Systematic module and interface definition using component design structure matrix

Robert Helmer a, Ali Yassine b & Christoph Meier b

a Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA
b Institute of Astronautics, Technical University Munich, Garching bei München, Germany

Available online: 08 Dec 2008

To cite this article: Robert Helmer, Ali Yassine & Christoph Meier (2010): Systematic module and interface definition using component design structure matrix, Journal of Engineering Design, 21:6, 647-675

To link to this article: http://dx.doi.org/10.1080/09544820802563226
Systematic module and interface definition using component design structure matrix

Robert Helmer\textsuperscript{a}, Ali Yassine\textsuperscript{b}\textsuperscript{*} and Christoph Meier\textsuperscript{b}

\textsuperscript{a}Department of Industrial and Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, USA; \textsuperscript{b}Institute of Astronautics, Technical University Munich, Garching bei München, Germany

(Received 5 November 2007; final version received 19 October 2008)

Modular product architectures can offer many benefits, but require carefully chosen interfaces as early as possible in the development process to exploit their full potential. Late or erroneous definition of modules and interfaces results in excessive design iterations and consequently causes unanticipated delays and cost overruns in product development. In this article, an improved method for product architecting (i.e., the identification and definition of modules and their interfaces) using clustering of component-based Design Structure Matrices (DSMs) is presented and demonstrated. Various researchers contributed to the improvement of clustering techniques in the past. However, the approaches used are mostly unsatisfactory with respect to the definition of modules for products resulting in suboptimal or even wrong definitions. In lieu of investigating single steps of DSM clustering, a comprehensive approach is presented in this article, covering all relevant aspects, from data acquisition, and handling of multiple perspectives in DSM clustering, to a post-processing phase, where results are corrected with respect to technical feasibility. By discussing various topics neglected in previous literature, many deficiencies are also revealed and solutions to these problems are proposed, resulting in a systematic and practical procedure for product architecting based on DSM clustering.

Keywords: product development; product architecture; DSM clustering; module and interface definition

1. Introduction

Modularity has been widely recognised as a means to manage complexity in product design processes and to achieve competitive advantages, mostly related to 'business performance' (Sharman and Yassine 2004, Hölttä-Otto and de Weck 2007, Yassine and Wissmann 2007). Successful modular designs are highly dependent on well-defined interfaces between the different modules. One important feature of interfaces is their physical or geometrical design, i.e., properties like size, shape, etc. However, in order to design these interfaces correctly and sensibly, the definition of the interactions that have to be exchanged through these interfaces is at least as important as the physical or geometrical design itself. In particular, the definition of these system level interactions between different modules is a topic that has not been addressed satisfactorily in previous
literature. Rechtin (1991) states ‘the greatest leverage in systems architecting is at the interfaces’, whereas Jackson (1997) remarks, ‘interfaces are [...] areas which, in traditional engineering, suffer from insufficient treatment’. Sosa et al. (2003) found that a significant number of design interactions across modules were not recognised properly by the development organisation and that this is generally a problem of modular designs.

Design Structure Matrix (DSM) clustering algorithms can help to identify clusters, i.e. potentially good modules, and their interfaces in complex products (Sharman and Yassine 2004, Yu et al. 2007). Unfortunately, several problems can be attributed to the various DSM-based clustering methods available. First, a high degree of arbitrariness exists in the application of DSM-based clustering due to the utilisation of different dependency rating schemes (or simply rating schemes) and the focus on different types of dependencies among system elements. While the distinction between different dependency types in component DSMs increases the accuracy of the product model (Pimmler and Eppinger 1994), no generally applicable method to handle these various dependencies in DSM-based clustering has been presented to date. Furthermore, unawareness of the information type required for module and interface definition as well as data of poor quality used to model systems with DSMs reduce the benefits gained from the use of these algorithms yielding improper module definitions.

Secondly, another issue hardly investigated is the topic of overlapped clusters, buses, and minibuses (Sharman and Yassine 2004). These structures can occur in the clustered DSM, depending on input data, the clustering algorithm used or expert preference. These constructs have been reported to be ‘areas requiring integration effort across chunks’ (Pimmler and Eppinger 1994). However, the intent of these structures in a technical context has not been discussed so far. By looking at possible technical equivalents, we point out that the identification of overlapped clusters, minibuses, and buses is unnecessary.

Lastly, previous clustering strategies focus on maximising intra-module interactions and minimising inter-module interactions without considering any functional constraints. While usually constraining the assignment of elements to modules, the functionality of the product as a whole is often completely neglected. As an example, consider an automotive parking assistant system, which requires a set of distance sensors (for parking assistance) integrated in both the front and rear bumpers, a computer controller (situated in the automobile interior), and the connections (e.g. electric wires) of these sensors to the controller. Performing clustering based on documented signal exchanges exclusively would group all sensors to the controller. Thus, these components would form a signal module that highlights the strongest signal interactions, but cannot be built as a module, since the sensors have to be at the front and rear ends in order to fulfill the task of distance measuring.

Therefore, simply focusing on signal interactions is not sufficient to capture all aspects of the architecture. It is obvious in this example that the sensors can only form modules with the front and rear bumpers. Hence, the sensors have two interactions with other components of the automobile: a signal type interaction with the controller in the vehicle interior and a structural interaction with the front or rear bumpers, respectively. For this simple problem, it was easy to realise that the sensors’ interactions with the bumpers require sensors and bumpers to be spatially adjacent, whereas the interactions between sensors and controller do not require spatial adjacency. Nevertheless, this simple example clearly demonstrates how spatial requirements constrain the assignment of elements to modules: regardless of the intensity of sensor–controller interactions, the assignment to possible modules is dictated by requirements for spatial adjacency, which prohibits the sensors to be in one module with the controller. Thus, weak interactions can outweigh stronger interactions and restrict or determine the possible assignment of elements to modules. These restrictions can be regarded as constraints on the product layout and consequently on the product architecture. Hence, we consider the spatial distribution of elements over the product as
decisive for the definition of assembly modules and their interfaces, determined by adjacency requirements and further driven by the desire of minimizing interactions between modules.

In this article, we present solutions to the above three problems based on a new DSM dependency rating scheme that focuses on the spatial aspect for module identification. A case differentiation between different dependency types, performed in each cell of a DSM, allows handling multiple interaction types. Moreover, the proposed rating scheme and the case differentiation are the basis for a post-processing step to correct and refine the automated clustering result. Throughout the procedure we focus on assembly modules, i.e. modules that can actually be built and therefore are decisive for the definition of interfaces, as opposed to design dependency modules, e.g. signal modules, which result from merely clustering signal design dependencies, but usually constitute infeasible solutions. The result is an improved DSM clustering procedure that takes into account input DSM data consistency, assembly modules, and (in contrast to all DSM clustering algorithms) a structured post-processing step is included. Although the proposed clustering procedure is fully automated, it allows human expert intervention, which is particularly important in the post-processing phase, to correct, refine, and confirm the automated clustering results obtained. This results in a more practical DSM-based clustering that provides system architects the required flexibility to include various ‘soft’ constraints that are not possible to include in earlier DSM clustering algorithms.

The proposed DSM-based clustering approach is composed of five major steps described in Sections 3–6 and each in turn consists of several substeps (Figure 1). The procedure starts with the acquisition of data for the DSM model (Section 3). Then, these data are pre-processed to ensure consistency in the input DSM data prior to clustering (Section 4). An existing clustering algorithm was improved and implemented in order to accommodate the new rating scheme (Section 5). A semi-automated post-processing procedure, used to refine the clustering results obtained by the clustering algorithm, is discussed in Section 6. Finally, in Section 7 we present a case study of a generic jet engine to demonstrate the proposed clustering technique.

