Production Engineering
https://doi.org/10.1007/s11740-022-01176-1

COMPUTER ASSISTED APPROACHES IN PRODUCTION ENGINEERING

Creation and validation of systems for product and process configuration based on data analysis

Alex Maximilian Frey[1] · Marvin Carl May[1] · Gisela Lanza[1]

Received: 12 September 2022 / Accepted: 28 November 2022

Abstract
In the course of increasing individualization of customer demand, configurable products are gaining importance. Nowadays, variant-specific bills of materials and routings for configurable products are created with the help of rule-based configuration systems, so-called low-level configuration systems. The rules and generic structures on which such configuration systems are based are created manually today. This is challenging because it can be difficult and sometimes impossible to directly transfer expert knowledge into those systems. Furthermore documents that have already been created by experts in the past such as bills of material and routings contain relevant information as well which may be exploited to compose configuration systems. However, in the literature, there are no approaches yet to systematically transfer expert knowledge into configuration systems or to consider existing documents. In addition, the creation of such configuration systems is prone to error due to their complexity. Although there are already numerous approaches to the formal testing of configuration systems, approaches based on data analysis to support the validation of such systems have not yet been considered. Therefore, in this paper an approach is presented to automatically create low-level configuration systems by means of exemplary variant-specific bill of materials and routings using machine learning. The super bill of materials and the super routing as well as the dependencies between the product characteristics and the components respectively the operations are learned. Furthermore, it is shown how errors in the input data as well as errors in the resulting low-level configuration system can be detected by means of anomaly detection.

Keywords Product configuration · Process configuration · Machine learning · Mass customization · Bill of materials · Routings

1 Introduction

The individualization of customer demand is a continuing trend [1]. In order to achieve the vision of mass customization, the production of customized products at costs similar to mass production [2] on the one hand must be combined with economies of scale on the other. This is the advantage of configurable products [3]. Configurable products are products that meet customer requirements with predefined technical solutions within a predefined solution space [4]. The Configurator Database (https://www.configurator-database.com/) represents the entirety of product configurators available online and shows how diverse the industries are in which configurable products are used today, in B2B as well as in B2C. Comparing the number of 1636 configurators currently listed with 1200 in 2017 and 1050 in 2015 [5] an upward trend can be seen in the use of configurable products.

The variants of configurable products are described in practice by predefined characteristics, such as the throughput of a pump, for which the customer can choose between predefined values, such as 45, 90 or 160 gallons per minute. The combinatorics of the characteristics' values give rise to numerous configuration possibilities, e.g., in the order of 10^{21} in the automotive industry [6]. This is a challenge for production, since production-related documents such as, in particular, manufacturing bills of materials (manufacturing BOM, mBOM) and routings are required for each variant. Therefore, systems for the automatic configuration of variants, variant-specific bills of materials and variant-specific routings have become established in practice, especially as

✉ Alex Maximilian Frey
alex.frey@kit.edu

1 wbk-Institute of Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany

Published online: 13 December 2022

part of ERP systems [7]. These configuration systems represent a form of rule-based expert systems that, among other things, establish a relationship between characteristic values selected by the customer and the design of the associated BOM and routing [8]. For certain use cases, 18,000 rules exist for BOM elaboration alone [9]. Creating such expert systems is costly and error-prone [10], with one challenge being the transformation of domain knowledge into a formal model [11]. Due to the extensive use of configuration systems in industry, this is a problem with high practical importance. In the following, an approach is presented to support the creation of systems for the configuration of BOMs and routings, referred to in the literature as product and process configuration. Furthermore, it is shown how errors in the input data as well as errors in the resulting configuration system can be detected by means of anomaly detection. For this purpose, methods from the state of the art are combined and suitably complemented by own methods.

2 State of the art and state of research

2.1 The challenge of transferring expert knowledge into models

According to Haug et al. [11], the process of creating configuration systems consists first of all of transferring knowledge from domain experts to knowledge engineers (elicitation), who then transfer it via an analysis model (translation) to a design model (formalization) before finally implementing it in a configuration program. Numerous works show challenges in transferring expert knowledge, referred to as externalization of tacit knowledge (see [12] for a comprehensive review). This term corresponds to Nonaka and Takeuchi's SECI model according to which tacit knowledge can be transformed into explicit knowledge through externalization [13]. Besides fundamental criticism of the empirical validation of the SECI model by Gourlay [14], Tsoukas [15] shows that the SECI model is based on a wrong interpretation of tacit knowledge according to Polanyi [16] and that tacit knowledge should actually be understood as knowledge that cannot be transformed into explicit knowledge. According to this use of the term, which also corresponds to Haug's [12] understanding, tacit knowledge can only be represented by a simplifying model which mimics decisions based on tacit knowledge but is not a one-to-one depiction of the tacit knowledge. Furthermore, Haug [12] shows that tacit knowledge in this sense plays a minor role for the creation of configuration systems. Thus, the overarching challenges in transferring expert knowledge as described in the literature are to transfer knowledge that is externalizable, but whose externalization is difficult. Aldanondo et al. [17], based on 10 use cases, attribute this to communication between the domain expert and the knowledge engineer. The coincidence of the two roles is desirable, but hardly realizable in reality, since the qualification effort is high in each case. A method to transfer domain knowledge directly into a configuration system without the domain expert needing knowledge of formal modeling therefore seems promising. A secondary challenge is to represent tacit knowledge in a model even if this is of subordinate importance. Decisions made by experts may be based on tacit knowledge even if this knowledge cannot directly be transferred. Therefore, Haug [12] mentions the approach of observing decisions of domain experts to create a model that mimics those decisions.

2.2 High and low level configuration systems

Configuration systems can be divided by their function into high-level and low-level configuration systems (HLCS and LLCS, respectively) [18]. In high-level configuration systems, the characteristics to be defined by the customer and their predefined values are stored [18]. Furthermore, there are constraints regarding the combination of characteristic values. I.e., certain characteristic values cannot be combined with other characteristic values or must always be combined with certain other characteristic values [18]. E.g., the customer cannot order a convertible car with a sunroof. High-level configuration systems allow to check the permissibility of the characteristic values selected by the customer or to suggest only permissible combinations of values to the customer. They, thus, form the basis for interaction with the customer. The product variant is defined by specifying the characteristic values in the high-level configuration. Low-level configuration systems, on the other hand, enable the creation of production-related documents, such as bills of materials and routings in particular, depending on the selected variant. Low-level configuration systems store super BOMs and super routings on the one hand and dependencies on the other hand [8]. Super BOMs are BOMs that contain all components that can occur in the configurable product across all variants. Similarly, super routings are routings that contain all operations that can occur in the configurable product across all variants. Dependencies are Boolean expressions that establish a relationship between the characteristics of a variant and the presence of a particular element in the BOM or routing. The focus of the presented approach is on the creation and validation of low-level configuration systems i.e., specifically on the creation of super BOMs, super routings, and dependencies. An approach for the creation and validation specifically of low-level configuration systems does not exist in the literature, which is why all approaches for the creation, validation or verification of models that allow the derivation of BOMs or routings are included in the following overview.

2.3 Previous approaches for the creation and validation of systems for product and process configuration

The creation of models for the derivation of BOMs is done in the literature via super BOMs. Numerous approaches exist that address an efficient representation of super BOMs e.g., with plan skeletons [19] or object oriented [20]. Kashkoush and ElMaraghy [21] show an approach to generate a master assembly sequence that is comparable to a super BOM. It is created by calculating, through mathematical optimization, the master assembly sequence that has the smallest distance to all variant-specific assembly sequences entered. Here, however, alternatives regarding the structure cannot be depicted. Furthermore, an approach by Moussa and ElMaraghy [22] exists for the creation of master assembly sequences. However, when transferring this approach to the creation of super BOMs, the assemblies contained in the BOMs would not be preserved, which has the disadvantage for configurable products that assembly-specific configurable routings can no longer be assigned to the assemblies of the super BOM.

