Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 100 (2021) 397–402

www.elsevier.com/locate/procedia

31st CIRP Design Conference 2021 (CIRP Design 2021)

AI-Based knowledge extraction for automatic design proposals using design-related patterns

Carmen Krahe[a,*], Maksym Kalaidov[a], Markus Doellken[b], Thomas Gwosch[b], Andreas Kuhnle[a], Gisela Lanza [a], Sven Matthiesen[b]

[a]*Karlsruhe Institute of Technology, wbk Institute of Production Science, Kaiserstrasse 12, 76131 Karlsruhe, Germany*
[b]*Karlsruhe Institute of Technology, IPEK Institute of Product Engineering, Kaiserstrasse 10, 76131 Karlsruhe, Germany*

* Corresponding author. Tel.: +49-721-608-44011 ; fax: +49-721-608-45005. *E-mail address:* carmen.krahe@kit.edu

Abstract

Engineering competence and the digitization of all processes along the product development process are highly decisive for today's success of industrial companies. The design process is very individual and strongly based on design engineers' experience. Part of this knowledge and the result of the design approach are fixated in the existing variations of the product generations, but are difficult to extract and to formalize. Conclusions about design-related patterns between products of different generations or variants can be drawn from the model tree representing the design engineer's thinking process for each individual CAD model. However, the model tree has hardly been used so far. The aim of this paper is to examine whether there exist any common design patterns between CAD models of certain component classes by the exemplary use case in the area of mechanical engineering. To identify patterns and to extract knowledge out of complex data sets, Machine Learning (ML), especially Deep Learning, has proven an immense capability. Finally, based on the learned patterns, meaningful next design steps are to be proposed in the form of an assistance system. The results show that there exist common design patterns for various classes of components. It is illustrated on an exemplary component class that those patterns can be used to train an assistance system based on Recurrent Neural Networks (RNNs). The corresponding design patterns were extracted from data of an industrial application partner. By transferring these design patterns to the development of new product generations or variants, on the one hand the design process itself and thus the time to market can be shortened. On the other hand, the knowledge from previous product generations contained in those patterns can be preserved. For further research the design patterns of CAD models extracted by ML algorithms is a contribution to faster knowledge extrapolation.

Keywords: Artificial Intelligence; Pattern Recognition; Design; Product Development

1. Introduction

Today's companies face a multitude of challenges in the area of product development. On the one hand, increasing individualization leads to an increasing number of variants and thus to growing complexity. Due to shorter product life cycles and increasing global competition, on the other hand companies are forced to bring innovative products to the market in the shortest possible time [1]. The so-called time-to-market is an essential factor for the market success of a product. Product development plays a particularly important role here, both in terms of time and costs [1,2]. In order to maintain the market attractiveness of a product due to innovative features or to comply with changing specifications or standards, different generations or variants are usually launched on the market. For efficiency reasons, product development is therefore very often based on reference products, such as products from previous generations or products from competitors. According to [3], the majority of products are developed in generations. Reliably working components or subsystems are taken over from

10.1016/j.procir.2021.05.093

reference products and, if necessary, adapted slightly [3]. This reduces the development effort, possible risks with regard to functionality and manufacturability and required investments, e.g. investments in production facilities [3]. In many companies, a large number of existing product models of different generations and variants exist in the form of CAD data, which are very similar in their basic structure and often only differ in individual subsystems or components. Especially the reliable components and subsystem, which have been used for multiple product generations, contain experience knowledge accumulated over years. What is meant by this is for example knowledge about the relationship between embodiment design and product functionalities, manufacturability or economic efficiency. The problem is that this knowledge is mostly implicit and individual [4,5] and the manufacturing-specific knowledge is partly not sufficient [6], so it can only be described in a rule-based manner or sophisticated models of embodiment function relations [7] with high effort [8,9]. Machine Learning (ML) offers an enormous potential here to extract unknown patterns and correlations from data and to formalize implicit knowledge.

