biodegrade in a composting environment, turning into carbon dioxide, water, and biomass within a certain time frame without releasing toxins and for contributing to a sustainable nutrient cycle. They must meet specific standards (e.g., ASTM D6400 or EN 13432) that ensure they break down quickly and do not leave harmful residues. They must be also suitable for biological waste treatment through industrial or home composting systems. In brief, the use of compostable packaging is increasing due to the demand for eco-friendly solutions, promoting waste reduction and circular economy practices. However, certification and adequate composting infrastructure are essential for proper management of compostable waste [13]. Lastly, recyclable packaging refers to materials that can be processed and reused to create new products after their initial use, rather than being disposed of as waste. This type of packaging is designed in a way that allows it to be collected, sorted, and processed into raw materials for manufacturing new items. The recyclability of packaging depends on various factors including the material type, the presence of contaminants, and the local recycling facilities available to handle those materials [14]. Particularly, recycled materials come from post-consumer waste that has been processed and re-manufactured into new products. The recycling process helps reduce the demand for virgin resources and can mitigate environmental impacts compared to producing new materials from raw inputs. However, the recycling rates for many types of plastics remain low due to contamination and the complexity of different polymer types.

While compostable and biodegradable materials offer alternatives to traditional plastics by providing options for waste treatment, recycled materials contribute to sustainability by reusing existing materials. Each type has its own advantages and challenges, and the choice often depends on specific application requirements and waste management systems in place. Accordingly, companies face the challenge in designing effective Reverse Logistics (RL) systems, crucial for transitioning to a circular economy.

Although the literature has produced a variety of digital methodologies and decision-support tools to address CE principles, particularly in the domain of end-of-life management, there remains a gap in the integration of these principles within the early-stage design processes for packaging systems. A notable contribution in this regard is the Reverse Logistics Support Tool (RLST), recently proposed by Mallick et al. [15], which assists firms in evaluating strategic motivations, contextual product characteristics, regulatory conditions, and system design variables to comply with CE principles. The RLST framework puts key CE considerations into practice by incorporating variables such as stakeholder engagement, digital technologies, and consumer behavior. Although RLTS provides substantial guidance for downstream operations, particularly the recovery, sorting, and treatment of post-consumer packaging, its functionality in guiding upstream decision-making remains limited. Specifically, the RLST tool do not embed lifecycle-aware intelligence into the design phase, where up to 80% of a product's environmental footprint is determined [8]. Furthermore, they lack the capacity to support modular design thinking, traceability integration, and user co-design all of which are increasingly recognized as core enablers of circularity in packaging [7]. In this context, there is a critical need for digital tools implemented with systems that bridge CE- compliant logic directly into product development workflows. The present paper conceptualize that such systems could be extended beyond conventional rule-based configurators used in mass customization, evolving into intelligent, interactive platforms that simulate EoL pathways, assess regulatory compatibility, and support sustainable material selection at the point of packaging design to align design processes with environmental compliance requirements, consumer expectations, and circular material flows from inception.

2.2. Digital Configurators for Circular Co-Design

In the context of sustainable innovation and circular economy (CE), the implementation of digital configurators has emerged as a critical enabler for integrating co-design methodologies, stakeholder engagement, and consumer-centered innovation. Prior research has demonstrated that sustainability goals are more effectively achieved through multi-actor collaboration, particularly when stakeholders contribute distinct competencies and sectoral perspectives to a shared design process [16-17]. Digital tools serve not only as technological platforms but also as relational infrastructures that support resource-sharing, joint knowledge production, and circular-oriented innovation.

In this regard, consumer participation plays a crucial role, not only during the consumption phase but throughout the entire product lifecycle. The acceptance and active involvement of consumers are prerequisites for extending product longevity and maximizing material circularity [18]. Indeed, consumer co-design contributes to aligning product functionality with user values, facilitating behavioral change, and ultimately enhancing sustainability outcomes. Previous research has increasingly focused on how consumer perceptions of circularity influence both purchase and disposal behavior, highlighting the urgency of integrating behavioral science into lifecycle assessments [19].

Mass customization (MC), as a production paradigm, provides a powerful mechanism for aligning individual consumer needs with scalable and sustainable production systems. When embedded within a CE framework, MC enables firms, particularly SMEs, to reduce material waste, enhance resource efficiency, and stimulate demand for eco-designed products [20]. The process of co- innovation, whereby customers and producers co-create value, is significantly amplified by digital configurators that facilitate real-time feedback loops and participatory design choices [21-22].

Online Sales Configurators (OSCs), in particular, represent a mature instantiation of this model. They function as knowledge-intensive systems designed to support product development, delivery, and personalization [23-24]. OSCs reduce cognitive load during the decision-making process by guiding users through structured choice navigation paths, thereby minimizing complexity while preserving design freedom. By managing constraints, validating configurations, and simulating outcomes, these systems create highly engaging user experiences that are simultaneously efficient and satisfying [25- 26].

Specifically, when OSCs would be designed to support sustainable product attributes, they could significantly increase customer willingness to pay a premium for customized products [27]. This effect is amplified when configurators enable users to understand and visualize the environmental value embedded in their design decisions, fostering both individual agency and systemic alignment with circularity goals. Recent studies have confirmed that rewarding customization experiences not only increase perceived product value but also positively influence repurchase intentions and long-term brand loyalty [28]. In addition to enhancing user experience, configurators have demonstrated operational benefits for manufacturers. These include shorter lead times, reduction in design errors, improved product–market fit, and lower material consumption. From a system engineering perspective, configurators contribute to the optimization of product architectures and supply chain coordination, particularly in modular and variant rich environments [29-30]. When integrated with circular metrics, such as lifecycle data, recyclability indexes, and traceability modules, configurators evolve into adaptive platforms capable of orchestrating design choices that comply with environmental regulations and sustainability standards.

3. Methodology

The research design of the present study is structured on a multi-phase empirical research design. To address research aims (a) and (b), the investigation employs a combined Multi-Criteria Decision Analysis (MCDA) and Analytical Hierarchy Process (AHP) framework. Grounded on previous research [31-32] the adopted approach enables a quantitative evaluation of the relative importance of sustainability criteria and preferences for different EoL strategies by industry experts. Specifically, MCDA provides a prioritization of design criteria in terms of environmental, economic, social, and technical relevance, while AHP [33] facilitates the pairwise comparison of EoL alternatives, specifically reuse, mechanical recycling, chemical recycling, and organic recycling, against those criteria. To address the research objective (c) , the study proposes a conceptual framework for the development of a CE-compliant product configurator. The framework is grounded in existing literature on circular product design, eco-design regulation, and digital configuration systems, and is empirically informed by a test on a real case study analysis of Packstyle, a company operating in the food packaging sector. Packstyle (the real name) is a SME (Small Medium Enterprise) expert in online mass customization (Web-to-print). The idea came from the request to have customized flexible packaging for small runs. In this sector, the traditional machines have high operating costs and work only on large orders of food brands or manufacturers. Packstyle was created to satisfy a new niche market, that of small businesses that need packaging for their products but demand limited runs and in a very short time. The driving force came from the innovation culture of the company and the availability of one of its largest suppliers who had a machine to do experimentation on digital printing in flexible materials. The case is used to identify limitations in current customization platforms and to derive functional requirements for a next-generation configurator designed to align with CE principles.

As the packaging industry transitions toward compliance with upcoming EU directives such as the Packaging and Packaging Waste Regulation (PPWR) [2], the role of configurators is set to expand beyond customization. Their potential lies in supporting modular, traceable, and regulation- compliant packaging systems, thereby aligning technological innovation with circular economy imperatives. By embedding sustainability constraints and consumer-driven logic into digital design workflows, configurators can bridge the gap between technical feasibility and behavioral adoption, paving the way for more resilient and circular production ecosystems.

3.1. Multicriteria Decision Analysis

The MCDA is a decision-making method crucial when multiple criteria has to be considered since it enables a better valorization of multiple points of views (e.g. from stakeholders, experts, respondents) [33]. For the purpose of this study, we present the pre-test phase of the MCDA and AHP methodologies. This testing stage is preparatory, serving to refine the methodological design and validate it for subsequent development. As detailed in Table 1, this phase involved a sample of experts from the packaging sector holding high specialized expertise and professional roles.

On an initial step the industry expert was asked to perform the Multi Criteria Decision Analysis Which consist of an iterative pairwise comparison of 10 criteria on Eco-design sustainability (e.g. C1 versus C2 versus C3). The resulting matrix provides the average of experts' evaluations which identify a classification of the 10 criteria from the most relevant to the least one in terms of Eco-design for packaging and EOL strategies. A second step of the MCDA is experts' evaluation of the four alternatives of EoL strategies, namely: reuse, mechanical recycling, chemical recycling and organic recycling, to evaluate the preferred end-of-life strategies to be implemented by the company in responding to Eco design and circular regulatory requirements. The alternatives of EoL strategies are evaluated by the industry expert on a 10-point value scale as an extension of the Likert scale (1 to 5) to provide a more complete, industry centered perspective. The testing of the research design is performed by an expert from Packstyle with high specialized knowledge, experience and high-level responsible role in the packaging industry. The multilevel knowhow of the expert enables a unique opportunity to test the robustness of the empirical research design of the present study. While the MCDA step aims at identifying the sustainability criteria relevant for Eco design in the food packaging industry, the second step of AHP completed the scenario with the strategic perspective on EoL strategy.

Table 1
Experts' profiles

Expert	Role	Core Specialization	Key Sector
Expert id_NG	CEO, Packstyle	Business Strategy & Regulatory Compliance	Flexible Packaging (Food/Cosmetics)
Expert id_LG	Head of R&D, Plastigraf Trevigiana	Materials Engineering	Luxury & Cellulose-based Packaging
Expert id_SD	Researcher & Professor	Food Science & Eco-design	Academic Research & Food Innovation
Expert id_MS	Polymer Scientist	Macromolecular Science	Advanced Manufacturing (Aerospace/Medical)

3.2. Selection of the criteria and alternatives

For the purpose of the present study 10 criteria (C1-C10) were selected to include the sustainability dimensions of packaging. Grounded on previous study [31-32, 34]. Criteria were selected to represent the sustainability in terms of technological, environmental, social economic and transferal dimensions of sustainability, Table 2 reports a synthesis of the 10 criteria described in the following.

C1 Green production process refers to the adoption of technologies and machinery that allow for waste reduction, consumption reduction and productivity increase with the same resources. *C2 Durability* refers to the increase in the life cycle of the product (packaging). *C3 Green practices for End-of-life (EOL):* refers to the reuse of packaging materials in new products, the use of packaging multiple times for the same or different purposes and the biological decomposition of organic packaging materials (e.g. bioplastics and paper) into compost. *C4 Modularity* refers to the possibility of designing packaging with standardized elements that can be combined with each other in order to optimize space, resources and functionality along the entire supply chain (production, logistics, display and disposal). *C5 Eco-label reputation* (how a company is perceived with respect to its environmental commitment and sustainability). *C6 Social sustainability:* refers to how an organization, company or community promotes equity, well-being, human rights and social cohesion in the present and for future generations. *C7 Green Premium Price* refers to the additional price that the buyer (company/end consumer) is willing to pay for a sustainable product or service compared to a traditional, less environmentally friendly one. *C8 Green logistics optimization* refers to the set of strategies and solutions to optimize logistics (transport, storage, distribution) in order to reduce the environmental impact along the entire supply chain. *C9 Material knowledge* refers to the knowledge of innovative and sustainable materials. *C10 Traceability* refers to the importance of detecting, recording and tracking all information related to the path and history of the packaging.

Table 2
Synthesis of the criteria

Dimension	Criterion ID	Criterion Label
Technological	C1	Green production process
Technological	C2	Durability
Environmental	C3	Green practice for EoL
Environmental	C4	Modularity
Social	C5	Green brand reputation
Social	C6	Social sustainability
Economic	C7	Green premium
Economic	C8	Green logistic optimization
Transversals	C9	Materials Knowledge
Transversal	C10	Traceability

The alternatives of EOL strategies are identified in accordance with previous research [34] with the Packaging and Packaging Waste Regulation (PPWR) and the scope of the present study. Specifically, reuse refers to the EoL practices that enable the reuse of the packaging. Recycling is considered in both its options of mechanical and chemical processes of packaging breaking down.

Table 3 provides a summary of EoL practices.

Table 3
Alternatives of End-of life strategies

ID	EOL strategies	Description
A1	Reuse	Upcycling
A2	Mechanical Recycling	To mechanically break down of packaging
A3	Chemical Recycling	To chemically break down packaging
A4	Organic Recycling	Compostable packaging

To evaluate the quality of the answers provided by the expert, answers are evaluated using a "consistency ratio" indicator which measures the consistency within the set of answers of each expert. As its name says, the "consistency ratio" indicates whether the evaluations provided by a respondent are consistent with the entire set of his/her answers. The maximum threshold of the consistency value is 0.10. If the value is 0.10, the evaluation provided by the responded result is inconsistent and not valid. AHP step. Ten-point scale, as an extension of the Likert scale (1 to 5) is adopted to provide a more completed scenario of expert's evaluation on the EoL alternatives.

4. Results

In the MCDA phase, expert responses were evaluated using a consistency ratio threshold of 0.10 to ensure internal coherence. Criteria exceeding this threshold were excluded from the analysis. The evaluation of ten sustainability criteria revealed three dimensions with the highest average weights: modularity (0.30), materials knowledge (0.22), and traceability (0.15) (Tab. 3). These dimensions are considered, by the expert, as fundamental to the development of CE-aligned packaging configurations. Modularity was recognized for its role in enabling design-for-reuse and disassembly. Materials knowledge was highlighted as critical to determining recyclability, contamination risk, and material compatibility. Traceability was identified as a cross-functional enabler that supports regulatory compliance and facilitates transparent material flows (Table 4)

Table 4
Relevance of Criteria for sustainable Eco-design in food packaging

Dimensions	ID	Criterion Label		Score per dimension
Technological	C1	Green production process	0.06	0.17
	C2	Durability	0.11	
Environmental	C3	Green practice for EoL	0.27	0.30
	C4	Modularity	0.03	
Social	C5	Green Brand reputation	0.02	0.06
	C6	Social sustainability	0.04	
Economic	C7	Green premium	0.02	0.10
	C8	Green logistic optimization	0.09	
Transversals	C9	Materials Knowledge	0.22	0.37
	C10	Traceability	0.15	

After identifying the criteria with the highest and lowest weights for measuring end-of-life (EOL) performance, the empirical analysis proceeds by analyzing the values assigned by the company to the proposed alternatives of EoL namely: reuse, mechanical (MEC), chemical (CHEM) and organic (ORGC) recycling (Table 5).

From the AHP analysis of EOL strategies, all four pathways—reuse, mechanical recycling, chemical recycling, and organic recycling—received nearly equivalent aggregate scores (ranging from 55 to 57 out of 100). However, a more granular interpretation reveals a differentiated profile for each alternative.

Chemical recycling, although technologically complex and economically burdensome, scored highest in traceability (score: 10) and modularity (score: 10), indicating that its potential lies in managing heterogeneous material streams with high fidelity. Conversely, reuse strategies were most positively evaluated in terms of social sustainability (score: 8) and green premium acceptance (score: 9), illustrating their alignment with consumer values and market positioning. Remarkably, organic recycling, though often lauded for its biodegradability, performed poorly in terms of materials knowledge (score: 2) and traceability (score: 9), raising concerns about its compatibility with current data systems and compositional verification methods. Mechanical recycling, widely considered the most mature EOL solution, scored well across green logistics, green production, and durability but was not dominant in any single criterion.

Table 5
Relevance of EoL strategy in food packaging

		End-of-life alternatives	**Recycling**		
Criteria	ID	Reuse (A1)	MEC (A2)	CHEM (A3)	ORGC (A4)
Green production process	C1	5	7	5	7
Durability	C2	2	6	9	10
Green practice for EoL	C3	1	1	1	1
Modularity	C4	6	4	10	5
Green Brand reputation	C5	10	10	7	8
Social sustainability	C6	8	8	6	6
Green premium	C7	9	9	4	3
Green logistic optimization	C8	7	5	3	4
Materials Knowledge	C9	3	2	2	2
Traceability	C10	4	3	10	9
Tot		55	55	57	55

Taken together, these findings substantiate a multi-dimensional view of circularity where no single EOL strategy is inherently superior. Rather, the configurator must operate as a decision-support system capable of matching design choices to context-sensitive sustainability parameters.

