

Using cluster analysis to enhance a method for the management of disturbance factors via product structures

Richard Breimann , Laura Luran Sun and Eckhard Kirchner

Technische Universität Darmstadt, Germany

 richard.breimann@tu-darmstadt.de

Abstract

To achieve higher functionality in mechatronic systems, the management of disturbance factors plays a crucial role. For this purpose, a method was developed in prior works to address this management via the optimisation of product structures. However, this method lacks applicability due to the complexity of one step of the method. It is the goal of this paper to present a software tool, utilizing cluster-analysis to sort components into assemblies, with which this step is supported. Additionally, the method is generally adapted to address a wider spectrum of phenomena in mechatronic systems.

Keywords: computational design methods, design tools, design optimisation, product architecture, decision making

1. Introduction

To achieve higher functionality in mechatronic products, the management of occurring disturbance factors is crucial, cf. [Meyer zu Westerhausen et al. \(2023\)](#). Disturbance factors can originate from either the environment of the product or the product itself ([Taguchi et al. 2011](#)). To achieve robustness against the influence of disturbance factors on the functionality of products, a number of methods and methodologies exist. One of these methods by [Breimann et al. \(2023a\)](#) focuses on the management of disturbance factors via the product structure. In this method, the relation between individual disturbance factors and components of the product is analysed. Components can either emit, be affected by or sense a disturbance factor. Additionally, if the relation is not relevant, it can be ignored. Based on this analysis, groups of components can be developed to create a product structure, which increases the robustness of the product, or enables the application of other strategies for the management of disturbance factors. However, there is a conflict in the application of the method. On the one hand, the application of the method is only worthwhile for complex products, as the ordering of components in assemblies is trivial in less complex systems. On the other hand, the manual sorting step in the method becomes challenging for more complex products. Therefore, it is the aim of this publication to provide a tool for the application of the method on complex systems, via the introduction of cluster analysis in the form of a software tool called Bacchus – (German: **B**austruktur **A**uswahl durch **c**omputergestützte **C**lusteranalyse **m**ithilfe **U**ser-orientierter **S**oftware).

2. State of the art

In the following subsections, the relevant state of the art is presented. Firstly, an introduction into product architecture design is given to introduce the necessary terminology. Secondly, basics of robust design are explained, on which the method by [Breimann et al. \(2023a\)](#) is based. Finally, the method by [Breimann et al. \(2023a\)](#) is presented, for which a software tool has been developed for this publication.

2.1. Product architecture design

In the context of this contribution, a product can be anything, offered to a person with the aim of fulfilling a specific wish or need. These products are either tangible goods, intangible services or energy services (Kotler et al. 2011). In accordance with Breimann et al. (2023a), for this contribution, the term product is restricted to a system with technical functions, which do not exclusively include sensory and actuator functions. These products consist of components, which are placed in assemblies, which again form the entire product. The placement of the components in their respective assemblies is referred to as the product structure of a product (Krause und Gebhardt 2018). The function of a product is fulfilled via a number of partial functions. The overall function and partial functions result in the functional structure. Since partial functions in a technical system are realised by components, corresponding connections can be drawn in. The connection of functional structure and product structure results in the product architecture as illustrated in Figure 1. Depending on which components are placed in assemblies with other components, the resulting synergies or negative interactions of components determine the functionality of the overall product (Krause und Gebhardt 2018). That is why a number of methods have already been developed, aiming to create specific synergies and avoiding specific negative interactions. These methods can address various aims, such as interactions between components, see Steward (1981), synergies over all life-phases, see Erixon et al. (1996), and the implementation of R-imperatives to promote circular economy, see Breimann et al. (2023b).

