11th International Conference on Customization and Personalization MCP 2024
The Power of Customization and Personalization in the Digital Age
September 25-26, 2024, Novi Sad, Serbia

COMBINING KNOWLEDGE-BASED CAD AND ALGORITHMIC MODELING FOR DESIGN AUTOMATION

Paul Christoph Gembarski [0000-0002-2642-3445], Bilal Ibrahimi [0009-0006-1230-7052], Nikolas Plett [0009-0007-1987-1833], Maxim Stötzer [0009-0001-4140-5894],
Institute of Product Development, Leibniz Universität Hannover,
An der Universität 1, 30823 Garbsen, Germany

Abstract: *Computational design systems offer diverse opportunities for capturing and reusing engineering knowledge and design intent. Two approaches are knowledge-based computer-aided design (CAD) and algorithmic modeling. While the first uses mathematical and logical constraints, design rules, and reasoning to control the parameters of a CAD model, the latter abstracts the modeling process of the geometry itself. This paper explores the combination of both approaches and how their strengths contribute to efficient modeling of geometric solution spaces. This is investigated on the example of a generator for enclosures of printed circuit boards. The generator was implemented as a remote control for the CAD system Autodesk Inventor in C#.*

Key Words: ***Design Automation, Knowledge-Based Engineering, Algorithmic Design, Computer-Aided Design, Solution Space Modeling***

1. INTRODUCTION

Today's computational design systems offer plenty of possibilities to capture and reuse knowledge and design intent. Design automation does not only reduce the error rate and the time required for modeling tasks but it can also be scaled to optimize downstream development processes (Kuegler et al., 2023; Verhagen et al., 2012). Parametric and chronology-based computer-aided design (CAD) systems offer different techniques build geometric solution space models and so to automate especially variant design activities. These reach from mathematical and logical constraints, design rules, and reasoning up to knowledge-based CAD configuration (Poot et al., 2020; Gembarski 2018; Amadori et al., 2012). In contrast, algorithmic modeling emphasizes abstracting and modeling the modeling process itself, not just using simple sequential macros, but creating intelligent interactive model generators. (Fuchs et al., 2022; Müller et al., 2021; Tedeschi & Lombardi, 2018).

Knowledge-based CAD has advantages such as consistent subsequent geometry adaptation and including additional information to the parameters like tolerance data. Disadvantageous is the necessity of an initial geometric model with defined addressable parameters and references which may lead to an inefficient model setup. In contrast, the algorithmic modeling approach doesn't require a master model and is thus favourable for complex geometries (Gadeyne et al., 2014; Chakrabarti et al., 2011).

The question arises how a complementary application of both approaches could lead to a more efficient modeling and better performance of geometric solution space models. This paper explores the combination of knowledge-based CAD and algorithmic modeling in printed circuit board (PCB) enclosure design. The generator was implemented as a remote control for the CAD system Autodesk Inventor in C#. The scientific and practical contribution is an application case to illustrate the applicability of algorithmic modeling in parametric, chronology-based CAD systems to exploit the advantages of both technologies.

2. THEORETICAL BACKGROUND

The concepts of knowledge-based engineering (KBE) and design automation involve a paradigm shift in computer-aided product modeling. Instead of designing a single product variant, the goal is to develop a solution space, from which a variant can easily be derived based on requirements. This includes two perspectives, first how to model the solution space and second how to explore it (Gembarski, 2020; Zhang, 2014; Skarka 2007).

2.1. Knowledge-Based CAD

Knowledge-based CAD is a 3D modeling principle which adds reasoning capabilities to a geometric model that enables it to draw conclusions from the design context (Ortner-Pichler & Landschützer, 2022; Hirz et al., 2013). It is grounded on parametric and feature-based design (Fig. 1). The first means to differentiate between shape and its describing parameters, the latter describes semantic objects that represent geometric building blocks enhanced with behaviour (VDI2209:2009).

Today's CAD systems offer different options for creating knowledge-based product models. Defining logical and mathematical relations between parameters allows to differentiate between leading and driven parameters. As a consequence, the designer additionally

has to plan the configuration concept and parameterization of the component (Aranburu et al., 2022; Camba et al., 2016).

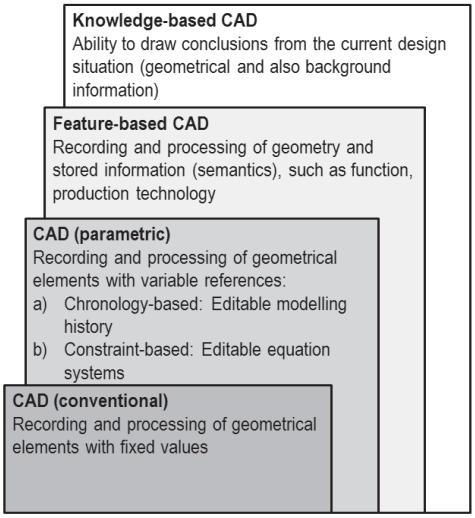


Fig. 1. *Overview of the principles of 3D modeling (VDI2209:2009)*

Supporting this, users can define additional parameters for length or angular dimensions and for, e.g., forces or moments of inertia. This incorporates integrating extensive mathematical formulas into the CAD model, e.g. for dimensioning or proof calculation, streamlining user workflow (Raffaeli & Germani, 2010).

