10th International Conference on Mass Customization and Personalization – Community of Europe (MCP - CE 2022)
Toward the Sustainable, User-Centric and Smart Industry 5.0
September 21-23, 2022, Novi Sad, Serbia

DESIGN AUTOMATION CASE STUDY: MODULAR LOCATING FIXTURE

Paul Christoph Gembarski
Institute of Product Development, Leibniz Universität Hannover,
An der Universität 1, 30823 Garbsen, Germany

Abstract: *Design automation systems often mimic human designers and implement routine design activities. Beside this, the idea of such knowledge-based engineering is to support developers in the analysis and syntheses of complex engineering artifacts. An instance of this is product configuration. A central aspect of knowledge-based engineering is its ability to draw conclusions about the design context. For this inference, different reasoning techniques have been proposed. One uses constraint satisfaction problems as model-based approach. In the present contribution, the authors report about a case-study about the implementation of a constraint-based configuration system with onboard resources of a computer aided design system on the example of a locating fixture.*

Key Words: ***Knowledge-Based Engineering, Product Configuration, Constraint Satisfaction Problems, Computer Aided Design, Solution Space Modeling***

1. INTRODUCTION

One of the most intense reported use cases of knowledge-based engineering (KBE) systems is, from the beginning, Computer aided fixture design (CAFD) [1-3]. KBE is an approach to automate and support design tasks, like adapting a product to new requirements, choosing a variant out of a solution space or automaticly designing products and components with a rich body of explicit and formalized laws of creation. [4-6].

The ever increasing amount of product configurators indicates a widespread application of KBE techniques especially in configuration design, i.e., the composition of a system out of a set of predefined building blocks that are linked via standardized interfaces [7]. Basically, KBE can serve to get to a product specification or bill of materials, independently from geometric models [8, 9]. But KBE may also integrate geometric modeling tools, like computer aided design (CAD) systems either for visualization purposes or for further processing of the geometry [10, 11].

To effectively find solutions for configuration design problems, constraint satisfaction problems (CSP) are applied as basis for model-based inference [12]. In short, a CSP consists of a set of finite domains as containers for variables and their possible parameter values, and a set of constraints that relate the domains and specify which combinations of values of each variable are allowed [13, 14].

A multitude of proprietary software packages is available on market for implementing CSP and CAD. Many CAD systems offer functionalities for knowledge-based product modeling, but there is still a lack of application examples and case studies in literature. Thus, this contribution follows the question to what extent and with which scripting a CSP-based configurator can be implemented directly into a CAD system, using its onboard resources. The case study that was performed for this article on the example of a locating fixture was meant to test the above methods and illustrate their application in a comprehensible way in order to motivate other designers and engineers to use constraint-based configuration in their everyday business.

The remainder of the article is organized as follows: Section 2 presents a brief theoretical background for knowledge-based CAD as well as constraint-based reasoning and configuration design. The subsequent section 3, conceptualizes the implementation of the CSP applying the application programming interface of a CAD system in general. Section 4 then presents the locating fixture configurator before the final section 5 discusses findings and gives a conclusion.

2. THEORETICAL BACKGROUND

2.1. Knowledge-Based CAD

Knowledge-based CAD is to be understood as paradigm shift in solution oriented product modeling. It forces designers not to have a single variant in mind that has to be designed but a solution space from which this variant is reasoned [15]. The basis for this is the ability to add pieces of domain and control knowledge to the geometric model [16, 17]. Amongst others, domain knowledge in the form of geometric and logical constraints, parameter tables, features, templates and design rules are common techniques that are available in today's CAD systems.

In contrast to this domain knowledge, control knowledge formalizes the way to explore the solution space. Therefore, the three traditional reasoning types known from expert systems, i.e. rule-based, model-based and case-based reasoning are also applicable in knowledge-based CAD [16, 17].

2.2. Constraint-Based Reasoning and Configuration Design

For modeling physical or engineering contexts, constraint networks use a representation consisting of domains for variables and their possible values and constraints as relation between them. This structure can be written as graph where the domains form the nodes and the constraints the arcs [12].