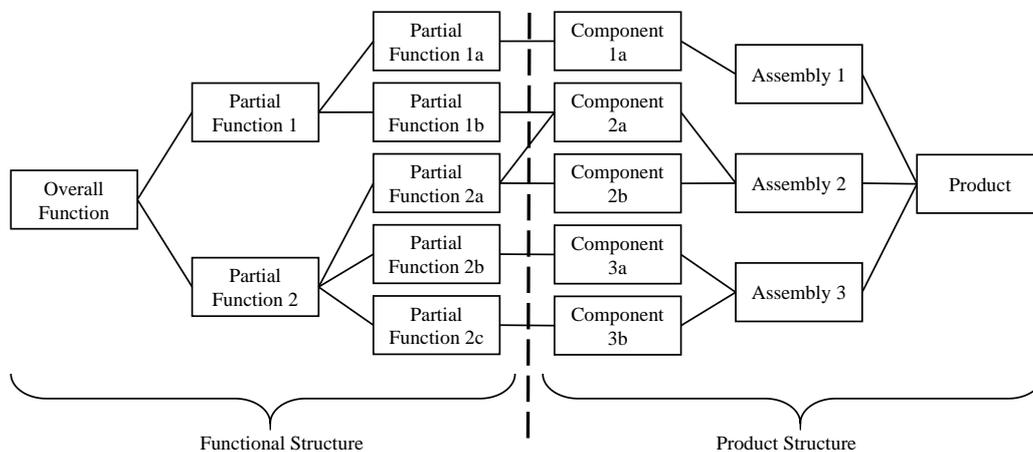


Figure 1. Schematic representation of a product architecture, based on Krause und Gebhardt (2018)

2.2. Robust design

According to Taguchi et al., a process or product is considered robust, when it has limited or reduced functional variation, even in the case of occurring disturbance factors. The discipline in design aiming to create such robust systems, is called robust design. (Taguchi et al. 2011) In the context of robust design, a number of different methods have been developed, which are not explained further here. Based on the point at which the strategies prevent the influence of the disturbance factor on the system, three categories of strategies can be identified (Mathias et al. 2010):

- Eliminate disturbance factor: The occurrence of the disturbance itself is prevented.
- Reduce/eliminate disturbance factor influence: The disturbance factor occurs, but the system is designed in a way, that prevents the disturbance factor from affecting the product or process.
- Eliminate/reduce disturbance factor impact: The existence of the disturbance factor and its influence on the system is inevitable, but the system is designed so that the impact of the disturbance factor does not limit the functionality of the process or product.

According to Mathias et al. the effectiveness or feasibility of these strategies does not have to be present under any circumstances (Mathias et al. 2010). Breimann et al. state, that among other factors, the product structure determines whether these strategies can be applied and how laborious their application

is (Breimann et al. 2023a). The optimisation of the product structure to create robust products or enable robust design strategies is subject to the method presented in the next sub-section.

2.3. Management of disturbance factors via the optimization product structures

Breimann et al. (2023a) have stated, that a component can have either of four relations regarding a specific disturbance factor. These relations are as follows:

- Emitter [E], which means that the component emits a disturbance factor,
- Affector [A], which means that a disturbance factor has a negative effect on the component,
- Sensor [S], which means that the component is used to detect a physical quantity which acts as a disturbance factor on other components,
- Irrelevant [I], which means that the disturbance factor has no relevant impact on the component in this specific system.

Based on these relations, Breimann et al. (2023a) developed a set of characteristics to group or separate components. These characteristics can be summarised by the following goal:

Groups of components should be formed in a way, so that components in the same assembly have the same relation to the same disturbance factors.

This goal leads Breimann et al. (2023a) to a method, consisting of six steps, which are explained using the example of a sensor-integrated timing belt. First, the system is divided into the individual components. These components are an acceleration sensor, a temperature sensor, a microcontroller, a bluetooth-transmitter, a battery, and a timing belt, see Figure 2 at the top. In the second step the disturbance factors are determined, which are heat, electric charge, vibration, and deformation, see Figure 2 on the left. Then, the components are classified according to their relation to the occurring disturbance factors, see above, and these relations are plotted into a matrix, see Figure 2 centre. In the resulting matrix, groups of components are identified, which have the same behaviour regarding the same disturbance factors. These groups of components then form the assemblies in the resulting product architecture. In the example of the timing belt, the first group of components forming an assembly are the acceleration sensor, the temperature sensor, the microcontroller, the battery and the bluetooth-transmitter. These components are placed in one assembly because these are all affected by electric charge, vibration, and deformation. Additionally, the battery is affected by heat. The timing belt is the only component emitting disturbance factors, so it is also placed in its own assembly.

Whether a component is classified as an Emitter or an Affector, or whether the relation can be ignored, is up to the discretion of the designer and is dependent on the specific system. Here, this method relies on support from other methods, which specialise on the evaluation of the criticality of individual disturbance factors, see Welzbacher et al. (2021; 2022; 2023).

		Set of components					
		Acceleration sensor	Temperature sensor	Micro-controller	Bluetooth-Transmitter	Battery	Timing Belt
Disturbance Factors	Vibration	S	A	A	A	A	E
	Deformation	A	A	A	A	A	E
	Electric Charge	A	A	A	A	A	E
	Heat	I	S	I	I	A	I

Figure 2. Component disturbance factor-relation matrix sorted for the optimisation of disturbance factor management

Due to the subjectivity, it can be stated that the assignment of relationships between disturbances factors and components is not unambiguous. In addition, the example of the battery shows that a component can play several relations in a system, as a battery is damaged by heat that it emits itself. This means that the relationships between components and disturbance factors determined by Breimann et al. (2023a) are not complete.