5. Discussion

The empirical results of this study provide preliminary insights into how digital configurators may function as strategic enablers for the circular transition for a company operating in food packaging. The double evaluation steps, based on Multi-Criteria Decision Analysis (MCDA) and the Analytical Hierarchy Process (AHP), yields a structured classification of sustainability criteria and end-of-life (EOL) preferences grounded on company perspective from a specialized industrial stakeholder (Packstyle). Based on the achieved results, this section outlines a conceptual extension of the Packstyle online customization system into a next-generation digital product configurator designed specifically for Eco-design requirements and circular (CE) compliance. The conceptual framework is provided by merging results from expert evaluation with the current state of the art of Packstyle packaging configurator.

5.1. Regulatory framework and gaps in food packaging configurators

Currently, Packstyle offers customizable flexible packaging through its online platform, enabling users to configure the visual identity, dimensions, and accessory options of pouches such as doypacks, flat pouches, and pillow bags. However, the existing system is oriented primarily toward graphical personalization and format selection, without integrated support for environmental criteria, end-of- life (EOL) strategies, or sustainability logic. Packstyle configurator it is still needed to be programmed accordingly to implement those features necessary for aligning with the European Union's regulatory framework on sustainable packaging design. Particularly, within the context of the upcoming Packaging and Packaging Waste Regulation (PPWR) [2] and the Food Contact Materials (FCM) Regulation (EU) No. 10/2011. These normative instruments require that packaging be designed for recyclability, traceability, and food safety across the entire lifecycle, including end-of-life (EOL) processing. Accordingly, several functional and structural gaps can be identified in the present system.

First, the platform does not currently integrate material traceability logic. Traceability is not only a prerequisite for compliance with Regulation (EU) No. 1935/2004, on materials and articles intended to come into contact with food, but also a strategic enabler of circularity. For Packstyle to align with both the FCM framework and forthcoming digital product passport (DPP) mandates under the EU Green Deal, each configured pouch should be linked to a unique identifier that includes information about the base polymer, barrier layers, adhesives, inks, and functional additives. This would also ensure compatibility with audits by food safety authorities and recyclers, particularly where multilayer or metallized materials are involved.

Second, the current system of tools lacks eco-design integration [15], (i.e., real-time feedback mechanisms that inform the user about the recyclability class, reuse potential, or contamination risk of the selected configuration). This omission prevents designers and users from evaluating whether their pouch aligns with Article 6 of the proposed PPWR, which requires packaging to be "designed for recycling" according to harmonized criteria. Incorporating an LCA-based scoring mechanism or recyclability simulation module, based on CEPI or RecyClass guidelines would allow the configurator to dynamically assess whether a given pouch meets threshold recyclability requirements (e.g., >90% mono-material by mass).

Another major gap to be addressed is the absence of modularity assessment, which is essential to enable reuse or material separation in post-consumption phases. Although Packstyle provides accessory options like spouts, valves, and zippers, the configurator does not indicate whether these additions impact recyclability, nor does it recommend alternative combinations that would facilitate disassembly or reuse. To support circular design principles, the system should incorporate disassembly scoring and suggest configurations where pouch and closure are made from the same polymer family, in line with CEN/TC 261/SC 4 standards for material compatibility.

5.2. Conceptualizing a next generation of circular food pouch configurator

Although Packstyle already offers a robust customization platform, transforming it into a CE-compliant and regulation-ready configurator requires embedding traceability, disassembly planning, EOL simulation, and food safety compliance directly into the design logic. Such an extension would not only ensure alignment with EU directives and Italian consortia (e.g., CONAI, COREPLA), but also position Packstyle at the forefront of digital eco-design innovation in flexible packaging.

To coherently implement these improvements, Packstyle's configurator (in advance for shortness: *P Configurator*) should evolve into a multi-layered digital decision-support system, where regulatory logic, material specification, and design modularity converge. Technically, this may involve integration with LCA platforms (e.g., OpenLCA), real-time material libraries (e.g., Matmatch, Granta MI), and compliance databases (e.g., EU FCM Positive List). Functionally, the interface should provide real-time validation of circularity metrics, simulate material degradation under expected use conditions, and visualize compliance risks through accessible dashboards.

The envisioned new circular P Configurator would be designed to integrate some interrelated features proposed in the following.

(a) Future for user engagement with a 3D visualization module that reflects real-time configuration changes, such as pouch type, closure system, or material choice. This feature builds directly on upgrading P current configurator, which already enables high- resolution previews of packaging formats. However, in the proposed circular configurator, the visualization would be dynamically linked to environmental metadata, such as recyclability indicators and carbon footprint metrics. This approach addresses a key limitation in current P configurator, which rarely incorporate material references into visual engaging representation.
(b) To support EoL decision-making, the next P configurator would include a strategic selector that evaluates the four EoL alternatives: reuse, mechanical recycling, chemical recycling, and organic recycling. Based on the materials selected, contamination risks, and local infrastructural constraints each option is algorithmically assessed against user-defined priorities, including regulatory compliance (e.g., EU PPWR), supply chain conditions, and intended market geography. This allows the next P configurator to simulate alternative scenarios and recommend design configurations that maximize the compatibility between the final configuration and the EoL strategy preferred. Materials such as polyethylene (PE), used widely by Packstyle in its mono-material recyclable films, are well- suited for mechanical recycling. In contrast, multilayer composite structures, while offering superior barrier properties, may require more advanced processing such as chemical depolymerization or solvolysis [14].
(c) Another core future would be the integration of traceability logic, operationalized through a digital passport that records component origin, recyclability classification, batch data, and additive presence. This feature directly supports regulatory goals under the European Green Deal and Digital Product Passport (DPP). By assigning a persistent digital identifier to each configuration, the configurator ensures that packaging complies with traceability mandates and facilitates reverse logistics. In this respect, the proposed system aligns with recent industry efforts to harmonize digital and physical product identities [35].
(d) Furthermore, a future that includes a modularity and disassembly module that enables users to simulate separability and detachment logic based on the selected closures, materials, and sealing methods. For instance, the inclusion of degassing valves or spouts which are options currently available in P catalog, could trigger a disassembly analysis that evaluates recyclability trade-offs, energy input for separation, and potential contamination vectors. The integration of features that support this decision-making possibility would be crucial for generating realistic circular products designed-for-disassembly.

Another feature would be a real time performance dashboard that provides real-time feedback through weighted indicators derived from the MCDA, allowing users to observe how different configurations perform across multiple sustainability dimensions. Rather than offering binary "valid/invalid" judgments, the interface could present clear trade-off visualizations that reflect system-level interactions among material properties, environmental impact, and operational feasibility. This design approach fosters user learning and internalization of CE logic, in line with recent findings on configurator-assisted behavior change.

The proposed new P Circular Configurator embodies a shift from conventional product customization to lifecycle-oriented design by embedding criteria such as modularity, traceability, and material knowledge which are identified in this study by packaging experts as core enablers of CE- aligned packaging. The proposed conceptualization of the configuration system extends the functional features of existing digital packaging tools and establishes a blueprint for how new digital product configurators can become strategic enablers of circularity in the packaging industry.

From a user point of view the configurator could implement a 3d products visualization while providing real time feedback on sustainable dimensions and circular pattern of the configuration. The proposed new P Circular Configurator, aims at evolving the current version from a passive interface into an active sustainability orchestrator that aligns packaging design processes with the Eco-design circular principle, the operational demands and the regulatory requirements in the packaging sector. As derived from expert-driven MCDA, next user interface of circular configurators should integrate four core capabilities such as (i) dynamic modularity: enabling flexible design variants that facilitate reuse and effortless disassembly; (ii) material traceability: ensuring each component is tagged with provenance, compliance, and recyclability metadata; (iii) embedded material data: incorporating polymer properties, contamination resistance, and process compatibility; (iv) strategy-adaptive logic: enabling prioritization of EOL pathways tuned to regulatory and contextual variables (Figure 1).

Figure 1: Simulation of new P Circular Configurator

source: our elaboration

6. Conclusion

The present study outlines a conceptual extension for packaging configuration systems into a next-generation digital product configurator designed specifically for circular economy compliance. Preliminary results from the present study multistep research design which includes a MCDA and AHP- evaluations performed by industry expert, the study provides preliminary result in researching on: (a) which sustainability criteria are considered most relevant by packaging industry when evaluating EOL strategies for packaging i.e. modularity, traceability, and materials knowledge results as critical enablers of sustainable packaging design. (b) Moves an initial step on exploring how do different EOL strategy (i.e., reuse, mechanical recycling, chemical recycling, organic recycling) are preferred by packaging industry experts which result to be chemical recycling however findings pointed out the importance of a multi-dimensional view of circularity where no single EOL strategy is inherently superior. Rather, the configurator must operate as a decision-support system capable of matching design choices to context-sensitive sustainability parameters. As well as exploring on (c) potential features to be implemented on a next-generation product configurator to support eco- design packaging. Furthermore, the preliminary results underscore that the choice of EOL strategies must be dynamically aligned with design features, environmental constraints, and user priorities. There are no single optimal solutions, only context-sensitive configurations capable of maximizing circular value retention. Even in its exploratory stage the present study reveals the role of product configurators as critical enablers of circular packaging systems that could support each stage of the packaging life cycle design, manufacturing, distribution, collection, and recovery. Future research should validate this framework through prototype development and field trials in actual packaging design workflows.

Acknowledgements

This study was carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from the European Union Next-Generation EU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1551.11-10-2022, PE00000004). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

Declaration on Generative AI

The Authors used Gemini to improve language proof. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] MacArthur, E. (2013). Towards the circular economy. Journal of industrial ecology, 2(1), 23-44.

[2] European Parliament (2023). Circular Economy: Definition, Importance and Benefits, Retrieved on may 2025 at: https://www.europarl.europa.eu/topics/en/article/20151201STO05603/circular-economy- definition-importance-and-benefits

[3] Felfernig, A., Hotz, L., Tiihonen, J., & Bagley, C. (2014). Knowledge-based Configuration: Overview of the Book. In Knowledge-based Configuration: From Research to Business Cases (1 ed., pp. 35-37). ISBN: 978- 0-12-415817-7

[4] Du Rietz, S., & Kremel, A. (2024). Consumer behavior as a challenge and opportunity for circular food packaging—A Systematic Literature Review. Circular Economy and Sustainability, 4(1), 413-438.

[5] Zhu, Z., Liu, W., Ye, S., & Batista, L. (2022). Packaging design for the circular economy: A systematic review. Sustainable production and consumption, 32, 817-832.
[6] McDonough W., & Braungart M. (2009). Cradle to Cradle: Remaking the Way We Make Things. London: Vintage Books. ISBN 9780865475878 / 0865475873
[7] Kirchherr, J., Reike, D., & Hekkert, M. (2017). Conceptualizing the circular economy: An analysis of 114 definitions. Resources, conservation and recycling, 127, 221-232.
[8] European Environment Agency (2020). Green economy [WWW Document]. URL https://www.eea.europa.eu/soer/2015/europe/green-economy#:~:text=Whereas a circular economy focuses, resilience and human well-being. (accessed 11.1.23).
[9] European Investment Bank (2020). The EIB Circular Economy Guide: Supporting the Circular Transition. Luxembourg, Retreived on may at: https://www.eib.org/files/publications/thematic/circular_economy_guide_en.pdf
[10] Kumar, M., Raut, R. D., Jagtap, S., & Choubey, V. K. (2023). Circular economy adoption challenges in the food supply chain for sustainable development. Business Strategy and the Environment, 32(4), 1334- 1356.
[11] Song, J. H., Murphy, R. J., Narayan, R., & Davies, G. B. H. (2009). Biodegradable and compostable alternatives to conventional plastics. Philosophical transactions of the royal society B: Biological sciences, 364(1526), 2127-2139.
[12] Carus, M. (2017). Biobased economy and climate change—important links, pitfalls, and opportunities. Industrial Biotechnology, 13(2), 41-51.
[13] Lynch, D. H., Klaassen, P., & Broerse, J. E. (2017). Unraveling Dutch citizens' perceptions on the bio- based economy: The case of bioplastics, bio-jet fuels and small-scale bio-refineries. Industrial Crops and Products, 106, 130-137.
[14] Hopewell, J., Dvorak, R., & Kosior, E. (2009). Plastics recycling: challenges and opportunities. Philosophical Transactions of the Royal Society B: Biological Sciences, 364(1526), 2115-2126.
[15] Mallick, P. K., Salling, K. B., Pigosso, D. C., & McAloone, T. C. (2024). Towards a circular economy: Development of a support tool for designing reverse logistics systems. Journal of Environmental Management, 351, 119819.
[16] Albino, V., Balice, A., & Dangelico, R. M. (2009). Environmental strategies and green product development: an overview on sustainability-driven companies. Business strategy and the environment, 18(2), 83-96.
[17] Xu Song, Murugesan, Thulasi, Elfar, Abdullah, Durairaj, M. (2024). Evaluation of sustainable manufacturing performance – A case illustration with multistakeholder perspective, Journal of Cleaner Production, 458.
[18] Chenavaz, R. Y., & Dimitrov, S. (2024). From waste to wealth: Policies to promote the circular economy. Journal of Cleaner Production, 443, 141086.
[19] Corona, B., Tunn, V. S., & van den Broek, K. L. (2024). Integrating consumer behaviour into the environmental assessment of circular packaging: a scoping review. The International Journal of Life Cycle Assessment, 29(1), 80-98.
[20] Rajendran, R.; Ranjitharamasamy, S. (2024). A Decision Framework for Selecting Highly Sustainable Packaging Circular Model in Mass-Customized Packaging Industry. Appl. Sci. 14, 10224. https://doi.org/10.3390/ app142210224
[21] Tsou, H. T., Cheng, C. C., & Hsu, H. Y. (2015). Selecting business partner for service delivery co-innovation and competitive advantage. Management Decision, 53(9), 2107-2134.
[22] Khan, S. A. R., Shah, A. S. A., Yu, Z., & Tanveer, M. (2022). A systematic literature review on circular economy practices: challenges, opportunities and future trends. Journal of Entrepreneurship in Emerging Economies, 14(5), 754-795.
[23] Forza, C., & Salvador, F. (2002). Managing for variety in the order acquisition and fulfilment process: The contribution of product configuration systems. International Journal of Production Economics, 76(1), 87-98.

[24] Antonelli, D., & Bruno, G. (2017). Product configuration for order acquisition and fulfillment. In Mass Customized Manufacturing: Theoretical Concepts and Practical Approaches, pp. 167-192.
[25] Trentin, A., Perin, E., & Forza, C. (2014). Increasing the consumer-perceived benefits of a mass-customization experience through sales-configurator capabilities. Computers in Industry, 65(4), 693- 705.
[26] Sandrin, E., Trentin, A., Grosso, C. and Forza, C. (2017). Enhancing the consumer-perceived benefits of a mass-customized product through its online sales configurator: An empirical examination. Industrial Management & Data Systems, 117(6), 1295-1315.
[27] Turner, F., Merle, A. and Gotteland, D. (2020). Enhancing consumer value of the co-design experience in mass customization. Journal of Business Research, 117, 473-483.
[28] Trentin, A., Aichner, T., Sandrin, E., & Forza, C. (2020). Competing through manufacturing: countering a product's liability of foreignness through mass customization. International Journal of Operations & Production Management, 40(11), 1661-1683.
[29] Suzić, N., Sandrin, E., Suzić, S., Forza, C., Trentin, A. and Anišić, Z. (2018). Implementation guidelines for mass customization: A researcher-oriented view. Int. J. Ind. Eng. Manag, 9(4), 229-243.
[30] Trentin, A., Sandrin, E., Suzic, S., Grosso, C., & Forza, C. (2025). Reconciling Product Flexibility with Cost, Delivery, and Quality: The Importance of Bundling Mass Customization Practices. Global Journal of Flexible Systems Management, 26(2), 269-293.
[31] Barca, A., D'Adamo, I., Gastaldi, M., & Leal Filho, W. (2024). Managing waste packaging for a sustainable future: a strategic and efficiency analysis in the European context. Environment, Development and Sustainability, 1-23.
[32] D'Adamo, I., Gastaldi, M., Giacalone, R., & Kazancoglu, Y. (2024). A strategic and social analytics model for sustainable packaging in the cosmetic industry. Supply Chain Analytics, 8, 100090.
[33] Saaty, T. L. (2013). Analytic hierarchy process. In Encyclopedia of operations research and management science (pp. 52-64). Springer, Boston, MA.
[34] Bonifazi, G., D'Adamo, I., Grosso, C., & Palmieri, R. (2025). Advancing Business Strategy in End-Of-Life Management for the Fashion Industry. Business Strategy and the Environment, 34(6), 6814-6827.
[35] D'Adamo, I., Fratocchi, L., Grosso, C., & Tavana, M. (2025). An Integrated Business Strategy for the Twin Transition: Leveraging Digital Product Passports and Circular Economy Models. Business Strategy and the Environment, 34(7), 9008-9022.