Additionally, many CAD systems allow the definition of design rules as if-then-else statements. These rules link e.g. the suppression state of features or components to parameters (Grković et al., 2020; Gembarski, 2018; Myung & Han, 2001).

A well-structured model setup with parameters at various hierarchy levels is imperative as components become more complex. For this, CAD models can relate to skeletons that define the positioning of components and features and their geometrical characteristics (Li et al. 2018; Demoly & Roth, 2017).

The control of parameters can be further externalized by using spreadsheets, providing additional mathematical and statistical capabilities beyond the CAD system. E.g., by integrating lookup tables, it becomes possible to efficiently select standard parts based on geometric or load information and integrate basic reasoning (Peng & Ridgway, 1993). Another way to integrate reasoning in CAD systems is through script languages and macros or external knowledge-based systems to determine parameters (Gembarski, 2018; Hirz et al., 2013). In such a way, it is also possible to include non-geometric components such as service features of a product-service system (Guillon et al., 2022; Kloock-Schreiber et al., 2020; Elgammal et al., 2017).

2.2. Algorithmic Modeling

In algorithmic modeling, the focus is on automating the design process rather than pre-formulated solutions. Therefore, a design problem needs to be formalized in a way that algorithms can get from an initial situation, e.g. the statement of requirements, to generating a geometric model (Zuo et al., 2023; Müller et al., 2021; Chakrabarti et al., 2011). Algorithms are used to extract product properties from requirements, build product design rules also taking into account external data or numerical simulations, and alter the geometry in a rule-based manner (Caetano et al., 2020; Zboinska, 2015).

There are multiple ways to create an algorithm-based product model. One method is to use an application programming interface (API), design language or expert system shell to drive a CAD kernel (Gembarski, 2022; Frank et al., 2014; Ma et al., 2012). Another method is the generation of polygonal or NURBS-based geometric models (Zboinska, 2015). An example is the Grasshopper add-on for Rhinoceros (Oxman, 2017). Additional application of algorithms for evaluation and optimization of the geometry during generation is possible (Boretti et al., 2023; Cubukcuoglu et al., 2019; Holzer, 2016).

Algorithmic modeling aims to generate an individual product for each set of customer requirements. This is particularly favorable for complex geometries and pure configuration designs but largely restricts subsequent parametric editing of the variant. Instead, requirements and design rules need to be altered to create the updated version (Biedermann & Meboldt, 2020; Queiroz et al., 2015).

3. RESEARCH OBJECTIVES AND METHODS

The body of literature concerning KBE is widespread. Nonetheless, there are only few case studies that report in detail about knowledge bases, the implementation in CAD, and the transfer from the case under consideration to other ones (Kuegler et al., 2023; Plappert et al., 2020; Verhagen et al., 2012). By nature, parameter planning of knowledge-based CAD models concentrates on defining and constraining parameters in the sense of single variable domains. Since many feature definitions contain multiple parameters for dimensions, references, and orientations, designers often use multiple instances of them. If, e.g., a cut-out feature is situated once on the left and once on the right side depending on the variant, the necessary feature is activated by a topological parameter that controls the suppression state of the feature.

The question arises if a better performance can be achieved when features with such higher-order degrees of freedom, i.e., compound parameters and references, are algorithmically modeled. Studies about such a joint application of knowledge-based CAD and algorithmic modeling are scarce. To explore this, we follow design science research (Peffers et al., 2007; Hevner, 2007). As a case study we chose a generator for PCB enclosures as this combines a parametric design – the actual cuboid enclosure with both halves and the connecting screws – and an algorithmic part that generates an arbitrary number of cut-outs for plugs, controls, displays, and LEDs. For the case, a corresponding generator is developed and validated with different PCB assemblies.

4. CASE: PCB ENCLOSURE GENERATOR

The idea of the generator is to use any PCB assembly built in the CAD system Autodesk Inventor (Fig. 2) and create an individual enclosure for it. To limit the solution

space, only rectangular boards with one mounting hole per corner are considered. After using the generator, the user should receive 3D print-ready files for the parts of the enclosure.