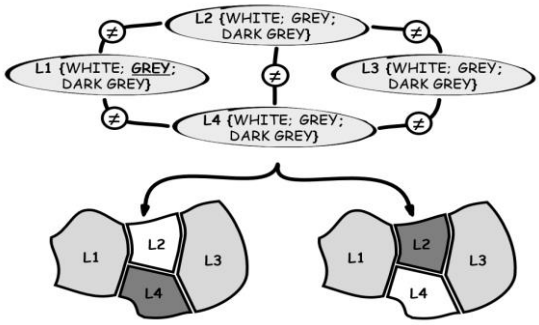

Fig. 1. *Map Coloring Problem modeled as CSP*

If now input values are applied to the constraint network the values of the other domains can be calculated on their basis which is knowln as constraint propagation. Beside this, the solution of configuration problems is a common task where constraint networks are used for reasoning. The map coloring problem (Fig. 1) is an example for such a task [13].

There exist different solutions strategies, some of them use the graph nature and the principle of local consistency of the constraint network to raise the performance of the solution process. E.g., arc consistency takes the value assignment of a domain and tests it against all other neighboring domains that are directly connected by constraints. Inconsistent values are then immediately removed. Referring to the example of Fig. 1, when the domain L1 is occupied with grey as input value, arc consistency would remove grey from L2 and L4. The advantage is that enforcing local consistency does not generate obviously erroneous value assignments, like e.g. generate-and-test would do [14].

Configuration design belongs to the synthetic design tasks. Particularly here, developers compose a system out of predefined building blocks or modules which are assembled by known, ideally standardized interfaces [6].

Still popular and compatible with template techniques, propose-and-revise is a suitable problem-solving technique (Fig. 2). E.g., the KBE system uses its formalized knowledge about system creation and templates to suggest an initial configuration which is then verified against formalized requirements. The Truth-Maintenance-System as essential part of the problem-solving mechanism, detects these violations and is then able to counteract this by changing the configuration in a predefined way out of a reaction pool [23].

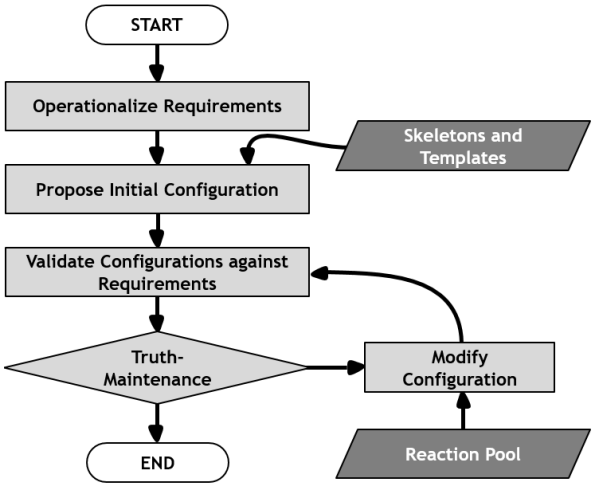

Fig. 2. *Propose and Revise as Problem-Solving Strategy*

If alternative reactions are available, these contain additional weighting so that the Truth-Maintenance-Systems is able to prioritize. If a reaction does not lead to constraint satisfaction, the Truth-Maintenance-System will return to this point and test the next reaction [24].

To use such problem-solving and reasoning mechanisms in a KBE system, they can be attached e.g. to a product model. In the simplest stage this can be a variant bill of materials where the constraint-based reasoning is used to determine product options. Advanced approaches work directly on the geometric CAD model [11, 16]. Additionally, constraint-based approaches are also capable of configuring non-geometric data, e.g. corresponding manufacturing processes for chosen product features [25] or service components of a product-service system [26, 27].

3. IMPLEMENTING A CSP-BASED IN-CAD CONFIGURATOR

The basic goal of a constraint-based configurator is to collapse the solution space to a single (best fitting) variant. Before that, it is necessary to translate the design problem into a configuration problem.