In the literature, the creation of models for generating routings can be rule-based (i.e., the user enters the model manually) or data-based (i.e., the model is created based on data using data analysis). On the one hand, rule-based creation is done for part manufacturing applications based on manufacturing feature models [23–25]. Here, a technology database is built that assigns a manufacturing technology and a processing station to individual manufacturing features. Inference is performed by creating a feature model for a new part—e.g., via feature recognition [26]—and deriving a routing via the technology database. On the other hand, rule-based creation for assembly applications is based on CAD assembly models [27–30]. Here, the assembly sequence and assembly technology for an assembly are determined based on the interaction of its individual CAD models. All approaches assume manual creation of the underlying set of rules and do not address the difficulties described above when transferring expert knowledge into models. They further do not consider that previously created BOMs and routings may contain relevant information that could be used to deduct rules. Data-based approaches generate models for the creation of routings based on routings already created in the past. For this purpose, there are approaches that also assume feature models and make data-based assignments of manufacturing technologies and processing stations to features [31–33]. In addition, approaches exist that assume that product variants are described by parameters [34, 35]. Even if this is not the focus of these approaches, they can in principle be applied to configurable products whose variants are also described by one-dimensional characteristics. However, low-level configuration systems cannot be directly derived from the models of these approaches, as they are non-interpretable machine learning models in the sense of Rudin [36]. In addition, there are approaches for learning assembly precedence graphs [37–39], which have some analogy to a super routing. However, no dependencies are learned, which is why no variant-specific inference is possible. A different approach is taken by Tseng et al. [40] and Denkena et al. [41], who propose routings and BOMs for variants not yet produced using case-based reasoning based on variants already produced. However, the creation of routings and BOMs is done by the user. Lastly, approaches from decision mining [42–45], in which process models with variant-specific branches are learned, are particularly relevant for the presented work. This is analogous to a super routing with associated dependencies. All in all the existing data-based approaches can only consider knowledge that is already documented in existing bill of materials, routings or case descriptions. Knowledge that is already explicitly described in the form of rules, cannot be taken into account, nor can expert knowledge that has not yet been externalized. Furthermore the results of those approaches aren't LLCS as they are common in industry.

When checking models, a distinction must be made between verification and validation. Verification ensures that a model is correct in itself, i.e., that it does not contain any logical contradictions [46]. Validation, however, ensures that the model correctly represents reality [46]. Several approaches to verification of high-level configuration systems and of product configuration systems exist in the literature [47–49]. In this context, methods for checking the satisfiability of Boolean expressions are usually used (so-called SAT problem) to ensure that contradictory expressions of BOMs, such as double assignment of a BOM item, are not satisfiable. However, whether the resulting BOMs are correct with respect to reality cannot be checked automatically so far. To the best of the authors' knowledge, there are no approaches for checking process configuration systems and, thus, in particular, no approaches for their validation.

Table 1 summarizes on the one hand the necessary aspects to be covered by the presented approach and on the other hand to what extent existing approaches already take them into account. First, it should be possible to create inference models with the existing approach (1). Both, the data used for this purpose and the resulting model shall be validated (2, 3). For the sake of completeness, previous work is also classified with respect to verification of models (4), although this is not considered in the presented approach. The resulting LLCS should be suitable to predict the elements and the structure of the bill of materials and the routings of a variant (5–8) and, thus, enable a complete creation of production related documents. For the creation of LLCS different types of knowledge shall be considered which have been differentiated in Sect. 2.1. Firstly there is

Table 1 Relevant aspects for the approach at hand and overview on the state of the art

		Generate inference models (1)	Validate input data (2)	Validate resulting model (3)	PVerify resulting model (4)	Predict BOM elements (5)	Predict BOM structure (6)	Predict Process elements (7)	Process structure (8)	Knowledge in form of rules is implementable(9)	Existing documents can be used (10)	Expert knowledge difficult to transfer or tacit (11)	Interpretable model (12)
Rule based approaches	[28]	◐	○	○	○	◐	◐	◐	●	◐	○	○	◐
	[29]	◐	○	○	○	◐	◐	◐	●	◐	○	○	◐
	[27]	◐	○	○	○	◐	◐	◐	●	◐	○	○	◐
	[30]	◐	○	○	○	◐	◐	◐	●	◐	○	○	◐
	[23]	◐	○	○	○	○	○	●	●	◐	○	○	●
	[24]	○	○	○	○	○	○	●	●	◐	○	○	●
	[25]	○	○	○	○	○	○	●	●	●	○	○	●
	[48]	○	○	○	●	○	○	●	●	○	○	○	○
	[47]	○	○	○	●	●	●	○	○	○	○	○	○
	[49]	○	○	○	●	●	●	○	○	○	○	○	○
Data based approaches	[34]	●	○	○	○	○	○	●	●	○	●	○	○
	[35]	●	○	○	○	○	○	●	●	○	●	○	◐
	[32]	●	○	○	○	○	○	●	●	○	●	○	○
	[31]	●	○	○	○	○	○	●	●	○	●	○	○
	[33]	●	○	○	○	○	○	●	●	○	●	○	○
	[38]	◐	◐	○	○	◐	◐	●	●	○	●	○	●
	[39]	◐	○	○	○	◐	◐	●	●	○	●	○	●
	[37]	◐	○	○	○	◐	◐	●	●	○	●	○	●
	[41]	○	○	○	○	○	○	●	○	○	●	○	●
	[40]	○	○	○	○	○	○	●	●	○	●	○	●
	[43]	○	○	○	○	○	○	●	●	○	◐	○	●
	[42]	●	○	○	○	○	○	●	●	○	●	○	●
	[44]	○	○	○	○	○	○	●	●	○	◐	○	●
	[45]	○	○	○	○	○	○	●	●	○	◐	○	●
	[21]	◐	○	○	○	●	●	○	○	○	◐	○	●
	[22]	◐	○	○	○	●	●	○	○	○	◐	○	●
	[20]	◐	○	○	○	●	●	○	○	○	◐	○	●
	[19]	◐	○	○	○	●	●	○	○	○	◐	○	●

●Considered

◐Partially considered

○Not considered

knowledge which is already externalized or is readily available for externalization in the form of rules (9). Secondly there are existing documents (10) i.e., BOMs and routings which contain relevant information and may have been created by experts using their tacit knowledge. Thirdly there is expert knowledge (11) which may be difficult to transfer into a model or which is tacit and can therefore not directly be transferred but only indirectly as described in Sect. 2.1. Last, the resulting LLCS should be interpretable (12) i.e., consist of super BOM and super routings as well as rules as is common in industry and not be replaced by e.g., a black box machine learning model. This shall ensure the acceptance of the approach in practice.

Altogether it can be stated that in the literature there are no approaches yet.

- For the creation of systems for product and process configuration that consider knowledge in form of rules, hardly transferable or tacit expert knowledge and existing documents, or
- For the systematic validation of such configuration systems in the sense described above as well as the validation of the data used to create such configuration systems.

In the presented approach, these aspects are to be implemented and, thus, these deficits are to be remedied. The verification of the resulting models may also require further research, especially with respect to process configuration, that is not addressed in the paper at hand.