A lifecycle- and sustainability-aware product configuration model for modular industrial systems

Gottfried Schenner[1,*], Giray Havur[1,2], Sophie Rogenhofer[1], Stefan Wallner[1], Erwin Filtz[1] and Tassilo Pellegrini[2]

[1]*Siemens AG Österreich, Siemensstraße 90, 1210 Vienna, Austria*
[2]*Fachhochschule St. Pölten, Campus-Platz 1, 3100 St. Pölten, Austria*

Abstract

The incorporation of sustainability and lifecycle information is an important aspect of modern product configurators. In this paper, we describe how to enhance a classic component-based product configuration model by integrating sustainability and lifecycle data. We also identify the relevant external data sources—such as lifecycle assessment databases, product lifecycle management systems, and environmental product declarations—that provide the necessary input. Using a prototypical MiniZinc implementation, we demonstrate how to estimate lifecycle indicators when precise values are unavailable.

Keywords

Green configuration, Sustainability, Minizinc, Power supply

1. Introduction

Building on the Green Deal [1] and its sub-policy, the Circular Economy Action Plan (CEAP) [2], the EU's Clean Industrial Deal aims to address climate and environmental challenges while enhancing Europe's competitiveness and promoting a cleaner, more sustainable future. This also affects industry and industrial production, and from a product configuration point-of-view, the Ecodesign for Sustainable Products Regulation (ESPR) [3] needs to be considered in the product configuration phase. To comply with upcoming legal requirements established through delegated domain- and sector-specific acts complementing the ESPR, methods for measuring and documenting product sustainability indicators — such as ISO-certified, LCA-based Product Environmental Footprints (PEFs) or corresponding Environmental Product Declarations (EPDs)-will eventually become mandatory in the EU. According to the ESPR, sustainability-related product information must be provided through a Digital Product Passport (DPP), which aims to facilitate more circular product and material flows by promoting transparency, accountability, and environmental governance throughout a product's lifecycle.

Following this trend in the product configuration community, the term "green configuration" [4] has been established, referring to a product customization service which also considers sustainability aspects, e.g., in the form of carbon footprinting. In light of these regulatory trends, we argue that lifecycle-related product characteristics — such as total cost of ownership, environmental impact, repairability, reusability, and recyclability — should be considered during the configuration phase.

So far, the problem has been acknowledged and theoretically analyzed from various angles (e.g., [4, 5, 6]). In this paper, we discuss how a green configuration model differs from a traditional configuration model based on an example from the industry sector. We develop a conceptual product configuration model enriched with sustainability and circularity information, which helps us identify the challenges for getting the information from external sources like sustainability databases or product lifecycle

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
*Corresponding author.
gottfried.schenner@siemens.com (G. Schenner); giray.havur@siemens.com (G. Havur); sophie.rogenhofer@siemens.com (S. Rogenhofer); stefan.wallner@siemens.com (S. Wallner); erwin.filtz@siemens.com (E. Filtz); tassilo.pellegrini@fhstp.ac.at (T. Pellegrini)
0000-0003-0096-6780 (G. Schenner); 0000-0002-6898-6166 (G. Havur); 0000-0002-9755-6632 (S. Wallner); 0000-0003-3445-0504 (E. Filtz); 0000-0002-0795-0661 (T. Pellegrini)

management (PLM) systems. For this paper, we focus on the sustainability information at a level typically found in Environmental Product Declarations (EPDs). For a more thorough discussion of the dimensions of sustainability, especially in software engineering, see [7]. Furthermore, we show how to encode the conceptual model in MiniZinc [8]. Therefore, our contribution can be summarized as follows:

1. A conceptual model for sustainability-aware product configuration, which formally integrates component data with material compositions, lifecycle phases (from manufacturing to end-of-life), and key environmental performance indicators (KPIs).
2. A practical implementation of the model in MiniZinc, demonstrating how to encode complex sustainability constraints and objectives for a real-world industrial system, enabling both validation and optimization based on environmental criteria.
3. An analysis of the data-sourcing challenges and solutions for lifecycle-aware configuration, identifying key external data sources (PEF, EPDs, DPPs, LCA services) and outlining a tiered approach for acquiring and estimating the necessary data to populate the model.

The remainder of the paper is structured as follows: Section 2 summarizes the state of the art related to this paper and introduces the fundamental concepts and definitions used throughout the paper. The lifecycle- and sustainability-aware product model is described in Section 3 and followed by our Minizinc encoding in Section 4. Section 6 concludes the paper.

2. State of the art

In the following section, we summarize the state of the art and introduce the most important concepts.

2.1. Green Configuration

The combination of "green" and "configuration" usually describes an approach that combines configuration and sustainability. For example, in [9, 10] the term "Green Configurations" appears in an approach that leads to a greener design and implementation of cyber-physical systems. In [11], the term "Green Configuration" refers to a system to reduce the energy consumption of configurable software systems. In this paper, we use "Green Configuration" in the context of product configuration as it has been defined in [4].

Green Configuration represents an innovative approach that combines conventional product configuration systems with environmental impact assessments while incorporating circular economy principles such as recyclability, repairability, and reusability. By providing immediate feedback on environmental consequences of configuration choices, stakeholders are enabled to make informed decisions. This approach supports the transition toward more environmentally conscious product designs and circular business models, optimizing resource efficiency and minimizing waste throughout the product lifecycle.

One prominent example of environmental impact assessment used in Green Configuration is Life Cycle Assessment (LCA). LCA is an ISO-certified methodology [12, 13] that evaluates environmental impacts throughout a product's complete lifecycle - from raw material extraction ("cradle") through manufacturing, distribution, and use, to final disposal or recycling ("grave"). The process encompasses a detailed analysis of energy and material flows across supply and value chains, calculating associated environmental impacts and emissions. LCAs are fundamentally based on Bill of Materials (BOM) and Bill of Processes (BOP) throughout a product's lifecycle. For decades, LCA has served as the standard for environmental impact assessment according to ISO 14040, with results typically documented in Environmental Product Declarations (EPDs) following ISO 14025. Traditionally, LCA methodologies have operated independently from product configuration processes.

Recent research in green configuration has focused on describing requirements and architectures for integrating LCA into product configurators. Comploi-Taupe et al. [4] have identified four key architectural approaches for combining configurators, knowledge bases, and LCA tools:

1. Sequential Approach: LCA is performed manually after configuration.
2. Loosely Coupled Architecture: Automated but separate LCA calculations requiring synchronization between configurator and LCA tool.
3. Tightly Coupled Architecture: Configurator manages LCA data and directly interfaces with the LCA tool, providing a unified interface.
4. Integrated Architecture: LCA calculation is fully embedded within the configurator, enabling direct environmental data usage during reasoning and optimization.

While the integrated approach offers the most seamless user experience, it demands significant development resources and continuous maintenance to ensure compliance with standards.

Wiezorek and Christensen [5] follow a similar argumentation line that configurators and LCA tools must be integrated and propose extensions to existing product configurators to support green configuration. Jakobsen et al. [6] go one step further and argue that the sustainability aspect already needs to be considered in the product configurator design phase and provide a comprehensive overview of product configurator architectures and sustainable product configuration systems.

2.2. Legal

Although product configuration can be seen as a purely technical task of combining different components to fulfill technical and user requirements (constraints), it is also necessary to consider legal requirements in the configuration process, if they were not already addressed in the product design phase. Such legal requirements are not limited to isolated aspects of product configuration but cover different topics, such as information and documentation requirements, restrictions on the usage of hazardous materials or the disassembling and disposal of products. Additionally, there might be no single legal framework to be considered in a particular product configuration project but multiple national and international legal frameworks.

The rising importance of sustainability and related topics also triggered regulatory activities from the European Union. All of the regulatory acts are supporting overarching goals as laid out in the Clean Industrial Deal [14] and its sub-policies fostering climate-neutrality and the reduction of greenhouse gas emissions. Of special interest for industry is the Green Deal Industrial Plan [15], which aims to simplify the regulatory environment, get easier and faster access to funding, enable the improvement of skills and to foster fair and open trade. Another regulatory framework is the Ecodesign for Sustainable Products Regulation (ESPR) [3] focusing on improving circularity, durability and energy performance by defining ecodesign requirements to better meet the material and procedural demands of circular product design and end-of-life handling. Measures are laid out in the ESPR to achieve these requirements, such as the Digital Product Passport (DPP), which serves as a digital identity for products (including components).

In addition to the more general initiatives regarding sustainability and circular economy, prominent regulatory acts are the Waste from Electrical and Electronic Equipment (WEEE) [16] outlining the requirements on how waste has to be handled to protect humans and the environment. In particular, there are more specific regulations regarding different types of waste, for instance glass cullet [17] or metal scrap [18, 19]. Since there are more and more devices equipped with batteries, there is also a regulation laying out the requirements for the safe operation and disposal of batteries [20]. The Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH) regulation [20] is another example to enhance safety by putting restrictions on the handling of chemicals. Similarly, the restriction of the use of certain hazardous substances in electrical and electronic equipment directive (EEE) [21] focuses on hazardous substances in electric equipment. Furthermore, the Green Claims Directive [22] will require companies to substantiate environmental claims which will also affect services such as product configurators.

In addition to regulatory acts from different legislative bodies, there are also activities from standardization organizations, for instance the International Standardization Organization (ISO) to be considered.

The standards ISO 14020 and 14025 are relevant for the generation of Environmental Product Declarations, ISO 59040 is dealing with circular economy and ISO 59014 with material sustainability.

2.3. Data Sources for Green Configuration

Product configuration typically relies on multiple interconnected data sources that provide the structural, commercial, and logical foundation required to define and validate a specific product variant. In configuration environments, especially those aligned with sustainability goals, these core data categories are increasingly complemented by sustainability and lifecycle data. The following sections outline the key data categories and their typical sources.

2.3.1. Configuration Rules & Constraints

These represent the logical and business dependencies that govern valid product configurations. Constraints define which combinations of components are allowed, required, or excluded. This information is usually maintained in knowledge bases, rule engines, or Product Lifecycle Management (PLM) systems and ensures that only technically valid and manufacturable configurations are generated.

2.3.2. Product Master Data

Sourced from Enterprise Resource Planning (ERP) or Product Information Management (PIM) systems, this includes product identifiers, descriptions, technical attributes, lifecycle status, and classification. It forms the foundation of the configurable product catalogue.

2.3.3. Bill of Materials (BOM)

Maintained in ERP or PLM systems, the BOM describes the hierarchical structure of a product, listing its components and subcomponents. It establishes the link between the configuration process and downstream manufacturing and procurement operations. For Product Configuration, a Maximum Bill of Materials (also known as 150% BOM) is required to encompass all possible product variations and options within a single structure. This comprehensive 150% BOM acts as the foundation for product configuration, enabling the definition of variants and options while managing dependencies between components. Through configuration rules and variant conditions, specific 100% BOMs can be derived for individual product variants, ensuring accurate representation of each product configuration.

2.3.4. Pricing Data

Originating from ERP systems or dedicated pricing engines, pricing data includes base prices, option surcharges, discount rules, regional pricing, and tax logic. This data supports real-time, customer-specific price calculation during the configuration process. Product configuration systems integrate sophisticated pricing mechanisms that dynamically adjust prices based on component combinations and their interactions. The systems process customer-specific pricing agreements and implement volume-based pricing tiers while supporting multi-currency calculations for global operations. For customized configurations, specific pricing models ensure appropriate pricing of unique product variants, while maintaining consistent margin calculations. The pricing engine adheres to established business rules and manages approval workflows for special configuration requests, ensuring accurate pricing across all possible product variants.

2.3.5. Inventory & Availability Data

Sourced from supply chain management or ERP systems, this includes real-time stock levels, lead times, and supplier availability. It enables feasibility checks and supports delivery date estimation for configured products. The system continuously monitors component dependencies to ensure that proposed configurations can be manufactured with available materials. Real-time inventory checks during the configuration process help prevent the creation of product variants that cannot be delivered within acceptable timeframes. Additionally, the system considers production capacity constraints and alternative sourcing options when determining component availability. This integration enables accurate promise dates for customized products while maintaining efficient inventory management across different configuration scenarios.

2.3.6. Sustainability and Lifecycle Data

In addition to data for traditional product configuration, Green Configuration requires sustainability and lifecycle data as a crucial data category that captures key environmental and circular economy-related information. This data category can include various environmental impact metrics such as carbon footprint, energy and water consumption, and material toxicity. It also might cover circular economy aspects like recyclability rates, material recovery potential, and product durability. Additionally, it encompasses regulatory compliance information, including supplier declarations and certifications. Such data can be sourced from various providers and is increasingly critical for aligning product configurations with sustainability goals and legal requirements. However, significant challenges remain in the practical implementation of these data sources. Many companies do not yet disclose environmental data for their products, partly because they do not know them themselves. This results in missing environmental data concerning the supply chain, usage, and end-of-life processing. Furthermore, the required data is often incomplete, with some components needing to be manually disassembled and weighed because suppliers do not provide corresponding data. The calculation of lifecycle assessments relies on comprehensive databases that contain environmental impact data for materials, processes, and energy flows. Key databases include Sphera (GaBi)[1], which provides detailed lifecycle inventory data for thousands of materials and processes across industries. The ecoinvent database[2] is another widely used source containing over 17,000 datasets with cradle-to-gate and cradle-to-grave environmental impacts. These databases include information on greenhouse gas emissions, resource depletion, water consumption, land use changes, and other environmental indicators. They follow standardised methodologies like ISO 14040/44 and are regularly updated to reflect technological advances and improved data quality. Regional databases like the European Life Cycle Database (ELCD) or the U.S. Life Cycle Inventory Database (USLCI) provide location-specific environmental impact factors. These databases are essential for conducting scientifically sound LCA calculations during product configuration and enable the comparison of different material choices based on their environmental impacts.

AAS-Based Data Provider The Asset Administration Shell (AAS)[3] is a standardized digital representation of a physical or logical asset, as promoted by the Industrial Digital Twin Association (IDTA) in Germany [23]. The AAS encapsulates all relevant data and services across the asset's Lifecycle providing a digital twin of a product. AAS supports a modular structure through submodels, which can represent specific sustainability aspects, such as carbon footprint or recyclability scores of a component. Thus, AAS-based services can be used to expose sustainability data as part of a product configuration.

Digital Product Passports from 3rd parties The Digital Product Passport (DPP) is a standardized, uniquely identifiable, digital record of a product introduced by the UN (as part of the UN Transparency Protocol [24] and currently adopted by the European Union as part of its ecodesign regulations [3]. It shall facilitate the sharing of product information among the stakeholders of a product's lifecycle by providing - among other things - highly granular, structured, machine-readable data on circularity-related product parameters such as material composition, substances of concern, environmental impacts, repairability, and end of life (EoL) treatment. Leveraging DPP data within the configuration process enables more informed, sustainable product choices, especially when selecting materials and components from 3rd party providers during the manufacturing phase.

[1] https://sphera.com/solutions/product-stewardship/life-cycle-assessment-software-and-data/
[2] https://ecoinvent.org/database/
[3] https://reference.opcfoundation.org/I4AAS/v100/docs/4.1

LCA Service A Life Cycle Assessment (LCA) service evaluates the environmental impact of products across their entire lifecycle — from raw material extraction to end-of-life. In product configuration, it enables the calculation of product-specific environmental impact indicators such as carbon footprint, energy use, and water consumption for different variants along pre-specified product category rules [25]. This allows for instant feedback on the sustainability impact of the user decisions and supports environmentally responsible choices. LCA services also provide verified data for integration into Digital Product Passports (DPPs), ensure compliance with regulations like the ESPR, and can generate standardized documentation such as ISO 14025 compliant Environmental Product Declarations (EPDs) or Product Environmental Footprints (PEFs) as mandated by the European Union [26]. Overall, they support informed decision-making for eco-design and sustainability optimization.