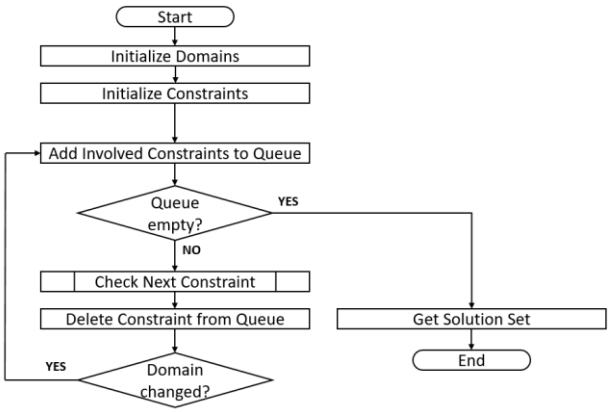

Fig. 3. *General Routine for Constraint Handling*

The first step is to model the domains which can be abstracted to a list of variables and their values as groups of related data. Many of today's CAD systems share Visual Basic for Applications (VBA) as Application Programming Interface, e.g. for macro programming. In

VBA, domains could be basically implemented as either collections / array lists or dictionaries. The latter offer some advantages since all dictionary properties, including the key values, are writable and retrievable. Beside strings, dictionaries also can handle nearly every other type of key except arrays and dictionary methods offer the possibility to check if a key value is already existing. Compared to collections / array lists, dictionaries are a bit slower in creation but significantly faster in computation.

Dictionaries are also suitable to represent the constraints and their handling in a queue. A distinction is made between unary and binary constraints. The first propagate input values for collapsing their involved domains. The latter relate two neighboring domains to each other. In order to implement the above mentioned arc consistency, the change of a domain must trigger the addition of attached constraints to the queue (Fig. 3).

4. LOCATING FIXTURE CONFIGURATOR

In the following, the above methods are tested on the example of a locating fixture. The implementation is further illustrated on an exemplary implementation into the CAD system Autodesk Inventor. Sub-section 4.1 contains the description of the configuration task itself, 4.2 then presents preliminary considerations about data repositories and constraint handling before 4.3 illustrated the CAD implementation.

4.1. Task Description

Before a part can be machined properly, it needs to be located explicitly so that machine coordinate system and workpiece coordinate system coincide. Locating fixtures that are positioned and calibrated on the machine table support workers here. Usually, such fixtures do also contain elements for clamping which are left out in the further example. In literature, locating fixtures are organized according to their locating principle which allocates locating machine elements to the reference surfaces of the part. This assignment is a suitable basis for the propose-and-revise logic to get to an initial configuration of the fixture.

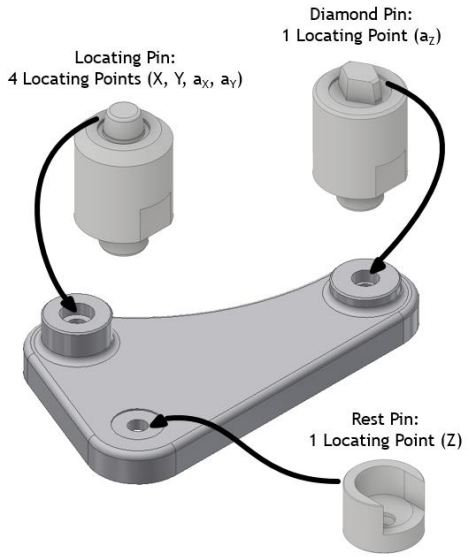

Fig 4. *Locating Principle Pin/Pin/Rest*

In this case study, the fixture follows the principle pin/pin/support (Fig. 4), where the locating pin is the primary locating datum, the diamond pin the secondary and the rest pin the tertiary one. The corresponding workpiece surfaces are also depicted in Fig. 4, i.e. the cylinder face of primary and secondary locating datum, as well as the plane face of the tertiary one. The plane surfaces of primary and secondary datum function as reference for the determination of the height position of the single machine elements and the global positioning on the base plate. The component manifold for the assembly is illustrated in Fig. 5.