The product development process is highly complex and relies heavily on the individual experience of design engineers. Nevertheless, there is a structured procedure in accordance with VDI 2221 [10], which design engineers can use as a guideline. The implementation of the proposed phases as well as the choice of the methods required for this is up to the design engineer's experience.

The idea of systematically using existing experience knowledge of design engineers has been pursued for some time in the field of knowledge-based engineering (KBE). KBE systems are a combination of object-oriented programming, artificial intelligence and CAD [11]. According to [12] such a KBE system can capture and reuse product and process engineering knowledge in a convenient and maintainable manner. According to [12], the ultimate objective of KBE is the reduction of time and costs in product development by automating repetitive tasks that do not require the creativity of the design engineer. However, the use of KBE systems is limited to design tasks where knowledge can be expressed in explicit rules [13] that must first be formalized for example by experts for the specific application. Other approaches use existing knowledge through design reuse or shape retrieval of existing CAD models, e.g. [14]. The idea of Group Technology also aims at using and modifying existing designs of a product family to generate a new component [15].

Current approaches using ML to process 3D data are particularly concerned with classification, e.g. [16–19] and shape completion of 3D models, e.g. [20–23]. An overview of advances of Deep Learning on different 3D data representations is given in [24]. In general, the approaches attempt to learn relevant geometric properties from the 3D objects. The necessary design steps required to create a corresponding model are disregarded. Nevertheless, shape completion methods could be used to automatically complete semi-finished components. However, these approaches are not yet applied in the design process.

In the area of engineering, various methods have already been tested to generate recommendations based on complete components [25,26]. In [27] a classification and similarity search of components is implemented on the basis of the model tree. The existing approaches are therefore only a passive support for the design engineer in order to give him a good overview of already existing CAD models, for example to avoid duplicates. The model tree on component basis as representation for the design process of a design engineer via design features has hardly been used so far.

Therefore, the aim of this paper is to use ML methods to extract implicit knowledge from CAD models in the form of design patterns, as suggested in [28], to make it usable. With this, the design engineer is supported in designing new variants or generations with existing knowledge and moreover the design process is accelerated. The basis for learning the design patterns is the model tree in CAD, which is kind of a log file representing the conceptual approach of the design engineer [27]. Recurrent Neural Networks (RNNs) are suitable for processing such sequential data because they can store information over time. The specific research questions are:

- The definition of the term design pattern as well as an examination on the basis of a data set from the industry whether design patterns exist at all
- Development of a deep learning-based assistance system trained on the design patterns, which suggests next design steps to support the design engineer in following proven procedures, thereby avoiding errors and speeding up the design process
- Evaluation of the developed architecture in a case study for an industrial application using data from the area of mechanical engineering

2. Methodology

The basic prerequisite for the application of ML is first of all the presence of patterns in the design process. For this purpose, the term "design pattern" is defined in chapter 2.1 and a graph-based survey is used to check whether patterns are present within and across different classes of components in chapter 2.2. Next, chapter 2.3 shows how the corresponding patterns are prepared in order to finally apply the RNN architecture shown in chapter 2.4.

2.1. Definition of design patterns

CAD models of components in their original format are the basis for finding design patterns. The model tree contained in these models provides information about the design features

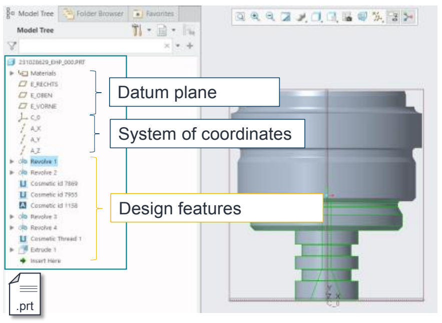


Fig. 1: Exemplary model tree of a CAD model in

and their parameterisation, which the design engineer used to create the 3D model. In Fig. 1 an exemplary model tree is shown in the software PTC Creo®. The design features used can be divided into five different categories, such as shape features (e.g. extrusion), engineering features (e.g. chamfer), editing features (e.g. pattern), datum features (e.g. datum plane) and surfaces. Each of these defined design features also has additional parameters, such as dimensions and tolerances. There are different ways to create a geometrically identical component. For example, a cylinder can be represented by extrusion or rotation. The model trees for one and the same 3D model are therefore not necessarily identical. Nevertheless, typical sequences of design features can be found even between different model trees. These feature sequences are called design patterns.