3. A sustainability enhanced configuration model

Stumptner et al. [27] define product configuration as the assembly of a complex system from simpler predefined components to satisfy some given user requirements. We add sustainability requirements to the basic product configuration model and describe the conceptual sustainability-aware product configuration model with UML [28].

The evolution of product configuration systems reflects a significant shift in focus over time. While early configurators primarily concentrated on ensuring technical feasibility – configurators were designed to validate whether a specific combination of components could function together effectively from a technical perspective – modern configuration approaches have expanded to address multiple optimization criteria. Today's configuration systems take a more comprehensive approach, considering various optimization goals beyond technical requirements. These include economic factors such as cost minimization, operational aspects like energy efficiency, and practical considerations such as ease of maintenance and serviceability. The optimization criteria have further evolved to include environmental impact, resource efficiency, and lifecycle considerations.

Green Configuration represents a holistic approach that integrates sustainability aspects into the configuration process. This approach considers not only the immediate technical and economic factors but also long-term environmental impacts, resource consumption, and end-of-life scenarios. By incorporating sustainability metrics into the configuration process, organizations can optimize their products for both performance and environmental responsibility. This includes considerations such as carbon footprint, material recyclability, energy efficiency during operation, and the overall environmental impact throughout the product's lifecycle. The goal is to find configurations that balance technical requirements, economic viability, and environmental sustainability in an integrated way.

In the following, we will make the information needed for sustainable product configuration more explicit. This way we can provide feedback, how the user decisions influence the sustainability of the configured product. We can not expect to assess the sustainability of a configured product in the same detail as it is done in a full lifecycle assessment process (LCA).

Still our main goals are:

- Compare configurations based on environmental KPIs across lifecycle phases
- Verify compliance with environmental regulations
- Allow specification of material constraints (e.g., hazardous substance restrictions)
- Identify key components and phases with highest sustainability impact
- Evaluate impact of various usage scenarios
- Represent end-of-life, recycling, and circular economy options

3.1. Example SITOP PSU8600 Power Supply System

As a running example, we use the task of configuring the industrial SITOP PSU8600 power supply system by Siemens.[4] This advanced modular and expandable system efficiently converts alternating current (AC) to stable direct current (DC) output, featuring high conversion efficiency and reliability for industrial applications.

An optimal SITOP PSU8600 system variant can be configured based on several critical technical requirements:

- **Input voltage specifications** – Supporting diverse power grid standards
- **Output parameters** – Precise current and voltage requirements for connected equipment
- **Environmental factors** – Operating temperature constraints and installation conditions
- **Power reliability** – Buffer load capabilities for system stability
- **Industrial networking** – Connectivity features including PROFINET or standard Ethernet integration

A SITOP PSU8600 variant comprises multiple components called modules that can be combined according to defined technical constraints. The UML class diagram in Figure 1 illustrates the components of the SITOP PSU8600 system considered in this paper and their interrelationships. Each SITOP PSU8600 system requires exactly one basic module. Up to four expansion modules can be added to the system. To safeguard the system against small power failures (up to several seconds) buffer modules can be added. For longer power outages, up to two Uninterruptible Power Supply (UPS) modules with max. five batteries are possible.

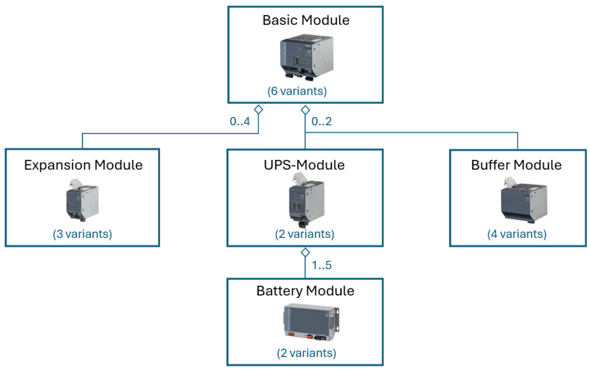


Figure 1: SITOP PSU8600 UML diagram

3.2. Materials

The material composition is an important part of the sustainability of a product. In the manufacturing phase the used materials impact the KPIs, e.g., CO_2 emission caused by providing the material. Problematic and hazardous substances impact the end-of-life phase. The materials of a component might either be fixed for supplied parts or variable for generic components, e.g., components whose dimensions can be configured. Figure 2 depicts a configuration model augmented with material information. To keep the model simple the class `Component` represents anything from products, assemblies to supplied (hardware) parts. Components can have materials and sub components.

`LCAScope` defines the lifecycle phases (`LCAPhase`) considered in the current `Configuration`. The class `Material` defines the amount of a Material used in a component or in a `MaterialAllocation`.

[4] See the SITOP PSU8600 product information at: https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10251281.

The class `MaterialAllocation` corresponds to additional material that cannot be assigned to a component, but is required only during one of the lifecycle phases, e.g., packaging material, consumable materials...

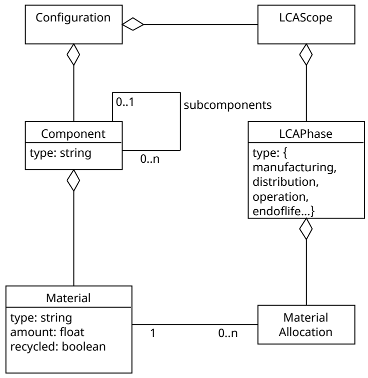


Figure 2: Configuration model with materials

Each component is composed of an arbitrary number of materials. The level of granularity regarding the used materials depends on the available information. In cases, where the material composition of sub-components is not known, the material information of a component just contains the aggregated values of the used materials in the sub-components. The aggregated materials of the whole configuration corresponds to the material composition that is reported in EPDs. For instance, in the PLM model (Siemens Teamcenter) the SITOP PSU8600 basic module of a given type is comprised of hundreds of sub-components, such as electronic parts, housing and so forth.

This detailed information is only relevant, if there are some constraints on the sub-components or the user wants to have insights on the material composition of the product. An simplified example for the material composition of a SITOP PSU8600 basic module is shown in Figure 3. The information about the materials is taken from the EPD of the basic module and lists the different types of materials and their weights.

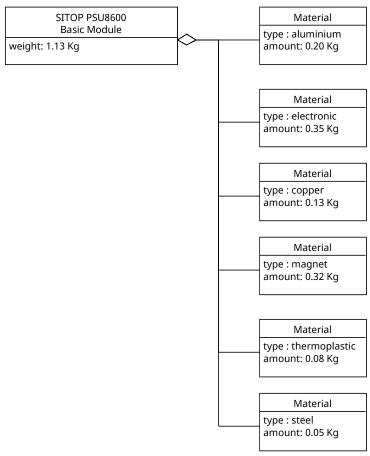


Figure 3: Basic module material example

3.3. KPIs and Lifecycle Phases

Another important aspect of LCAs and EPDs are key (environmental) performance indicators (KPI). They indicate the environmental impact and resource consumption of the configured product during specific lifecycle phases.

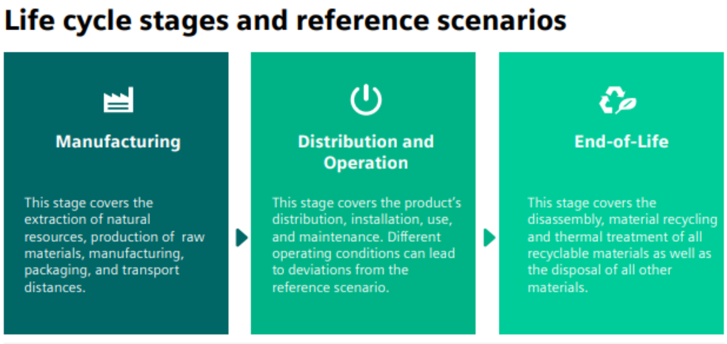


Figure 4: SITOP PSU8600 lifecycle phases (as defined in the EPD)

For the running example of this paper we use the lifecycle model of the EPDs of the SITOP PSU8600 system.[5] (Figure 4). In the EPD different phases are aggregated into one stage. For example raw material extraction, production of raw materials, manufacturing, packaging and transport are summarized in one manufacturing stage.

The LCA of a product typically covers the entire lifecycle, from cradle-to-grave. In a product configurator not every lifecycle phase will be considered depending on the configuration scenario.

For instance, in a sales configurator the manufacturing phase and the usage phase are the most important phases. Information about the detailed end-of-life options are very customer specific and may not be available to the sales configurator. However, at least information about the circularity and recyclability of the product can be provided. In contrast, in an in-house engineering configurator not only the usage but also circularity and end-of-life aspects are typically known as they are managed inside the organisation.

As can be seen in Figure 5, a KPI is assigned to a component and a LCA phase. On the configuration level these values are aggregated to KPIs per lifecycle phase and subsequently a total KPI can be computed. For our SITOP example, the estimation of the global warming potential (GWP) of the manufacturing phase of the configured SITOP PSU8600 system is the sum of the (manufacturing) GWPs of the components used in the configuration.

3.3.1. Manufacturing

For supplied components/products, we can expect to get data from existing EPDs or if available from a DPP (on the model level). In the case of the SITOP PSU8600 system the data can be taken from the published EPDs or from in-house tools like the green digital twin (GDT).

In the case of third-party components where this data is not available, we could still use approximate data for the type of product from environmental databases. The data is expected to be more accurate if a more specific type of product is considered. For example, when estimating values for a specific variant of a SITOP PSU8600 basic module, using data from another variant of the same basic module would be more accurate than using data from a SITOP CNX8600 expansion module. For the estimation of KPIs related to transportation, information about the shipping routes for the supplied parts and materials as well as the location where the configured product is assembled, is required. Data for common ways of transportation (air, container ship, rail) are standard in all environmental databases and LCA tools. Although one could define very sophisticated transport models, for the sake of the configuration scenario considering the distances and the mode of transport should be sufficient. Remember that accurate values are only required if it helps to find the most sustainable configuration among the possible configurations satisfying all the user requirements. If, for instance, in an engineering context there is only a single supplier with a single shipping site the exact values are not important. A sophisticated approach would be to analyze existing transport data and create a machine learning model for the most

[5]The EPD can be downloaded from https://support.industry.siemens.com/cs/ww/en/view/109824794.

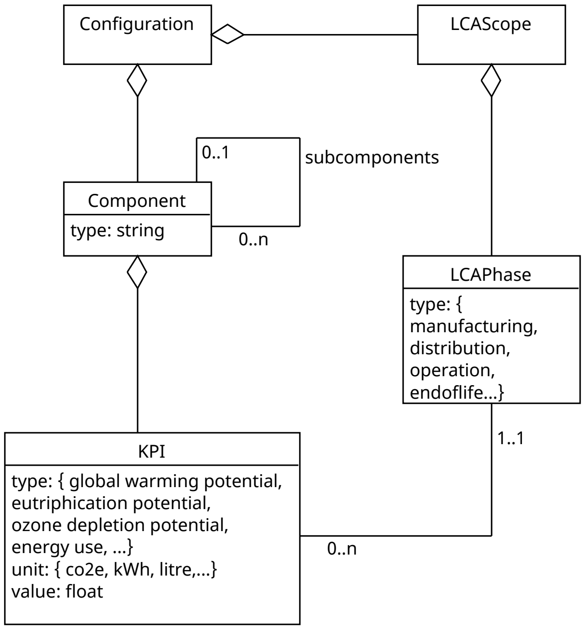


Figure 5: Configuration model with KPIs

likely mode of transport of products of type X from A to B.

The KPIs for production of raw materials and manufacturing the configured product can range from simple (consumer products) to highly complex, e.g., the production of a configured railway interlocking system. LCA of production and manufacturing must consider bill of processes (BOP), factory data (assembly lines). The LCA of complex systems, such as railway interlocking systems, involves additional factors like construction work, road work, and specialized equipment. Modeling this information within a product configuration scenario is unrealistic. Therefore, a product configurator must either access data from existing (parameterized) LCA calculations or rely on rough estimates.

3.3.2. Distribution and Operation

The transportation aspect of distribution is essentially the same as previously discussed in the manufacturing phases. An additional aspect is packaging as this requires additional (hopefully recyclable or reusable) material.

The impact of the usage phase is very specific to the configured product and the intended usage of the configured system. This is the phase where user requirements typically have the greatest impact. The KPIs for the usage phase are often specified for a defined time period (e.g., 10 years) and usage scenario (e.g., 24/7 operation). For electrical components the most basic calculation of the GWP is the energy demand of the component multiplied with the usage time multiplied by the GWP of the energy source. However, the situation is more complex in practice. Under a naive calculation a configuration with less components will have a better GWP KPI. But introducing components such as buffer modules increases the reliability of the entire system. Without buffer modules short drops in electricity (brown out) can lead to failures in industrial processes and negatively impact the sustainability of the production process as a whole. One way to communicate this to the user is through multi-objective optimization; specifically, showing the relationship between system reliability and sustainability in the case mentioned above.

For long-running systems, obsolescence considerations are a critical aspect of this phase, particularly in determining the number of components that will require replacement within the given time-frame based on their expected life expectancy. Repairability and spare parts availability significantly influence the usage phase. However, developing metrics to quantify these aspects remains challenging. Upcoming standards like the DPP will define some standard KPIs to measure these circularity aspects.

3.3.3. End of life

The minimum requirement for end-of-life treatment of products are landfills or thermal dissipation. The more sustainable option is to disassemble the product and recycle as much of the components as possible. Still better one of the R-strategies of the circular economy [29], i.e., reuse, remanufacture or refurbish should be applied to the product or its sub-components.

4. MiniZinc Encoding

In this section, we show the implementation of the running example in MiniZinc using our conceptual model.

4.1. Components and BOM generation

The MiniZinc encoding is simple and only serves the purpose of illustrating the sustainability enhanced aspects of the configuration model. In practice, a more generic and sophisticated encoding should be used, e.g., an encoding based on an object-oriented formalism.

Listing 1 shows how components and their quantities are encoded for the SITOP PSU8600 example. The cardinality constraints are taken from Figure 1.

Listing 1: MiniZinc Encoding of components

```
enum SITOPComponent = {
  % top level component
  BaseModule,
  ExpansionModule,
  BufferModule,
  UPSModule,
  % sub component
  BatteryModule
};

array[SITOPComponent] of var 0..10:
  component_quantity;

var 0..5: UPSModule1_nrofbatteries;
var 0..5: UPSModule2_nrofbatteries;

constraint
  component_quantity[BaseModule] = 1;

constraint
  component_quantity[ExpansionModule] <= 4;

constraint
  component_quantity[BufferModule] +
  component_quantity[UPSModule] <= 2;

constraint
  UPSModule1_nrofbatteries>0 <->
    component_quantity[UPSModule]=1;

constraint
  UPSModule2_nrofbatteries>0 <->
    component_quantity[UPSModule]=2;
```

```

constraint
  component_quantity[BatteryModule] =
    UPSModule1_nrofbatteries +
    UPSModule2_nrofbatteries;
```

4.2. Encoding of Materials

The materials of components are modelled with a table that contains the amount of material that is included in every component. This amount is considered fixed, i.e., for this simple example there is no variability. In a more realistic example the dimension of a component might be configurable, e.g., the length of a cable, and therefore the materials would also be dynamic. Based on the selected components and the material table the total amount is computed. This allows the easy formulation of constraints about the material content of the configuration, like the one in Listing 2 which states that the configuration should not contain any hazardous materials.