![Figure 1. Steps of the proposed clustering procedure.](image_url)
2. Literature review

2.1. Overview of previous dependency rating schemes

Different rating schemes have been used to model products or systems with component DSM models in previous literature, ranging from simple binary ones over weighted rating scales to ones that discriminate between different dependency types. Binary DSM models are rather coarse and do not allow discriminating different dependency strengths or types. Either the user is confined to note only one interaction type (e.g., structural) or to blend several ones (e.g., structural, energy, signal, material). These models can be used in very early phases of product development or, more generally, whenever no detailed information is available. In order to include more information in the DSM model, weighted rating scales can be used. For example, a 0 to 10 rating scale representing dependency strengths from ‘no dependency’ over weak dependencies to strong dependencies. More important than these values is the interplay of the used rating scale with the algorithm used for the clustering. Although these simple weighted rating scales constitute an improvement over binary rating scales, their application often results in very arbitrary and inconsistent assessments of the dependencies in products. For example, with the 0 to 10 rating mentioned earlier, it is very difficult to decide whether a relation is an 8 or a 6, or maybe a 9. Furthermore, it is very difficult to compare different kinds of interactions. How important is a 10 Mbit/s data flow compared with a fluid mass flow of 1 kg/s? As similar issues are very difficult to handle, weighted rating scales result in lots of arbitrariness in the input data and thereby reducing the efficiency of clustering algorithms.

An important step toward more consistency and more accuracy in DSM models came with a rating scheme introduced by Pimmler and Eppinger (1994). The rating scheme is depicted in Figure 2, exhibiting four general types of interactions. This rating scheme has three important advantages compared with the ones previously mentioned: (1) every value in the rating scale is linked to a statement and all statements are clearly distinguishable; (2) the rating scale allows not only for consideration of positive, i.e., desired, interaction but also for negative, i.e., undesired or detrimental, ones; (3) four general types of dependencies are distinguished. Therefore, accuracy and consistency of the DSM model are increased due to the linkage of scale and statements facilitating the decision of the most appropriate value. The distinction between different types of dependencies is important, too. It helps to perform model updates, to review clustering results and to enter more than one interaction between two elements. Furthermore, by including information about possible negative effects, elements can be actively separated in the clustering. Moreover, possible conflicting situations or design challenges are pointed out, especially when −2 and −1 values end up inside clusters. However, negative interactions between elements of different clusters require attention as well, since these may influence or constrain the final product layout. Later on, the rating scheme was extended by Sosa et al. (2000), who added structural as a fifth dependency type.

Although this rating scheme provides good DSM models in general, it has not been widely adopted in recent clustering literature. One problem with the applicability of this rating scheme

---

**Figure 2.** Rating scheme by Pimmler and Eppinger (1994).
is its tailoring to the case study in the original paper of Pimmler and Eppinger (1994). Furthermore, Pimmler and Eppinger suggested extending the list of interactions, e.g. by taking different interaction types into account. While a substantial extension of interaction types is possible for special applications, we do not recommend it in general as it hinders the development of a standard procedure. In addition, more than four or five different types of dependencies typically make the DSM model more difficult to overview while not enhancing the quality of the DSM model with respect to clustering.

2.2. DSM clustering: a multi-objective optimisation task

The introduction of different dependency types by Pimmler and Eppinger (1994) did not only increase accuracy of the DSM model, but also presented new problems. Unlike previous rating schemes featuring only one value per cell in the DSM, there was now a set of values in each cell. Therefore, the clustering problem was no longer one-dimensional with exactly one optimum, but a multi-dimensional one. Multi-objective optimisation means that an optimum has to be found considering all of the selected dependency types (i.e. structural, energy, signal, and material). Unfortunately, it is very unlikely to find one clustering arrangement that is optimal for all of the dependency types. Rather, there is a set of Pareto-optimal solutions constituting the set of optimal trade-offs between the different goals (Kim and de Weck 2005).

The question of how to handle these multiple objectives in the clustering problem has not been thoroughly discussed yet. Pimmler and Eppinger (1994) noted that weighted sum approaches (see description below) muddle the perspectives. Hence, they decided to focus only on the material aspect for clustering their Climate Control System (CCS) case study, arguing that the material aspect is deemed most important for the CCS. While this point of view works relatively well in the climate control example (as most of the elements are involved in material flows), the proposed method is not generic. In fact, it is always possible to look at different types of interactions separately, as Pimmler and Eppinger (1994) proposed in their paper, but the practical use of this possibility is not very high. Even for rather small DSMs, it is extremely difficult to compare different clustering arrangements obtained by individually clustering each type of interaction and then conclude with an overall optimal clustering arrangement.

Realising that the clustering problem is an optimisation task, one is typically tempted to tackle multiple types of interactions with either multi-objective optimisation algorithms or to reduce the problem to a single-objective one. In the latter case, the objective function is separately applied to each objective and these objectives are weighted and aggregated resulting in one single combined measure. On the other hand, real multi-objective optimization algorithms could be used, regarding different objectives (i.e. interaction types) individually and determining the Pareto-optimal set instead of only one single solution on the Pareto-front as obtained with weighted sums.

Both methods have advantages and disadvantages. Single-objective techniques allow ‘choosing’ a certain product perspective by setting weights and using a single-objective clustering algorithm. In contrast, it is all but trivial to find sensible weights to define the contribution of each dependency type’s individual objective value to the overall objective function a priori. Furthermore, the importance assigned to all interactions of one type is identical in combination with Pimmler and Eppinger’s (1994) rating scale. Another problem is that weighted sums distort the clustering result due to double counting of one and the same interaction and similar effects. Therefore, such an approach is unlikely to yield a true optimal clustering result and is one reason why such approaches distort the clustering.

A benefit of multi-objective clustering is the identification of many different perspectives of the product in a single optimisation run, each optimal in its own merit (solution on the Pareto-front), without the need to specify weights. Specifying no weights for dependency types is certainly
beneficial. However, this advantage is attained by the sacrifice of unambiguous clustering results, \textit{i.e.} the user obtains many different results that have no clearly specified perspective. Although these clustering results can potentially provide new insights, it is hardly possible to compare different clustering arrangements of the same product related to their dependency marks. To conclude, multi-objective clustering suffers from similar distortion effects as weighted sums for the purpose of DSM clustering.

2.3. Clustering algorithms

In recent years, several DSM clustering algorithms have been proposed (Baldwin and Clark 1999, Fernandez 1999, Thebeau 2001, Whitfield \textit{et al.} 2002, Yu \textit{et al.} 2003, 2007, Wang and Antonsson 2004, 2005). A full discussion of these algorithms is beyond the scope of this article; however, for good summary discussion, the reader is referred to Sharman and Yassine (2004) and Helmer (2007). During this research, several of these algorithms were investigated with respect to their proposed dependency representation schemes, utilised clustering objective functions, solution search procedures, robustness of the results, and their practical usefulness (Helmer 2007). The relative superiority of a particular algorithm was difficult to show mainly due to the different DSM dependency rating scales used. However, based on insights gained during this research, we were able to develop a new clustering algorithm, which is an improvement over the one proposed by Yu \textit{et al.} (2003, 2007).

The objective function proposed by Yu \textit{et al.} (2003, 2007) implements the Minimal Description Length principle. The basic idea is to develop a model description such that the amount of data describing the model is especially small when the model has a modular architecture. Then, the search for clusters can be pursued by minimising Model Description Length (MDL). In order to derive the MDL, the two contributing parts Optimal Cluster Description (OCD) and Mismatch Data Description (MDD) must be encoded. The OCD describes the DSM model as if its clusters were perfect, \textit{i.e.} completely filled and no dependency marks outside clusters. The OCD is simply defined by a cluster number followed by the numbers of the elements that are inside the cluster.\textsuperscript{3} This can be captured with Equation (1):

\begin{equation}
\sum_{i=1}^{n_c} (\log_2 n_c + s_i \cdot \log_2 n)\text{c}
\end{equation}

where \(s_i\) is the size of cluster \(i\) and \(n_c\) is the maximum number of clusters entered by the user. The first logarithm determines the number of bits required to describe the cluster number, the second one determines the number of bits required to describe all \(s_i\) elements in cluster \(i\).