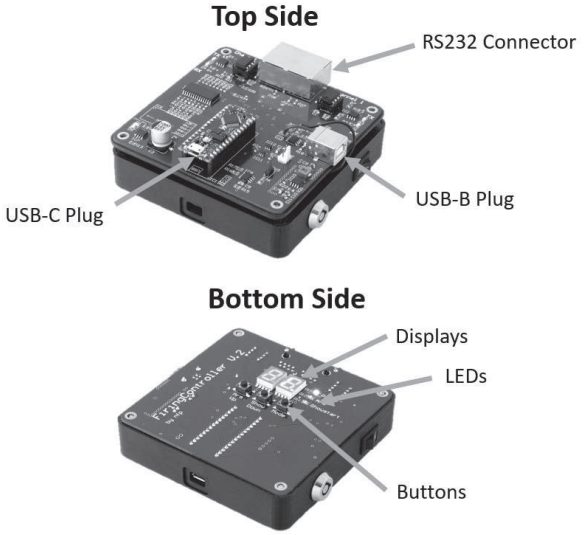


Fig. 2. *PCB assembly*

4.1. System Specification

The user has to supply a suitable PCB assembly in Autodesk Inventor. The assembly file needs to contain the board (with mounting holes as hole features) and the electronic components as parts. The generator should function optimally when utilizing SMD components. Plugs and connectors need to be placed on the edge of the board since the initial version of the generator does not check for their accessibility. Additional inputs include wall thickness, corner radius, clearances between enclosure and electronic components as well as the clearance between the enclosure and the board.

The generated enclosure should be exported in formats that are usual for 3D printing, such as .stl, .stp, or .obj. Basic manufacturing restrictions, e.g. minimum wall thicknesses, bar widths, and radii, need to be followed.

4.2. Program Flow

The generator is based on a parametric model of an enclosure half, which is fitted to the supplied PCB. The altered enclosure half is then augmented by the addition of cut-outs for the connectors and LEDs/displays.

Fig. 3 shows the program flow of the enclosure generator. Once the generator has been started, a suitable assembly file must be loaded into the generator. Once this has been completed, a preview image of the complete circuit board will appear. The next step is to select and confirm the board part, after which the user can select and confirm the connector, LED, and display parts, each of which must be confirmed. The user can then enter the user-configured dimensions, which are confirmed by pressing the Next button. Finally, the enclosure halves are created and joined together. A preview of the enclosure is now visible. To save the generated enclosure, a storage folder must first be selected. Several export formats are available for 3D printing and can be chosen. The user may then trigger the saving and exporting processes via the Export button. The 'PCB enclosure' folder, which contains all files, is now located in the selected storage location.

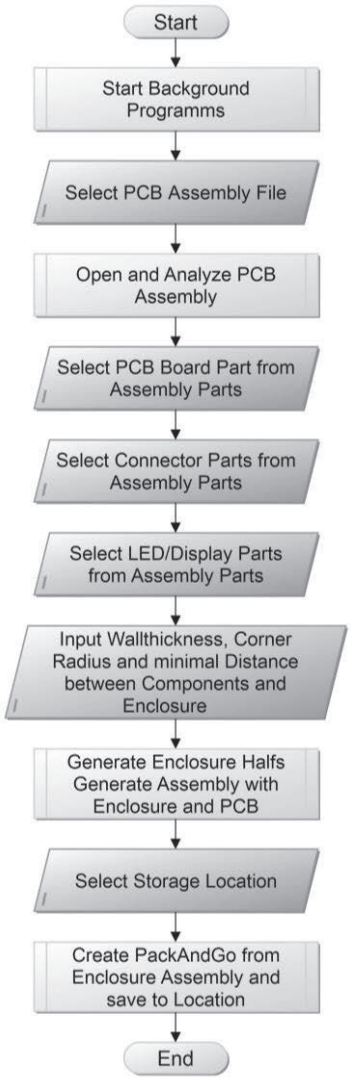


Fig. 3. *Flowchart of the general program*

4.3. Implementation

The implementation was carried out using C# for the underlying logic and Inventor for the 3D modeling. The logic was divided into different classes.

To analyze the given PCB assembly, a circuit board class was employed. The class constructor is used to obtain the background instance of Inventor and to open the assembly file. The method is used to generate a list of all parts contained in the assembly. This list is then used to select the parts that are the board, connectors, or displays/LEDs. The method AnalyseBoard takes a string that represents the name of the board part and provides the board's measurements, which include the general size of the board, hole sizes and hole positions, the height of the components on top and on the bottom of the board, and the position of the board in the assembly. A list of CutOuts and the methods AddConnectorToCutOuts and AddLEDToCutOuts were created for the purpose of storing connectors and displays/LEDs that should be cut out of the enclosure. CutOut is a class that is used to store the data needed to generate the hole in the enclosure. This data includes the size, position, and type

of the underlying component. The AddConnectorToCutOuts and AddLEDToCutOuts methods accept a string representing the desired component name and generate a new instance of the CutOut class, populated with the required dimensions, and add it to the CutOuts list.

In order to ascertain the dimensions of the standardized parts utilized in the enclosure, a StandardizedParts class was created. This class utilizes a spreadsheet file as its database, which contains the measurements of the ISO 7045 screws and the inserts used to join the enclosure halves. The class constructor creates a new background instance of Excel that will be employed to access the database. The following methods were created for obtaining the measurements: GetInsertHoleDia, GetScrewHeadDia, GetScrewHeadHeight, and GetScrewDiameter. These methods accept the diameter of the circuit board hole as an argument and return the corresponding dimension of the part with the largest thread diameter that is still able to fit through the holes in the circuit board.