3 Approach

Based on the deficits identified in the state of the art, this chapter presents an approach for the creation and validation of low-level configuration systems for BOMs and routings based on different types of knowledge as described above. The overall approach is illustrated in Fig. 1. The modules of the approach are detailed in the following section. In contrast to existing approaches according to the state of the art, in the presented approach LLCS are not to be built by entering rules and structures, but by entering variant-specific BOMs and routings from which, in turn, super BOMs, super routings and dependencies are to be derived. Since experts can use both knowledge that is difficult to transfer and tacit knowledge for the creation of variant-specific BOMs and routings, these forms of knowledge can also be indirectly used in the creation of LLCS. As described above, no direct transfer of tacit knowledge takes place in this way and the resulting LLCS is not a one-to-one representation of the expert's tacit knowledge, but rather mimics decisions the expert makes based on tacit knowledge. For cases where the introduction of a HLCS precedes the introduction of the LLCS, some data is already available, i.e., BOMs and routings have already been created manually for certain variants. For cases where the HLCS and LLCS are introduced simultaneously, this data does not exist for the time being. In principle, this data can be generated in response to customer orders. However, the creation of BOMs and routings

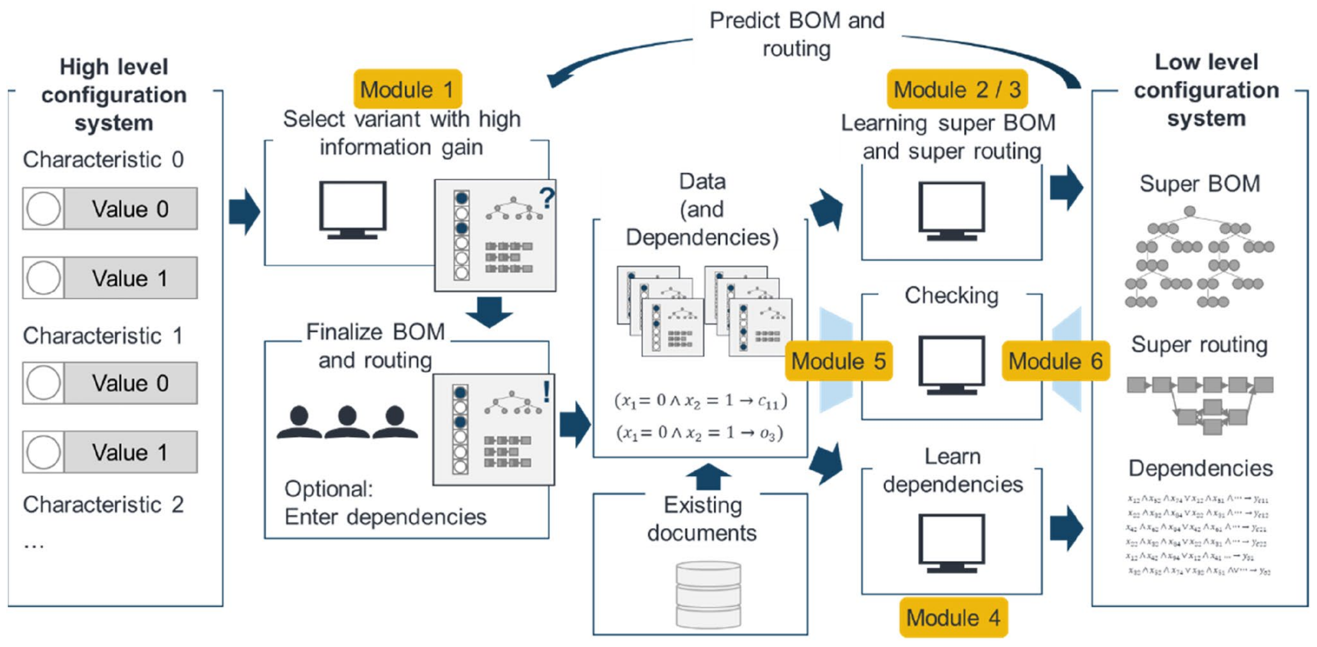

Fig. 1 Overall approach with modules

would then fall within the order lead time. Alternatively, the LLCS can be created before the introduction of the configurable product by having experts create BOMs and routings for systematically selected variants in advance, from which the LLCS can be derived as a whole, as shown in Fig. 1 (module 1). The data created in this way is transferred to a data pool. Based on this data pool, the super BOMs (module 2) and super routings (module 3) as well as the dependencies (module 4) can now be determined by means of data analysis. This process takes place iteratively in an interaction between a computer program and one or more experts. Individual variants are repeatedly selected, the associated BOMs and routings are fed into the data pool, and the LLCS is generated. Thus, the LLCS can already be used during its generation to suggest BOMs and routings for selected variants and thus support the experts. Since there is still knowledge that is explicitly known to the experts, it is also possible, although not necessary, to enrich the data pool with manually entered dependencies, which are also taken into account during the creation of the LLCS. By replacing the manual creation of the LLCS with the input of BOMs and routings, errors when entering rules and structures into the LLCS can be avoided. Nevertheless, errors can occur during the creation of variant-specific BOMs and routings. These errors shall be found by validating the data i.e. the variant-specific BOMs and routings (module 5). Since dependencies can be entered in addition to data, errors can occur in the resulting LLCS. These can be found on the one hand by verifying the model according to the state of the art and on the other hand by validating the model using anomaly detection (module 6). In the following sections, we will use the following notation. x_i denotes the value of characteristics i of the product. y_{c_j} denotes the presence of component c_j from the super BOM in the variant-specific BOM. y_{o_k} denotes the presence of operation o_k from the super routing in the variant-specific routing. Dependencies may then be written as Boolean expressions such as $(x_1 = 1) \vee \left(\left(x_3 < 4\right) \wedge (x_4 = 3)\right) \rightarrow y_{c_1}$ meaning that if characteristic 1 is chosen to be equal 1, characteristic 3 is chosen to be lower than 4 and characteristic 4 is chosen to be 3, component 1 is part of the variant-specific BOM.

3.1 Systematic selection of variants for the generation of the LLCS (module 1)

Each selected variant is accompanied by an effort for the expert team for creating or checking the associated bill of materials and the associated routings. Therefore, the number of necessary variants for achieving a sufficiently precise LLCS should be kept as low as possible. For the creation of the LLCS, the BOMs and routings take the function of labels in supervised learning. Thus, the task of *Module* 1 can be generalized to the request of using as few labels as possible for the creation of a machine learning model with high prediction quality. Thus, active learning strategies can be used to systematically select those variants that promise the highest possible information gain. Those variants can then be handed over to the expert team for creating the variant specific BOM and routings. There are several active learning strategies in the literature, but not all of them are suitable for the present case (see Table 2).

The LLCS is an inference model based on Boolean expressions. Therefore, it can only predict if e.g., a certain item is part of the BOM or not but is not able to give probabilities for predictions. Thus, strategies based on uncertainty sampling that require information on probabilities for different predictions are excluded. Likewise, strategies based on variance reduction and expected error reduction, which require analytical optimization procedures and thus analytical models, are excluded because an LLCS is not an analytical model. All other methods are applicable. Density weighted strategies can be used when constraints are present in the HLCS and thus an inhomogeneous distribution of variants exists over the space spanned by the product characteristics. Otherwise, the space is homogeneous and the density is the same everywhere. As an example demonstration of the approach, a configuration of pumps from the company Liquiflo is used here and in the following, as in the work of Kashkoush and ElMaraghy [50]. For these pumps, the customer can choose between type 620 $(x_1 = 0)$

Table 2 Methods of active learning and their applicability for selecting variants for generating LLCS (Settles 2009; Wu et al. 2019)

Active learning method	Query the sample…	Applicability
Uncertainty sampling	… for which the model has the highest uncertainty	Not applicable
Query-by-committee	… for which two or more separately trained models disagree the most	Applicable
Expected model change	… which leads to the biggest expected change in the model	Applicable
Variance reduction/Expected error reduction	… which is expected to lead to highest variance or expected error reduction of the model	Not applicable
Density weighted	… from the region with the highest density	Applicable
Greedy sampling	… furthest away from all previously queried samples	Applicable

and 621 ($x_1 = 1$), between a single seal ($x_2 = 0$), a double seal ($x_2 = 1$), a packing seal ($x_2 = 2$) and a sealless magnetic coupling ($x_2 = 3$). Other features include the presence of a replacement cartridge (x_3), base plate mounting (x_4), S-adapter (x_5), power frame (x_6) and accessories & spare parts (x_7). Full details can be found on the manufacturer's website (http://www.liquiflo.com/v2/centro/centry/index.html) and in the work of Kashkoush and ElMaraghy [50]. From the HLCS $(x_1 = 1) \wedge ((x_2 = 1) \vee (x_2 = 2)) \leftrightarrow False$ is required i.e. type 621 cannot be combined with a double seal or a packing seal. If, for example, 4 variants are selected according to the greedy sampling, starting with type 621 single seal, the data pool contains the variants ($x_1 = 1, x_2 = 0, \ldots$), ($x_1 = 0, x_2 = 1, \ldots$), ($x_1 = 0, x_2 = 2, \ldots$), ($x_1 = 1, x_2 = 3, \ldots$), hereafter referred to as (1,0), (0,1) etc. according to the values of their first two characteristics, with the associated BOMs and routings.