Listing 2: MiniZinc

```
enum Material = { aluminium,
  electronic,
  copper,
  lead };
enum Component_Material =
  { componentid, materialid, gram };

array[int,Component_Material] of int:
  material_table;

material_table =
     [| BaseModule, aluminium, 20,
      | BaseModule, electronic, 35,
      | BaseModule, copper, 13,
      %...
|];

int : material_table_rows =
  length(material_table) div 3;

array[Material] of var int: total_weight_material;

constraint
  (forall (m in Material)
  (total_weight_material[m] =
    sum([material_table[r, gram] *
         component_quantity[
         to_enum(
         SITOPComponent,
         material_table[r,
           componentid])] |
        r in 1..material_table_rows where
        material_table[r,
          materialid] = m])));

var set of Material:
  hazardous_materials = { lead };

% example:
% configuration should contain no
```

```
% no hazardous materials
constraint
  (forall (m in Material)
  (if total_weight_material[m]>0
  then not (m in hazardous_materials)
  else true endif));
```

4.3. Distribution

For the distribution phase, we can model the impact of transporting the final product from the assembly location to the customer. This involves defining different modes of transport, their respective environmental impact factors (e.g., kg CO_2-eq per ton-kilometer), and the total weight of the configured system. Listing 3 shows a simple implementation where the model can choose a transport mode based on user requirements or optimization goals.

Listing 3: MiniZinc

```
% available transport modes
enum TransportMode = {TRUCK, TRAIN, AIR};

% transport mode
var TransportMode: transport_mode;

% GWP in kg CO2-eq per ton-km for each transport mode
array[TransportMode] of float: transport_gwp_factor = [0.08, 0.02, 0.5];

% distance to customer in km (can be a user requirement)
float: distance_km = 1000.0;

% total weight of the configuration in kg (calculated from materials)
var float: total_weight_gram = sum(m in Material)(total_weight_material[m]);

% calculated GWP for the distribution phase
var float: distribution_gwp =
  transport_gwp_factor[transport_mode] * (total_weight_gram / 1000.0) * distance_km;

% example constraint: for urgent deliveries, air freight is required.
% constraint transport_mode = AIR;
```

4.4. Usage

The impact of the usage phase is highly dependent on the efficiency of the product and the operating scenario of the user. For the SITOP PSU8600, the primary environmental impact during usage stems from energy loss (heat dissipation), not the energy it delivers to other components. We can calculate this by taking the energy consumed by the power supply itself and multiplying it by an impact factor for the electricity grid. Listing 4 demonstrates this calculation.

Listing 4: MiniZinc

```
% efficiency of the basic module (can depend on the selected type)
param float: base_module_efficiency = 0.95;

% user requirements for usage profile
param float: avg_power_output_kw = 2.0; % Avg. power delivered
param float: lifetime_h = 43800; % e.g., 5 years of 24/7 operation

% environmental factor for the electricity grid (e.g., from EPD or database)
% kg CO2-eq per kWh
```

```
param float: grid_gwp_factor = 0.4;

% total energy delivered over the lifetime
var float: total_energy_delivered_kwh = avg_power_output_kw * lifetime_h;

% total energy consumed by the PSU
var float: total_energy_consumed_kwh = total_energy_delivered_kwh /
    base_module_efficiency;

% total energy lost as heat
var float: energy_loss_kwh = total_energy_consumed_kwh - total_energy_delivered_kwh;

% calculated GWP for the usage phase
var float: usage_gwp = energy_loss_kwh * grid_gwp_factor;
```

4.5. End of life

The end-of-life phase considers the environmental effects of disposing of, recycling, or reusing the product's materials. Different treatments yield different impacts; for instance, recycling metal often results in an environmental credit, avoiding emissions from virgin material production. Listing 5 models this by allowing a choice of end-of-life option for each material and calculating the resulting environmental impact. The effects of more sophisticated R-strategies like reuse, remanufacturing, refurbish on the GWP are too difficult to calculate in a configuration model. Regardless components to which these R-strategies can be applied should be preferred in the configuration either by modeling the options as boolean or giving them an "estimated" GWP value that is lower than the other EoL options.

Listing 5: MiniZinc

```
% End-of-Life options
enum EoL_Option = {LANDFILL, INCINERATION, RECYCLING};

% choose an EoL option for each material
array[Material] of var EoL_Option: eol_choice;

% GWP impact per kg of material for each EoL option (kg CO2-eq/kg).
% negative values represent credits from recycling.
% this data would come from LCA databases.
array[Material, EoL_Option] of float: eol_gwp_matrix =
  [| % Columns: LANDFILL, INCINERATION, RECYCLING
      0.02, 0.05, -1.5,   % for Aluminium
   |  0.05, 0.20, -0.8,   % for Electronics (simplified)
   |  0.02, 0.04, -2.8,   % for Copper
   |  0.04, 0.06, -3.5,   % for Steel
   |  0.5, 0.70, 0.70,    % for Lead
     %...
   |];

% calculated GWP for the end-of-life phase
var float: eol_gwp = sum(m in Material) (
    (total_weight_material[m] / 1000.0) * eol_gwp_matrix[m, eol_choice[m]]
);

% example constraint: Maximize recycling
constraint forall(m in {aluminium, copper, steel})(eol_choice[m] = RECYCLING);
```

5. Discussion

From a representational standpoint, constraints concerning sustainability are not different from constraints expressing technical restrictions or user requirements. The main difference is that exact values might not be available for the sustainability parameters. This is not a problem as long as the orders of magnitude are correct.

Since we cannot call external functions when solving a MiniZinc model, we have to gather the required sustainability data (material composition, carbon footprint, etc.) before we start the solving process. Then we can use multi-objective optimization to find the most sustainable (Pareto-optimal) configuration(s). In practice, there can always be contradictory objectives, e.g., avoiding hazardous substances vs. low carbon footprint. The final decision about which configuration to select is up to the user.

If the number of possible configurations for the given user inputs is relatively small (<=10) , we can alternatively ignore the sustainability aspects during solving and assess the sustainability of the found configurations with an external API afterwards.

6. Conclusions

The purpose of our sustainability-enhanced configuration model is to give the user an indication of how their requirements and selections affect the sustainability of the configured product.

While this approach does not replace a full LCA, carefully modeling sustainability parameters allows the configurator to suggest solutions that are likely to be sustainable both in the LCA and in real-world usage.

Currently the necessary sustainability data for the possible components of a configured product must be collected from various sources (EPD, LCA, sustainability databases, in-house tools). Sometimes this data is not even available in machine readable form, e.g., the documents of the EPDs.

Upcoming standards like the DPP ease this process as the necessary data will then be available in a digital form via standardized APIs. This should enable us to get most of the required sustainability information of the configuration model in an automated manner.

The DPP will also contain life data from the usage and end-of-life phase, which allows the comparison of the expected values for the KPIs with the actual KPIs measured in the product lifecycle.

The configuration model is not limited to first-time configuration. Once a sustainability-enhanced configuration model is established interesting reconfiguration scenarios are possible, e.g., replacing sub-components with more sustainable components that might not have been available at the time of the initial configuration. The changes in the configuration will then be reflected in an updated DPP.

Acknowledgments

This work has been partially funded by the Austrian Research Promotion Agency (FFG) under the project grants FO999915294 (ECO-TCO) and FO999917177 (PACE-DPP).

Declaration on Generative AI

During the preparation of this work, the author(s) used generative AI in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] European Commission, The European Green Deal, Technical Report COM(2019) 640 final, European Union, 2019.

[2] European Commission, A New Circular Economy Action Plan for a Cleaner and More Competitive Europe, Technical Report COM(2020) 98 final, European Union, 2020.
[3] Regulation (EU) 2024/1781 of the European Parliament and of the Council of 13 June 2024 establishing a framework for the setting of ecodesign requirements for sustainable products, amending Directive (EU) 2020/1828 and Regulation (EU) 2023/1542 and repealing Directive 2009/125/EC (Text with EEA relevance), 2024.
[4] R. Comploi-Taupe, A. Falkner, K. Müller, S. Rogenhofer, Requirements and Architectures for Green Configuration, in: E. Vareilles, C. Grosso, J. M. Horcas, A. Felfernig (Eds.), Proceedings of the 26th International Workshop on Configuration (ConfWS 2024), volume 3812 of *CEUR Workshop Proceedings*, CEUR, Girona, Spain, 2024, pp. 33–40.
[5] R. Wiezorek, N. Christensen, Integrating Sustainability Information in Configurators, in: Proceedings of the 23rd International Configuration Workshop (CWS/ConfWS 2021), volume 2945 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2021, pp. 65–72.
[6] A. Jakobsen, T. Tambo, M. Kadenic, Greener Information Systems for Product Configuration Management: Towards Adaptation to Sustainability Requirements:, in: Proceedings of the 26th International Conference on Enterprise Information Systems, SCITEPRESS - Science and Technology Publications, Angers, France, 2024, pp. 100–109. doi:10.5220/0012737200003690.
[7] S. Betz, B. Penzenstadler, L. Duboc, R. Chitchyan, S. A. Kocak, I. Brooks, S. Oyedeji, J. Porras, N. Seyff, C. C. Venters, Lessons Learned from Developing a Sustainability Awareness Framework for Software Engineering Using Design Science, ACM Trans. Softw. Eng. Methodol. 33 (2024) 136:1–136:39. doi:10.1145/3649597.
[8] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, G. Tack, MiniZinc: Towards a Standard CP Modelling Language, in: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007, volume 4741, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 529–543. doi:10.1007/978-3-540-74970-7_38.
[9] D.-J. Munoz, J. A. Montenegro, M. Pinto, L. Fuentes, Energy-aware environments for the development of green applications for cyber–physical systems, Future Generation Computer Systems 91 (2019) 536–554. doi:10.1016/j.future.2018.09.006.
[10] J.-M. Horcas, M. Pinto, L. Fuentes, Green Configurations of Functional Quality Attributes, in: Proceedings of the 21st International Systems and Software Product Line Conference - Volume A, SPLC '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 79–83. doi:10.1145/3106195.3106205.
[11] N. Siegmund, J. Dorn, M. Weber, C. Kaltenecker, S. Apel, Green Configuration: Can Artificial Intelligence Help Reduce Energy Consumption of Configurable Software Systems?, Computer 55 (2022) 74–81. doi:10.1109/MC.2021.3120048.
[12] ISO, ISO 14040:2006 - Environmental management — Life cycle assessment — Principles and framework, https://www.iso.org/standard/37456.html, 2025.
[13] ISO, ISO 14044:2006 - Environmental management — Life cycle assessment — Requirements and guidelines, https://www.iso.org/standard/38498.html, 2025.
[14] Communication from the commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. The Clean Industrial Deal: A joint roadmap for competitiveness and decarbonisation, 2025.
[15] European Commission, A Green Deal Industrial Plan for the Net-Zero Age, Technical Report COM(2023) 62 final, European Union, 2023.
[16] Directive 2012/19/EU of the European Parliament and of the Council of 4 July 2012 on waste electrical and electronic equipment (WEEE) (recast) (Text with EEA relevance), 2018.
[17] Commission Regulation (EU) No 1179/2012 of 10 December 2012 establishing criteria determining when glass cullet ceases to be waste under Directive 2008/98/EC of the European Parliament and of the Council, 2012.
[18] Commission Regulation (EU) No 715/2013 of 25 July 2013 establishing criteria determining when copper scrap ceases to be waste under Directive 2008/98/EC of the European Parliament and of the Council, 2013.

[19] Council Regulation (EU) No 333/2011 of 31 March 2011 establishing criteria determining when certain types of scrap metal cease to be waste under Directive 2008/98/EC of the European Parliament and of the Council, 2011.
[20] Regulation (EC) No 1907/2006 of the European Parliament and of the Council of 18 December 2006 concerning the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH), 2006.
[21] Directive 2011/65/EU of the European Parliament and of the Council of 8 June 2011 on the restriction of the use of certain hazardous substances in electrical and electronic equipment (recast) (Text with EEA relevance), 2016.
[22] Proposal for a DIRECTIVE OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on substantiation and communication of explicit environmental claims (Green Claims Directive), 2023.
[23] IDTA, IDTA – Der Standard für den Digitalen Zwilling - Startseite, https://industrialdigitaltwin.org/, 2025.
[24] United Nations Economic Commission for Europe, UN Transparency Protocol, https://uncefact.github.io/spec-untp/, 2025.
[25] E. International, The PCR | EPD International, https://www.environdec.com/pcr/the-pcr, 2025.
[26] E. Commission, PEF METHOD - European Commission, https://green-forum.ec.europa.eu/environmental-footprint-methods/pef-method_en, 2025.
[27] M. Stumptner, G. E. Friedrich, A. Haselböck, Generative constraint-based configuration of large technical systems, Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12 (1998) 307–320. doi:10.1017/S0890060498124046.
[28] A. Felfernig, G. E. Friedrich, D. Jannach, UML as Domain Specific Language for the Construction of Knowledge-Based Configuration Systems, International Journal of Software Engineering and Knowledge Engineering 10 (2000) 449–469. doi:10.1142/S0218194000000249.
[29] P. Morseletto, Targets for a circular economy, Resources, Conservation and Recycling 153 (2020) 104553. doi:10.1016/j.resconrec.2019.104553.

Sustainability Evaluation Metrics for Configuration Systems

Alexander Felfernig[†], Damian Garber[†], Sebastian Lubos[†] and Thi Ngoc Trang Tran[†]

Institute of Software Engineering and AI, Graz University of Technology, Graz, Austria

Abstract

Sustainability-oriented evaluation metrics offer a means to assess the quality of configuration systems beyond conventional metrics such as accuracy of personalized configurations or sales-related conversion rates. Aligned with the United Nations' Sustainable Development Goals (SDGs), these metrics enable a structured analysis of the environmental, social, and economic impacts of configuration systems. In this paper, we explore sustainability-focused evaluation metrics tailored to configurators and examine applications and implications.

Keywords

Configuration, Configuration Systems, Sustainability, Evaluation Metrics, Sustainable Development Goals

1. Introduction

Configuration can be regarded as a key technology supporting the mass customization paradigm [1, 2, 3]. Traditionally, the effectiveness of configurators is assessed on the basis of performance-related metrics such as configuration accuracy (of personalized recommendations) and sales-related conversion rates [4, 5]. However, due to societal relevance, there is a growing need to integrate evaluation criteria that also take into account sustainability aspects [6, 7, 8, 9, 10, 11, 12].

To address this need, we propose a basic set of sustainability-aware evaluation metrics. These metrics go beyond immediate system performance and focus more on embedding long-term impacts of configurations into the evaluation process. In this context, our aim is not only to optimize the utility of configurations but also to take into account global sustainability goals such as the United Nations' Sustainable Development Goals (SDGs).[1] By considering, for example, the environmental impact of selected components, fairness and inclusivity of configuration options, and economic fairness, such metrics offer a more holistic understanding of the impact of configuration systems.

In this paper, we provide an overview of basic sustainability-oriented evaluation metrics (being aware of the fact that many further variants are possible). We map these metrics to the three core sustainability areas of the United Nations' SDGs: environmental, social, and economic. We also illustrate application scenarios and discuss topics for future research.

The remainder of this paper is structured as follows: in Section 2, we present evaluation metrics related to aspects of environmental sustainability. Section 3 addresses metrics for social sustainability, followed by economic sustainability metrics which are discussed in Section 4. Sections 5 and 6 discuss cross-dimensional metrics and outline open research issues. The paper is concluded with Section 8.

2. Environmental Metrics

Environmental sustainability metrics extend the evaluation of configuration systems beyond traditional performance-based measures by assessing their contribution to environmental objectives.

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy

[†]These authors contributed equally.

alexander.felfernig@tugraz.at (A. Felfernig); damian.garber@tugraz.at (D. Garber); sebastian.lubos@tugraz.at (S. Lubos); trang.tran@tugraz.at (T. N. T. Tran)

0000-0003-0108-3146 (A. Felfernig); 0009-0005-0993-0911 (D. Garber); 0000-0002-5024-3786 (S. Lubos); 0000-0002-3550-8352 (T. N. T. Tran)

[1]https://sdgs.un.org/goals

2.1. Carbon Footprint of Configurations

The carbon footprint of a set of configurations $\mathcal{CONFS}$ proposed to users over a specific time period measures the average greenhouse gas emissions (expressed, e.g., in tons of CO_2 (equivalent) produced over the full configuration lifecycle) associated with the configurations $conf \in \mathcal{CONFS}$. Let CarF($conf$) denote the estimated overall carbon footprint of components in $conf$. Then, the *average carbon footprint of offered configurations* (AvgCarFConf) can be defined as follows:

$$\text{AvgCarFConf} = \frac{1}{|\mathcal{CONFS}|} \sum_{\text{conf} \in \mathcal{CONFS}} \text{CarF(conf)} \tag{1}$$

In this context, lower values of AvgCarFConf indicate that the configuration system tends to favor components with lower carbon footprints.