To generate the enclosure halves from the collected data in the circuit board class, a parametric model was employed as a template for an enclosure half and an Enclosure class containing the underlying logic. The template consists of a generic enclosure half that uses parameters for all of its dimensions. Furthermore, the inside walls were named in order to enable easy referencing when creating the cut-outs. The constructor of the Enclosure class is utilized to obtain the parameters of the enclosure, which originate from user inputs and the instances of the CircuitBoard and StandardizedParts classes used within the main part of the program. Moreover, the constructor is passed the background instance of inventor to manipulate the template. The parameters can be applied to the model via the _SetParameters method, which also creates through-holes for screws if the instance is for a lower enclosure half. The cutouts created using an instance of the CircuitBoard class can be passed to the class by the method AddCutOut, which takes a CutOut as a parameter. The method _AddCutOutsToModell takes the saved CutOuts and applies them to the model. This is achieved by first determining the side on which the cut-out should be placed. Connector CutOuts are placed in the face that is intercepted by the underlying connector, while LED/display CutOuts are placed in the face across from the underlying component. Thereafter, a sketch is initiated on the referenced face and a rectangle contour is drawn with the size and position of the CutOut that is being processed. Subsequently, the rectangle is extruded through the wall of the enclosure to create the cut-out, this process is repeated for each CutOut saved in the instance. To create and save the enclosure, the Save method is employed. This method first opens the template and then employs the methods _SetParameters and _AddCutOutsToModel to create the desired model. The Save method takes a file path as a string, which is where the customized model is saved. Additionally, the class contains methods for exporting the saved models as .stl, .obj, and .step files.

To generate an assembly containing the PCB, both enclosure halves, and the screws, the class MergeAssembly was created. The constructor is passed the background instance of Inventor, which is then used to create the assembly document. The method AddPCB is used to add the PCB in the center of the assembly. Therefore, the file path and a position matrix are passed to the method. The AddTopEnclosureHalf and AddBottomEnclosureHalf methods are employed to position the enclosure halves. These methods are once again passed the filepaths of the enclosure halves and additionally the height of the PCB. The methods then generate a position matrix for positioning the enclosure halves, with the PCB height used as the offset in the z-direction. Finally, the AddScrews method is used to add the screws to the assembly. The method is then passed the dimensions of the board, the enclosure, and the screws. With the dimensions of the screws, the appropriate part is identified in the Inventor Content Center. This part is then placed in each screw hole using position matrices and dimensions of the enclosure and board. The method Save is used to save the assembly to the given path. To obtain all the necessary files for using the assembly on another machine, the method PackAndGo was created. This method employs the Pack and Go feature present in Inventor. This feature is used to package a file and all of its referenced files into a single location. The method is passed the path of the assembly and the path where the files should be saved.

In order to facilitate the management of file paths, a Save class was implemented. This class is employed to generate temporary paths for the storage of files, to delete the aforementioned temporary paths once they have been used, to generate a folder structure and the associated paths for the saving of documents, and finally to generate a method for the creation of a .zip archive of the aforementioned documents. The constructor is used to obtain the temporary folder on the machine. A number of methods were implemented that returned the corresponding path as a string. For example, GetPathTop was responsible for returning the temporary path of the top enclosure half. To clear the temporary files created by the program, the method DeleteFiles was employed. To generate the file structure used to save the generated files, the method ExportFiles was implemented. This method generates a main folder and subfolders for saving the CAD files and the exported enclosure halves. To generate a .zip archive from the generated folder structure, a MakeZip method was created.

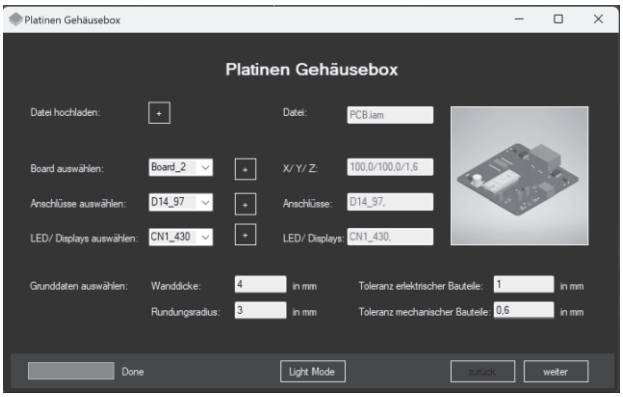


Fig. 4. *PCB enclosure generator input GUI*

In order to facilitate seamless user interaction, a graphical user interface (GUI) was implemented using the Windows Forms framework. Fig. 4 shows the input form of the generator.

4.4. Testing

To be able to use the generator, it must first be installed from the GitHub repository. After installation, the enclosure generator can be started via the shortcut on the desktop.

For testing, different PCB assemblies were input in the generator and the look and feel as well as the performance and validity of the generated enclosures was assessed. Fig. 5 and Fig. 6 show examples of the test PCBs. The test PCBs cover connectors and displays from various orientations and counts.

The generator could create the corresponding enclosures error-free and efficiently. Besides the declaration of the PCB, no further user interaction was needed and no rework of the enclosures. In order to check the manufacturability, a selection of enclosures was input into a slicer and was processed without warnings. Nonetheless, the generator reaches its limits with the wall thickness, as only a narrow wall thickness range is possible.