3.2 Learning of super BOMs (module 2)

When learning super BOMs, the aim is to create a super BOM that contains every variant-specific BOM available in the data pool as one specification. In addition, the variables are to be defined by whose assignment a variant-specific BOM can be created from the super BOM. The method presented in the following is intended to eliminate the deficits for the creation of super BOMs described in the state of the art. Figure 2 shows the BOMs of the example case. Figure 3 illustrates the procedure by means of the variant-specific BOMs (0, 1) and (0, 2) as introduced in chapter 3.1. The steps described in the following are shown as horizontal arrows in the figure.

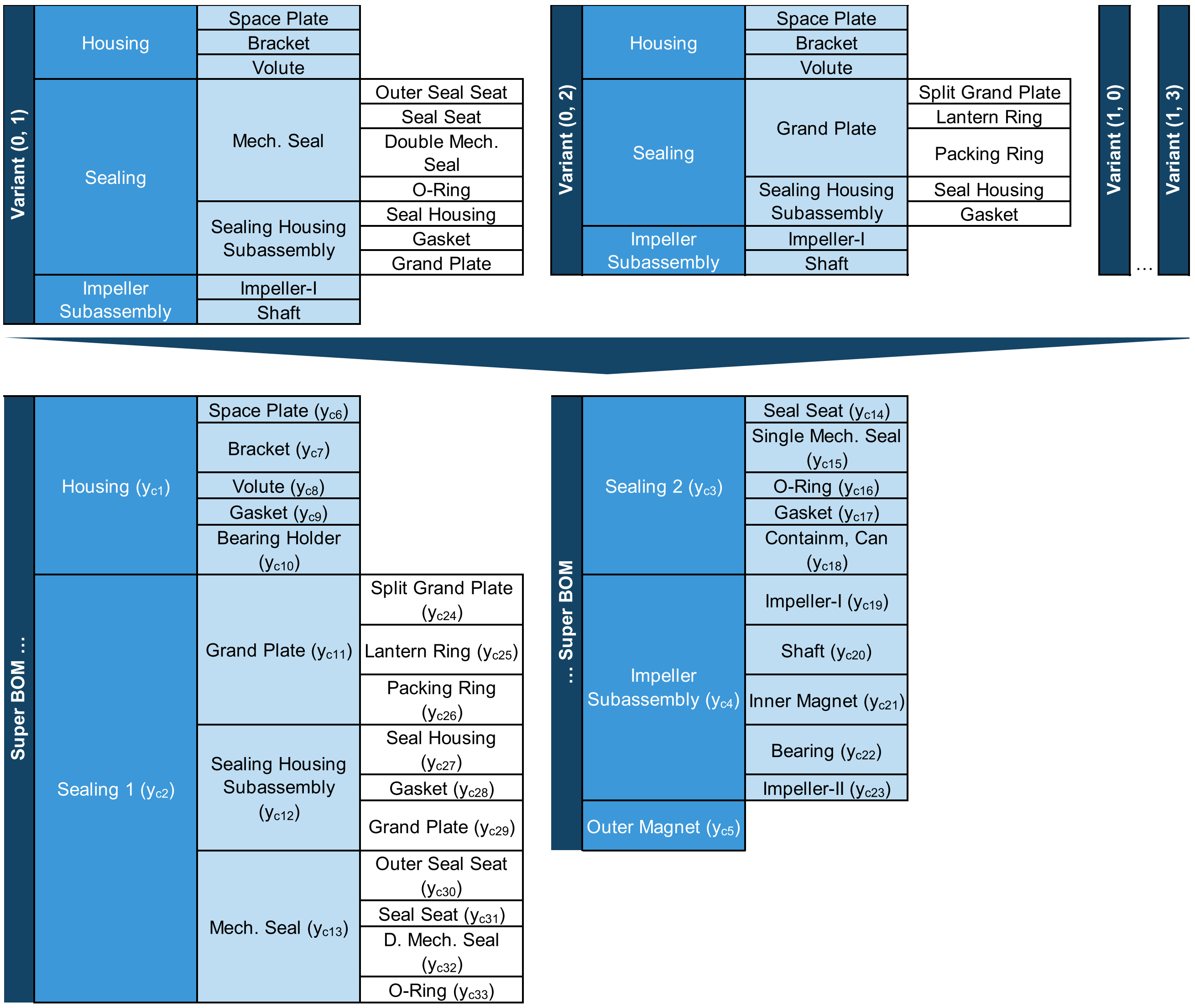

Fig. 2 Variant-specific bills of materials and resulting super bill of materials for the demonstration use case

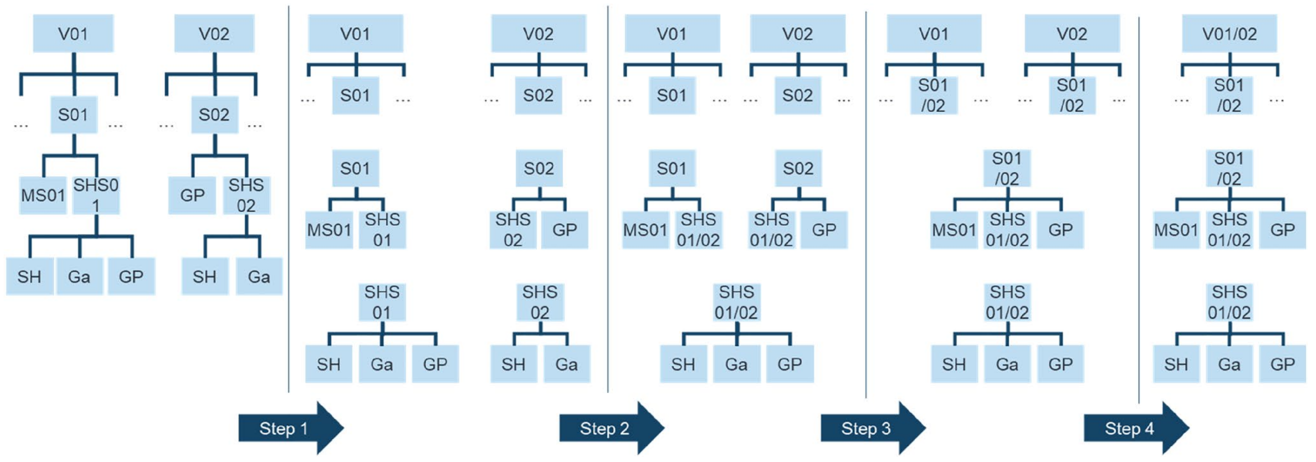

Fig. 3 Procedure for generating super BOMs

1. *Transforming bills of materials into single-level bills of materials* I.e., separate consideration of the assemblies of the BOMs. E.g. the BOM of variant (0, 1) is separated into its assemblies and subassemblies S01, SHS01 etc. as shown in Fig. 3.
2. *Compare and unify assemblies pairwise if the percentage of identical subcomponents is* $\geq \epsilon$. Here, ϵ is a machine learning hyperparameter that can be tuned with appropriate methods such as tree-structured parzen estimator [51]. As objective value the similarity of the resulting super BOM to the individual variant-specific BOMs as described by Kashkoush and ElMaraghy [21] can be used. If the subcomponents themselves are assemblies, they are identical only if they in turn contain the same subcomponents. In the present case $\epsilon = 0.5$ is chosen for demonstration. This means that for variants (0, 1) and (0, 2), the sealing housing subassemblies (SHS01 and SHS02) are combined.
3. *Repeat step 2 until there are no more changes* After the sealing housing subassembly has been combined, the sealing assembly of variants (0,1) and (0,2) contain the identical sealing housing subassembly and thus 50% identical sub components, which means that the two sealing assemblies (S01 and S02) can be combined. Afterward there are no further combinations possible because all similarities are lower than 0.5. E.g., the similarity of S01/02 and SHS01/02 is 0.33.
4. *Combine all assemblies on product level* In the example, V01 and V02 are combined.