2.4. Cluster analysis

Clustering is essentially a method for finding groups in a set of objects, which have similar properties, e.g. colour, size etc., while each of these properties can have different characteristics, e.g. blue, green, yellow etc. While humans are able to find clusters in a set of objects intuitively, numerical approaches originated in biology and natural sciences, trying to provide objective and stable classifications of objects. (Everitt et al. 2011)

In numerical approaches, these analyses can be based on similarity-, or dissimilarity-matrices, the latter also being called distance-matrices (Runkler 2015). These matrices contain information on how similar/dissimilar two objects are. The more similar two objects are, the higher the value of similarity or lower the value of dissimilarity. As similarity and dissimilarity-matrices have the same proportionality, they can be converted into each other. The specific similarities are calculated comparing the different properties of the objects. (Runkler 2015) In the case that a property can have several characteristics for different objects, calculations rules need to be defined for the different combinations of properties of two objects (Everitt et al. 2011). Finally, the individual properties can be weighted to focus the calculation of the similarity matrix and thus the cluster analysis (Everitt et al. 2011).

Additionally to the calculation of similarity between individual objects to form clusters of two objects, cluster analysis also needs to consider which cluster to merge into bigger clusters. Common approaches to do so are inter-group proximity measures and an additional linkage method by Ward (1963). (Müller 2004, Everitt et al. 2011) Inter-proximity measures calculate similarity of two clusters based on the similarities of the objects in the two clusters. Here, the highest occurring similarity can be used (Single linkage), the lowest occurring similarity can be used (complete linkage), all similarities can be used equally (group average linkage) or the different similarities can be weighted creating specific approaches. Finally, the method by Ward (1963) uses an error sum-of-squares criterion. The aim is to minimise the resulting error sum within a cluster when merging two clusters (Everitt et al. 2011). Based on the selected method, objects are combined to clusters and these resulting small clusters to bigger clusters until a specified termination condition is reached. This condition can e.g. be the number of clusters or the tolerated value of dissimilarity within a cluster. This approach forming clusters from single objects is called agglomerative. (Bacher 2010)

When comparing the above mentioned linkage methods, it is apparent that the single linkage method will aggregate the data mostly around one cluster as it only considers the closest object without regard to the other objects in its cluster. Group average linkage has shown to only be best for special occasions, such as very uneven or over specified cluster sizes. The method by Ward has proven to be accurate in a number of studies, as none of the problems of other linkage methods occur. While complete linkage also performs well, it is overall outperformed by the Ward's method. (Saraçlı et al. 2013)

The result from a hierarchical cluster analysis can be illustrated in a dendrogram, which is a graph showing all objects in the cluster analysis on the abscissa. The ordinate shows the value of dissimilarity tolerated to form that cluster. (Everitt et al. 2011) A dendrogram can be seen in Figure 5.

Apart from hierarchical cluster analysis, there are also partitioning-based and density-based approaches. K-Means is a partitioning-based method which minimises the sum of squared distances between objects and cluster centroids, which are the focal points of a cluster. Instead of looking at the location of the centroids, the DBSCAN method separates clusters by receiving a maximum distance and minimum cluster size as input and then looking at the number of neighbours each object has, making it a density-based approach. (Dudik et al. 2015) As these approaches are not implemented into the software, presented in section 5 at the present time, a more detailed explanation is not provided here.

Common cluster analysis techniques can be found in online libraries. The specific code used for the development in this contribution is cited at a later point in the paper.

3. Research gap

The application of the method, presented by Breimann et al. (2023a), is only worthwhile when applied to complex systems, as the ordering of components in assemblies is trivial in less complex systems. However, with an increasing complexity in the system, the fifth step of the method becomes challenging. The resulting matrix, used for this step becomes unwieldy. This in turn leads to a conflict, since these

two findings in combination mean that the method can only be applied efficiently to systems with a specific complexity. Additionally, the method and the relations of components and disturbance factors, used for the method, do not cover all the relations a component can have to a specific disturbance factor. Specifically, there are components that emit disturbance factors by which they themselves are affected. Therefore the aim of this contribution is:

First, to change the method in a way, that the list of possible relations between components and disturbance factors is more comprehensive and identified gaps are closed.