Additionally, the generator was tested with an assembly containing different through-hole-technology components. Here, the cut-out orientation was not reliably detected.

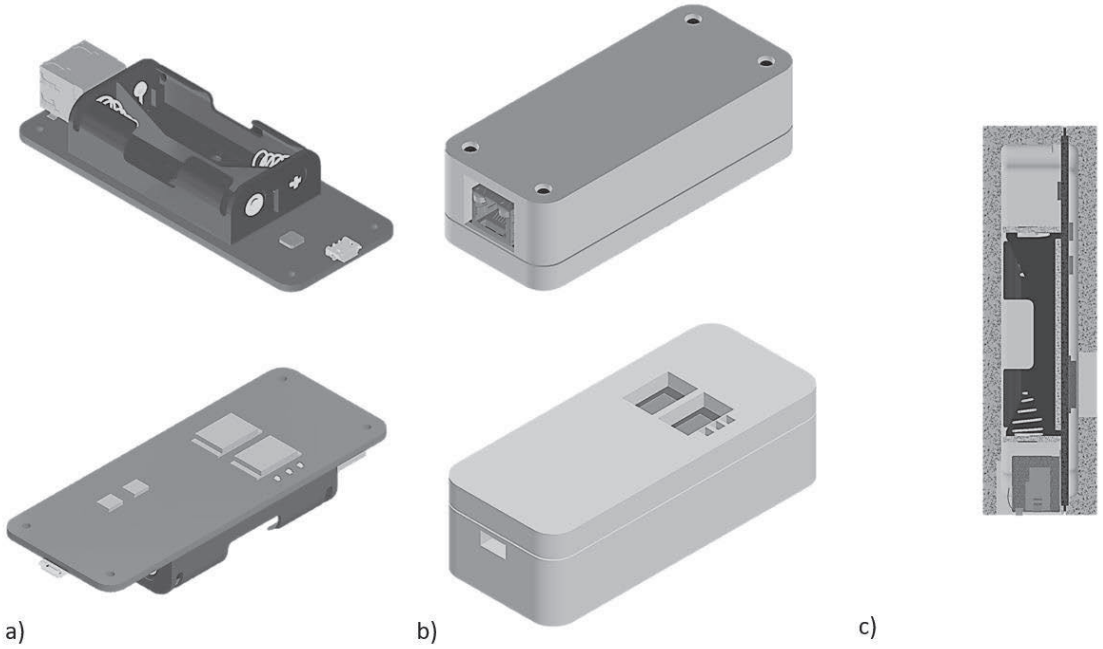


Fig. 5. *Test Case #1: a) PCB assembly b) PCB with enclosure c) Section view*

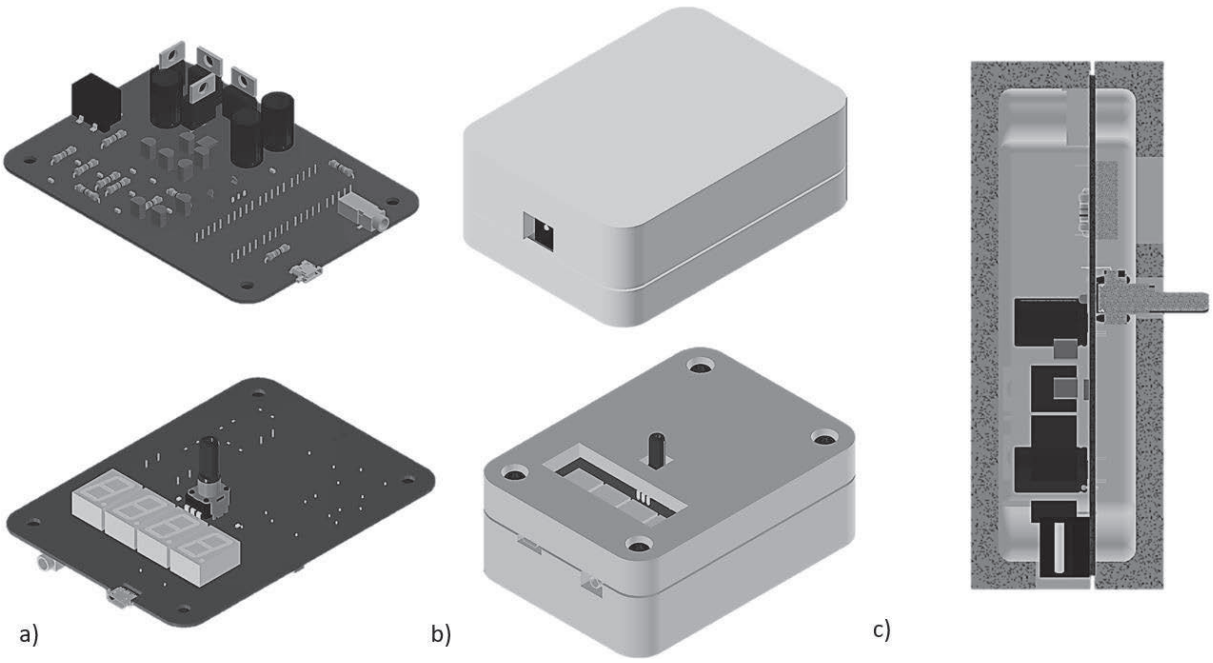


Fig. 6. *Test Case #2: a) PCB assembly b) PCB with enclosure c) Section view*

5. DISCUSSION

The intended functionality of the enclosure generator is given. Nonetheless, there are several avenues for improvement. Besides upgrades regarding functionality like mentioned above with the through-hole-technology components, an issue that was discovered during testing is the declaration of the cut-out relevant components. For heavily equipped PCBs, the list of components for selection gets quite long and sometimes confusing. An organizational solution to this would be the recommendation of a naming scheme for the components or the integration of a library. The generator could then process the information coded in the name or properties and decide by itself if a cut-out is obliged for the component without further user interaction.

The example use case of the PCB enclosure is a place holder for designs that include both first-order and higher-order degrees of freedom in the geometric model. The first-order ones comprise the dimensions of the enclosure halves such as length, width, height, wall thickness, and corner radius.

The higher-order degrees of freedom encompass count, position, and orientation of the needed cut-outs which the algorithmic part of the generator adds to the enclosure. It would be possible to integrate these into a parametric CAD master model. Therefore, a maximum count of cut-outs on each face would be included as single features and suppression states would have to be controlled additionally to the position and orientation. This seems to be a suitable solution for rectangular cut-outs as implemented in the current version of the generator. Considering that the shape of the cut-outs usually varies according to the corresponding component (like slots for USB-C connectors), a master model would have to include all of them in sufficient count. Since the definition of these features is an inherent part of the digital master, the model grows and gets slower in loading, processing, and rebuilding. In contrast to this, the algorithmic part of the generator could easily be enhanced by different cut-out shapes so that this behaviour could be avoided.

From an implementation point of view, other definitions of the cut-outs would be possible. The current version uses a very basic approach of defining a new sketch, creating the 2D polygon for the cut-out shape and applying the extrusion command to this. User-defined features (in Inventor iFeatures) represent a different approach. To apply them to the CAD model, reference geometry needs to be declared, such as insertion plane and point. If this is beneficial for the later model performance or for reducing coding effort in the algorithmic part of the generator was not tested during the case study.