After the super BOM has been created, variables y_{c_j} are introduced that can be used to derive variant-specific BOMs from the super BOM. As can be seen, the variant-specific BOMs resulting from the super BOM can differ not only in the elements they contain, but also in their structure, e.g., the O-ring can appear as a sub-component of sealing or of mech. seal, depending on the variant. Therefore, identical components at different positions must be coded with different variables y_{c_j}. Assigning values to all y_{c_j} thus results in a complete variant-specific BOM.

3.3 Learning super routings (module 3)

When learning super routings, the aim is to create a super routing that contains every variant-specific routing in the data pool as one specification. In addition, the variables by whose assignment a variant-specific routing can be created from the super routing must be defined. In current ERP systems, super routings are represented as a sequence of operations. For the final assembly of the pump described above, this would result in the super routing shown in Fig. 4. For the purpose of demonstration, we assume that there are alternative sequences for the operation Mount Outer Magnet, depending on the variant. Such structural alternatives are represented in current ERP systems by alternative operations. I.e., the structure results from the elements at the same time. For instance if a certain variant requires the operations Assemble Housing, Mounting

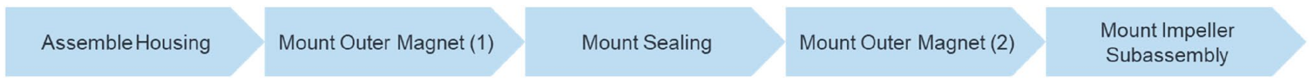

Fig. 4 Representation of super routings in state of the art ERP systems

Outer Magnet (1) and Mount Impeller Subassembly this also determines that they have to be executed in this order. This representation has the disadvantage that explicit knowledge about the existence of elements cannot be entered without entering knowledge about the structure, which may not be explicitly available. For instance if it is explicitly known that Mounting Outer Magnet is required for a certain variant this cannot be represented as a rule if it is not clear as well if the operation has to be executed at position 1 or 2. Furthermore, this representation has the disadvantage that in case of faulty systems the execution of both alternative operations is not generally excluded. For the presented approach, therefore, two alternative forms of representation (A and B, respectively) are presented, as shown in Fig. 5. Representation form A in YAWL representation has variables y_{o_k}, which indicate the contained elements and variables y_{s_l}, which indicate the structure independently and thus solves the two problems mentioned above. For example there could be a rule in the LLCS that determines that y_{s_1} is equal to 1 for a certain variant which would lead to the sequence represented by the lower path. Representation form B in precedence graph representation specifies precedence relationships between operations that exist or do not exist depending on the variant, which is encoded by variables y_{s_l}. I.e., elements and structure are also specified independently here. For instance if y_{s_2} is equal to 1 for a certain variant, there is a precedence relationship between the operations Mount Sealing and Mount Outer Magnet for this variant meaning that the operation Mount Outer Magnet may not be started before the operation Mount Sealing is finished. In addition to alternative structures resulting from differences between variants, there are also alternative structures in the disposition for individual variants themselves. All alternatives that are not violating the precedence relationship can be chosen depending on the current situation in production. In other words, disposition alternatives can be represented directly by representation form B. On the other hand, representation form B, in contrast to A, does not fundamentally exclude contradictory variant-specific routings. Therefore, depending on the case, one or the other is to be preferred. To learn super routings of representation form A, process mining approaches can be used. The commonly used alpha algorithm [52] is not applicable, since it may generate cyclic graphs, which cannot be automatically evaluated in an LLCS. In contrast, heuristic mining approaches are suitable, which use metaheuristics to vary the process model to find a model that has the best possible fit to the input paths. For learning super routings of representation form B, the method of Klindworth et al. [37] can be used, which, starting from a maximum constrained precedence graph, drops precedence relations one by one while reading in sequences of operations until finally only the necessary ones remain. The calculation of all y_{o_k} and y_{s_l} results in complete, variant-specific routings in each case. Routings for assemblies can go beyond joining operations and may contain operations such as cleaning, adjustment and inspection. These operations depend on the configuration of the variant, but not necessarily on its bill of materials. Finally, it should be noted that super routings can exist not only for the final product, but also for each assembly of the super BOM, if there are variant-specific differences for this as well.

3.4 Learning of dependencies (module 4)

After the super BOM and the super routing for the selected variants are known from modules 2 and 3, the dependencies of the target variables y_{cj}, y_{ok} and y_{sl} on the product characteristics x_i are to be established in module 4 using

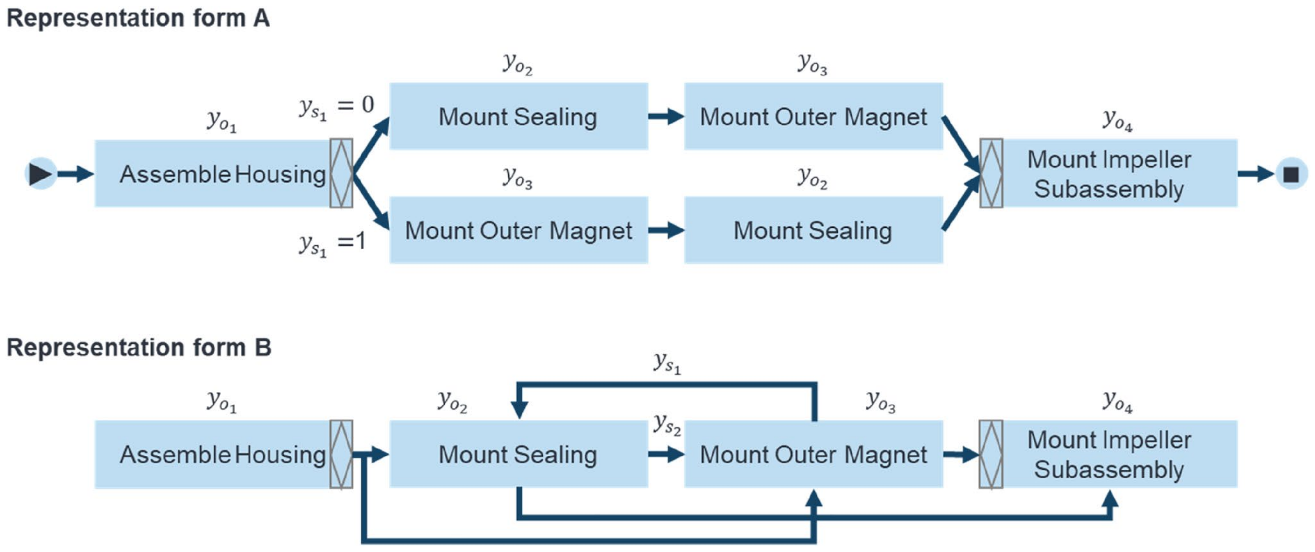

Fig. 5 Representation forms for super routings. Representation form A is shown in YAWL representation and B as precedence graph

machine learning. Since the data is in the form of BOMs and routings, it must first be converted into a tabular form suitable for machine learning. Since all product characteristics x_i are defined by the HLCS and all target variables y_{cj}, y_{ok} and y_{sl} are known from modules 2 and 3, a table can be created using the product characteristics as feature columns and the target variables as target columns. Non-binary product features must be transformed into a binary representation by one-hot coding. Table 3 shows an extract of the transformed data for the example case. For instance row 1 represents the variant for which product characteristic 1 is chosen to be 0 (type 620), product characteristic 2 is chosen to be 1 (double seal) etc. The variant specific BOM was defined to contain component c_1 (Housing), component c_2 (Sealing 1) etc. The routing was defined to contain operation o_1 (Assemble Housing) etc. and was defined to follow the sequence $y_{s_1} = 0$ (the upper path of representation form A in Fig. 5).