2.2. Examination of the presence of patterns in the design process

A prerequisite for the application of ML is the existence of design patterns across different model trees of the considered CAD model data base. For this investigation, the model trees are first displayed as a graph. Each design feature forms a node in the graph, and the corresponding dependencies are represented via the edges. Fig. 2 shows an exemplary graph. Parameters of the design features are not considered here. To determine the similarities of the graphs, the Neighbourhood Matching algorithm according to [29] is used. This compares the graphs for structural and contextual similarities, i.e. for the structure of the model tree and the design features used. The final result is a matrix that reflects the similarities between the individual model trees represented by graphs. If there are similarities between the individual model trees, it can be assumed that design patterns do exist.

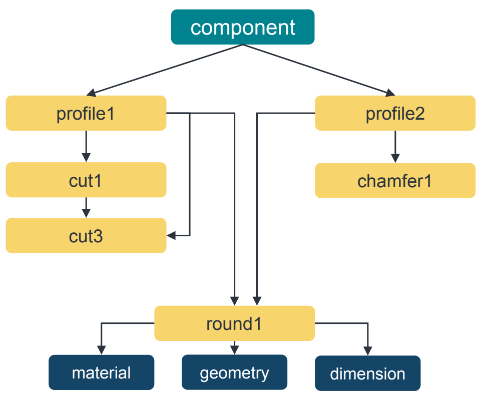


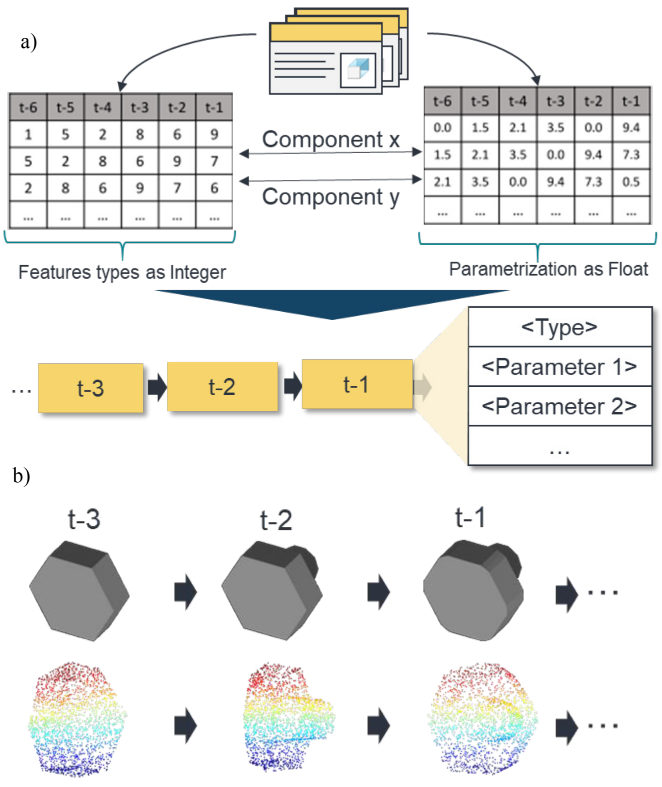
Fig. 2. Graph based representation of the model tree.