2.2. Energy Consumption of Configuration

Energy consumption of *configuration* (i.e., the generation of configurations) refers to the energy required to compute configurations ($conf \in \mathcal{CONFS}$) for users of a configuration system. Let $E_{\text{configuration}}$ denote the total energy (e.g., in kilowatt-hours) consumed by the configuration system over a defined evaluation period, and let N_{conf} be the total number of configurations generated. The *Energy Consumption per Configuration* (ECConf) can be defined as follows:

$$\text{ECConf} = \frac{E_{\text{Configuration}}}{N_{\text{conf}}} \tag{2}$$

This metric is particularly relevant for large-scale configuration systems that involve complex reasoning and consistency management, where an optimization for energy efficiency is important.

2.3. Energy Consumption of Model Building

Energy consumption of *configuration model building* refers to the total energy consumed during the construction of a configuration knowledge base. Let EC_{dev} represent the cumulative energy consumed throughout the entire configuration model development process, and let N_{versions} denote the number of developed configuration model versions over a specific time period. The *Energy Consumption per Model Version* (ECModVer) can be defined as follows:

$$\text{ECModVersion} = \frac{EC_{\text{dev}}}{N_{\text{versions}}} \tag{3}$$

Such metrics are particularly relevant for evaluating the environmental and computational efficiency of different configuration environments.

2.4. Energy Savings Through Configuration

Energy Savings through Configuration (ESTConf) refers to the reduction in energy consumption or resource usage achieved as a result of applying a configuration system [13]. Configuration systems can enhance efficiency by guiding users toward environmentally friendly component selections, reducing over-dimensioning, or optimizing system designs. Let EC_{baseline} denote, for example, the energy consumption of a system (e.g., annual energy usage) designed without the support of a configuration system, and let EC_{withconf} represent the energy consumption observed when a configuration system has been used (e.g., a configuration tool supporting energy-efficient component selection). Related energy savings can be expressed as follows:

$$\text{ESTConf} = \frac{EC_{\text{baseline}} - EC_{\text{withconf}}}{EC_{\text{baseline}}} \tag{4}$$

Such metrics are particularly relevant in domains where configuration decisions can significantly influence consumption patterns [10, 13].

Environmental sustainability metrics reflect a paradigm shift from short-term optimization goals—such as maximizing user engagement—toward long-term ecological considerations. However, implementing these metrics in practice presents challenges such as limited access to reliable carbon footprint data and the lack of standardized definitions for the sustainability of components.

3. Social Metrics

Social sustainability in configuration systems emphasizes the fair and inclusive generation of configurations. These metrics go beyond traditional performance measures to ensure that the system's design and resulting configurations support social equity, accessibility, and community well-being. In this context, configuration processes should avoid bias, promote inclusive component ($comp \in conf$) selection, and ensure that all users can effectively engage with and benefit from the configuration system.

3.1. Fairness and Bias

Related metrics can be used to assess whether configuration outcomes (components) are equitably distributed across different demographic groups. A commonly discussed fairness criterion is *demographic parity*, which requires that the share of selected components ($comp \in conf$) in generated configurations is similar across sensitive attributes (e.g., gender, age group). Let G denote a set of demographic groups, and let $P_g(comp)$ represent the probability that component $comp$ is included in a configuration for users belonging to group $g \in G$. In this context, demographic parity is satisfied if the following condition holds:

$$P_g(\text{comp}) \approx P_{g'}(\text{comp}) \quad \forall\, g, g' \in G \quad (g \neq g') \tag{5}$$

3.2. Diversity

To promote exposure to diverse perspectives, diversity in a configuration list ($\mathcal{C}_u$) presented to a user u is crucial. *Configuration diversity* ConfDiv_u for a user u can be measured, for example, based on the average pairwise similarity (sim) among configurations ($\{conf_i, conf_j\} \subseteq C_u$, $i \neq j$) presented to user u where similarity values (between two configurations) are assumed to be in the interval $(0, 1)$:

$$\text{ConfDiv}_u = 1 - \frac{\sum_{\text{conf}_i \in \mathcal{C}_u} \sum_{\text{conf}_j \in \mathcal{C}_u} \text{sim}(\text{conf}_i, \text{conf}_j)}{|\mathcal{C}_u| \times (|\mathcal{C}_u| - 1)} \tag{6}$$

In this context, $\text{sim}(\text{conf}_i, \text{conf}_j)$ denotes the similarity between configurations conf_i and conf_j (e.g., based on component equality). Higher values of ConfDiv_u indicate greater diversity in C_u which reflects a lower average similarity among configurations. The metric *ConfDiv* would then represent the average calculated over all user-specific values (ConfDiv_u).

3.3. Accessibility and Inclusivity

Accessibility aims to ensure that both, configurators and configurations can be effectively used by users with diverse abilities and backgrounds, represented by different groups $g \in G$. Let $\mathcal{AC}$ denote a set of *accessibility criteria* (e.g., understandability of configuration steps, clarity of component descriptions, or usability for users with visual or cognitive impairments), and let sat be a function that measures the extent to which the components and/or user interface elements in a set $\mathcal{Q}$ satisfy these criteria for a group g on a scale from 0 to 1. Then, the *accessibility score* (ACC_g) for a group $g \in G$ can be defined as follows:

$$\text{ACC}_g = \frac{\sum_{q \in \mathcal{Q}} sat(q, \mathcal{AC}, g)}{|\mathcal{Q}|} \tag{7}$$

A higher value of ACC_g indicates better accessibility for group g. *Inclusivity* is considered to be achieved when accessibility is fulfilled equitably across different demographic or ability-based groups $g_i \in G$:

$$\text{ACC}(g_i) \approx \text{ACC}(g_j) \quad \forall\, g_i, g_j \in G \quad (i \neq j) \tag{8}$$

3.4. Health Improvement through Configuration

Health Improvement through Configuration (HIConf) refers to the enhancement of individual or population health outcomes achieved through the use of configuration systems. Configuration systems can support healthier decisions by guiding users toward appropriate selections of components related, for example, to diet plans and training plans.

Let $\mathcal{M}_{\text{with}}$ denote a health outcome metric (e.g., average activity level or body mass index) for users of a configuration system, and let $\mathcal{M}_{\text{without}}$ represent the same metric for users not using such a system. The health improvement across all users can be defined as:

$$\text{HIConf} = \frac{\mathcal{M}_{\text{with}} - \mathcal{M}_{\text{without}}}{\mathcal{M}_{\text{without}}} \tag{9}$$

This metric is particularly relevant in application domains such as digital health platforms, wellness configurators, and preventive healthcare services, where personalized configurations can positively impact well-being [14, 15].

4. Economic Metrics

Economic sustainability metrics assess the role of configuration systems in fostering inclusive, resilient, and locally grounded economic ecosystems. These metrics extend traditional evaluation dimensions by considering how configuration systems influence market fairness and the visibility of small and/or local component suppliers. Such metrics help evaluate whether configurations promote equitable access to market opportunities and support economically sustainable choices.

4.1. Support for Local Businesses

This metric quantifies the proportion of components in the configuration knowledge base that originate from small or local businesses. Let $\mathcal{C}_u \subseteq \mathcal{COMPS}$ denote the components from local or small-scale providers available to user u. The *Local Business Promotion Rate* (LBPR) can be defined as:

$$\text{LBPR} = \frac{\sum_{u \in \mathcal{U}} |\{\text{comp} \in \mathcal{COMPS} : \text{comp} \in \mathcal{C}_u\}|}{|\mathcal{U}|} \tag{10}$$

Higher LBPR values indicate that the configurator supports community-level economic development by promoting components supplied by small and/or local businesses.

4.2. Fairness in Exposure

Configuration systems can inadvertently concentrate exposure and revenue on a smaller subset of producers supplying specific types of components (i.e., we regard fairness in exposure as a context-dependent metric). The reasons behind could be, for example, specific variable (value) orderings specified in the underlying constraint solver. To foster economic fairness, we define fairness in the context of *component producer exposure* as *Component Producer Exposure Fairness* (CPEF):

$$\text{CPEF} = \frac{avgdist_2(\mathcal{P})}{maxdist_2(\mathcal{P})} \tag{11}$$

In this context, $\mathcal{P}$ denotes producers associated with components appearing in user configurations. The term $avgdist_2(\mathcal{P})$ represents the average pairwise distance in exposure counts between two producers, while $maxdist_2(\mathcal{P})$ is the maximum observed distance between any two producers in $\mathcal{P}$. Both are calculated based on how often a producer's components are presented to users across all configurations.

The discussed economic sustainability metrics can offer valuable insights into how configuration systems influence the distribution of economic value. These metrics help to evaluate whether a system promotes equitable market exposure and supports small or local businesses.

5. Cross-cutting Metrics

Cross-cutting sustainability metrics capture and assess the multifaceted effects of configurators spanning environmental, social, and economic aspects.

5.1. Sustainable User Behavior

This metric evaluates sustainability-related user interaction behavior. Let B_u represent the set (more precisely, the bag) of user behaviors over a specific time period (of user u) when interacting with the configurator (e.g., inspecting component details, reading explanations, or selecting a component). Furthermore, let $\mathcal{S}$ be a set of sustainable behaviors (e.g., selecting eco-friendly components). The *Sustainable Configuration Behavior Score* (SCBS) can be defined as:

$$\text{SCBS} = \frac{\sum_{u \in \mathcal{U}} |\{b \in B_u : b \in \mathcal{S}\}|}{\sum_{u \in \mathcal{U}} |B_u|} \tag{12}$$

Higher SCBS values indicate a higher degree of sustainability-related user interaction behaviors.

5.2. Interpretability of Configurations

Interpretability (*IntCS*) of configuration support is essential for enabling informed user decision-making. Let $\mathcal{E}_u$ denote the set of explanations e provided to user u over a specific time period (e.g., justifications for selecting a particular component *comp*), and let interpret(e) quantify the interpretability of explanation e. The *Average Explanation Interpretability* (IntCS) across all users can be defined as:

$$\text{IntCS} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{E}_u|} \sum_{e \in \mathcal{E}_u} \text{interpret}(e) \tag{13}$$

Interpretability may be estimated on the basis of explicit user feedback, information complexity scores, or automated assessments (e.g., using large language models).

5.3. Life Cycle Impact of Configurations

Life cycle impact analysis considers both, upstream and downstream effects in the production, distribution, usage, and disposal of configurations. Let LCIC(*comp*) denote the total estimated life cycle impact score for component *comp* (including aspects such as carbon footprint or the potential for reuse and recycling). The *Average Life Cycle Impact of Configurations* (AvgLCIC) can be defined as:

$$\text{AvgLCIC} = \frac{\sum_{\text{conf} \in \mathcal{CONFS}} \sum_{\text{comp} \in \text{conf}} \text{LCIC(comp)}}{\sum_{\text{conf} \in \mathcal{CONFS}} |\text{conf}|} \tag{14}$$

Lower values of AvgLCIC indicate that the configuration system favors components with lower ecological and social burdens throughout their life cycles.

The deployment of such cross-cutting sustainability metrics also depends on the availability and reliability of life cycle metadata for the involved components.

6. Challenges and Research Directions

Despite growing interest in sustainability-aware configuration [6, 10, 11, 12], several challenges hinder a widespread adoption and evaluation.

6.1. Multi-objective Optimization

The incorporation of sustainability goals into configuration systems often introduces trade-offs between traditional performance metrics (e.g., e.g., accuracy with regard to recommended/included components of a configuration) and sustainability-related outcomes. Formally, this leads to the optimization of a vector-valued objective function:

$$\max_{\theta} \quad \mathbf{FOPT}(\theta) = [\text{Accuracy}(\theta), \text{Sustainability}(\theta)] \tag{15}$$

where θ denotes the configuration parameters. This requires the definition of specific multi-objective optimization problems, typically resulting in Pareto-efficient solutions that aim to balance competing goals.

6.2. Data Availability and Labeling

Most sustainability metrics rely on fine-grained metadata, such as the carbon footprint of a component, ethical sourcing labels, or the classification of vendors (e.g., local or small-scale). Let $\mathcal{COMPS}$ be the set of components available in the configuration catalog, and let s_{comp} be a binary sustainability label for a component comp $\in \mathcal{COMPS}$. The share of labeled components is defined as:

$$\text{CLabelCov} = \frac{|\{\text{comp} \in \mathcal{COMPS} : s_{\text{comp}} \text{ is known}\}|}{|\mathcal{COMPS}|} \tag{16}$$

Low CLabelCov values limit the applicability and accuracy of sustainability-related evaluations.

7. Productive Usage of Metrics

Developers have to integrate sustainability indicators such as carbon footprint or component origin into evaluation workflows of the configuration environment. This includes activities such as extending existing logging frameworks with the goal to capture relevant data such as energy usage, component source information, and demographic data of users. In addition, configuration algorithms can be enhanced to prioritize sustainable choices, for example, by applying “green component variable value ordering” that favor environmentally friendly or ethically produced components. Beyond implementation, companies have the opportunity to promote transparency by reporting the sustainability performance of their configuration engines.

8. Conclusions

Sustainability-oriented evaluation metrics are essential for advancing configuration systems beyond conventional performance criteria. By embedding environmental, social, and economic considerations into system assessment, these metrics help to align the development of configuration systems with global sustainability goals, particularly those outlined in the United Nations Sustainable Development Goals (SDGs). Configuration systems have the potential to promote eco-friendly component selection, ensure equitable access to configurable solutions, and support local and responsible value chains. With this, these systems can contribute to more sustainable forms of product customization. A central focus of our future work will be to provide software components that will support the application of our proposed metrics in real-world contexts.

Declaration on Generative AI

While preparing this work, the author(s) used ChatGPT-4 (GPT-4-turbo) and Grammarly to check grammar and spelling and improve formulations. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] A. Felfernig, L. Hotz, C. Bagley, J. Tiihonen, Knowledge-based Configuration: From Research to Business Cases, 1 ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[2] D. Sabin, R. Weigel, Product configuration frameworks-a survey, IEEE Int. Sys. and their Applications 13 (1998) 42–49. doi:10.1109/5254.708432.

[3] M. Stumptner, An overview of knowledge-based configuration, AI Commun. 10 (1997) 111–125.

[4] A. Falkner, A. Felfernig, A. Haag, Recommendation technologies for configurable products, AI Mag. 32 (2011) 99–108. doi:10.1609/aimag.v32i3.2369.

[5] S. Lubos, V.-M. Le, A. Felfernig, T. N. T. Tran, Analysis operations for constraint-based recommender systems, in: RecSys'23, RecSys '23, ACM, New York, NY, USA, 2023, pp. 709–714. doi:10.1145/3604915.3608819.

[6] C. Fagarasan, C. Cristea, M. Cristea, O. Popa, A. Pisla, Integrating sustainability metrics into project and portfolio performance assessment in agile software development: A data-driven scoring model, Sustainability 15 (2023) 13139. doi:10.3390/su151713139.

[7] A. Felfernig, M. Wundara, T. N. T. Tran, S. Polat-Erdeniz, S. Lubos, M. El Mansi, D. Garber, V.-M. Le, Recommender systems for sustainability: overview and research issues, Frontiers in Big Data 63 (2023). doi:10.3389/fdata.2023.1284511.

[8] A. Konys, Knowledge based approach to sustainability assessment, Procedia Computer Science 176 (2020) 3626–3635. doi:10.1016/j.procs.2020.09.023, 24th Int. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems.

[9] H. Koziolek, Sustainability evaluation of software architectures: a systematic review, QoSA-ISARCS '11, ACM, New York, NY, USA, 2011, pp. 3–12. doi:10.1145/2000259.2000263.

[10] S. Lubos, A. Felfernig, L. Hotz, T. Tran, S. Polat-Erdeniz, V.-M. Le, D. Garber, M. E. Mansi, Responsible configuration using llm-based sustainability-aware explanations, in: ConfWS2024, CEUR, 2024, pp. 68–73.

[11] K. Medini, T. Wuest, D. Romero, V. Laforest, Integrating sustainability considerations into product variety and portfolio management, Procedia CIRP 93 (2020) 605–609. doi:10.1016/j.procir.2020.04.147, 53rd CIRP Conf. on Manufacturing Sys.