Classification methods from the field of machine learning can be used to create the dependencies by using existing BOMs and routings as well as BOMs and routings created by experts within the overall procedure (see Fig. 1) for training after they have been converted into a table representation as described above. However, in order to allow users to manually edit the LLCS as is common today, only methods that generate a model from Boolean expressions or models that can be transformed into such expressions can be used. This excludes non-interpretable models such as Deep Neural Networks. Table 4 provides an overview of the methods under consideration. In order to take into account rules explicitly entered by users, these must be incorporated into the model. For Quine-McCluskey classification [53], this is possible by including the rules as minterms within the method. In Associative Classification, the rules can be added to the learned model and the resulting model simplified using the Quine-McCluskey method. For decision trees, conversion to Boolean expressions is done first, followed by addition of rules, and finally simplification of the model. The greedy Quine-McCluskey classification introduced by Safaei and Beigy [53] is an adaption of the Quine-McCluskey classification which does not search the whole solution space in order to improve processing speed. It can therefore be assumed that the quality of its results is at most as good as that of the Quine-McCluskey classification. However, it can be shown that it nevertheless achieves better results for binary classification problems than a Decision Tree Classification. In this we see circumstantial evidence that a Quine-McCluskey classification achieves superior results, which is why we use it here and in the following. However, the approach described here is not limited to the use of this method. If we apply the Quine-McCluskey classification to the example case, we obtain the expression $\neg x_1 \rightarrow y_{c_{20}}$ for $y_{c_{20}}$, for example. Using the two previously unselected variants as test data results in an accuracy of 100% for this expression, i.e., this relationship is valid on all test data. Across all target variables, however, the accuracy is only 76.3%. This rather low value is due to the fact that in this case only few training data are available, since all but 4 data points do not contain any additional information. Therefore, a review for general cases will take place in chapter.

When applying the method, any number of data points i.e. variants with related BOM and routings can be generated in practice via module 1. The choice of the number of generated data points leads to a tradeoff between the prediction quality of the model and the effort of label generation. The prediction quality of the model can only be evaluated on the data for which labels are available. It follows that part of the

Table 3 Tabular representation of BOMs and routings for the example case

Row	x_1	$x_2=0$	$x_2=1$	$x_2=2$	$x_2=3$	x_3	…	x_7	y_{c1}	y_{c2}	…	y_{c33}	y_{o1}	…	y_{o4}	y_{s1}
1	0	0	1	0	0	0	…	0	1	1	…	0	1	…	1	0
2	0	0	1	0	0	0	…	1	1	1	…	0	1	…	1	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Table 4 Possible methods of supervised learning for creating dependencies

Supervised learning method	Short explanation	Applicability
Associative classification [54]	Considers how often certain items (here from x and y) occur together and how specific this correlation is	Applicable. Generates Boolean expressions
Quine-McCluskey classification [53]	Learns the simplest Boolean expression that explains the known labels	Applicable. Generates Boolean expressions
Greedy Quine-McCluskey classification [53]	Same approach as above but heuristic and therefore faster	Applicable. Generates Boolean expressions
Decision tree classification	Learns a decision tree that explains the known labels	Applicable. Decision trees can be converted to Boolean expressions

labeled data must not be used for training, but must remain available for testing. Which data is used for training and which for testing has an influence on the result. Therefore, cross-validation is used. When a certain prediction quality is reached in the cross validation, the iterative procedure terminates. Optimal prediction quality can be determined as optimization of the total costs resulting from label creation and errors.

3.5 Checking the input data (module 5)

The input data for the creation of the LLCS results from the input of BOMs and routings by experts. Errors can occur in the process, which is why the data is to be checked for anomaly using unsupervised learning. To do this, the data from Table 3 must first be suitably transformed. Anomalies can be detected by two different views on the data.

1. Relate values of product characteristics and target variables. Example for an anomaly: only for one variant a certain characteristic value does not occur together with a certain value of a target variable e.g., a certain component is always used if the customer selects a certain characteristic value except for one case.
2. Relate the values of different target variables. Example for an anomaly: only for one variant certain target variables take the same value e.g., certain operations are never performed in the same routing except for one case.

In the first case, a method of Muller and Markert [55] can be used. Step by step, all variants are considered that have been classified into a certain class with respect to a certain target variable. These variants are now clustered and checked for outliers, i.e., variants that are unusual for the selected class. E.g., in case of Liquiflo pumps all variants for which y_{c_6} is equal to 1 (i.e., the space plate is contained in the BOM) have product characteristic 1 chosen to be 0 (i.e., type is 620). By clustering all variants for which y_{c_6} is equal to 1 a variant with product characteristic chosen as 1 (i.e., type 621) would be recognized as an anomaly. There may be a technical reason for such an anomaly or it may be the result of a mistake. In the second case, the values of the target variables themselves are considered as data points and are also checked for anomalies by clustering, i.e. for cases in which the target variables have unusual combinations of values. For instance components c_{17} (gasket) and c_{18} (containment can) always go along with each other. If there would be a BOM that would include c_{17} but not c_{18} this would be seen as an anomaly which can be detected by clustering as described above. Found anomalies are reported to the user and require a manual check.

3.6 Checking the LLCS (module 6)

The LLCS is learned from data on the one hand and is based on rules entered by experts on the other. Errors can occur when rules are entered, which is why the LLCS itself also needs to be checked. Since the LLCS corresponds to a model of Boolean expressions, the methods for verification known from the state of the art can be applied to it. In addition, the model can be validated by checking for anomalies using unsupervised learning. For this purpose, the Boolean expressions of the model are first transformed into two tables. Table 6 shows the literals on which the minterms occurring in the LLCS depend. Next, Table 5 shows the minterms on which the target variables occurring in the LLCS depend. For demonstration purposes, an error was introduced into the system learned in module 4, representing an incorrect manual user input, the expression $x_1 \wedge x_4 \rightarrow y_{c_1}$. This expression consists of only one minterm which has never occurred before and is therefore added as last row of Table 6. This row shows a 1 in column x_1 and a 1 in column x_4 because this minterm depends on x_1 and x_4. Furthermore, the minterm is added as a new column to Table 5 which shows a 1 for y_{c_1} because y_{c_1} depends on this minterm. When clustering the entries of the two tables we see an anomaly in Table 5, as it results in the only entry out of 38 that depends on the $x_1 \wedge x_4$ minterm. All other minterms have influence on several target variables. In general, both datasets can be analyzed for

Table 5 Relevant minterms per target variable

	x_1	$\neg x_1$	$(x_2=0)$	$(x_2=1)$	$(x_2=2)$	$(x_2=3)$	$\neg(x_2=3)$	$x_1 \wedge x_4$
y_{c1}	0	0	0	0	0	0	0	1
y_{c2}	0	1	0	0	0	0	0	0
y_{c3}	…	…	…	…	…	…	…	…
y_{c4}	0	0	0	0	0	0	0	0
y_{c5}	0	0	0	0	0	1	0	0
y_{c6}	0	1	0	0	0	0	0	0
y_{c7}	0	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…
y_{c38}	0	0	0	0	0	1	0	0

Table 6 Relevant literals per minterm

	x_1	$\neg x_1$	$(x_2=0)$	$\neg(x_2=0)$	$(x_2=1)$	$\neg(x_2=1)$	$(x_2=2)$	$\neg(x_2=2)$	$(x_2=3)$	$\neg(x_2=3)$	x_3	$\neg x_3$	x_4	$\neg x_4$	x_5	$\neg x_5$	x_6	$\neg x_6$	x_7	$\neg x_7$
x_1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\neg x_1$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$(x_2=0)$	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$(x_2=1)$	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$(x_2=2)$	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
$(x_2=3)$	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
$\neg(x_2=3)$	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
$x1 \wedge x4$	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

anomalies using unsupervised learning. Thus, also more complex anomalies can be detected.