The second data set, MDD, captures the deviations of the real DSM from this perfect assumption and is computed by simply counting the number of type-I and type-II mismatches and storing them in the variables \(S_1\) and \(S_2\).\textsuperscript{4} Mathematically this can be written as follows:

\begin{equation}
S_1 = \sum_{i,j} DS\textsc{m}(i, j) | d_{ij} = 0, d_{ij}' = 1 \right\}
\end{equation}

\begin{equation}
S_2 = \sum_{i,j} DS\textsc{m}(i, j) | d_{ij} = 1, d_{ij}' = 0 \right\}
\end{equation}

Where, \(DS\textsc{m}(i, j) = d_{ij}\) and \(DS\textsc{m}'(i, j) = d_{ij}'\). In order to describe these mismatches, their position in the DSM as well as their type is noted. This requires the following number of bits.

\begin{equation}
\log_2 n_c + \log_2 n_c + 1 = 2 \log_2 n_c + 1
\end{equation}
The first logarithm describes the row, the second one the column of the cell with the mismatch. The additional bit allows distinguishing between the mismatch types. Hence, the total MDD is described with Equation (5).

\[ S_1 \cdot (2 \log_2 n_c + 1) + S_2 \cdot (2 \log_2 n_c + 1) = (S_1 + S_2) \cdot (2 \log_2 n_c + 1) \] (5)

Yu et al. (2003) furthermore introduced the weights \( \alpha \) and \( \beta \) to tune the clustering algorithm, where \( \alpha > 0, \beta > 0, \) and \( 0 \leq \alpha + \beta \leq 1 \). Thus, the entire objective function for the clustering can be written as follows:

\[ f_{DSM} = (1 - \alpha - \beta) \cdot \left( n_c + \sum_{i=1}^{n_c} s_i \right) \cdot \log_2 n_c + \alpha \cdot S_1 \cdot (2 \log_2 n_c + 1) + \beta \cdot S_2 \cdot (2 \log_2 n_c + 1) \] (6)

In order to consider non-binary DSMs, a normalisation procedure is utilised (so that the DSM values become between 0 and 1), which can be accomplished by applying Equation (7) to every DSM cell:

\[ p_{ij} = \frac{d_{ij} - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}} \] (7)

where \( d_{\text{max}} \) constitutes the maximum value in the DSM dependency rating scale, \( d_{\text{min}} \) the minimum value, and \( d_{ij} \) is the entry in the DSM in row \( i \) and column \( j \). By changing the original definitions of \( S_1 \) and \( S_2 \) (Equations 2 and 3), the gravity of a mismatch can be taken into account as well:

\[ S_1 = \sum_{d_{ij}=1} (1 - p_{ij}) \] (8)

\[ S_2 = \sum_{d_{ij}=0} p_{ij} \] (9)

The deficiencies in Yu et al.’s (2003, 2007) clustering optimisation stem from two reasons: (1) multiple cluster membership is allowed and therefore elements can be in several and non-adjacent clusters5 and (2) a maximum number of clusters must be specified. Especially the former point leads to distortions, since more dependency marks can be shown inside clusters than actually being possible from a technical point of view. This again can cause further elements to join multiplied elements in such clusters. In the most extreme case, an entire cluster would be exclusively created by duplicated elements despite the fact that only one element is reasonable or possible in the technical realisation. Furthermore, it makes review more difficult and especially conceals interactions between clusters which appear to be inside clusters. Additionally, the detection of buses and overlapping clusters is arbitrary as further explained in Section 6.

3. Data acquisition: a new dependency rating scheme

The type of data used for the DSM model and its quality are decisive for the quality of module definition. When setting up DSM models, issues related to the type of information being included or to the level of detail of this information may arise. Component DSM models can be set up with lots of types of information, as well as binary and weighted, symmetrical and asymmetrical data. Furthermore, indirect dependencies can be noted in addition to direct ones.6 Yet not all possible
ways to model a product are appropriate for the correct definition of modules. In this section, a new rating scale is introduced, capturing information required for the definition of modules based on the explanations given in Section 1. It is worth noting that an asymmetrical DSM model is assumed in the remainder of this article since the information content in these models is higher than in symmetrical ones. For instance, a distinction of uni- and bi-directional interactions is possible and different values below and above the diagonal can be entered.

As pointed out earlier, we consider the spatial aspect to be the driving factor for the definition of ‘assembly modules’, which in turn are decisive for the identification and definition of module interfaces. To account for the spatial distribution of elements within the product, the rating scale by Pimmler and Eppinger (1994) was modified as shown in Figure 3. The modified scale captures not only the necessity of a flow or possible negative influences, but also allows for a distinction between ‘interaction and spatial adjacency required’ (+2) and ‘interaction, but not spatial adjacency required’ (+1) on the positive end of the rating scale. That is, unlike with the original rating scale of Pimmler and Eppinger, where every required interaction results in the same value (+2), the new rating scale allows distinguishing between interactions requiring spatial adjacency and required interactions without need for spatial adjacency.

In Figure 3, the inner bright parts (−1 through +1) correspond to Pimmler and Eppinger’s rating, the values ±2 constitute the extension. Furthermore, the statements linked to the values in the rating scale have been modified, whereas the values and limits themselves are arbitrary. A further extension to Pimmler and Eppinger’s rating scale by ±3 would have been an option. However, the ±1 of the original rating scale referred to interactions that might occur and are either beneficial or might cause negative effects (Figure 2). By compressing the values, these former ±1 constitute now ±0.5, with the ‘0’ intending to point out that these interactions might, but do not have to occur. Moreover, the same types of interactions introduced by Pimmler and Eppinger (1994) and extended by Sosa et al. (2003) are used, i.e. structural, energy, signal, material, and spatial (Figure 4).

The meaning of the small black arrows pointing toward the spatial entry becomes clearer in Section 4.1. For now, it suffices to understand that not only structural interactions and requirements such as accessibility, etc., as noted by Pimmler and Eppinger (1994) and Sosa et al. (2003) can require spatial adjacency of two elements, but also energy, signal, and material exchanges. For instance, when a material flow is required to flow through structural parts like between different segments or cases in a gas turbine, these elements need to be adjacent to confine the flow and form a continuous conduit. The new rating scale captures this fact due to the distinction between +2 and +1. When looking at the meaning of the entries in the new rating scale, we note that any value is regarded not only with respect to required or undesired interactions, but also with respect to implications on the spatial aspect resulting from the interactions. In other words, the rating scale, in combination with its interplay with a clustering algorithm, is intended to find the optimal spatial distribution of elements over the product to identify assembly modules.

![Figure 3. New rating scheme.](image-url)
Since +2 values represent the adjacency of two elements, thus constraining the product layout, it is obvious that these elements are potential candidates to be members of the same module. Consequently, these +2 values constitute constraints on the product layout, although they should not be used as constraints in the clustering algorithm forcing them into the same module, because +2 marks can be outside cluster bounds, too. Simple examples for +2 being outside cluster bounds are structural connections between two modules.

The distinction between −2 and −1 is less important, as both values indicate that an interaction cannot be coped with without taking special measures to mitigate the influence if the elements are close or adjacent. For example, cooling techniques are required to cope with detrimental heat exchanges in gas turbines. Thus, this distinction mainly permits the possibility to rate the gravity of the negative influence. The choice can influence the final clustering result in three ways: (1) by the higher or lower value itself, (2) through the Perspective Reduction, described in the next section, the choice can influence the overall dependency value of different dependency types, and (3) there is the possibility to give ±2 values a higher weight in the clustering optimisation. These influences become clearer in subsequent sections of the article. It is important to note that except for −1 and −2 no distinction between interaction strengths exists. This could be deemed a deficiency in the proposed approach, but is not easy to overcome. Nevertheless, the proposed method is consistent and deficiencies are reduced due to the clustering post-processing (Section 6). Furthermore, this deficiency becomes more important for very dense DSMs. Fortunately, most product architectures are sparse enough on the system level, so that this deficiency does not significantly influence the clustering result.