[12] R. Wiezorek, N. Christensen, Integrating Sustainability Information in Configurators, in: M. Aldanondo, A. Falkner, A. Felfernig, M. Stettinger (Eds.), ConfWS'21, volume 2945, CEUR, 2021, pp. 65–72.

[13] G. Leitner, M. Stettinger, Configuration of domotic systems based on constraint solving, in: 26th ACM International Systems and Software Product Line Conference, SPLC '22, ACM, New York, NY, USA, 2022, pp. 145–149. doi:10.1145/3503229.3547045.

[14] A. Felfernig, M. Wundara, T. N. T. Tran, V.-M. Le, S. Lubos, S. Polat-Erdeniz, Sports recommender systems: overview and research directions, J. Intell. Inf. Syst. 62 (2024) 1125–1164. doi:10.1007/s10844-024-00857-w.

[15] T. N. Trang Tran, M. Atas, A. Felfernig, M. Stettinger, An overview of recommender systems in the healthy food domain, J. Intell. Inf. Syst. 50 (2018) 501–526. doi:10.1007/s10844-017-0469-0.

The Role of Generative AI in the Future of Smart Home Configuration*

Gerhard Leitner

University of Klagenfurt, Universitaetsstrasse 65-67, 9020 Klagenfurt, Austria

Abstract

Since the concept of the smart home was announced, several waves of ups and downs in its adoption have been observable, but the big breakthrough promised frequently has yet to happen. There are several reasons for that, which are addressed in this paper from the perspective of configuration problems. One key reason is the overwhelming complexity and dimensionality of smart home solutions, which are not easily graspable, particularly for laypersons in their role as homeowners or dwellers. Artificial intelligence (AI), specifically conversational generative AI / Large Language Models (LLMs), could help overcome the problem and contribute to the spread of these respective technologies. In this paper, the current possibilities and future potential are exemplified.

Keywords

Smart Home, Configuration, AI

1. Introduction

Since the announcement of the concept of smart homes by the Association of Homebuilders in 1984 [1], the spread of smart homes has experienced several ups and downs, but never the big breakthrough that was frequently promised in the last five decades. In comparison, around the same time as the concept of the smart home, in 1983, IBM introduced the personal computer. The penetration of the PC and its descendants is over 100%, meaning that, statistically, all of us have more than one PC, smartphone, and/or tablet. Compared to that, the percentage of smart home penetration is poor. What we consider real smart homes are living environments where at least two smart sub-infrastructures are interconnected. Closed ecosystems or island solutions, such as a power socket with a proprietary remote control or light bulbs that can be directly operated in a smart speaker's ecosystem, do not constitute appropriate examples in our understanding. According to [2], the penetration of smart environments that fulfill the described requirements is around 30%.

There are several reasons why the adoption of smart home technology, compared to the abovementioned success story of the PC, is low. One of them is probably that, because of the dimensionality (e.g. on sensor level, on component level, on utilization level) and complexity (e.g. direct control, (conventional) remote control, cross-infrastructure control via local gateways or clouds, interoperability) of smart home systems, they are difficult to grasp, specifically for laypersons, resulting in suboptimal adaptation and utilization of the respective technologies. This reminds of the original goal of Usability (Engineering) brought to the point on the cover of IEEE Computer magazine in 1992, designed by P. Simpson: To hide the complexity of a backend system from the user.[1] The spread of AI, more concretely, conversational generative AI/large language models (LLM) such as ChatGPT could bring a revolutionary change in the field, specifically for complex tasks such as configuration in a smart home context, resulting in a situation illustrated - of course by AI - in Figure 1

ConfWS'25: 27th International Workshop on Configuration, Oct 25–26, 2025, Bologna, Italy
gerhard.leitner@aau.at (G. Leitner)
https://www.aau.at/en/isys/ias/ (G. Leitner)
0000-0002-3084-0727 (G. Leitner)

[1] https://archive.org/details/computer-magazine-1992-03/mode/2up

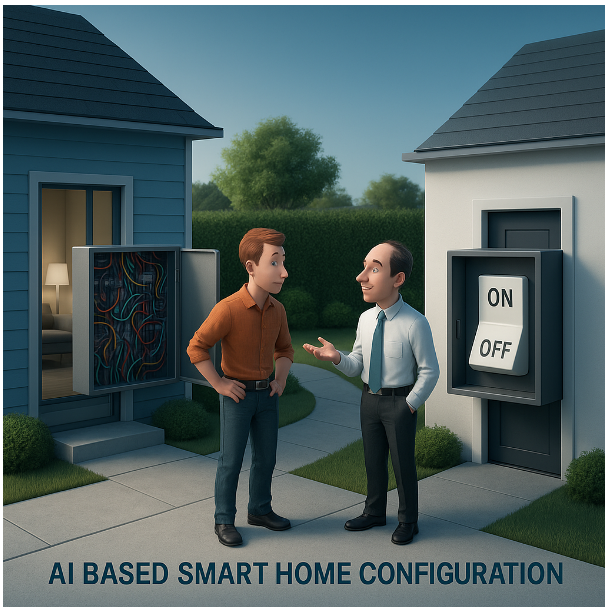


Figure 1: The ideal situation of smart home configuration, of course, also designed by AI (ChatGPT 4.0 Plus, July) - inspired by the cover design of IEEE Computer Magazine 1992/03 by P.Simpson

2. Motivation

A private home is characterized by individuality/customization. Homeowners and dwellers strive to adapt their homes to their needs and preferences. Even in living contexts that are somewhat standardized (such as apartment buildings characterized by repeating floor plans and room sketches), individualization needs are obvious; see, e.g., the example of the famous LeCorbusier Building in Berlin where the inhabitants *"behaved like moles"* to undermine the structural limitations [3]. In the past, end-consumer markets have addressed related needs by offering customers a wide range of choices and possibilities across various sectors, including home equipment and furniture. Interested persons could choose from several styles, price ranges, and materials, and combine furniture and accessories in almost arbitrary combinations, as long as basic constraints, such as dimensions/measures, were appropriately considered. The possibilities of individualization were also present with conventional components of the infrastructure, such as lighting, heating, and shading, which could be combined almost arbitrarily as long as the components met certain standards. For example, light bulbs could be exchanged (e.g., for LEDs that consume less energy) as long as the correct socket type out of a few commonly used ones was correctly identified.

With the advancement of smart home systems, the situation has become more challenging (or even unmanageable) for laypersons, as devices or components that physically fit might not function automatically due to software restrictions that are not always obvious or understandable. No wonder that consumers are reluctant to let technology into their homes; they are skeptical about being able to domesticate it.

To overcome this problem on a technical level, several attempts have been made in the past to address the challenges by establishing integrative platforms. These platforms, for example, include Home Assistant [4], OpenHAB [5], Domoticz [6], and, most recently, MATTER [7]. However, due to various factors, these platforms were and still are only usable for tech-savvy users who possess a genuine interest in the technology and its adaptation. The respective problems are illustrated, for example, in an article in an Austrian newspaper [8] titled *"The Matter Debacle - When Nothing Works in a smart home as it should"*. Consumers with an average interest in or knowledge of smart technology would still either have to completely abandon the technology or invest significant amounts of money to hire professionals to do the job. The individualization and customization possibilities that consumers have been familiar with in other market segments, for example, as observed in the spread of bricolage or furniture stores and their approach to private customers (DIY and IKEA's *"philosophy"*), are quite limited in the smart home domain. Customization would still require a high level of knowledge or expertise in different fields, specifically on a software application level [2]. In this paper, we aim to understand/define customization needs as configuration problems, focusing on related potentials, challenges, and limitations, as well as the role Artificial Intelligence (AI) could play in the near future to address these issues. We, in a simplified manner, differentiate between two categories of configuration-related tasks/ problems: 1) Configuration tasks relevant at the *"design time"* of a smart home, i.e. when a smart home is initially planned, and 2) Configuration tasks relevant at *"run time"*, i.e., when the smart home is already in operation. Before delving into the specifics of AI integration, we exemplify state-of-the-art approaches to such configuration tasks from our own work.

3. Pre-AI Smart Home Configuration

The variants of smartness offered are vast; however, the possibilities for adopting the respective technology for an average end consumer are relatively limited. Related information is available, but it is heavily distributed across various websites, community forums, brochures, and other sources. The majority of the offered solutions are based on single-manufacturer systems characterized by several shortcomings. First, the solutions are presented from the supplier's perspective, emphasizing the functional range of the products and rarely taking into account the user's perspective (e.g., regarding the characteristics of their living environments and needs). A future requirement would therefore be to increase the overlap between available functionality and individualization and customization needs. A second related shortcoming is that functionality not within the supplier's portfolio or product range is neither offered nor discussed. Attempts to overcome different aspects of this problem began before the advancements in AI, and the resulting solutions can be classified as falling somewhere between conventional approaches to smart home systems and prospective AI-based tools. These solutions offered a configuration based on the integration of smart components from different manufacturers (as well as their descriptions and characteristics), i.e. on linking information and possibilities that were very scattered over different online sources before these tools became available.

3.1. Example Configuration - Design Time

To help specifically naïve users better understand the respective possibilities, the idea of the configurator solution presented below is to guide them through a configuration task on the basis of the user's own floor plan, identifying/selecting conventional equipment present in their household and smartifying it with components that support certain needs. A secondary goal of the approach is to educate users on smart home functionality by showcasing different possibilities and comparing their pros and cons (e.g., in terms of installation effort, complexity, or price). The Figure 2 shows a snippet of a configurator system developed in the course of a Master's Thesis [9], which represents a further development of our past work [10, 11].

The system is based on a dialogue that starts by asking users about their smartness-related needs, such as increased comfort (through remote management with a smartphone), energy savings, and safety (e.g., burglar prevention, activity deviation recognition). Based on this initial selection, a backend system pre-computes appropriate example solutions. In the example, the user has selected energy savings, and the system and user have cooperatively identified an existing radiator in the living room as equipment that should be made smarter. The system proposes connecting a compatible Shelly smart thermostat to this radiator. The process can be repeated for each room and a number of equipment items and components to generate a satisfactory solution. In most cases, however, smart components are available from different manufacturers. To ease the choice between them, the system would analyze the necessity of add-ons (e.g., gateways, adaptors) and contain explanations and ratings, which could

[2]In this position statement, the legal restrictions and requirements, e.g. certificates to be allowed to integrate components in electrical wiring, are not addressed explicitly but are, of course, relevant.

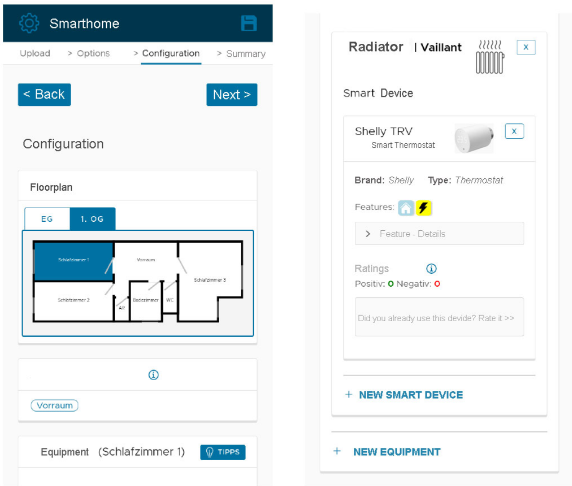


Figure 2: Example Configuration Step, adapted from Leustik, 2025

stem from other consumers or professionals who have experience with certain components. This add-on information should help in identifying the smart components that optimally meet the user's requirements.

3.2. Example configuration - Run time

When consumers are in a comfortable situation where their smart system works as needed and expected, situations may arise when functionality needs to be changed. In our view, this situation would constitute a reconfiguration problem, with the central goal of reusing existing infrastructure and, in this way, also contributing to sustainability. For example, a family member is an adult daughter who started studying abroad, resulting in significantly changed heating requirements in certain rooms. Current smart thermostats typically allow for local programming or switching between pre-programmed standard settings (e.g. weekdays, weekends, holidays). Therefore, simply selecting "holiday" for the rooms the daughter typically uses when she is absent would likely resolve the issue. Instructions for implementing these changes could be provided in the configurator shown in 2, as in the example above, provided that all components and their specifications have been integrated in the floor plan.

Another example could be that the inhabitants are not satisfied with their lighting situation; they want to exchange a light source that is already smart but only offers an *"on* and an *"off"* status for a dimmable light source. Such a change would already initiate a configuration problem with different variants; for example, would only the light bulbs be exchanged for dimmable ones, and the lamp itself would be kept? This approach would probably require exchanging the light switch for a dimmer; if the respective light can be switched from several positions (as this is a typical case in many living rooms), the switch/dimmer components would have to be coordinated, for example, by centrally managing dimming from a device in the household's fuse box. This basic use case already involves several configuration problems, and we have not even touched on software aspects (e.g., relevant when the goal is to allow dimming from a smartphone). These kinds of problems can be not only complex and confusing but also elaborate and expensive (e.g., when hiring professionals). Moreover, in a critical view of the pre-AI configuration approach discussed above, the system would require a comprehensive knowledge base containing all alternatives (e.g., dimmers in the room, in the fuse box, or elsewhere) and would need to visualize/compare the pros and cons in an understandable manner.

4. AI-based smart home configuration

4.1. Related work

Already before the current hype of generative AI / LLMs, AI was thematized in the context of smart homes. For example, Kastner et al. [12] emphasized the potential of AI-based automation (in the domain of energy efficiency) based on Artificial Neural Networks (ANN) in 2010. In 2020, Bicakci [13] highlighted the relevance of AI-generated algorithms used for automating smart home functionality in simulated environments, while also pointing out the potential limitations of such algorithms in real-world environments (e.g., in terms of errors).

Kopytko et al.[14] considered the connection between smart homes and AI as a promising area for new implementations/applications in 2018, for example, to overcome problems related to the variety/diversity/incompatibility of offered solutions. Jahanbakhsh et al. [15] 2025, propose retrieval-augmented generation of LLM for the automation of daily routines, or as they put it, *"smart-home orchestration"*. Adaptive automation is exemplified on the basis of OSGi-based[3] smart home platforms in combination with LLMs. Anik et al. [16] show the possibilities and limitations of automated configuration/programming of smart home functions in the context of YAML(*"a human-readable data serialization language"*)/Home Assistant. Although not explicitly addressed, the work can be considered relevant for deriving use cases for the utilization of AI (specifically in the context of platforms such as Home Assistant).

The explicit utilization of conversational generative AI, to our knowledge, is not addressed in the related literature in the context of Smart Home Configuration. However, there are a few works from related fields. One example from the field of CSP (Constraint satisfaction problems) by [17] analyzes the application of LLMs for constraint modelling. Another example from the field of conceptual modelling is provided by Fill et al. [18], who used ChatGPT to generate ER (entity relationship) diagrams.