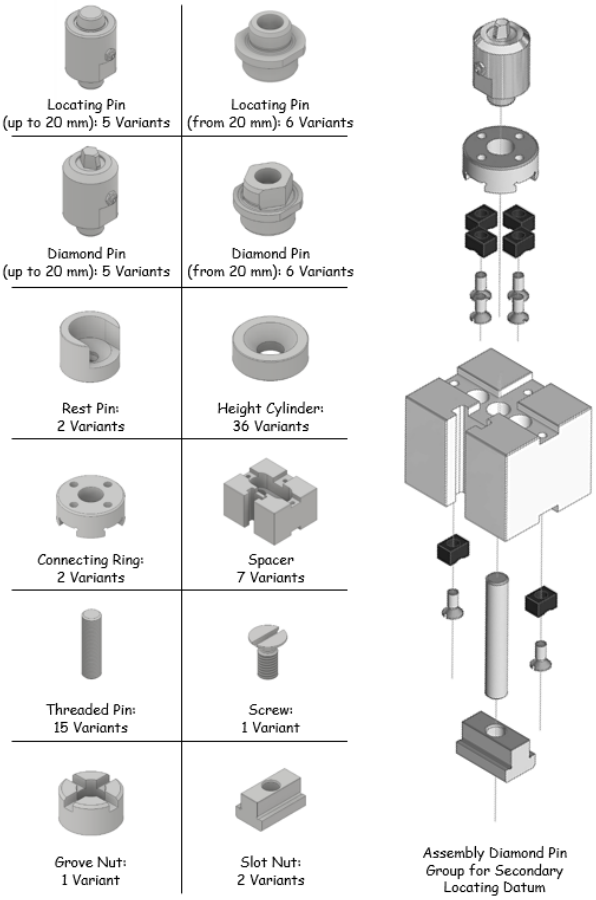

Fig. 5 *Component Manifold and Assembly Structure*

4.2. Preliminary Considerations

From the above it can be concluded that the following parameters need to be processed by the CSP:

- Diameter of primary and secondary datum face
- Height difference between primary and secondary datum
- Height difference between secondary and tertiary datum
- X-Y-position of secondary datum related to primary datum (for later collision check and positioning on the base plate)
- X-Y-Position of tertiary datum related to primary datum (for later collision check and positioning on the base plate)

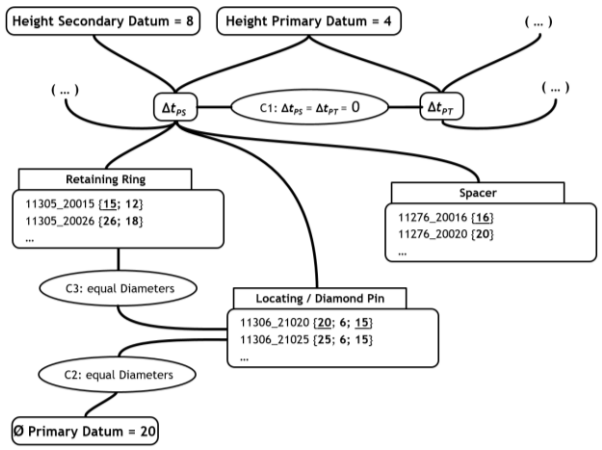

Fig 6. *Constraint Network (Excerpt)*

The CSP itself is constituted by domains containing the relevant machine elements. As shown in the excerpt of the constraint network in Fig. 6, the domains contain beside the article number of the machine element as key all other relevant dimensional data to reason based on mounting diameters, interface dimensions between the single machine elements and their particular height. Two additional domains are introduced for the calculation of the height differences between the three locating datums which later will serve for the height comparison of the assembled locating groups.

From a process perspective, the first configuration step after initializing the pin/pin/rest template is to propagate unary constraints in order to reason about the necessary locating machine elements. This is done by checking e.g. constraint C2 from Fig. 6, which collapses the domain of the locating / diamond pin to a suitable variant. If no one is found, an error should be fed back, if there is a siutable variant, the next constraint to be propagated is C3 which connects the locating mchine element to its connecting ring via their diameter parameters. Parallel to this, the according heights are added to the domain Δtps. After determining the locating machine elements and their connecting parts, the constraint C1 is propagated to calculate the heigth differences between the single assemblies. The reasoner then adds up to four spacers and height cylinders where necessary to compensate this.

4.3. VBA Implementation in Autodesk Inventor

```
Option Explicit

'Part Number
Public PartID As String

'_____________________________________________________
'Properties Unary Constraints
Public pdl As Integer 'Locating Diameter

'_____________________________________________________
'Properties Constraints Machine Elements
Public pD As Integer 'Diameter: Pin to Retaining Ring
Public poD As Integer 'Diameter: Outer Retaining Ring
Public ph As Double 'Component Height
Public px As Integer 'Position x
Public py As Integer 'Position y

'_____________________________________________________
'Properties for Component Handling
Public LastID As Integer
```

Fig 7. *Class Module for Machine Element Domain Declaration*

A necessary step in the VBA implementation is the declaration of public class modules for domains and constraints and their respecting properties as shown in Fig. 7. Note that not every property needs to be filled in when the CSP is initialized as the dictionaries allow empty arguments.