Second, to present a tool that makes it possible to apply the method to systems of greater complexity.

4. Alterations to the method

In the existing method, the classification of components regarding the disturbance factors distinguishes between four different relations. However, this classification is not necessarily unambiguous for all components. That means a component, which is emitting a specific disturbance factor might also be affected by the same disturbance factor, resulting in a need for a fifth type of classification. Subsequently, this additional classification must be considered in the method and alterations to the method need to be identified. This need for an additional relation can be illustrated by the component battery from the example in section 2.3. Since a battery not only emits heat, but is also affected by heat, a battery can have the relation of an emitter as well as an affector at the same time. This new relation is called Emitter-Affector. To ensure robustness in the system, this component needs to be split from any component, being affected by, or emitting that specific disturbance factor. Whether a component needs to be classified as Emitter-Affector or a classification as Emitter or Affector is sufficient, is up to the discretion of the developer of the overall system and depends on that specific system. To illustrate the changes to the method, the example from section 2.3 is changed accordingly. The relation between the battery and the disturbance factor heat is changed to Emitter-Affector, see Figure 3. Subsequently, the assemblies are changed. In the new product structure, the battery is placed in an isolated assembly, since the heat emitted by the battery affects the acceleration sensor, the temperature sensor, the micro-controller, and the Bluetooth-transmitter.

		Set of components					
		Acceleration sensor	Temperature sensor	Micro-controller	Bluetooth-Transmitter	Battery	Timing Belt
Disturbance Factors	Vibration	S	A	A	A	A	E
	Deformation	A	A	A	A	A	E
	Electric Charge	A	A	A	A	A	E
	Heat	I	S	I	I	E-A	I

Figure 3. Component disturbance factor-relation matrix considering the changed relation of the battery and the disturbance factor heat

5. Development of Bacchus

In the following, the development process is presented, with the help of which the software is developed. First, the requirements for the software are presented. Secondly, the sorting procedure is selected. Thirdly, the logic and mathematics on which the software is based is presented. Finally, the Graphical User Interface (GUI) is presented and the user application of the software is shown.

5.1. Definition of requirements

Prior to the development of the logic and the software, a set of requirements is established for the development process. The requirements focus on the usability of the software by the user. However, additional requirements are included, which address the accessibility of the code, the extension of the software, as well as the flexibility in the algorithms used. The most relevant requirements are presented in the following. **First**, the software must be automated in a way that no user interventions are required

during the sorting process. **Second**, the user needs to be able to weight the different disturbance factors in the Component Disturbance Factor-Relation Matrix differently when using the programme. **Third**, the user must be able to choose between different approaches to calculate the similarity matrix. **Fourth**, the user must be able to choose between different cluster analysis algorithms in the software. That is necessary, because the algorithms can change the outcome of the cluster analysis. **Fifth**, the software modules used must be stored locally. Furthermore, any code used must be inspected to ensure its desired functionality. **Sixth**, the interface must be understandable for the user. That means that the user does not need acquire any knowledge beyond the understanding of the method. **Seventh**, the results must be presented visually as well as in the form of text. **Eighth**, the reasons for clustering components into assemblies must be apparent from the GUI.

5.2. Logic and mathematics of the software

Before developing a logic for sorting components, the basic procedure for performing the sorting is selected. Optimisation algorithms and cluster analysis are identified as possible methods. Cluster analysis attempts to map intuitive sorting that can be carried out by humans using software. On the other hand, optimisation algorithms work by maximising or minimising an objective function considering given constraints. Since cluster analysis comes closest to the sorting step in the existing method, this procedure is chosen for the software. This also means that the requirement for comprehensibility of the software is better met, as the software only imitates the steps that would otherwise be carried out by humans. A comparison of the quality of results of cluster analysis and optimisation algorithms is postponed to subsequent research.

5.2.1. Similarity matrix

To perform the cluster analysis, the component disturbance factor-relation matrix has to be transformed into a similarity matrix first. To do so, the components are compared in pairs. When two different components are compared, the relationships between all disturbance factors and the components are considered. The goal is to increase the similarity of components which have the same relation to the same disturbance factors. If the relation of a disturbance factor with at least one of the two components is either “Sensor” (S) or “Irrelevant” (I), the similarity between these two components stays unchanged (-), see Table 1. Two “Emitters” (E), as well as two “Affectors” (A) of a disturbance factor each increase the two components similarity, indicated in Table 1 by an upward facing arrow. In the case of at least one component being “Emitter-Affector” (E-A), any comparison to E, A, or another E-A will decrease the similarity between the two components, indicated by a downwards faced arrow. Table 1 shows an overview of how the similarity of two components changes depending on the pairing of relations regarding disturbance factors.