4. Data pre-processing

During data pre-processing the DSM model is prepared for the subsequent clustering and is composed of two main steps: Perspective Reduction (PR) and Data Reconciliation (DR). While the former constitutes a new and easy way to handle multiple objectives in component DSM clustering resulting from the distinction between different types of dependencies, the latter one further ensures consistency and prevents wrong bias.

4.1. Perspective reduction

The concentration on the spatial aspect with the modified rating scale is the starting point of a new way to handle multiple types of interactions in DSM clustering. This approach, that we termed PR, is a case differentiation between the different types of interactions performed in every
single cell of the DSM in order to collapse the four entries per cell to a single overall dependency mark. This overall dependency mark, indicated by the black arrows in Figure 4, describes the overall spatial requirement or desire due to all types of interactions and is the value used for the subsequent clustering.

This reduction is only possible due to the new rating scale that allows a comparison of the ‘importance’ of different interaction types. As an example, imagine a negative energy exchange between two elements if the elements are close (−1) and a required material exchange between the same elements requiring spatial adjacency (+2). It is obvious that the requirement for spatial adjacency prevails in order to achieve proper functionality, while the negative interaction has to be accepted in this product concept and constitutes a design challenge. The result of this trade-off would be a +2 spatial requirement, so that the dependency between the two elements in consideration would be as strong as any other +2 dependency without a negative mark due to another type of interaction. It is clear that the clustering result is different from the one obtained, if the clustering would have been exclusively performed based on the energy dependency. In this case, elements that actually have to be close would probably end up in different clusters due to the negative dependency mark and the fact that further aspects are ignored.

In the following paragraphs, general rules to be applied for the PR are described. These rules can be implemented as general rules in software, so that the PR can be performed automatically although conflicting situations might occur. It is important to be acquainted with the trade-offs discussed below and to keep these in mind to apply the rating scale properly. In the case of a single interaction type (e.g. Figure 5), no trade-off is required and the entry is directly taken as overall value.

In all other remaining cases a trade-off is required. Generally, all marks with higher absolute value prevail against other marks of the same algebraic sign as these act supportive. For instance, if one interaction has a +0.5 spatial need and another one in the same cell has a +1 spatial need, the +1 prevails. Likewise, a +2 (i.e. the corresponding interaction requires spatial adjacency of the corresponding elements) prevails against +1 and +0.5 entries, which express another dependency’s desire for adjacency or proximity, but do not require it. This rule is also applicable to negative marks. A −2 due to a certain interaction is more important than a −1 due to another aspect and both are more important than a weak −0.5 mark. Furthermore, the rule is always applicable, independently of the number and type of interactions with a certain mark as shown in Figure 6.

![Figure 5. PR function 1.](image1.png)

![Figure 6. PR function 2.](image2.png)
However, more critical to handle are trade-offs between marks of different algebraic signs. If any of the aspects require spatial adjacency (\(+2\)), this interaction prevails always, \textit{i.e.} a \(+2\) entry is the strongest entry possible, because it requires two elements to be adjacent to fulfill a certain function. Possible negative interactions caused by this adjacency, no matter how many, are overruled. Even \(-2\) marks which normally state that the elements have to be spatially separated can be overruled by a \(+2\) spatial adjacency requirement and constitute extreme design challenges to allow functionality of the product. As an example, consider a jet engine’s High Pressure Turbine (HPT). One stator row is required at the exit of the combustion chamber to divert the fluid flow in circumferential direction. The stator vanes have to be mounted in this position, so that a \(+2\) structural entry is required. There is also a required fluid \textit{i.e.} material flow from the combustion chamber to the stator vanes, which requires adjacency \((+2\) material). Unfortunately, the vanes are subjected to extremely high heat loads, both due to heat radiation from the combustion itself (consequence: \(-2\) energy mark; the interaction is so detrimental that spatial separation is required, otherwise the vanes would melt) and the hot material flow itself. This extreme example is ideal to show that the \(+2\) marks overrule the \(-2\) mark for the clustering and, despite the \(-2\) mark, it is obviously possible to solve the problem by applying elaborate and expensive measures (in this case, cooling techniques and choice of material). So \(+2\) marks always prevail, no matter how many negative marks are in the same cell of the DSM.

Since a \(+2\) mark can prevail against a \(-2\) mark, as just explained, it consequently also prevails against \(-1\) and \(-0.5\) as well as \(+0.5\) and \(+1\) marks. Conversely, a \(-2\) mark prevails against all other marks, except for the \(+2\) mark. The reason is obvious: \(-2\) marks are used in cases where very detrimental interactions occur if two elements are adjacent or even if in the same module. If possible, this interaction must be prevented, otherwise it must be mitigated. Other negative or positive marks (except \(+2\)) merely support the spatial separation or only express a desire for spatial proximity, but no requirements, so that they are of minor importance.

All cases without \(\pm 2\) marks and which also do not fall into the simple cases of equal algebraic signs, discussed earlier, are more difficult to handle and general rules are difficult. Imagine a \(+1\) material and a \(-1\) energy mark. The \(+1\) mark represents a required material flow without the need for adjacency, \textit{i.e.} often via a hose. The negative influence might be rather strong, without absolutely requiring spatial separation. One can assign the general indifferent mark 0 for the overall spatial aspect, but there may be cases where a \(+1\) or \(-1\) is more appropriate. This decision cannot be made automatically, because the DSM model with the proposed rating scale is too abstract, \textit{i.e.} does not comprise enough information to decide over such cases. Similar statements can be given with \(\pm 0.5\) marks. The trade-off becomes even more complicated in the case of three or even all four interactions at the same time.

What is the best overall value for constellations as shown in Figure 7a? The answer to this is all but trivial. Therefore, in all cases where such conflicts with equally strong and various numbers of interactions occur, a default ‘0’ is given (see Figure 7). While theoretically possible, a completely automated procedure can never provide the optimal answer in all cases. Hence, a manual trade-off is highly recommended in conflicting cases. Fortunately, in most products the number of such difficult cases is expected to be rather low, which is supported by the case studies we performed. Sometimes, it may also help to increase the level of resolution for the element under consideration, so that the various interactions are assigned to an element’s sub-elements and the conflicting situation does not occur any more.

To summarise, the following ranking of dependency marks shown in Figure 8 can be given. Each value at a higher level prevails against any number of values at a lower level within one cell. For instance, one or several \(+2\) entries of any interaction type in one cell overrule any number of \(\pm 1\) entries in the same cell, so that the resulting overall value is a \(+2\). Conflicting situations are solved with a default indifferent ‘0’ mark (Figure 7b–e). However, manual decisions can be made and are often required in conflicting situations to take knowledge into account that is not
captured by the abstract DSM model. Furthermore, the possibility to manually overrule values with other values exists due to the unique nature of information possessed only by the designer, but that is not included in the DSM model with the basic four types of interactions. For instance, it can be sensible to overrule other values, due to requirements for accessibility, as captured by the spatial aspect as used by Pimmler and Eppinger (1994) and Sosa et al. (2000). However, often such requirements fall into the category of structural aspects as an element has to be mounted in a certain place to attain accessibility, for instance.

4.2. Data reconciliation

Most component DSMs models are built based on a symmetrical assumption since symmetrical DSMs are easier to set up because this requires filling out half of the DSM; thus half the effort. However, this comes at the expense of model accuracy. Asymmetrical DSM models are generally more accurate and contain more information than symmetrical DSM models simply because both ‘directions’ of interactions are checked independently. Therefore, we suggest using asymmetrical DSMs to model the product. However, this may result in different values below and above the diagonal for the same interacting components, which would require reconciliation before clustering begins. This section describes how the overall dependency values created during PR are reconciled; that is, the asymmetrical DSM, obtained through data acquisition, is made symmetrical.