In a first version of the configuration system, the constraint list was hardcoded as shown in the excerpt in Fig. 8. As VBA does not allow for creating variable names as combination of strings and integers at runtime, this approach is cumbersome as every argument combination needs to be forseen and coded.

Following this, a second version of the configuration system works with a domain and constraint list which is stored in an external data repository, here an excel spreadsheet. After the configuration system is started, the system imports the domains and constraints from the spreadsheet to the working memory and then populates the domains from this representation. Additionally to the data to be handled by the CSP, the worksheets contains adjacent data for the later CAD assembly, e.g. names of iMates (semi-automatic geometric constraints in Inventor) that are then triggered after adding a component.

The constraints are imported in a similar way. As operators, the system handles equality, inequality and math constraints.

```
'Initialize Unary Constraints
Dim Key As Variant
Dim argl As Dictionary
Dim ConVal As Double
Do While UnConList.Count <> 0
    For Each Key In UnConList
        Set ConPa = UnConList(Key)
        Set argl = ConPa.argl
        ConVal = ConPa.ConVal
        Call Un_Constraint(argl, ConVal)
        UnConList.Remove (Key)
        'Collapsed Domain: Meldung und Programmende
        If argl.Count = 0 Then
            Select Case Key
                Case 1
                    MsgBox "No locating pin available."
                Case 2
                    MsgBox "No diamond pin available."
                Case 3
                    MsgBox "No rest pin available."
            End Select
            Exit Sub
        End If
    Next
Loop
```

Fig 8. *Unary Constraints derived from Locating Data*

After the domains have been populated, the system waits for the first user input. After a locating surface was chosen and submitted to the configuration system, the corresponding unary constraint is added to the queue and propagated. The command call un_constraints from Fig. 8 shows the respective comparison of the domain properties and the geometric parameters. After propagation, the constraint is removed from the queue.

Following the principle of local consistency, all constraints attached to the affected domain are added to the queue as well and propagated afterwards. If another domain is restricted, e.g. as mentioned for the constraint C2 before, the corresponding domain is collapsed and the firther involved constraints are added to the queue.

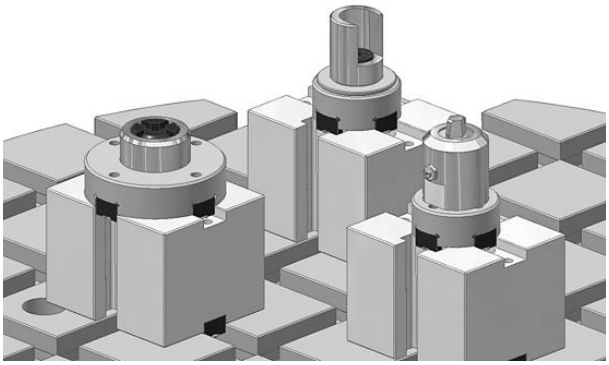

Fig 9. Completely Configured Locating Fixture

The solution strategy for bridging the height difference between the three locating machine elements is a bit different and follows a backtracking strategy. Backtracking means in this context, that the system proposes an initial combination of two spacer elements as such a combination is in most cases sufficient to connect the machine elements to the base plate. The system then test sequentially the combinations of the spacers. If the resulting solution set gets empty, the configuration is discarded and the system returns to the last known state where all constraints could be satisfied. Afterwards it tests the next alternative. Again, if the solution set is empty the process is repeated. If in contrast a combination of spacers is found that does not violate any constraint, it is checked whether a combination within a turret assembly can be substituted by a single spacer. If multiple spacer configurations are available, the one with minimum height is chosen.

The later CAD integration functions as a model generator, which means that the system starts with an empty assembly file. To this, all chosen machine elements are added and automatically related to each other using the usual geometric constraints like mate, flush or concentric. Fig. 9 shows an exemplarily configured assembly, where the spaces could be collapsed to one in each turret assembly. The user has already arranged the turrets on the base plate as this step has not been automated so far.