Table 1. Influence of specific disturbance factor-component relation on similarity between two components

	E	A	S	I	E-A
E	↑	↓	-	-	↓
A	↓	↑	-	-	↓
S	-	-	-	-	-
I	-	-	-	-	-
E-A	↓	↓	-	-	↓

After the relations of the two components to all disturbance factors are compared, the overall similarity between these two is stored in the similarity matrix, which is a symmetric component-component matrix. The principal diagonal is set to zero, as that is the similarity of a component with itself that is of no relevance for the subsequent cluster analysis. The quantitative change of the similarity between two components depends on the weight of the disturbance factor that is being looked at. The default weight is 1.0, but can be increased or decreased depending on the relevance of that disturbance factor, which is at the discretion of the user. That allows the user to consider the varying relevance of different

disturbance factors for the design of the system. The specific values can be assigned based on experience or be defined iteratively. The three approaches, linear, quotient and squared quotient, chosen to calculate the similarities and respective matrices are as follows. These approaches are intuitively derived from the desired dependencies, shown in table 1. $S_{i,j}$ is the similarity between components i and j , $S_{i,j,k}$ is the specific similarity between components i and j for disturbance factor k .

- Linear, which means that the default similarity $S_{i,j}$ is zero and linearly increases or decreases by the weight of the current disturbance factor, see formula 1. That means, that when two components are either both Emittor or Affector, $S_{i,j,k}$ is the positive weight. If one of the components is Emittor and the other one is Affector or if any of the two is Emittor-Affector and the other one is Emittor, Affector or Emittor-Affector, $S_{i,j,k}$ is the negative weight. If any of the two components is Sensor or Ignore, $S_{i,j,k}$ is zero.

$$S_{i,j} = \sum_k S_{i,j,k} \text{ with } S_{i,j,k} = \pm \text{weight or zero} \quad (1)$$

- Quotient, which means that the default $S_{i,j}$ is 1.0, with both the numerator and denominator at 1.0. An increase in similarity, see Table 1, increases the numerator by the disturbance weight, indicated by the index p . A decrease in similarity, see Table 1, increases the denominator by the disturbance weight, indicated by the index n , see formula 2. That means, that when two components are either both Emittor or Affector, $S_{i,j,k}$ is the positive weight and added to the numerator. If one of the components is Emittor and the other one is Affector or if any of the two is Emittor-Affector and the other one is Emittor, Affector or Emittor-Affector, $S_{i,j,k}$ is the positive weight and added to the denominator. If any of the two components is Sensor or Ignore, the quotient remains unchanged.

$$S_{i,j} = \frac{\sum_k S_{i,j,k,p}}{\sum_k S_{i,j,k,n}} \quad (2)$$

- Squared quotient, which follows the same principle as quotient, only that the denominator is squared in the end to create the overall similarity, see formula 3.

$$S_{i,j} = \frac{\sum_k S_{i,j,k,p}}{(\sum_k S_{i,j,k,n})^2} \quad (3)$$

Figure 4 shows the exemplary calculation of the similarity of two components for the three different approaches presented earlier. The data used in the calculation is shown in the left in the form of weights and relationships between disturbance factors and components. The summands $S_{i,j,k}$ determined from this are shown in the centre. Finally, the resulting similarities are presented on the right as a function of the three possible approaches.

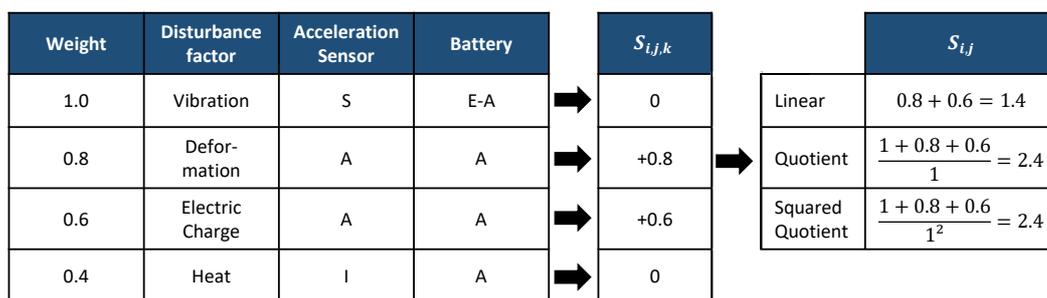


Figure 4. Specific similarity calculation for two exemplary components

As the test data required for a comparison of the different approaches is not available, a comparison and evaluation of the quality of the approaches is not conducted at this point.