The higher accuracy of asymmetrical DSMs is the result of extra work effort required to fill out the asymmetrical DSM: every type of interaction has to be checked twice, once for every direction. This is not unnecessary extra work, but helps to set up a correct DSM model due to the independent consideration of each direction. As a simple example, imagine a required energy (e.g. electrical energy) flow from an element A to an element B represented by a +1 mark, and another −1 energy flow going from B to A (e.g. heat energy) if both elements are adjacent. With a symmetrical DSM, it is very likely that one of the energy types will be concealed so that either a required interface is not identified or a conflicting situation cannot be pointed out. Also, updates of asymmetrical DSM models and revisions are easier. Nevertheless, in this step, the overall
dependency values shown in the middle of each cell (Figure 4) above and below the diagonal are reconciled.

The procedure to change the asymmetrical and reduced DSM to a symmetrical one for the purpose of clustering uses the same rules as presented in Section 4.1. The only difference is that instead of comparing different entries in one cell, the trade-off is performed with each pair of cells above and below the diagonal. This is depicted in Figure 9 for only some of the DSM cells.

Reading from the upper left corner toward the bottom right corner, the first case shows that the +2 overrules the overall −2 again. In this case, the clustering algorithm will not distinguish between these two entries treating both equally as two symmetrical +2 entries. In the second case, the −2 overrules the 0, so that the algorithm will handle it as two −2 entries, and so on. This type of reconciliation is accomplished for all pairs, although it is demonstrated only for four of the cells in Figure 9. Further discussion and examples of this procedure can be found in Helmer (2007).

5. Clustering algorithm

We overcome the deficiencies noted with Yu’s and other clustering algorithms using a slightly different Genetic Algorithm (GA) in combination with the MDL-based objective function described earlier. The new algorithm constitutes a simple GA design with a different encoding of chromosomes, so that the multiplication and insertion of the same element at different locations is prevented. This is achieved by the use of Random Keys (RK) (Bean and Norman 1993). Here, every chromosome is not represented by the element order itself, but by RK as shown in Figure 10. Thereby, the position of an element is determined by the value of its RK.
Then, we apply common crossover and mutation operators not to the elements, but to their RKs. Thus, the elements change their position in the chromosome while the number of elements remains constant and each element appears only once in every single chromosome. Thus, RKs constitute a very simple solution to the duplication problem of combinatorial problems.

Due to the fact that we do not use overlapping and bus detection for the identification of modules, the dependency of the clustering algorithm on the level of DSM resolution is reduced. The post-processing methods presented in Section 6 further reduce the influence of the level of resolution chosen for the model. In order to use the objective function with the new rating scale, the calculation of $S_2$ has to be slightly modified as well, so that negative entries and zeros outside of cluster bounds are not considered type-II mismatches:

$$S_2 = \sum_{d_i = 0, d_j > 0} p_{ij} \quad (10)$$

By calculating $S_2$ in this way, we prevent the creation of large and empty clusters. Furthermore, no parameters like maximum cluster size, maximum number of clusters, or minimal number of clusters have to be specified in our GA strategy. In order to ensure the selection of potentially good individuals independent of absolute fitness values as well as to achieve an appropriate selection pressure, Tournament Selection without Replacement with a tournament size of four was selected. Different replacement operators were tested as well. The best results with respect to best fitness values and robustness were obtained replacing the entire parent population by their offspring and using single-point crossover. Finally, the ability to constrain the algorithm to place elements in certain clusters was implemented.

6. Post-processing

Post-processing is the correction and improvement of the results obtained by application of the clustering algorithm with respect to technical feasibility and implementation. This is a mandatory task as clustering algorithms cannot be tuned to provide optimal results in all cases from a technical point of view. To date, little effort has been made to provide assistance during this phase. In this section, we introduce a post-processing step that follows automated clustering. The method mainly focuses on the investigation of $+2$ interaction marks outside of cluster bounds, as these marks indicate that an element may have been assigned to a wrong cluster, or may have to be cloned or split.

6.1. Elements with bus character

In this article, the term bus is used not only for classical buses (i.e. elements that have interactions with a majority of elements in the DSM) but also for all elements that possess a bus character (i.e. have interactions with multiple clusters) independent of the number of interactions. This includes the notions of minibuses and overlapped cluster elements. Knowledge of these elements (e.g. distinction of different bus types) help in deciding the appropriate post-processing action required to improve the clustering result (i.e. product architecture).

In this section, we demonstrate that buses in the DSM can be single technical elements, but also coupled systems or even decoupled entities. In the figures shown in this and the subsequent sections, buses are usually depicted as elements featuring many interactions with other elements or overlapped cluster elements to demonstrate the procedure for didactical reasons, but it is important to note that the post-processing procedure does not require the detection of overlapping clusters or buses by the automated clustering algorithm.
6.1.1. +2 Buses

This bus type has only +2 interactions with other elements in the DSM. Recalling the intention of the +2 interaction as a constraint for spatial adjacency, this element must be adjacent to every other element exhibiting a +2 interaction with it. Several different cases can now be distinguished. Element 6 in the DSM of Figure 11a and Figure 13a as well as element 13 in Figure 12a correspond to the dark element(s) in Figures 11b–13b. The light elements in the figures represent other elements shown in the DSM.\textsuperscript{11} It can be seen that a single element in the DSM can be one or several real elements and that these can be coupled or decoupled, depending on the dissection of the product chosen for the model. The buses shown in Figure 11a and Figure 13a seem identical when looking at the DSM. For the system shown in Figure 12, a larger number of light elements have been chosen in order to better show the various interactions the bus (\textit{i.e.} element 13) has. Thus, the DSM model at a certain level of resolution can conceal information which becomes important for the post-processing procedure.

One possibility would be the bus element representing a large component, such as a platform that many other components are mounted to, or a housing where other components are inside and/or attached to the outside. Thus the element in the DSM represents exactly one continuous physical element (as in the case of Figure 11).

Another possibility is shown in Figure 12. Here, the dark element in the DSM does not represent a single physical element, but a distributed system, depicted as dark components in Figure 12b. It can be seen that these sub-elements are \textit{coupled} among each other, both structurally and with conduits such as hoses and wires. This coupling, \textit{i.e.} the internal connections between the sub-elements of this system, cannot be seen in the DSM in Figure 12a as they represent internal relations of element 13 (and only external dependencies are shown in DSMs). An example of such a system could be any type of control system with one or several controllers, sensors, and actuators.

![Figure 11. Continuous +2 bus. (a) DSM model of +2 bus and (b) schematic representation.](image-url)
Figure 12. Coupled +2 bus. (a) DSM model of +2 bus and (b) schematic representation.

Figure 13. Decoupled +2 bus. (a) DSM model of +2 bus and (b) schematic representation.
A third possibility of a +2 bus is depicted in Figure 13, which shows that the single element in the DSM represents several decoupled elements. These elements in different locations of the product can have lots of +2 interactions with many elements in the product. An example could be an entry ‘bearings’ or ‘sensors’ in the DSM. The sub-elements of such an element are not coupled among each other, but can be spatially adjacent to many elements of the product.

6.1.2. +1 Buses

This type of bus has interactions that do not require spatial adjacency with other elements. Such interactions can take place via conduits (cables, wires, hoses, pipes, tubes, etc.) or through space, such as infrared connections, radiation (heat, laser/light beam, theoretically even radioactive, etc.). Latter types of interactions might constitute constraints on the final product layout as a straight visibility line between the involved elements is usually necessary. Different cases are depicted in Figures 14.

Again, we note that the dark entities in Figure 14 are usually represented by a single bus element in the DSM, which can be a single continuous component, subassembly, or module (Figure 14a), a coupled system (Figure 14b), or several decoupled entities (Figure 14c). Since no requirement for spatial adjacency exists, this element is generally ‘free’, i.e. it is the designer’s decision where to put it (e.g. such that the number of interactions between modules is minimised). Most of the bus types in reality are mixed forms of the extremes presented earlier, i.e. contain a number of +2 and/or +1 dependencies, in addition to zeros and possibly some negative marks.