6. DISCUSSION AND CONCLUSION

The implemented configurator fulfills its purpose reliably. An extension to other locating principles beside pin/pin/rest would be a first point for future extension of the system. The management of additional machine elements or other sizes is convenient by adding them to the corresponding part catalogue spreadsheets.

Autodesk Inventor basically offers different mechanics to implement reasoning. One of them is the iLogic scripting language that is meant to code design rules. Such a rule-based representation is, in comparison the the illustrated constraint-based approach, requires more code and usually leads to a less flexible programm flow as could be realized by the constraint queueing. Additionally, the maintainability of rule-based approaches suffers as in the case of a modification of the rule-base each rule needs to be checked for consistency.

The restriction the the onboard ressources of the CAD system has also disadvantages: As mentioned before, the code is less economic as VBA does not allow to create variables (and user interface elements) at runtime. The choice of advanced programming environments that attach to the CAD system and remote control its functions would be favorable. A possible avenue here is the use of the phyton scripting language.

The example of the locating fixture is a good placeholder for many configuration problems as it uses equalities, inequalities and math constraints to get to a solution. This is transferable for many other configurable CAD assemblies. The design knowledge in this particular case was already formalized to a large extend and easy obtainable. As CAFD is an early working field for KBE systems, the abstraction of the design problem to a configuration problem was easy using the restriction that the locating principle was fixed. If the system should be able to abstract the locating principle by itself, different geometric data is necessary is to be included into the analyzis, not only the diameters and height positions. A piece of data is e.g. the orientation of drillings. In the example of the case study these are all parallel to each other, which is not necessarily the case.

Another avenue for future work in this field is the coupling of different solutionstrategies and reasoning mechanisms to more complex solvers. E.g. for performance optimizations and the application for large solution spaces, the integration of shortcuts and e.g. case-based reasoning is promising. Another question is to integrate the CAD model e.g. by reobtaining geometric data as further input to the CSP after the initial configuration has been proposed. Modeling guidelines and best practices for such applications could promote the application of KBE techniques in general.

7. REFERENCES

[1] I. Boyle, Y. Rong, and D.C. Brown, “A review and analysis of current computer-aided fixture design approaches,” *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 1-12, 2011. DOI: 10.1016/j.rcim.2010.05.008

[2] R.H. Alarcón, J. Ríos Chueco, J.M. Pérez García and A.Vizán Idoipe, “Fixture knowledge model development and implementation based on a functional design approach”, *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 1, pp. 56-66, 2010. DOI: 10.1016/j.rcim.2009.02.001

[3] Y. Rong, and Y. Zhu, Computer-Aided Fixture Design, Taylor&Francis: Milton Park, 1999.

[4] G. La Rocca, “Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design” *Advanced Engineering Informatics*, vol. 26, no. 2, pp. 159-179, 2012. DOI: 10.1016/j.aei.2012.02.002

[5] W.J.C. Verhagen, P. Bermell-Garcia, R.E.C. van Dijk, and R. Curran, “A critical review of Knowledge-Based Engineering: An identification of research challenges,” *Advanced Engineering Informatics*, vol. 26, no. 1, pp. 5-15, 2012. DOI: 10.1016/j.aei.2011.06.004