5.2.2. Cluster analysis

The software modules on which Bacchus is based have not all been developed from the ground up. In order to make use of existing knowledge, online databases were used and the code taken from there was

revised according to the new application. Namely, code for the different cluster analysis methods was taken from [Github \(2023a, 2023b, 2023c\)](#) as well as software modules specific for hierarchical cluster analysis ([Github 2023d](#)).

The three exemplary cluster analysis methods selected for implementation are the following (see 2.4):

- Agglomerative
- K-Means
- DBSCAN

The initial main approach focuses on agglomerative clustering, as it best meets the requirements of a comprehensible method. That is due to its hierarchical nature, since it allows the illustration of the results in form of a dendrogram, which neither the K-Means nor the DCSCAN method do. Initially, the linkage method used is the Ward's method, as it is proven to be well performing (see 2.4). The option to select other linkage methods is not implemented yet. The provided code is here modified to be able to use the similarity matrix as input, as it originally requires a condensed distance matrix. To do so, the similarity matrix is first normalised to values between 0 and 1, and then transformed into a distance matrix by subtracting each value from 1, mirroring the values of similarity around the value of 0.5. This symmetric distance matrix is condensed, so no redundant entries are left. Afterwards, a base distance of 1 is set for the components with the highest similarity, shifting the normalisation to values between 1 and 2. The reason for that is that due to the normalisation, the highest similarity between components in the distance matrix equals a distance of zero. This however, is not the case, so a base distance is artificially added. That allows the comparability between applications, as the similarities are in the same number range, allowing users to gain experience with the software. Depending on the linkage method, that number range can shift upwards, as with higher dissimilarity, the distance can become higher than 2. This condensed distance matrix is then finally used as input for the cluster algorithm. This modified code was then tested using a test data set as described below.

5.3. Development of the Graphical User Interface

All functions of the software Bacchus are integrated into a Graphical User Interface (GUI) as seen in Figure 5. A CSV file containing the component disturbance factor-relation matrix can be uploaded (1) and then altered within the GUI if necessary. If the CSV file does not include the “Weights” (2) column,

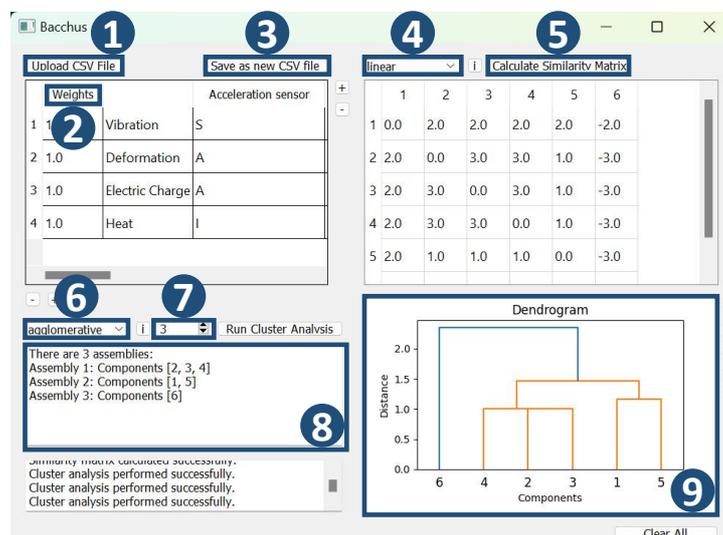


Figure 5. Graphical User Interface

it will be automatically added upon uploading. All entries including weights, the components and disturbance factors and the relations can be edited. The new, edited CSV file can then be saved as a new CSV file (3). Afterwards, it can be chosen which approach (as described in 5.3.1) (4) should be used to calculate the similarity matrix (5). In this step, if there are any invalid entries in the component disturbance factor-relation matrix, a descriptive error appears in the text field (8). Following that, the

cluster analysis method is chosen (see 5.3.2) (6), including the number of clusters desired (7). The text field (8) shows the assemblies including the respective components that belong in that group. In case of hierarchical clustering, the dendrogram (9) showing the results is presented as well. The data use in the GUI are equivalent to the data from section 4. However, the components are numbered and therefore do not have their original names. It can be seen that the results of the cluster analysis from the GUI differ from the assemblies identified in section 4. This is due to the fact that the acceleration sensor has the same similarity to the temperature sensor, bluetooth transmitter and microcontroller as it does to the battery. Therefore, both variants are possible.