6. CONCLUSION

Approaches for implementing knowledge-based engineering methods in geometric modeling contribute to reducing effort for variant creation and increasing the quality of design artefacts. In the case study presented in this article, both knowledge-based CAD and algorithmic modeling was applied to model a solution space of cuboid PCB enclosures. The model addresses first-order and higher-order degrees of freedom. To raise the efficiency, the algorithmic part of the enclosure generator is applied to the higher-order degrees of freedom since their translation into features and parameters goes along with a loss of performance of the resulting CAD master model.

Still, the implemented generator works on a parametric CAD kernel, here the Autodesk ShapeManager used in Inventor Professional. This means that the geometry description follows strict rules and methods, which enable parametrics, feature history, and various internal consistency checks. Additionally to the geometry model, the so called model-based definition adds information about tolerances, surface treatment, and material to distinct geometric elements. In the case of the enclosure, the intended output is a 3D printable design in a neutral printable format, e.g. .stl, for which the model-based definition is unnecessary.

Other design approaches, such as those known e.g. from computer graphics, reduce the geometric model to the definition of vertices, edges, and faces. The necessary commands to model the cuboid enclosure halves are straightforward. A suitable environment for such an implementation would be Rhino Grasshopper. The performance assessment and comparison of both implementations is an obvious next research aim.

7. REFERENCES

Amadori, K., Tarkian, M., Ölvander, J. & Krus, M. (2012) Flexible and robust CAD models for design automation. *Advanced Engineering Informatics*. 26 (2), 180-195. DOI: 10.1016/j.aei.2012.01.004

Aranburu, A., Cotillas, J., Justel, D., Contero, M., & Camba, J.D. (2022). How does the modeling strategy influence design optimization and the automatic generation of parametric geometry variations?. *Computer-Aided Design*, 151, 103364. DOI: 10.1016/j.cad.2022.103364

Biedermann, M., & Meboldt, M. (2020). Computational design synthesis of additive manufactured multi-flow nozzles. *Additive Manufacturing*, 35, 101231. DOI: 10.1016/j.addma.2020.101231

Boretti, V., Sardone, L., Bohórquez Graterón, L. A., Masera, D., Marano, G. C., & Domaneschi, M. (2023). Algorithm-aided design for composite bridges. *Buildings*, 13 (4), 865. DOI: 10.3390/buildings13040865

Caetano, I., Santos, L., & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *Frontiers of Architectural Research*, 9 (2), 287-300. DOI: 10.1016/j.foar.2019.12.008

Camba, J. D., Contero, M., & Company, P. (2016). Parametric CAD modeling: An analysis of strategies for design reusability. *Computer-Aided Design*, 74, 18-31. DOI: 10.1016/j.cad.2016.01.003