2.3. Extraction of design patterns

To process the model trees from the CAD models with ML, they must first be prepared accordingly. In principle, there are two ways of representing the design process (see Fig. 3). On the one hand, the sequence of the design features and their parametrizations can be read out from the model considering parent-child relationships between those features. However, it should be noted that the order of the features does not necessarily correspond to the order in which the designer created them. Different extraction methods can lead to different sequences even considering the same model tree. Hence, the assumed model tree hierarchy is not the only possible order, but at least one possible order. Further, the features' parameters (e.g. dimensions) are represented as floats, while feature types are categorical values and first are coded as integers from "1" to the number of unique feature types (dictionary size). Although it is possible to use categorical values as input to the RNN, it is common practice to transform them into float vectors, i.e. embeddings, which can be understood as semantic representations. Intuition behind the embeddings is that, for example, an extrusion is semantically much closer to a rotational body than to a cosmetic thread. This kind of relationships between different features should help the RNN to learn sequential feature dependencies faster and more robust. In order to do so, an embedding layer is preconnected to the RNN. Therefore, a linear transformation from integer values to corresponding embeddings is learned during the training phase.

On the other hand, the geometric development of the 3D model can be tracked via the model tree by gradually removing features and storing the corresponding intermediate states. Not all features, however, do result in a geometrical change. A new geometric state is therefore only spoken of if the volume of the 3D model varies. In a next step, the corresponding intermediate states are converted into point clouds to make them processable for ML.

To sum up, the design process can be represented by the sequence of encoded design features and the geometric development of the 3D model. In order to learn patterns from



t-6	t-5	t-4	t-3	t-2	t-1
1	5	2	8	6	9
5	2	8	6	9	7
2	8	6	9	7	6
...	...	...	...	...	...

t-6	t-5	t-4	t-3	t-2	t-1
0.0	1.5	2.1	3.5	0.0	9.4
1.5	2.1	3.5	0.0	9.4	7.3
2.1	3.5	0.0	9.4	7.3	0.5
...	...	...	...	...	...

Fig. 3. Representation of the design process for each design state t via a) the sequence of the design features used and b) the corresponding geometric development of the 3D model in form of point clouds.

the resulting feature sequences, they are now divided into shorter sequence sections. Via a sliding window sequence sections of a given length can be created. In addition, the data set can be significantly enlarged, since several observations can be created from one sequence.

2.4. RNN-based architecture to predict next design steps

The aim is to use RNNs to identify patterns based on the sequence data created in chapter 2.3 in order to suggest next design steps. In a first step only the design feature types is predicted. A further expansion stage then additionally considers the dimensions of each design feature. At the highest stage of development, next design steps should include the corresponding design feature (e.g. extrusion) with associated dimensioning (e.g. extrusion depth) and the visual representation of the next component state (see Fig. 4). Besides the sequence of design features, this approach also requires the geometric development of the 3D model as training data basis (see chapter 2.3). This paper considers the first two stages of development.

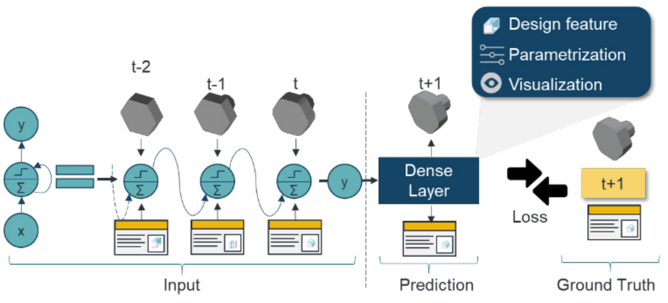


Fig. 4. General RNN architecture to predict next design steps.