6. Conclusion

In this publication, significant weaknesses were identified in a published method for the management of disturbance factors. To remedy these weaknesses, the method was adapted in a first step by integrating further possibilities of how components and disturbance factors can relate to each other. In addition, a software was developed that allows a time-consuming step in the application of the method to be carried out computer-aided. For this purpose, different basic approaches of computer support were first identified, evaluated and then cluster analysis was selected, implemented and a GUI was developed. This tool significantly increases the applicability of the method by [Breimann et al. \(2023a\)](#).

7. Outlook

Following this publication, there are a number of opportunities to further develop the software. First, the software needs to be completed with the cluster analysis algorithms that have not yet been implemented. Furthermore, the traceability of the steps of the cluster analysis can be improved by linking the individual branches in the dendrogram with information that shows why components are combined into a cluster. In addition, the software can be extended to include further methods of modularisation or general assembly design. This makes it possible to pursue several goals at the same time in the design of assemblies when using the software. Here, it must be examined whether the use of cluster analysis is more sensible than the use of optimisation algorithms or other approaches. For this, it must be first clarified whether the method on which the software is based can be supported by other methods, such as Axiomatic design, see [Suh \(2001\)](#). Furthermore, the software has so far been designed for use in a design if the assemblies are developed from scratch or a complete redesign of the assemblies is undertaken. It is necessary to examine how the software can be optimised so that systems can be redesigned with minimal effort to improve its properties under given objectives.

Finally, the efficiency and effectiveness of the presented software and its potential extension must be validated. In terms of effectiveness, it must be checked whether the proposed cluster analysis is able to generate applicable solutions for the underlying design problem. If too many boundary conditions beyond robust design are ignored in the method and the software, this could mean that the solutions proposed by the software are not applicable, as conflicts arise that cannot be mapped by the method and the software and may not be easy to resolve. In terms of effectiveness, the extent to which the software is able to generate not only good quality solutions, but also to what extent this supports the developer in developing or iterating more quickly, must be checked. For both efficiency and effectiveness, however, a study that can only be realised after the software has been expanded as described above is required.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 466493340 & 426030644

References

- Bacher, Johann (2010): Clusteranalyse. Anwendungsorientierte Einführung in Klassifikationsverfahren. 3., erg., vollst. überarb. u. neu gestalt. Aufl. München: Oldenbourg.
- Breimann, Richard; Fett, Michel; Küchenhof, Jan; Gombert, Ilja; Kirchner, Eckhard; Krause, Dieter; Trieu, Hoc Khiem (2023a): A method for optimizing product architectures for the management of disturbance factors. In: *Procedia CIRP* 119, S. 1041–1046. <https://dx.doi.org/10.1016/j.procir.2023.02.179>.