[6] N.R. Milton, *Knowledge Technologies*, Monza, Italy: Polimetrica sas, 2008.
[7] P. Blazek, C. Streichsbier, C., and M. Partl, *Configurator Database Report 2016*. Morrisville, USA: lulu.com, 2017.
[8] U. Blumöhr, M. Münch and M. Ukalovic, Variant configuration with SAP. Galileo Press, Bonn, 2012.
[9] J. McDermott, "R1: A rule-based configurer of computer systems," *Artificial intelligence*, vol. 19, no. 1, pp. 39-88, 1982. DOI: 10.1016/0004-3702(82)90021-2
[10] C.B. Chapman, and M. Pinfold, "The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure," *Advances in Engineering Software*, vol. 32, no. 12, pp. 903-912, 2001.
[11] P.C. Gembarski, "Three ways of integrating computer-aided design and knowledge-based engineering", Proceedings of the Design Society: DESIGN Conference, vol. 1, pp. 1255-1264, Cambridge University Press: Cambridge, 2020. DOI: 10.1017/dsd.2020.313
[12] C.J. Petrie, Automated Configuration Problem Solving, 1st ed.; Springer: Berlin/ Heidelberg, 2012.
[13] V. Kumar, "Algorithms for constraint-satisfaction problems: A survey", AI Magazine, vol. 13, no.1, pp. 32-32, 1992. DOI: 10.1609/aimag.v13i1.976
[14] R. Barták, M.A. Salido and F. Rossi, "Constraint satisfaction techniques in planning and scheduling", Journal of Intelligent Manufacturing, vol. 21, no.1, pp. 5-15, 2010. DOI: 10.1007/s10845-008-0203-4
[15] P.C. Gembarski and R. Lachmayer, "Solution space development: conceptual reflections and development of the parameter space matrix as planning tool for geometry-based solution spaces", International Journal of Industrial Engineering and Management, vol. 9, no. 4, pp. 177-186, 2018. DOI: 10.24867/IJIEM-2018-4-177
[16] M. Hirz, W. Dietrich, A. Gfrerrer, and J. Lang, Integrated Computer-Aided Design in Automotive Development, 1st ed.; Springer: Berlin/ Heidelberg, 2013.
[17] W. Skarka, "Application of MOKA methodology in generative model creation using CATIA". Engineering Applications of Artificial Intelligence, vol. 20, no. 5, pp. 677–690, 2007. DOI: 10.1016/j.engappai.2006.11.019
[18] A. Chakrabarti, K. Shea, R. Stone, J. Cagan, M. Campbell, N.V. Hernandez and K.L. Wood, "Computer-based design synthesis research: an overview", Journal of Computing and Information Science in Engineering, vol. 11, no. 2, pp. 1 – 10, 2011. DOI: 10.1115/1.3593409
[19] T. Brockmöller, R. Siqueira, P.C. Gembarski, I. Mozgova and R. Lachmayer, "Computer-Aided Engineering Environment for Designing Tailored Forming Components", Metals, vol. 10, no. 12, pp. 1589-1611, 2020. DOI: 10.3390/met10121589
[20] D. Sabin and R. Weigel "Product configuration frameworks - a survey", IEEE intelligent systems, vol. 13, no. 4, pp. 42-49, 1998. DOI: 10.1109/5254.708432
[21] L. Hvam, N.H. Mortensen and J. Riis, Product Curtomization, Springer, Berlin, Heidelberg, 2008.
[22] A.A. Hopgood, Intelligent Systems for Engineers and Scientists: A Practical Guide to Artificial Intelligence. CRC press, Boca Raton, 2021.
[23] B. Chandrasekaran, "Design problem solving: A task analysis", AI magazine, vol. 11, no. 4, pp. 59-71, 1990. DOI: 10.1609/aimag.v11i4.857
[24] S. Marcus and J. McDermott, "SALT: A knowledge acquisition language for propose-and-revise systems", Artificial Intelligence, vol. 39, no. 1, pp. 1-37, 1998. DOI: 10.1016/0004-3702(89)90002-7
[25] P. Pitiot, M. Aldanondo, and E. Vareilles, "Concurrent product configuration and process planning: Some optimization experimental results", Computers in Industry, vol. 65, no.4, pp. 610-621, 2014. DOI: 10.1016/j.compind.2014.01.012
[26] P.C. Gembarski and R. Lachmayer, "Mass customization und product-service-systems: Vergleich der Unternehmenstypen und der Entwicklungsumgebungen", Smart Service Engineering, pp. 214-232, Springer Gabler, Wiesbaden, 2017. DOI: 10.1007/978-3-658-16262-7_10
[27] D. Kloock-Schreiber, L. Domarkas, P.C. Gembarski, and R. Lachmayer, "Enrichment of geometric CAD models for service con-figuration", Proceedings of the 21st International Configuration Workshop, Hamburg, Germany, 21.09.2019, pp. 22-29. DOI: 10.15488/92

CORRESPONDENCE

Dr.-Ing. Paul Christoph Gembarski
Institute of Product Development,
Leibniz Universität Hannover,
An der Universität 1
30823 Garbsen, Germany
gembarski@ipeg.uni-hannover.de