In order to investigate possible sequential patterns two approaches are tested. First, one only takes a sequence of integers (feature types) as input, passes it through the embedding layer and then calculates an internal state of the RNN layer, which is the basis for the next feature prediction. Both during training and inference the neural net gets past features as input and has one output only (next feature prediction). Nevertheless, it has a many-to-many architecture, since the prediction can continue until the end of sentence (<EOS> token) is produced as output. Hereinafter, this architecture will be referred to as vanilla RNN.
The second approach also takes into account features' dimensions both for input and output. Since the neural net takes two different types on data as input (categorical integers and float vectors), it has two separate RNN branches for each of them. After the internal states of both RNN branches are calculated, they are concatenated to create a single state vector. However, since the dimension of this vector is not suitable for either of the tasks (to predict the next feature type and the corresponding dimensions), it is passed through two additional separate dense layers with proper output dimensions. Although the neural net technically creates two outputs, it still takes a sequence of features and the corresponding dimensions as input and can make predictions until <EOS> token is produced, therefore it can also be understood as a many-to-many net. Hereinafter, this type of architecture is referred to as multivariate RNN.

3. Case Study

The architecture of the vanilla RNN introduced in chapter 0 for predicting next design steps is shown as proof of concept using an exemplary data set from the field of mechanical engineering. After examining the data set for the presence of design patterns, the corresponding patterns are extracted for an exemplary component class and the RNN is trained.

3.1. Considered data base

Within the framework of the case study, CAD models of an industrial application partner in the field of mechanical engineering are used, which were created with PTC Creo® Parametric. The CAD models in .prt format come from the four component classes flange, housing, cover and adapter. Exemplary representatives of the classes are shown in Fig. 5. All four classes are used to examine the design samples. The implementation of the vanilla RNN is illustrated by the class cover.

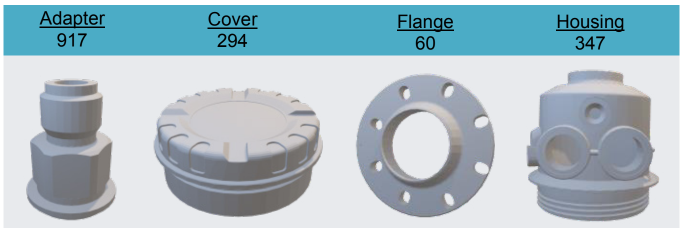


Fig. 5. Exemplary objects from the considered component classes and respective number of models.

3.2. Investigation of design patterns

First the model trees are extracted from the CAD models of the considered data base with the help of PTC Creo® Object TOOLKIT Java. A short overview over the lengths of sequences extracted from the for different component classes is given in Table 1.

Table 1. Overview of feature types and sequence lengths.

Class	Number of unique features	Sequence lengths			
		min	max	mean	std
Adapter	18	10	107	30.87	11.68
Cover	25	15	262	111.28	68.72
Flange	17	10	159	69.54	49.36
Housing	26	11	553	217.62	117.36

An overview over the distribution of these feature types and the most frequent features from each group is represented in Table 2 for the class of covers. The overall number of unique features is 25.

Table 2. Overview of distribution of feature types and most frequent features for cover class.

Feature type	Counts in dataset	Share in dataset [%]	Most frequent feature	Counts in dataset	Share in dataset [%]
datum	18270	49.45	datum plane	15004	40.61
shapes	9131	24.72	cut	7216	19.53
engin.	8344	22.59	round	5311	14.38
editing	1074	3.25	pattern	697	1.89
surfaces	125	0.34	surface	125	0.34

In a first investigation, the similarity matrix of the model trees is calculated according to chapter 2.2 for the individual component classes. Fig. 6 (a) shows the results for the class adapter in form of the frequencies of the occurred similarity values. Obviously there is an overlap for a large part of the model trees, which suggests the existence of design patterns within a component class. For the investigation across different component classes, the data set is first divided into a subset consisting of the same number of model trees from each class. Subsequently, the similarity matrix is calculated and based on this, a density-based clustering is performed. The results are shown in Fig. 6 (b). Most clusters are class-compliant. Only in two clusters there seem to be similarities also between model trees from different component classes. It is also apparent that there can be several design approaches in one class of building up components. For example, the model trees of the "cover" class have been divided into three different clusters.