- Breimann, Richard; Rennpferdt, Christoph; Wehrend, Sven; Kirchner, Eckhard; Krause, Dieter (2023b): Exploiting the sustainability potential of modular products by integrating R-imperatives into product life phases. In: *Proc. Des. Soc.* 3, S. 1785–1794. <https://dx.doi.org/10.1017/pds.2023.179>.
- Dudik, Joshua M.; Kurosu, Atsuko; Coyle, James L.; Sejdíć, Ervin (2015): A comparative analysis of DBSCAN, K-means, and quadratic variation algorithms for automatic identification of swallows from swallowing accelerometry signals. In: *Computers in biology and medicine* 59, S. 10–18. <https://dx.doi.org/10.1016/j.compbimed.2015.01.007>.
- Erixon, Gunnar; Yxkull, Alex von; Arnström, Anders (1996): Modularity – the Basis for Product and Factory Reengineering. In: *CIRP Annals* 45 (1), S. 1–6. [https://dx.doi.org/10.1016/S0007-8506\(07\)63005-4](https://dx.doi.org/10.1016/S0007-8506(07)63005-4).
- Everitt, Brian S.; Landau, Sabine; Leese, Morven; Stah, Daniel (2011): Cluster analysis. 5th ed. Chichester: Wiley (Wiley series in probability and statistics).
- Github (2023a): `_agglomerative.py`. Online verfügbar unter https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/cluster/_agglomerative.py, zuletzt aktualisiert am 26.10.2023.
- Github (2023b): `_dbscan.py`. Online verfügbar unter https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/cluster/_dbscan.py, zuletzt aktualisiert am 26.10.2023.
- Github (2023c): `_kmeans.py`. Online verfügbar unter https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/cluster/_kmeans.py, zuletzt aktualisiert am 26.10.2023.
- Github (2023d): `hierarchy.py`. Online verfügbar unter <https://github.com/scipy/scipy/blob/main/scipy/cluster/hierarchy.py>, zuletzt aktualisiert am 26.10.2023.
- Kotler, Philip; Keller, Kevin Lane; Bliemel, Friedhelm (2011): Marketing-Management. Strategien für wertschaffendes Handeln. 12., aktualisierte Aufl., [Nachdr.]. München: Pearson Studium (Wi - Wirtschaft).
- Krause, Dieter; Gebhardt, Nicolas (2018): Methodische Entwicklung modularer Produktfamilien. Hohe Produktvielfalt beherrschbar entwickeln. 1. Aufl. 2018. Berlin, Heidelberg: Springer Berlin Heidelberg. Online verfügbar unter <http://nbn-resolving.org/urn:nbn:de:bsz:31-epflicht-1501590>.
- Mathias, J.; Kloberdanz, H.; Engelhardt, R.; Birkhofer, H. (2010): Strategies and principles to design robust products. In: *DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference, Dubrovnik, Croatia*, S. 341–350. Online verfügbar unter <https://www.designsociety.org/publication/29379/STRATEGIES+AND+PRINCIPLES+TO+DESIGN+ROBUST+PRODUCTS>.
- Meyer zu Westerhausen, Sören; Schneider, Jannik; Lachmayer, Roland (2023): Reliability analysis for sensor networks and their data acquisition: A systematic literature review. In: *Proc. Des. Soc.* 3, S. 3065–3074. <https://dx.doi.org/10.1017/pds.2023.307>.
- Müller, W. (2004): Multivariate Statistik im Quantitativen Marketing: Konzeption und Anwendungsbereiche der Clusteranalyse. Online verfügbar unter https://opus.bsz-bw.de/fhdo/files/36/iamm-_clusteranalyse_im_marketing.pdf.
- Runkler, Thomas A. (2015): Data mining. Modelle und Algorithmen intelligenter Datenanalyse. 2., aktualisierte Auflage. Wiesbaden: Springer Fachmedien (Lehrbuch).
- Saraçlı, Sinan; Doğan, Nurhan; Doğan, İsmet (2013): Comparison of hierarchical cluster analysis methods by cophenetic correlation. In: *J Inequal Appl* 2013 (1), S. 1–8. <https://dx.doi.org/10.1186/1029-242X-2013-203>.
- Steward, Donald V. (1981): The design structure system: A method for managing the design of complex systems. In: *IEEE Trans. Eng. Manage.* EM-28 (3), S. 71–74. <https://dx.doi.org/10.1109/TEM.1981.6448589>.
- Suh, Nam P. (2001): Axiomatic design. Advances and applications. New York, Oxford: Oxford University Press (CIRP design book series). Online verfügbar unter <http://www.loc.gov/catdir/enhancements/fy0610/00040635-d.html>.
- Taguchi, Genichi; Chowdhury, Subir; Wu, Yui; Taguchi, Shin; Yano, Hiroshi (Hg.) (2011): Taguchi's quality engineering handbook. Hoboken, N.J, Livonia, Mich: John Wiley & Sons. Online verfügbar unter <http://onlinelibrary.wiley.com/book/10.1002/9780470258354>.
- Ward, Joe H. (1963): Hierarchical Grouping to Optimize an Objective Function. In: *Journal of the American Statistical Association* 58 (301), S. 236. <https://dx.doi.org/10.2307/2282967>.
- Welzbacher, P.; Puchtler, S.; Geipl, A.; Kirchner, E. (2022): Uncertainty Analysis of a Calculation Model for Electric Bearing Impedance. In: *Proc. Des. Soc.* 2, S. 653–662. <https://dx.doi.org/10.1017/pds.2022.67>.
- Welzbacher, Peter; Geipl, Anja; Kraus, Benjamin; Puchtler, Steffen; Kirchner, Eckhard (2023): A follow-up on the methodical framework for the identification, analysis and consideration of uncertainty in the context of the integration of sensory functions by means of sensing machine elements. In: *Proc. Des. Soc.* 3, S. 141–150. <https://dx.doi.org/10.1017/pds.2023.15>.
- Welzbacher, Peter; Vorwerk-Handing, Gunnar; Kirchner, Eckhard (2021): A control list for the systematic identification of disturbance factors. In: *Proc. Des. Soc.* 1, S. 51–60. <https://dx.doi.org/10.1017/pds.2021.6>.