4.2. AI at Smart Home Design Time

We took the example use case presented in section 3 and asked different AI engines for solutions based on the following prompt:

```
I have a Vaillant radiator in my living room and I want to make it smart. What
    possibilities do I have?
```

Listing 1: Prompt "Smart Heating"

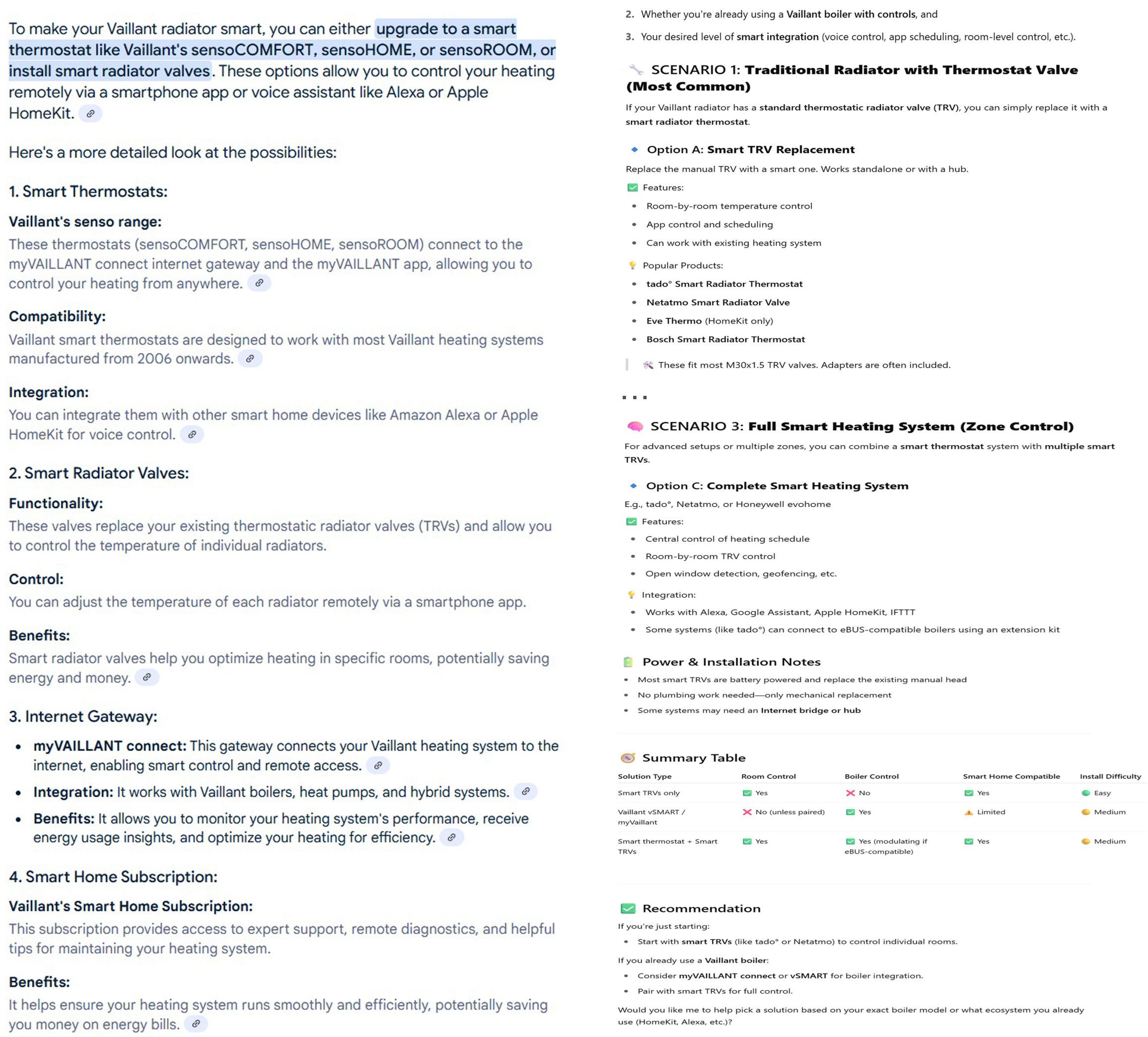
The answers of Google Gemini (Chrome Browser Integration) and Chatgpt 4.0 Dialogue Interface are as follows:

The provided answers differentiate themselves by several characteristics. Gemini directly focuses on advanced smartness, concretely, an integration in MyVaillant. The benefits are motivated, but not the disadvantages (e.g. being a closed system). In this regard, ChatGPT provides a broader variety of solutions, starting by proposing simple and independently working smart thermostats from different manufacturers, also mentioning the manufacturer platform MyVaillant (this part was exchanged in the figure by **"..."**, because it is comparable to the information provided by Gemini), and finally discussing the possibility of integration in cross-platform solutions (e.g. Alexa, Homekit, etc.). What can also be considered positive is showing the pros and cons of the proposed solutions in an overview table.

4.3. AI at Smart home Run Time

A suitable example, in a way similar to the dimmer problem described above, was brought to the author's attention and evaluated in the context of this paper. A homeowner who already utilizes a smart home environment (Alexa) wants to integrate another smart lighting system in a new living room wall unit

[3] OSGi Alliance (formerly Open Service Gateway Initiative) - a Java-based component platform that eases the development of complex systems, such as smart homes



To make your Vaillant radiator smart, you can either upgrade to a smart thermostat like Vaillant's sensoCOMFORT, sensoHOME, or sensoROOM, or install smart radiator valves. These options allow you to control your heating remotely via a smartphone app or voice assistant like Alexa or Apple HomeKit.

Here's a more detailed look at the possibilities:

1. Smart Thermostats:

Vaillant's senso range:

These thermostats (sensoCOMFORT, sensoHOME, sensoROOM) connect to the myVAILLANT connect internet gateway and the myVAILLANT app, allowing you to control your heating from anywhere.

Compatibility:

Vaillant smart thermostats are designed to work with most Vaillant heating systems manufactured from 2006 onwards.

Integration:

You can integrate them with other smart home devices like Amazon Alexa or Apple HomeKit for voice control.

2. Smart Radiator Valves:

Functionality:

These valves replace your existing thermostatic radiator valves (TRVs) and allow you to control the temperature of individual radiators.

Control:

You can adjust the temperature of each radiator remotely via a smartphone app.

Benefits:

Smart radiator valves help you optimize heating in specific rooms, potentially saving energy and money.

3. Internet Gateway:

- **myVAILLANT connect:** This gateway connects your Vaillant heating system to the internet, enabling smart control and remote access.
- **Integration:** It works with Vaillant boilers, heat pumps, and hybrid systems.
- **Benefits:** It allows you to monitor your heating system's performance, receive energy usage insights, and optimize your heating for efficiency.

4. Smart Home Subscription:

Vaillant's Smart Home Subscription:

This subscription provides access to expert support, remote diagnostics, and helpful tips for maintaining your heating system.

Benefits:

It helps ensure your heating system runs smoothly and efficiently, potentially saving you money on energy bills.

2. Whether you're already using a **Vaillant boiler with controls**, and
3. Your desired level of **smart integration** (voice control, app scheduling, room-level control, etc.).

SCENARIO 1: **Traditional Radiator with Thermostat Valve (Most Common)**

If your Vaillant radiator has a **standard thermostatic radiator valve (TRV)**, you can simply replace it with a **smart radiator thermostat**.

Option A: **Smart TRV Replacement**

Replace the manual TRV with a smart one. Works standalone or with a hub.

Features:

- Room-by-room temperature control
- App control and scheduling
- Can work with existing heating system

Popular Products:

- **tado° Smart Radiator Thermostat**
- **Netatmo Smart Radiator Valve**
- **Eve Thermo** (HomeKit only)
- **Bosch Smart Radiator Thermostat**

> These fit most M30x1.5 TRV valves. Adapters are often included.

...

SCENARIO 3: **Full Smart Heating System (Zone Control)**

For advanced setups or multiple zones, you can combine a **smart thermostat** system with **multiple smart TRVs**.

Option C: **Complete Smart Heating System**

E.g., tado°, Netatmo, or Honeywell evohome

Features:

- Central control of heating schedule
- Room-by-room TRV control
- Open window detection, geofencing, etc.

Integration:

- Works with Alexa, Google Assistant, Apple HomeKit, IFTTT
- Some systems (like tado°) can connect to eBUS-compatible boilers using an extension kit

Power & Installation Notes

- Most smart TRVs are battery powered and replace the existing manual head
- No plumbing work needed—only mechanical replacement
- Some systems may need an **Internet bridge or hub**

Summary Table

Solution Type	Room Control	Boiler Control	Smart Home Compatible	Install Difficulty
Smart TRVs only	Yes	No	Yes	Easy
Vaillant vSMART / myVaillant	No (unless paired)	Yes	Limited	Medium
Smart thermostat + Smart TRVs	Yes	Yes (modulating if eBUS-compatible)	Yes	Medium

Recommendation

If you're just starting:

- Start with **smart TRVs** (like tado° or Netatmo) to control individual rooms.

If you already use a **Vaillant boiler**:

- Consider **myVAILLANT connect** or **vSMART** for boiler integration.
- Pair with smart TRVs for full control.

Would you like me to help pick a solution based on your exact boiler model or what ecosystem you already use (HomeKit, Alexa, etc.)?

Figure 3: Side-by-side comparison of results provided by Google Gemini and ChatGPT 4.0

to be purchased. Following the improved performance in the previous example, ChatGPT was asked to provide a proposal. The results look similar to the ones presented in Figure 3 and can be summarized as follows:

- The light source in question is named Mittled, but cannot be made smart directly. ChatGPT explains that a controller (Tradfri) is required.
- The system could then be controlled via a proprietary remote control or connected to a smart gateway, which exists in two versions.
- The gateway has to be allowed to access the Alexa ecosystem, Chatgpt describes the procedure to be performed in the Alexa app.
- Examples of possible Voice commands are provided to show how the new components could be controlled in the context of the existing environment
- Finally, a video showing the necessary steps and a summarizing list are provided.

The example underlines the potential of AI and also gives an idea of sustainability in the context of smart homes (by combining existing with new smart components).

5. Discussion and Conclusion

In this paper, we tried to exemplify the potential of conversational generative AI platforms in supporting smart home configuration tasks. The preliminary conclusions to be drawn are mixed in several aspects.

The solutions provided by AI platforms are impressive in terms of combining almost all information that is distributed over different online sources, and, in many cases, tedious to find and difficult to mentally connect in the past. We tried out several problem prompts (which are not completely presented in the paper due to space constraints), for example, asking for the possibilities of connecting/integrating smart home systems of different suppliers. One of them is based on OAuth [4], allowing the reciprocal exchange of data between different cloud platforms. The solution, which took us several days in the past, was provided by AI within seconds, including information on how to register with the two platforms to be connected, how OAuth works, etc.

As mentioned in the context of the work by [9], we initiated our efforts at a time when generative AI was not widely available, with the goal of integrating related information distributed across multiple sources and storing it locally. Due to the easy accessibility of LLMs, this problem, meanwhile, appears to be obsolete or solved. The educational aspect, which we tried to cover in our approaches, is also covered by AI tools appropriately, at least by some of them. In the examples shown, specifically ChatGPT adequately explains the pros and cons of proposed solutions. Because of the quantity and variety of sources the different AI platforms can access, it is probable that even solutions for exotic combinations of smart and conventional devices can be made possible with the help of AI, and in this way contribute to sustainability, because devices considered as old or outdated would still not have to be thrown away. Such a problem occurred in our past work, in a field study within the context of active and assisted living (AAL)[19]. We had to manually find a solution to connect 380V-operated kitchen stoves (which are more or less standard in Austria) to a smart home system. The benefits of this approach are sustainability (because still-working devices do not have to be exchanged) and, even more important, usability and user ecperience; because of allowing people to keep their familiar devices while still benefiting from smartness. Not surprisingly, AI already offers a solution for such problems as well.

However, some weaknesses/gaps of AI-based results can be identified and will probably inspire future research. The proposed alternatives still require a certain level of knowledge in the field to be able to evaluate their usefulness. For example, alternative switching components that are principally equivalent might require different backend infrastructures with significantly different complexity. One component could be able to directly communicate via Bluetooth/Wifi/ Zigbee, while the other is based on a proprietary local gateway and/or the cooperation of separate cloud systems. This is something that laypersons might not be able to fully evaluate, but would require the consultation of experts. This is another aspect that should be investigated in future work, based on the following aspect. The majority of solutions in the context of smart homes are based on electric devices, the installation of which requires qualification and certification. This task is (also in conventional settings) covered by local SMEs. These SMEs are in a difficult role in several aspects: They are the first address for the end consumers because of their reachability and expertise and, probably existing customer relationship. However, they are - as in the pre-smart eras - responsible for the correct installation of components, their function, and - in case of problems - their adjustment or repair. At present, they have to deal with the problem that smart components not only require knowledge in their core expertise (e.g. electrical engineering, electronics) but also a certain level of knowledge, if not even expertise, in the field of informatics (software development, parametrization and maintenance), which they are (on average) only limitedly trained in. This presents a specific challenge when customers request solutions that require functionality or components not included in the SME's standard portfolio. On a non-representative and scientifically sound level, the authors have observed that SMEs advise their customers against smart solutions due to the expectation that they will be held responsible for problems by the customer, for which they may not expect support from the supplier's side. The target group of SME would probably

[4](short for Open Authorization), a standard for granting access and data exchange between different web-based systems as an alternative to user/password-based access

have to be involved in future solutions based on AI, to sort of *"moderate"* the proposed solutions.

A final aspect is the representation of results. Our past approaches were, as this seems to be the state of the art in the field, based on floor plan representations of smart home solutions. Suppliers such as Gira, Bosch, Feelsmart (for a comparison see [9]) also base their configuration solutions on this approach. In future work, we will investigate how well AI performs based on pictorial representations of homes and how this influences the results.

Acknowledgments

This paper was produced in the context of the Project Mass Customization 4.0 (MC 4.0), funded by the European fund for regional development and Interreg V-A Italy-Austria 2014-2020. We thank the reviewers for their valuable comments and proposals for enhancing this paper.

Declaration on Generative AI

During the preparation of this work, the author used Google Gemini Chrome Plug-In and ChatGPT 4.x Prompt System for deriving the AI-related examples presented in this paper. Further, the author used Grammarly for typo and grammar correction and text adaptation. After using these tools/services, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] R. Harper, Inside the smart home, Springer Science & Business Media, 2006.
[2] Statista, Statista: Global smart home penetration 2019-2028, 2023. URL: https://www.statista.com/forecasts/887636/penetration-rate-of-smart-homes-in-the-world.
[3] A. Quickert, Das Wort haben die Benutzer im Corbusierhaus, 2012. URL: https://web.archive.org/web/20120903040955/http://www.tipberlin.de/kultur-und-freizeit-theater-und-buehne/das-wort-haben-die-benutzer-im-corbusierhaus.
[4] H. Assistant, Home assistant, 2025. URL: https://www.home-assistant.io/.
[5] OpenHAB, Open home automation bus (openhab), 2025. URL: https://www.openhab.org/.
[6] Domoticz, Domoticz, 2025. URL: https://www.domoticz.com/.
[7] MATTER, Matter, 2025. URL: https://csa-iot.org/all-solutions/matter/.
[8] A. Proschofsky, Das Matter-Debakel: Wenn im Smart Home absolut nichts funktioniert, wie es soll, 2024. URL: https://www.derstandard.at/story/3000000248254/das-matter-debakel-wenn-im-smart-home-absolut-nichts-funktioniert-wie-es-soll.
[9] D. Leustik, Smart Home, Master's thesis, University of Klagenfurt, Klagenfurt, Austria, 2025.
[10] M. Pum, Personalisierung von Smart-Home-Systemen mit modernen (Web-)Technologien, Master's thesis, University of Klagenfurt, Klagenfurt, Austria, 2015.
[11] Selectionarts, The domotics system for you (DS4U), 2022. URL: http://smarthome.selectionarts.com/?lang=en.
[12] W. Kastner, M. J. Kofler, C. Reinisch, Using ai to realize energy efficient yet comfortable smart homes, in: 2010 IEEE International Workshop on Factory Communication Systems Proceedings, IEEE, 2010, pp. 169–172.
[13] S. Bicakci, H. Gunes, Hybrid simulation system for testing artificial intelligence algorithms used in smart homes, Simulation Modelling Practice and Theory 102 (2020) 101993.
[14] V. Kopytko, L. Shevchuk, L. Yankovska, Z. Semchuk, R. Strilchuk, Smart home and artificial intelligence as environment for the implementation of new technologies, Traektoriâ Nauki= Path of Science 4 (2018) 2007–2012.

[15] N. Jahanbakhsh, M. Vega-Barbas, I. Pau, L. Elvira-Martín, H. Moosavi, C. García-Vázquez, Leveraging retrieval-augmented generation for automated smart home orchestration, Future Internet 17 (2025) 198.
[16] S. M. H. Anik, X. Gao, H. Zhong, X. Wang, N. Meng, Programming of automation configuration in smart home systems: Challenges and opportunities, ACM Transactions on Software Engineering and Methodology (2025).
[17] L. Hotz, C. Bähnisch, S. Lubos, A. Felfernig, A. Haag, J. Twiefel, Exploiting large language models for the automated generation of constraint satisfaction problems, Configuration (ConfWS 2024) co-located with the 30th (2024) 91.
[18] H.-G. Fill, P. Fettke, J. Köpke, Conceptual modeling and large language models: impressions from first experiments with chatgpt, Enterprise Modelling and Information Systems Architectures (EMISAJ) 18 (2023) 1–15.
[19] G. Leitner, A. Felfernig, A. J. Fercher, M. Hitz, Disseminating ambient assisted living in rural areas, Sensors 14 (2014) 13496–13531.