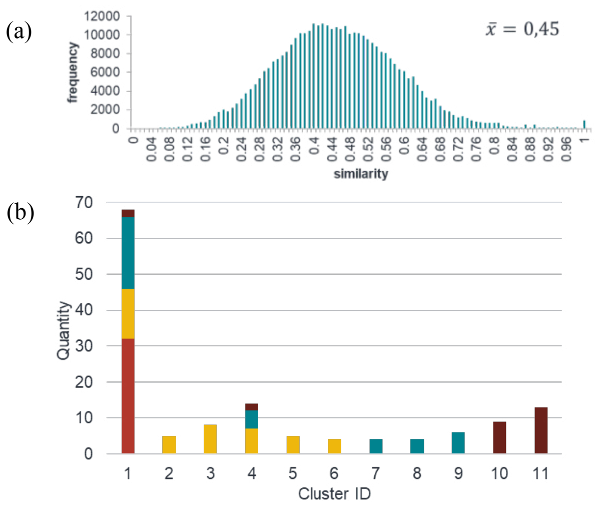


Fig. 6. (a) Histogram of similarities for adapter class and (b) obtained clusters based on the similarities of the model trees of the four investigated component classes.

3.3. Next step prediction

The architecture of the RNN presented in chapter 0 is exemplarily tested on the component class of covers (see chapter 3.1). For this purpose, the data set is prepared according to chapter 2.3. In the considered investigation, only the sequence of design features serves as input. Desired output is the next design step in form of the next design feature. An overview over the particular net architecture is presented in Table 3. As for recurrent layers, sigmoid activation function is used in all cases. No regularization is applied. Categorical cross entropy loss function is used as an optimization objective. For training, Adam algorithm with default TensorFlow settings is used.

Table 3. RNN Architecture

Layer	# of units	Activation function
Input	25	linear
Embedding	16	linear
RNN/ LSTM	16	sigmoid
Dense	25	linear

For the training, different input sequence lengths are tested. Fig. 7 (a) represents the achieved training and validation accuracy for according feature sequence lengths. Obviously, longer sequences make it easier for the net to make legit predictions. The maximum possible input sequence length depends on the minimal sequence length of the data set (see Table 1). For the given dataset, the accuracy does not saturate even by the maximum length of 15 features. Since the latter case shows the best accuracy with about 72 %, a typical learning curve for this training case is presented in Fig. 7 (b).

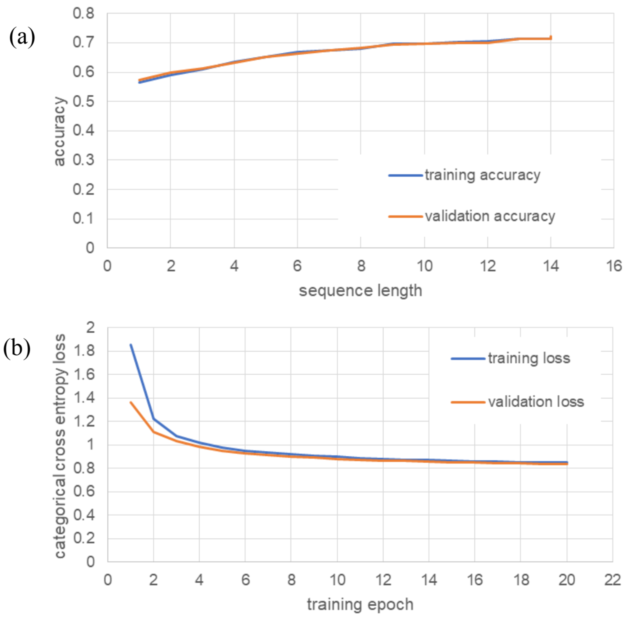


Fig. 7. (a) Training and validation accuracy for different lengths of input sequence and (b) course of loss over training epochs.