6.2. Correction and improvement of clustering results

The previous section has introduced different types of elements in the DSM that all have bus character. It is important to analyse these entities as they constitute integral patterns in the product. However, not only elements with several interactions with multiple clusters have to be investigated, but also elements featuring only a single +2 interaction to an element of another cluster (i.e. the +2 appears outside cluster bounds). This is due to the fact that +2 marks outside cluster bounds are indicators for possible wrong assignments of elements to modules, since +2 interactions point out that spatial adjacency of two elements is required. Overlapped clusters, if allowed, as well as minibuses and buses may have to be resolved in order to reveal true cluster memberships and interactions between clusters.

Hölttä-Otto (2005) tackles the problem of identifying module boundaries from a design for change perspective and notes several ways to deal with overlapped clusters: (a) merging of overlapped clusters, (b) assignment of elements to one of the clusters, and (c) duplication of elements and assignment of one twin-element to each of the clusters. A question arising from Hölttä-Otto’s approach is why the possible ways noted above are only suggested for overlapped clusters, but not for minibuses and buses, since overlapped cluster elements are only special cases of minibuses,
which are in turn special cases of buses, as mentioned earlier. The only difference between elements in overlapping regions and minibuses is that the former ones have most interactions with two clusters, instead of three or more like minibuses.

We argue that the treatment of all situations involving a bus character should be similar and suggest a common method for dealing with them. The following list shows several possibilities to deal with all elements that have a bus character (Figure 15):

1. Merging of clusters (also non-overlapped ones)
2. Assignment of elements to one of the clusters that they have interactions with or any other cluster
3. Assignment to none of the clusters → placement of an element in its own cluster
4. Cloning of elements (comprises duplication)
5. Splitting of elements

None of the above options can be claimed to perform better than any of the others. In some cases, two or more options might be sensible possibilities and need to be evaluated, in other cases there is only one. The right choice depends always on the type of product, the type of element, knowledge, and expertise, as well as available technologies and resources. Every element with a bus character should therefore be classified in the types presented earlier, especially in continuous, coupled, and decoupled.

6.2.1. Merging

Two or more clusters are merged forming one big cluster (Figure 15a). Theoretically, this can be accomplished with any two or more clusters that are next to each other in the product layout (not necessarily in the DSM!), so that the modules on the top level become nested modules. Practically, it means that two or more individually identified clusters are considered one module. Among others, this can be necessary, because one module can be identified as two clusters due to certain dependency mark constellations.

![Figure 15](image-url)
6.2.2. Assignment of elements to one of the clusters or another cluster

This assignment can often be done rather easily when looking at the spatial +2 dependencies representing requirements for spatial adjacency (Figure 15b). Thus, the element is assigned to the cluster it has +2 dependencies with. However, this cannot be done automatically, since +2 interactions may also occur between modules. Sometimes, it is also possible to put an element in another module while still complying with the spatial adjacency requirement which constrains the product layout. As the abstract DSM model does not capture information on element sizes, shapes, etc., these decisions have to be made manually.

6.2.3. Assignment to none of the clusters

Sometimes, depending on the type of element, it might be sensible to draw the element outside existing clusters so that the element can be considered a separate module.

6.2.4. Cloning of elements

Cloning is the multiplication of an element in the DSM. In Figure 15c, we observe that two identical elements 3a and 3b are created and assigned to clusters. Possible cases for cloning might be elements like ‘bearings’, ‘sensors’, ‘actuators’, or ‘controllers’. Cloning is not limited to decoupled buses, minibuses, or overlapped cluster elements, but can also be a possible option for entire systems, meaning that several modules are assigned their own and independent system (e.g. lubrication system, thermal management system, etc.). The identical elements are usually not connected among each other. Once the elements are assigned to the clusters, their interactions do not appear outside the cluster boundaries any more. Hence, clusters, which appear to be coupled before, can be decoupled in reality.

6.2.5. Splitting of elements

Splitting is intended to split distributed elements into several parts. Unlike cloning, not several identical elements are created but one element is partitioned in subsystems or segments. Figure 15d shows that element 3 is split in two subelements 31 and 32. Contrary to cloning, the created sub-elements remain coupled. This coupling is denoted as ‘x’ in the figure, indicating that the interface types are not defined yet. In case that the element represents a large structure or housing (e.g. Figure 11), this technique results in the segmentation of this element, which can be sensible for assembly and maintenance, but usually harmful for weight and part numbers. Thus the consideration of this option can reveal important design possibilities or necessities.

If the element represents a system (e.g. Figure 12b), the system is divided into subsystems which are interconnected, i.e. coupled. This division of systems and the correct assignment of their elements to the subsystems is a rather difficult procedure compared with the other ones, which will be described in detail next. Splitting the system into subsystems has the advantage that it allows parallel assembly, eases disassembly for maintenance or recycling, and helps to assign the systems’ elements in different modules. Furthermore, splitting is also required to determine the interactions between the modules involved as well as the bill of materials of each module.12 Splitting consists of four major steps as follows.

6.2.5.1. Step 1. The first step of this procedure is to split the original element (or system) in several sub-elements (or subsystems) (31 and 32 in Figure 15d). These sub-elements (or subsystems) are assigned to clusters which the original element has +2 interactions with, because this
usually means that components of the system (i.e. the original element) have to be in this cluster. In the first step, also all interactions with the corresponding cluster are drawn inside the cluster with the relevant element (Figure 15d). Finally, ‘x’s are noted between the sub-elements to denote any possible kind of interactions (including possible 0 interactions). These are to be defined next. Furthermore, it is not known so far which components of the system are in which subsystem.

6.2.5.2. Step 2. In Step 2, the level of resolution for the original element is increased, so that its sub-elements become visible. If most of the sub-elements that have +1, +0.5, or any negative interaction marks with elements of the original DSM, which is called 1st Level DSM, are subjected to spatial constraints due to +2 interactions with elements of the 1st Level DSM or there are not many interactions other than +2, the outer system interactions can be neglected and the entire system can be evaluated using a separate DSM, which is called 2nd Level DSM. +2 interactions to elements outside this system are taken into account as constraints, so that the sub-elements will be forced in clusters which they have +2 interactions with. These constraints are required since the system under consideration is embedded in the product.

6.2.5.3. Step 3. Some elements will typically be subjected to constraints forcing them in certain 1st level modules, whereas others will remain ‘free’, i.e. are not subjected to constraints. In order to optimise the 2nd Level DSM (i.e. the split system) in terms of modularity, the 2nd Level DSM can be clustered considering the constraints, so that only ‘free’ elements can change their cluster membership. The clustering might in turn have to be corrected, of course.

It can be seen in Figure 16 how the level of resolution for the original element (now split in 31 and 32) is increased, so that its sub-elements A through J are revealed. These are noted in the separate 2nd Level DSM on the right. Constraints on the sub-elements are shown by the dashed boxes in the 2nd Level DSM. The upper box denotes the affiliation of sub-elements to subsystem 31, the second box represents the affiliation with subsystem 32. In other words, all sub-elements of 31 subjected to constraints are in the first cluster, all sub-elements of 32 that are subjected to constraints are in cluster 2. Furthermore, some sub-elements of both or either elements 31 and 32 remain ‘free’, i.e. are not subjected to spatial constraints with 1st level elements, highlighted by the ellipse in Figure 16b.