4 Assessment of the required effort for applying the approach to general use cases

By applying the developed approach to the case of Liquiflo pumps as a proof of concept, it is shown that the method is feasible. Whether the application of the approach is economical, however, depends on the number of variant-specific bills of materials and routings that are proposed by the program and have to be corrected by the user. Therefore we provide an approach to approximate this number. Companies may use this information together with company specific information such as how much time was required in the past to create an LLCS and how long does it take to correct variant specific BOMs and routings to determine how beneficial the presented approach is for them. This may be different from company to company depending on how difficult it is to transfer their experts' knowledge into models and how many variant specific BOMs and routings are already available. Since the presented approach can be combined with the classical approach of entering rules into the LLCS companies may also decide to use it in addition to their classical approach to support the transfer of expert knowledge in certain areas were this is expected to provide the most benefit. How many variant-specific BOMs and routings are required depends on the prediction quality of the dependencies created on the basis of module 4 in relation to the amount of available data. If it is assumed that the real dependencies are completely random, they cannot be predicted. This is equivalent to predicting labels formed solely by noise. In practice, however, the dependencies are not random, but reflect technical relationships; in particular, it cannot be assumed that certain target variables depend on all characteristics of the product. Therefore, in the following it shall be assumed that each target variable depends on a maximum of k product characteristics (for the case of Liquiflo pumps, *k* is equal to 2). Besides that the dependencies were created randomly by assigning the target values 0 and 1 to truth tables for each target variable in an equally distributed manner, taking into account the above assumption, and creating the real dependencies from them using the Quine-McCluskey method. Scenarios with different numbers *n* of binary product characteristics were investigated. For each scenario 10 different dependencies were considered. For each dependency 10 different random sequences of regarded variants were considered. The complete random and therefore unsystematic sequence of regarded variants represents a worst case of sampling. The Quine-McCluskey classification presented above was used for classification. The amount of

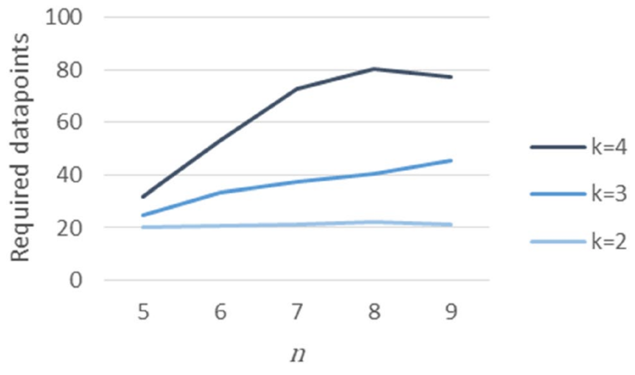

Fig. 6 Required data to learn a LLCS with n variables and k relevant variables per target

Table 7 Actual accuracy depending on n and k

		k		
		2 (%)	3 (%)	4 (%)
n	5	100.0	97.9	75.0
	6	99.3	97.2	98.2
	7	100.0	98.9	97.7
	8	99.7	98.6	98.0
	9	99.7	98.0	97.9

randomly selected variants was increased stepwise until a threefold cross-validation resulted in an accuracy of 100% with at least 20 variants considered. Then the actual accuracy of the found model was evaluated on all remaining test data. Figure 6 gives an overview of the required data. The sequence of the chosen variants as well as the randomly created dependencies had an effect on the amount of data required: the coefficients of variation were 15.63 and 13.55% respectively. Table 7 gives an overview of the achieved actual accuracies. It can be seen that the effort required for the application of the method grows approximately linearly. Linear extrapolation for e.g., k = 3 ($R^2 = 0.964$) would result in 339 variant-specific BOMs and related routings required on average for $n = 70$ which corresponds to about 10^{21} possible variants.

5 Summary, discussion and outlook

The presented approach enables the creation and validation of LLCS for product and process configuration. It is possible to consider different forms of knowledge in an integrated way, which are not yet considered holistically by approaches from the literature. By using interpretable LLCS, it remains possible to incorporate expert knowledge available in the form of rules into the LLCS, as is common practice today. In addition, the approach offers the possibility to use existing BOMs and routings for the creation of the LLCS by including them in the data pool, which is not possible with any state of the art approach. Lastly, knowledge that is difficult to transfer or is implicit can be taken into account by incorporating it into the creation of BOMs and routings by experts, which in turn serve as the basis for the creation of the LLCS. Thus, the research gap defined above regarding the creation of LLCS could be closed. With regard to the checking of LLCS, a distinction must be made between verification and validation. As shown, there are already approaches in the state of the art for the verification of configuration systems. The presented approach extends the state of the art when validating LLCS by using machine learning to detect anomalies in the defined rules. Furthermore, it could be shown how anomalies in the entered variant-specific BOMs and routings can also be identified by means of machine learning. Taken together, the above defined research gap concerning the validation of LLCS could be closed. By addressing the research gaps mentioned above, this paper contributes to the state of the art regarding the creation and validation of configuration systems. By applying the approach to a demonstration case as a proof of concept it could be shown that the approach is feasible. The reduction of the effort for the creation of the LLCS is partly compensated by the effort for the creation of variant-specific BOMs and routings. However, initial investigations show that this effort grows only linearly with the number of product features and thus seems justifiable even for large configuration systems. An economic comparison must be made on a case-by-case basis and depends, among other things, on how much effort is required to create variant-specific BOMs and routings and how much data is already available in the company. Since the input of explicit rules is still possible with this approach, it can be combined with approaches for the manual creation of the LLCS that are common today. Future research is needed to systematically compare the performance of the different methods of active learning for module 1. This is especially promising since the sequence of the chosen variants seems to have a significant effect on the required amount of data as shown in chapter 4. The different forms of representation described in chapter 3.3 may have an effect on the performance of the LLCS since they result in different target variables to be predicted by module 4.This needs to be systematically evaluated as well. Lastly, as explained above, there is a slight indication that the greedy Quine-McCluskey algorithm is well suited for solving this problem. However, it still needs to be thoroughly compared to other methods for the regarded case. Those further evaluations would be a good basis for selecting combinations of active learning techniques, representation types and machine learning techniques for the presented approach in practice.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