4. Discussion

For the investigated data set, it could be shown that there are obviously design patterns in the design procedure, especially within a component class. However, the occurrence of the patterns depends on the data set. Based on those patterns, an RNN architecture is trained and an accuracy of 72 % could be achieved for the considered component class. It is worth mentioning that the extracted feature sequences are not the only possible ones. The RNN's output is, however, still penalized every time it does not correspond to next feature that was extracted for building up sequences out of the model tree. Therefore, other metrics may be helpful to better evaluate the

RNN's performance. It is important to note that the validation loss is lower in the beginning of training which is the same for all investigated sequence lengths. This phenomenon can be explained by the way the loss is calculated: while the training loss is computed during the training, the validation is always computed after each training epoch, where the net is already trained, hence has better performance. Furthermore, it is assumed that the models, on the basis of which design patterns are extracted and the RNN is trained, are optimal and meet all requirements, e.g. manufacturability and cost-effectiveness.

5. Conclusion and Outlook

In this paper it is shown how implicit knowledge from existing CAD models can be extracted and formalized using ML. The implicit knowledge is represented by design patterns which, based on the CAD model tree, represent the design procedure of the designer via design features. By using those proven patterns in the development of new product generations and variants, the design process can be accelerated on the one hand and the implicit knowledge contained therein can be transferred on the other. In a first investigation, it is shown for an exemplary data set of an industrial application partner that patterns do exist in the design process. Based on the extracted model trees in the form of sequences, a Recurrent Neural Network (RNN) is applied to learn patterns in order to predict the next design steps. In a first case study the architecture is used to predict the next design feature based on the previous design steps. In further investigations, the approach is initially to be extended by the prediction of corresponding parameterizations of the individual design features. In a next step not only the sequence of the design features shall serve as input, but also the corresponding development of the 3D geometry. Based on this, it is also possible to visually represent the next design step.

Acknowledgements

This paper was also funded by the Federal Ministry of Education and Research (BMBF) project AIAx, Machine Learning-driven Engineering – CAx goes AIAx (01IS18048B).

References

[1]Fleischer, B., 2019. Methodisches Konstruieren in Ausbildung und Beruf. Springer Fachmedien Wiesbaden, Wiesbaden.

[2] VDI, 1987. Wirtschaftliche Entscheidungen beim Konstruieren: Methoden und Hilfen.

[3] Albers, A., Bursac, N., Wintergerst, E., 2015. Product generation development-importance and challenges from a design research perspective, 16 pp.

[4] Deigendesch, T., 2009. Kreativität in der Produktentwicklung und Muster als methodisches Hilfsmittel. Creativity in Product Development and Patterns as a Methodological Means of Support.

[5] Kirchner, E., 2020. Werkzeuge und Methoden der Produktentwicklung: Von der Idee zum erfolgreichen Produkt, 1st ed. 2020 ed., 1Online-Ressource (XII, 452 Seiten).

[6] Doellken, M., Zimmerer, C., Matthiesen, S., 2020. CHALLENGES FACED BY DESIGN ENGINEERS WHEN CONSIDERING MANUFACTURING IN DESIGN – AN INTERVIEW STUDY. Proc. Des. Soc.: Des. Conf. 1, 837–846.

[7] Grauberger, P., Bremer, F., Sturm, C., Hoelz, K., Wessels, H., Gwosch, T., Wagner, R., Lanza, G., Albers, A., Matthiesen, S., 2020. QUALITATIVE MODELLING IN EMBODIMENT DESIGN - INVESTIGATING THE CONTACT AND CHANNEL APPROACH THROUGH ANALYSIS OF PROJECTS. Proc. Des. Soc.: Des. Conf. 1, 897–906.

[8] Albers, A., Deigendesch, T., Turki, T., 2009. Design Patterns in Microtechnology. DS 58-5: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 5, Design Methods and Tools (pt. 1), Palo Alto, CA, USA, 24.-27.08.2009, 385–396.