Figure 16b shows that elements A through D of the 2nd level DSM are constrained to be in one cluster and elements E through G have to be together in another cluster. Figure 17a shows the result of the unconstrained clustering of the 2nd Level DSM, resulting in a violation of the constraints. That means this solution is infeasible. Its clusters are denser than the ones obtained

![Figure 16. The 2nd level DSM constrained and unconstrained elements. (a) 1st level DSM and (b) 2nd level DSM.](image-url)
with the consideration of constraints (Figure 17b), which reflect the technical reality. Figure 17b shows that, in this example, the sub-elements of subsystems 3₁ and 3₂, created in the first step, are identified and so are the interfaces between the subsystems and thus between the different modules (see ‘x’ in Figure 15d and Figure 16).

6.2.5.4. Step 4. In order to complete the definition of 1st level modules and interfaces, the elements of the 2nd Level DSM have to be transferred to the corresponding 1st level clusters, i.e. the level of resolution of the 1st Level DSM is increased. This is required in order to correct the dependency assignment performed in Step 1. Wrong assumptions about interaction marks are rather easily identified once the level of resolution in the 1st Level DSM is increased (Helmer 2007).

7. Case study

In this section, the clustering procedure of a generic jet engine is demonstrated. For brevity, the entire process cannot be shown. The generic jet engine is based on the layout of PW4000 series engines with generic externals. The data set for this case study was acquired by both knowledge of the authors and discussions with faculty, as well as numerous different sources (Rowles 1999, Cumpsty 2000, Bräunling 2001, Steinetz and Lattime 2002, Steffens et al. 2003, Steffens 2004, Boyce 2006, Kau 2006, Rolls-Royce 2007). The jet engine is decomposed into 49 elements on the first level. The external systems are modelled at a higher level of resolution in a separate 2nd Level DSM with 31 elements. Figure 18 shows the jet engine DSM at a random order of elements, which is a typical starting point for the clustering algorithm. This figure shows the DSM after the application of PR and DR. It is worth noting that there were no cases where trade-offs had to be performed manually due to conflicting situations that would have resulted in an overall 0.

7.1. DSM clustering

With PR and DR performed, the clustering algorithm can be applied to the data set. Recall that the algorithm only ‘sees’ the overall values shown in the middle of each cell. A very important factor for optimisation using GAs is the choice of the population size. Generally, it is recommended to
work with larger populations, because this ensures that the population is spread over the entire solution space, and computational time for clustering is not decisive. Since the algorithm used is a modified version of the one proposed by Yu et al. (2007), who chose a population size of 8500 for clustering the jet engine by Rowles (1999), which is of similar size and complexity (density), a value of similar order was used in this case study. As stated earlier, detection of overlapping clusters is not required, so that overlapping cluster detection was switched off. Bus detection is not sensible as well, as pointed out earlier. Therefore, the focus is on +2 interactions outside cluster bounds and elements with bus character.

In Figure 19, showing the result provided by the clustering algorithm, it can be seen that there are several elements that can be considered ‘distributed’, i.e. have +2 interactions with elements of several clusters, such as the Intermediate Case and the Fan Containment Case. Hence they have bus character. It can further be seen that many good (i.e. dense) clusters could be identified. The density could be increased or decreased by adjusting the $\alpha$ and $\beta$ parameters of the objective
function. A thorough discussion is beyond the scope of this article. The interested reader is referred to Helmer 2007. As expected, several very good clusters identified by the algorithm cannot be considered modules in a technical sense. This again shows the need for post-processing procedures to correct the assignments of elements to modules and obtain the true interfaces.

7.2. Post-processing

In Figure 19, the initially identified clusters have been given names to ease the review process. The most important corrections performed during post-processing are shown by the arrows. For example, the Inner Combustor Cluster (ICC) and the Combustor Cluster (CC) are merged. The assignment of the elements of the ICC to the CC simply was not performed automatically, since this would cause several 0 entries inside the combined cluster and this is penalised by the objective function used, which tries to find rather dense clusters. Recall that special attention is required.
whenever +2 entries are outside cluster bounds. But, of course, also +1 interactions have to be investigated, since no distinction between the importance of +1 interactions is possible with the rating scale proposed. Some of the clusters identified are not only good clusters, but also good modules from a technical viewpoint, such as the High Pressure Compressor (HPC) cluster and the HPT cluster, which both contain all basic elements that can be found in the corresponding modules of a real jet engine. Some further ones may not directly coincide with the technical modules found in a real jet engine, but the differences are neither grave nor critical in terms of interface identification. For instance, the Turbine Exhaust Case (TEC, element 31) in the Low Pressure Turbine (LPT) cluster; this is a correct assignment when looking at the dependencies the TEC has. However, due to its size and the mounting at the rear end of the LPT case, the TEC is usually considered an extra module, so that it can be drawn outside the module bounds. However, this assignment is not critical, since the interface design is in both cases exactly the same, although it appears one time inside the module and another time between modules. This shows that formal definitions do not always have implications on the technical design.

The Bleed Valves are an example to show that the clustering algorithm did not assign an element to a cluster, although it has only one single +2 interaction (with the HPC cluster). This is due to the fact that the assignment would cause many zeros inside the cluster, i.e. type-I mismatches of medium gravity, so that the algorithm does not perform the assignment automatically with the chosen optimisation settings. With different settings, the element could be

<table>
<thead>
<tr>
<th>Element</th>
<th>Spatial constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>EEC</td>
<td>–</td>
</tr>
<tr>
<td>Inlet air P/T probe</td>
<td>At fan containment case</td>
</tr>
<tr>
<td>Starter air valve</td>
<td>At fan containment case</td>
</tr>
<tr>
<td>Oil tank with oil level sensor</td>
<td>At fan containment case</td>
</tr>
<tr>
<td>Oil pumps</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>Oil filter (including pressure drop sensor)</td>
<td>–</td>
</tr>
<tr>
<td>ACOC with temperature sensor</td>
<td>At fan containment case</td>
</tr>
<tr>
<td>FCOC with temperature and pressure sensor</td>
<td>–</td>
</tr>
<tr>
<td>Chip detectors</td>
<td>–</td>
</tr>
<tr>
<td>Scavange pumps</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>Centrifugal breather</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>Alternators</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>LP fuel pump</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>LP fuel filter</td>
<td>–</td>
</tr>
<tr>
<td>HP fuel pump</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>Fuel metering unit</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>HP shut-off valve</td>
<td>–</td>
</tr>
<tr>
<td>Fuel flow meter</td>
<td>–</td>
</tr>
<tr>
<td>Fuel manifold</td>
<td>At combustor case</td>
</tr>
<tr>
<td>Drainage tank</td>
<td>–</td>
</tr>
<tr>
<td>Ejector pump</td>
<td>At outer gearbox</td>
</tr>
<tr>
<td>Filter blockage indicator</td>
<td>–</td>
</tr>
<tr>
<td>VSV servo control</td>
<td>–</td>
</tr>
<tr>
<td>VSV actuators</td>
<td>At HPC case (at VSV)</td>
</tr>
<tr>
<td>Ignition exciter</td>
<td>–</td>
</tr>
<tr>
<td>Igniter</td>
<td>Combustor case and flame tube</td>
</tr>
<tr>
<td>HP bleed control unit</td>
<td>–</td>
</tr>
<tr>
<td>HP air bleed valves</td>
<td>At HPC case</td>
</tr>
<tr>
<td>Sensors</td>
<td>In different places. Cannot be constrained to be in a single cluster. Either include several sensors and use one constraint for each of them, or correct the assignment manually once the 2nd Level DSM is included in the 1st Level DSM</td>
</tr>
<tr>
<td>Fan air valve</td>
<td>At fan containment case</td>
</tr>
<tr>
<td>Surge bleed butterfly</td>
<td>At splitter</td>
</tr>
</tbody>
</table>
drawn inside, but this would result in other clusters being suboptimal. Also, forcing it inside the cluster due to its single +2 entry would be beneficial in this case, but not in general, as there are cases where elements that have a single +2 mark with one cluster have to stay outside, since they constitute extra modules, as described for the TEC, above. Yet the identification of the non-assignment of the Bleed Valves to the HPC cluster is very simple when investigating the clustered DSM and +2 marks outside clusters, so that the assignment can be easily performed manually.