1. IFH (2017) Individualisierbare Produkte sind bei Konsumenten klar gefragt. https://www.ifhkoeln.de/individualisierbare-produkte-sind-bei-konsumenten-klar-gefragt/. Accessed 12 Sept 2022
2. Baranauskas G, Raišienė AG, Korsakienė R (2020) Mapping the scientific research on mass customization domain: a critical review and bibliometric analysis. J Risk Financ Manag 13:220. https://doi.org/10.3390/jrfm13090220
3. Vajna S, Weber C, Zeman K et al (2018) Übergreifende Informationsverarbeitung im Produktlebenszyklus. In: CAx für Ingenieure. Springer Vieweg, Berlin, Heidelberg, pp 515–547
4. Forza C, Salvador F (2006) Product information management for mass customization: connecting customer front-office and back-office for, fast and efficient customization. Springer
5. Shakeri M (2018) Variant configuration and management: challenges and opportunities. Doctoral Thesis, Massachusetts Institute of Technology. http://hdl.handle.net/1721.1/118520
6. Batchelor J, Andersen HR (2012) Bridging the product configuration gap between PLM and ERP - an automotive case study. In: 19th International product development management Conference, Manchester, UK, p 17–19
7. Sagegg OJ, Alfnes E (2020) ERP systems for manufacturing supply chains: applications, configuration, and performance. CRC Press
8. Ram Babu D, Lenin A, Bhaskar GB (2014) Advanced product configuration in manufacturing using enterprise resource planning variant configuration with optimization in manufacturing and assembly processes. Adv Manuf Res Intell Appl 591:94–97. https://doi.org/10.4028/www.scientific.net/AMM.591.94
9. Sinz C (2004) Verifikation regelbasierter Konfigurationssysteme. Doctoral dissertation, Universität Tübingen
10. Akhavan P, Shahabipour A, Hosnavi R (2018) A model for assessment of uncertainty in tacit knowledge acquisition. J Knowl Manag 22:413–431. https://doi.org/10.1108/JKM-06-2017-0242
11. Haug A, Hvam L, Mortensen NH (2012) Definition and evaluation of product configurator development strategies. Comput Ind 63:471–481. https://doi.org/10.1016/j.compind.2012.02.001
12. Haug A (2012) The illusion of tacit knowledge as the great problem in the development of product configurators. AI EDAM 26:25–37. https://doi.org/10.1017/S0890060410000533
13. Nonaka I, Takeuchi H (1996) The knowledge-creating company: How Japanese companies create the dynamics of innovation. Long Range Plan 29:592. https://doi.org/10.1016/0024-6301(96)81509-3
14. Gourlay S (2003) The SECI model of knowledge creation: some empirical shortcomings. In: 4th European Conference on Knowledge Management, Academic Conferences Ltd. https://eprints.kingston.ac.uk/id/eprint/2291/1/Gourlay%202004%20SECI.pdf
15. Tsoukas H (2005) Do we really understand tacit knowledge. Manag Knowl: Essent Read 107:1–18
16. Polanyi M (2012) Personal Knowledge. Routledge
17. Aldanondo M, Rougé S, Véron M (2000) Expert configurator for concurrent engineering: Caméléon software and model. J Intell Manuf 11:127–134. https://doi.org/10.1023/A:1008982531278
18. Haag A (1998) Sales configuration in business processes. IEEE Intell Syst 13:78–85. https://doi.org/10.1109/5254.708436
19. Günther S, Minkus A (2007) An integral model for mapping variant production in supply chains. In: Olhager J (eds) Advances in Production Management Systems: International IFIP TC 5, WG 5. 7 Conference on Advances in Production Management Systems (APMS 2007), September 17–19, Linköping, Sweden, Springer, New York, NY, p 265–272
20. Torkul O, Yılmaz R, Selvi İH et al (2015) Automatic generation of variants depending on changes of product properties in a flexible manufacturing environment. Comput Ind Eng 86:22–28. https://doi.org/10.1016/j.cie.2014.12.002
21. Kashkoush M, ElMaraghy H (2015) Knowledge-based model for constructing master assembly sequence. J Manuf Syst 34:43–52. https://doi.org/10.1016/j.jmsy.2014.10.004
22. Moussa M, ElMaraghy H (2018) Master assembly network generatio. Procedia CIRP 72:756–761. https://doi.org/10.1016/j.procir.2018.03.089
23. Jing X, Zhu Y, Liu J et al (2020) Intelligent generation method of 3D machining process based on process knowledge. Int J Comput Integr Manuf 33:38–61. https://doi.org/10.1080/0951192X.2019.1690687
24. Nonaka Y, Erdős G, Kis T et al (2013) Generating alternative process plans for complex parts. CIRP Ann 62:453–458. https://doi.org/10.1016/j.cirp.2013.03.048
25. Kang M, Kim G, Lee T et al (2016) Selection and sequencing of machining processes for prismatic parts using process ontology model. Int J Precis Eng Manuf 17:387–394. https://doi.org/10.1007/s12541-016-0048-2
26. Zhang Y, Zhang S, Huang R et al (2021) A deep learning-based approach for machining process route generation. Int J Adv Manuf Technol 115:3493–3511. https://doi.org/10.1007/s00170-021-07412-9
27. Wang Y, Tian D (2015) A weighted assembly precedence graph for assembly sequence planning. Int J Adv Manuf Technol 83:99–115. https://doi.org/10.1007/s00170-015-7565-5
28. Tariki K, Kiyokawa T, Nagatani T et al (2021) Generating complex assembly sequences from 3D CAD models considering insertion relations. Adv Robot 35:337–348. https://doi.org/10.1080/01691864.2020.1863258
29. Raju Bahubalendruni MVA, Bhusan BB (2016) Liaison concatenation – A method to obtain feasible assembly sequences from 3D-CAD product. Sadhana 41:67–74. https://doi.org/10.1007/s12046-015-0453-8
30. Ou L-M, Xu X (2013) Relationship matrix based automatic assembly sequence generation from a CAD model. Comput Aided Des 45:1053–1067. https://doi.org/10.1016/j.cad.2013.04.002
31. Natarajan KK, Gokulachandran J (2020) Artificial neural network based machining operation selection for prismatic components. Int J Adv Sci, Eng AND Inf Technol 10:618–628
32. Hussong M, Glatt M, Rüdiger-Flore P et al (2021) Deep Learning zur Unterstützung der Arbeitsplanung. Zeitschrift für

wirtschaftlichen Fabrikbetrieb 116:648–651. https://doi.org/10.1515/zwf-2021-0170

33. Amaitik S (2012) An integrated CAD/CAPP system based on STEP features. In: Proceedings of International Conference on Industrial Engineering and Operations Management 665: 665–673
34. Schuh G, Prote J-P, Hünnekes P (2020) Data mining methods for macro level process planning. Procedia CIRP 88:48–53. https://doi.org/10.1016/j.procir.2020.05.009
35. Schuh G, Prote J-P, Luckert M et al (2017) Knowledge discovery approach for automated process planning. Procedia CIRP 63:539–544. https://doi.org/10.1016/j.procir.2017.03.092
36. Rudin C (2019) Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1:206–215. https://doi.org/10.1038/s42256-019-0048-x
37. Klindworth H, Otto C, Scholl A (2012) On a learning precedence graph concept for the automotive industry. Eur J Oper Res 217:259–269. https://doi.org/10.1016/j.ejor.2011.09.024
38. de Giorgio A, Maffei A, Onori M et al (2021) Towards online reinforced learning of assembly sequence planning with interactive guidance systems for industry 4.0 adaptive manufacturing. J Manuf Syst 60:22–34. https://doi.org/10.1016/j.jmsy.2021.05.001
39. Otto C, Otto A (2014) Multiple-source learning precedence graph concept for the automotive industry. Eur J Oper Res 234:253–265. https://doi.org/10.1016/j.ejor.2013.09.034
40. Tseng H-E, Chang C-C, Chang S-H (2005) Applying case-based reasoning for product configuration in mass customization environments. Expert Syst Appl 29:913–925. https://doi.org/10.1016/j.eswa.2005.06.026
41. Denkena B, Schmidt J, Krüger M (2014) Data mining approach for knowledge-based process planning. Procedia Technol 15:406–415. https://doi.org/10.1016/j.protcy.2014.09.095
42. Scheibel B, Rinderle-Ma S (2021) Comparing decision mining approaches with regard to the meaningfulness of their results. https://arxiv.org/abs/2109.07335. Accessed 12 Sept 2022
43. Khannat A, Sbai H, Kjiri L (2021) Configurable process mining: semantic variability in event logs. In: proceedings of the 23rd international conference on enterprise Iinformation systems. SCITEPRESS - Science and Technology Publications 768–775
44. Sikal R, Sbai H, Kjiri L (2018) Configurable process mining: variability discovery approach. In: 2018 IEEE 5th international congress on information science and technology (CiSt) 137–142. IEEE
45. Buijs JCAM, van Dongen BF, van der Aalst WMP (2013) Mining Configurable Process Models from Collections of Event Logs. Business Process Management. Springer, Berlin, Heidelberg, pp 33–48
46. Boehm BW (1984) Verifying and validating software requirements and design specifications. IEEE Softw 1:75–88. https://doi.org/10.1109/ms.1984.233702
47. Walter R, Felfernig A, Küchlin W (2017) Constraint-based and SAT-based diagnosis of automotive configuration problems. J Intell Inf Syst 49:87–118. https://doi.org/10.1007/s10844-016-0422-7
48. Ait Wakrime A, Boubaker S, Kallel S et al (2019) A SAT-based formal approach for verifying business process configuration. Springer, Cham, pp 47–62
49. Voronov A, Tidstam A, Åkesson K et al (2012) Verification of item usage rules in product configuration. Springer, Berlin, Heidelberg, pp 182–191
50. Kashkoush M, ElMaraghy H (2016) Product family formation by matching bill-of-materials trees. CIRP J Manuf Sci Technol 12:1–13. https://doi.org/10.1016/j.cirpj.2015.09.004
51. Zhao M, Li J (2018) Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In: 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI). IEEE, pp 613–618
52. van der Aalst W (2016) Process mining. Springer, Berlin Heidelberg
53. Safaei J, Beigy H (2007) Quine-McCluskey classification. In: 2007 IEEE/ACS International Conference on Computer Systems and Applications. IEEE, pp 404–411
54. Kamsu-Foguem B, Rigal F, Mauget F (2013) Mining association rules for the quality improvement of the production process. Expert Syst Appl 40:1034–1045. https://doi.org/10.1016/j.eswa.2012.08.039
55. Muller NM, Markert K (2019) Identifying mislabeled instances in classification datasets. In: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8