[9] Albers, A., Turki, T., 2013. Supporting design of primary shaped micro parts and systems through provision of experience. Microsyst Technol 19 (3), 471–476.

[10] VDI, 2019. Entwicklung technischer Produkte und Systeme: Gestaltung individueller Produktentwicklungsprozesse.

[11] La Rocca, G., 2012. Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. Advanced Engineering Informatics 26 (2), 159–179.

[12] Cooper, D., LaRocca, G., 09182007. Knowledge-based Techniques for Developing Engineering Applications in the 21st Century, in: 7th AIAA ATIO Conf, 2nd CEIAT Int'l Conf on Innov and Integr in Aero Sciences,17th LTA Systems Tech Conf; followed by 2nd TEOS Forum, p. 333.

[13] Gembarski, P.C., Li, H., Lachmayer, R., 2017. KBE-Modeling Techniques in Standard CAD-Systems: Case Study—Autodesk Inventor Professional, in: Bellemare, J., Carrier, S., Nielsen, K., Piller, F.T. (Eds.), Managing Complexity, vol. 26. Springer International Publishing, Cham, pp. 215–233.

[14] Bai, J., Gao, S., Tang, W., Liu, Y., Guo, S., 2010. Design reuse oriented partial retrieval of CAD models. Computer-Aided Design 42 (12), 1069–1084.

[15] Mital, A., Desai, A., Subramanian, A., Mital, A., 2014. Product Development: A Structured Approach to Consumer Product Development, Design, and Manufacture, 2nd ed. ed. Elsevier Science, Burlington, 539 pp.

[16] Hegde, V., Zadeh, R., 2016. FusionNet: 3D Object Classification Using Multiple Data Representations.

[17] Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space.

[18] Simonovsky, M., Komodakis, N., 2017. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs.

[19] Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view Convolutional Neural Networks for 3D Shape Recognition.

[20] Sarmad, M., Lee, H.J., Kim, Y.M., 2019. RL-GAN-Net: A Reinforcement Learning Agent Controlled GAN Network for Real-Time Point Cloud Shape Completion.

[21] Tchapmi, L.P., Kosaraju, V., Rezatofighi, H., Reid, I., Savarese, S., 2019. TopNet: Structural Point Cloud Decoder, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. Proceedings : 16-20 June 2019, Long Beach, California, pp. 383–392.

[22] Wang, Y., Tan, D.J., Navab, N., Tombari, F., 2020. SoftPoolNet: Shape Descriptor for Point Cloud Completion and Classification.

[23] Xie, H., Yao, H., Zhou, S., Mao, J., Zhang, S., Sun, W., 2020. GRNet: Gridding Residual Network for Dense Point Cloud Completion.

[24] Ahmed, E., Saint, A., Shabayek, A.E.R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., Ottersten, B., 2019. A survey on Deep Learning Advances on Different 3D Data Representations, 35 pp.

[25] Chaudhuri, S., Koltun, V., 2010. Data-driven suggestions for creativity support in 3D modeling, in: ACM SIGGRAPH Asia 2010 papers on - SIGGRAPH ASIA '10, p. 1.

[26] Jaiswal, P., Huang, J., Rai, R., 2016. Assembly-based conceptual 3D modeling with unlabeled components using probabilistic factor graph. Computer-Aided Design 74, 45–54.

[27] Roj, R., 2016. Eine Methode für eine automatisierte Informationsextraktion aus großen CAD-Datenbeständen zur Bauteilsuche und Klassifikation. Dissertation, Wuppertal.

[28] Krahe, C., Iberl, M., Jacob, A., Lanza, G., 2019. AI-based Computer Aided Engineering for automated product design - A first approach with a Multi-View based classification. Procedia CIRP 86, 104–109.

[29] Nikolić, M., 2012. Measuring similarity of graph nodes by neighbor matching. IDA 16 (6), 865–878.References