Chakrabarti, A., Shea, K., Stone, R., Cagan, J., Campbell, M., Vargas-Hernandez, N., & Wood, K.L. (2011). Computer-based design synthesis research: An overview. *Journal of computing and information science*

in engineering, 11 (2), 021003-1. DOI: 10.1115/1.3593409

Cubukcuoglu, C., Ekici, B., Tasgetiren, M. F., & Sariyildiz, S. (2019). OPTIMUS: self-adaptive differential evolution with ensemble of mutation strategies for grasshopper algorithmic modeling. *Algorithms*, 12 (7), 141. DOI: 10.3390/a12070141

Demoly, F., & Roth, S. (2017). Knowledge-based parametric CAD models of configurable biomechanical structures using geometric skeletons. *Computers in Industry*, 92, 104-117. DOI: 10.1016/j.compind.2017.06.006

Elgammal, A., Papazoglou, M., Krämer, B., & Constantinescu, C. (2017). Design for customization: a new paradigm for product-service system development. *Procedia Cirp*, 64, 345-350. DOI: 10.1016/j.procir.2017.03.132

Frank, G., Entner, D., Prante, T., Khachatouri, V., & Schwarz, M. (2014). Towards a generic framework of engineering design automation for creating complex CAD models. *International Journal on Advances in Systems and Measurements*, 7 (1), 179-192.

Fuchs, D., Bartz, R., Kuschmitz, S., & Vietor, T. (2022). Necessary advances in computer-aided design to leverage on additive manufacturing design freedom. *International Journal on Interactive Design and Manufacturing*, 16 (4), 1633-1651. DOI: 10.1007/s12008-022-00888-z

Gadeyne, K., Pinte, G., & Berx, K. (2014). Describing the design space of mechanical computational design synthesis problems. *Advanced Engineering Informatics*, 28 (3), 198-207. DOI: 10.1016/j.aei.2014.03.004

Gembarski, P.C. (2018). *Komplexitätsmanagement mittels wissensbasiertem CAD: Ein Ansatz zum unternehmenstypologischen Management konstruktiver Lösungsräume.* Garbsen, TEWISS.

Gembarski, P.C. (2020). The meaning of solution space modeling and knowledge-based product configurators for smart service systems. In: Świątek, J., Borzemski, L., Wilimowska, Z. (eds) Information Systems Architecture and Technology: Proceedings of 40th Anniversary International Conference on Information Systems Architecture and Technology – ISAT 2019. Advances in Intelligent Systems and Computing, vol 1051. Springer, Cham. DOI: 10.1007/978-3-030-30440-9_4

Gembarski, P.C. (2022). Joining constraint satisfaction problems and configurable cad product models: a step-by-step implementation guide. *Algorithms*, 15 (9), 318. DOI: 10.3390/a15090318

Grković, V., Kolarevic, M., Petrović, A., & Bjelić, M. (2020). CAD configurator for automatic configuration of modular strongrooms. In *Proceedings of the 9th International Conference on Mass Customization and Personalization-Community of Europe (MCE-CE 2020)*, University of Novi Sad-Faculty of Technical Sciences: Novi Sad, Serbia, pp. 85–92.

Guillon, D., Ayachi, R., Vareilles, É., Aldanondo, M., Villeneuve, É., & Merlo, C. (2021). Product-service system configuration: a generic knowledge-based model for commercial offers. *International Journal of Production Research*, 59 (4), 1021-1040. DOI: 10.1080/00207543.2020.1714090

Hevner, A. R. (2007). A three cycle view of design science research. *Scandinavian journal of information systems*, 19 (2), 4.

Hirz, M., Dietrich, W., Gfrerrer, A., & Lang, J. (2013). *Integrated computer-aided design in automotive development*. Berlin, Heidelberg, Springer. DOI: 10.1007/978-3-642-11940-8

Holzer, D. (2016). Design exploration supported by digital tool ecologies. *Automation in Construction*, 72, 3-8. DOI: 10.1016/j.autcon.2016.07.003

Kloock-Schreiber, D., Domarkas, L., Gembarski, P. C., & Lachmayer, R. (2019). Enrichment of geometric CAD models for service configuration. In *CEUR workshop proceedings*; Vol. 2467, pp. 22-29. Aachen, Germany: RWTH Aachen. DOI: 10.15488/9272

Kuegler, P., Dworschak, F., Schleich, B., & Wartzack, S. (2023). The evolution of knowledge-based engineering from a design research perspective: Literature review 2012–2021. *Advanced Engineering Informatics*. 55, 101892. DOI: 10.1016/j.aei.2023.101892

Li, H., Gembarski, P.C., & Lachmayer, R. (2018). Template-based design for design co-creation. In *Proceedings of the fifth international conference on design creativity (ICDC 2018)*, University of Bath, Bath, UK, pp. 387-394.