Finally, splitting as discussed in Section 6.2.5 is performed in order to reveal interfaces between modules by assigning the systems’ elements to the 1st level clusters, i.e. dividing the systems into subsystems and also pointing out possible common interfaces. Splitting of systems is usually required with distributed systems. For brevity, not all steps can be shown and discussed.

Figure 20. Result of the *unconstrained* clustering of the external systems DSM.
in detail; rather, we show the implications of using constraints during clustering. Due to the interactions between external systems,\textsuperscript{15} comprising fuel system, oil system, starter system, and ignition system, as well as sensors, actuators, and Electronic Engine Control (EEC), we chose to incorporate all of these into one 2nd Level DSM. This 2nd Level DSM has 31 elements. Since the systems modeled with the 2nd Level DSM are part of the entire product, many of the elements are subjected to spatial requirements due to interactions with 1st level elements. These constraints are listed in Table 1 and have to be considered during clustering. For instance, oil and fuel pumps have to be at the Outer Gearbox, since they are driven by it. A more detailed description of the entire generic jet engine can be found in Helmer 2007.

Figure 20 shows the result obtained using the clustering algorithm presented in Section 5, but without consideration of the constraints. Figure 21, on the other hand, shows the clustering
8. Summary and conclusion

A comprehensive method for the definition and identification of modular product architectures was presented utilising component DSMs. Important topics that have been neglected in previous clustering literature were addressed, such as the question of symmetrical vs. asymmetrical DSM models and how to handle multiple aspects in DSM clustering, leading to a better understanding of DSM clustering and the development of an improved clustering technique.

The procedure is built on the basic definition of modularity that focuses on modules that can be built (‘assembly modules’), rather than isolated module definitions as often done in prior literature. Furthermore, these ‘assembly modules’ are the modules that eventually determine the interfaces between modules. Based on this understanding, a new rating scheme was introduced that augments the information content in the component DSM. Another important contribution is that of PR; a simple, yet consistent and bias-free way to deal with multiple aspects in component DSM clustering has been introduced. In a subsequent step, DR has been shown to further prevent bias in the clustering. Based on an existing clustering algorithm, an improved version was developed. An important innovation is the possibility of using constraints during clustering, which is a necessity for the application of the post-processing procedure and for obtaining practical results. The post-processing procedure gives designers an important tool for the correction and improvement of clustering results, revealing the true interfaces between modules and subsystems and increasing system understanding.

Also very important is the insight that the identification of overlapping clusters, minibuses, and buses is unnecessary, since not the pure number of interactions is not decisive, but the type of the element, its interactions with multiple clusters, and the interaction ‘strengths’ (+2, +1, ...). Hence, the systematic procedure presented reduces arbitrariness in DSM clustering by providing clear rules and a comprehensive approach, thus enhancing the potential of DSM as a tool for module definition and hopefully increasing acceptance of these techniques among practitioners.

The case study has demonstrated the functionality of the clustering procedure and the clustering algorithm both with 1st and 2nd Level DSMs, constrained and unconstrained, and has also shown result with consideration of constraints. Without going into details, it can be easily seen that the clustering results differ significantly. The clusters obtained with the unconstrained clustering were significantly denser, whereas the constrained clustering resulted in bigger emptier clusters. Although the unconstrained clustered DSM provided denser and thus better clusters from a pure clustering standpoint (i.e. without regard to implementation issues), the ones obtained using constraints are much closer or even match the assignment of system elements to modules found in a real jet engine. Nevertheless, the clustering result has to be reviewed and basic post-processing procedures applied to the 2nd Level DSM (i.e. a post-processing in the post-processing is required to correct the cluster assignments). This is often relatively easy, since common 2nd Level DSMs are smaller than the one shown here, because usually only one system is modelled per 2nd Level DSM, so that several small 2nd Level DSMs have to be analysed rather than few big ones. Since this is similar to the post-processing of the 1st Level DSM, this step is not shown here. To document the final modules and their interfaces, the information from the 2nd Level DSM has to be transferred to the 1st Level DSM of the entire engine. Although being time-consuming without software tools, this is very easy, since only the elements (including their interactions) from each 2nd level cluster have to be transferred to the corresponding 1st level clusters replacing the former system elements (see Step 1, Section 6.2.5), and possible further interactions of the new 1st level elements (former 2nd level elements) with original 1st level elements have to be noted.
that the consideration of spatial constraints is a necessity, especially when clustering systems that are embedded in a product.

A weakness of the procedure is that no distinction between dependency strengths of +1 interactions is possible; e.g. based on information about signal bandwidth or mass flows. However, taking this information into account increases model complexity and general statements about the importance of interactions during clustering based on additional information are difficult. Usually, at the system level, products are not dense enough for this weakness to have major effects on the clustering outcome. Furthermore, additional constraints on product layouts, for instance due to outer appearance, aerodynamic or ergonomic requirements, or limited space will cause deviations from the ideal assumptions proposed by the procedure. These decisions based on other information than interface requirements have to be taken into account by manually updating the DSM model. Also, it is generally important when considering a modular product architecture to keep cost reasonable and choose a sensible trade-off between technical and business performance.

The proposed approach is a tool to assist system architects during product architecture definition (based on interactions), but it is not to be taken as a tool for automatically making final decisions. That is, the suggested product architecture is not necessarily the ‘optimal’ product architecture, and must not be readily accepted without challenging it. However, it is a starting point for systems architects to inspect and experiment with (during the post-processing step) in order to either confirm that the automated output is ‘sufficient’, or to refine this initial output by including various constraints that were not noted or included in the automated clustering step. Additionally, the proposed method is mainly intended for use with mechatronic products and systems, where usually the four types of interactions (structural, energy, signal, material) occur, but cannot be applied, for example, in the electronic sector, where many interactions of the same type occur between two elements. These cannot be modelled with this approach, since only one interaction of a certain type can be noted between two elements and it may not be possible to increase the level of resolution for these elements.

Notes

1. The term ‘cluster’ refers to the notion that modules are identified by seeking groups of elements in the product that are highly coupled among each other, but relatively weak to other groups.
2. The name assembly modules is used here to distinguish the new modularity definition given here from other definitions in the literature.
3. Yu et al.’s algorithm is able to detect buses, which will be pointed out later. For now, it suffices to know that the last cluster in the representation is always a bus.
4. The first type of mismatches (type-I) are voids in the real clusters, which are not correctly captured by the OCD. The second mismatch types (type-II) are marks outside the real clusters, which are not captured by the OCD.
5. That is, an element is duplicated or multiplied and these resulting elements are considered independent elements during clustering.
6. Indirect dependencies are dependencies between two elements, yet with another element in between. That is, there is no direct relation between the two elements.
7. For example, single-point crossover, two-point crossover, or uniform crossover.
8. The level of resolution denotes the ‘zoom level’ chosen to dissect the product. For instance, a subassembly like a pump can be entered as one element, or could be dissected further into, for example, bolts, pistons, seals, etc. The choice can influence the clustering outcome, and not every choice is sensible.
9. That is, values of 0.5 and smaller after the normalisation.
10. For example, Goldberg et al. (1989).
11. Note that in the following figures only the spatial aspect, i.e. the overall value obtained through PR is depicted.
12. Splitting is more difficult than the other four techniques presented and requires rather detailed knowledge of the product. Thus, we may not be able to apply this technique to its full extent in very early phases of the development process.
13. Since these two procedures are very simple in their application, they are not described here.
14. However, due to the post-processing the correct modules and their interfaces can be determined either way and the differences in the clusters identified by the clustering algorithm are typically not very grave.
15. Not all systems in a real jet engine were modelled, since they do not provide any new insights.
References


Helmer, R., 2007. Defining modular product architectures using the design structure matrix and genetic algorithms. Diploma Thesis. University of Illinois at Urbana-Champaign, Technical University Munich, Urbana, IL, USA.