Ma, C. Y., Ma, C. H., Chu, D. Q., & Yang, Z. F. (2012). AutoLISP drawing using the second development of AutoCAD. *Advanced Materials Research*, 562, 993-996. DOI: 10.4028/www.scientific.net/AMR.562-564.993

Müller, P., Gembarski, P.C., & Lachmayer, R. (2022). Parametric topology synthesis of a short-shaft hip endoprosthesis based on patient-specific osteology. In *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems: Proceedings of the 8th Changeable, Agile, Reconfigurable and Virtual Production Conference (CARV2021) and the 10th World Mass Customization & Personalization Conference (MCPC2021)*, Aalborg, Denmark, October/November 2021. Springer International Publishing. pp. 669-676. DOI: 10.1007/978-3-030-90700-6_76

Myung, S., & Han, S. (2001). Knowledge-based parametric design of mechanical products based on configuration design method. *Expert Systems with applications*, 21 (2), 99-107. DOI: 10.1016/S0957-4174(01)00030-6

Ortner-Pichler, A., & Landschützer, C. (2022). Integration of parametric modeling in web-based knowledge-based engineering applications. *Advanced Engineering Informatics*, 51, 101492. DOI: 10.1016/j.aei.2021.101492

Oxman, R. (2017). Thinking difference: Theories and models of parametric design thinking. *Design studies*, 52, 4-39. DOI: 10.1016/j.destud.2017.06.001

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of management information systems*, 24 (3), 45-77. DOI: 10.2753/MIS0742-1222240302

Peng, C., & Ridgway, K. (1993). Integration of CAD/CAM and spreadsheet data processing. *Integrated Manufacturing Systems*, 4 (4), 29-36. DOI: 10.1108/09576069310044646

Plappert, S., Gembarski, P.C., Lachmayer, R. (2020). The Use of Knowledge-Based Engineering Systems and Artificial Intelligence in Product Development: A Snapshot. In: Świątek, J., Borzemski, L., Wilimowska, Z. (eds) Information Systems Architecture and Technology: Proceedings of 40th Anniversary International Conference on Information Systems Architecture and Technology – ISAT 2019. Advances in Intelligent Systems and Computing, vol 1051. Springer, Cham. DOI: 10.1007/978-3-030-30604-5_6

Poot, L.P., Wehlin, C., Tarkian, M., & Ölvander, J. (2020). Integrating sales and design: applying CAD configurators in the product development process. In: *Proceedings of the Design Society: DESIGN Conference.* 1, 345-354. DOI: 10.1017/dsd.2020.129

Queiroz, N., Dantas, N., Nome, C., & Vaz, C. (2015, November). Designing a Building envelope using parametric and algorithmic processes. In *Proceedings of the 19th Conference of the Iberoamerican Society of Digital Graphics,* pp. 797-801.

Raffaeli, R., & Germani, M. (2010). Knowledge-based approach to flexible part design. *Journal of Engineering Design*, 21 (1), 7-29. DOI: 10.1080/09544820802086996

Skarka, W. (2007). Application of MOKA methodology in generative model creation using CATIA. *Engineering Applications of Artificial Intelligence*, 20 (5), 677-690. DOI: 10.1016/j.engappai.2006.11.019

Tedeschi, A., & Lombardi, D. (2018). The algorithms-aided design (AAD). *Informed Architecture: Computational Strategies in Architectural Design*, 33-38. DOI: 10.1007/978-3-319-53135-9_4

Verein Deutscher Ingenieure. (2009) VDI 2209:2009. *3D product modeling - Technical and organizational requirements - Procedures, tools, and applications - Cost-effective practical use.* Berlin, Beuth.

Verhagen, W.J., Bermell-Garcia, P., Van Dijk, R.E., & Curran, R. (2012). A critical review of Knowledge-Based Engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26 (1), 5-15. DOI: 10.1016/j.aei.2011.06.004

Zboinska, M.A. (2015). Hybrid CAD/E platform supporting exploratory architectural design. *Computer-Aided Design*, 59, 64-84. DOI: 10.1016/j.cad.2014.08.029

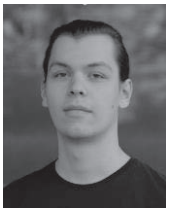
Zhang, L.L. (2014). Product configuration: a review of the state-of-the-art and future research. *International Journal of Production Research*, 52 (21), 6381-6398. DOI: 10.1080/00207543.2014.942012

Zuo, W., Chen, M.T., Chen, Y., Zhao, O., Cheng, B., & Zhao, J. (2023). Additive manufacturing oriented parametric topology optimization design and numerical analysis of steel joints in gridshell structures. *Thin-Walled Structures*, 188, 110817. DOI: 10.1016/j.tws.2023.110817

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

Bilal Ibrahimi
Institute of Product Development,
Leibniz Universität Hannover,
An der Universität 1
30823 Garbsen, Germany
bilal.ibrahimi@stud.uni-hannover.de



Nikolas Plett
Institute of Product Development,
Leibniz Universität Hannover,
An der Universität 1
30823 Garbsen, Germany
nikolas.plett@stud.uni-hannover.de

Maxim Stötzer
Institute of Product Development,
Leibniz Universität Hannover,
An der Universität 1
30823 Garbsen, Germany
maxim.stoetzer@stud.uni-hannover.de