Res Eng Design (2007) 18:91-109
DOI 10.1007/s00163-007-0030-1

ORIGINAL PAPER

An information theoretic method for developing modular architectures using genetic algorithms

Tian-Li Yu · Ali A. Yassine · David E. Goldberg

Received: 9 March 2006 / Revised: 10 July 2006 / Accepted: 11 April 2007 / Published online: 10 August 2007

Abstract Designing modular products can result in many benefits to both manufacturers and consumers. The development of modular products requires the identification of highly interactive groups of elements and arranging (i.e., clustering) them into modules. However, no rigorous clustering technique can be found in engineering design literature. This paper uses the design structure matrix (DSM) to visualize the product architecture and to develop the basic building blocks required for the identification of product modules. The DSM architectural representation and building blocks are then used for the development of a new clustering method based on the minimum description length (MDL) principle and a simple genetic algorithm (GA). The new method is capable of partitioning the product architecture into a set of modules where interactions within modules are maximized and interactions outside modules are minimized. We demonstrate the proposed clustering method using several examples of real complex products and compare our results to clustering arrangements proposed by human experts. The proposed method is capable of mimicking the clustering preference of human experts and yields competitive clustering arrangements.

T.-L. Yu · A. A. Yassine · D. E. Goldberg
Department of General Engineering,
Illinois Genetic Algorithm lab,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA

Present Address:
T.-L. Yu (✉)
Department of Electrical Engineering,
National Taiwan University,
Taipei 106, Taiwan, ROC
e-mail: tianliyu@cc.ee.ntu.edu.tw

Keywords Design structure matrix (DSM) · Product architecture · Modular · Integral · Genetic algorithm (GA) · Minimum description length (MDL)

1 Introduction

Modularity is an ambiguous and elusive notion that has been loosely used in different ways by different people at different times (Schilling 2000; Gershenson et al. 2003). In its architectural usage, it is easier to define as an antonym, i.e., modular is the opposite of integrated.[1] So at the two extremes, modular architecture is one made up of assemblies of components, while an integrated architecture is one made up purely of the lowest level of component without having intermediate assemblies. A fully modular architecture is one with clear clusters of elements, and where the relationships between the elements within an assembly are hidden to the elements outside the assembly. This incorporates the notion that a module not only contains elements, but also contains a higher density of relationships between those elements than to elements outside the module. The significance of being a modular cluster is that all the parts within the module possess the same relationships with each other and with parts outside of the module—this can be thought of as being the design rules of the module (Baldwin and Clark 2000).

Modularity as a strategy can be employed for different reasons (Gershenson et al. 2003). In engineering design, modularity can be exploited at the early stages of product development to significantly reduce development effort (i.e., time and cost) (Pahl and Beitz 1996). In other

[1] Webster's dictionary defines the noun "module" as "*a compact assembly functioning as a component of a larger unit*".

circumstances, it allows increased product variety (Gershenson and Prasad 1997; Marshall 1999), increased rates of technological innovation (Baldwin and Clark 2000), increased opportunities for market dominance through interface capture (Moore 1999), and increased specialization at the firm level which in turn may allow more flexible response to environmental change (Sanchez and Mahoney 1996).

Product architecture is defined as the scheme by which decomposed elements of a product are arranged in modules (Ulrich 1995). Thus, the development of modular product architecture requires the identification of highly interactive groups of elements and clustering them into modules. Clustering techniques in graph theory are abundant (Hartigan 1975; Jian 1999); however, they are scarce in engineering design literature. This paper is concerned with methods and applications of clustering techniques in engineering design. The few ones found share two major deficiencies. First, the algorithms are manual, very dependent on human expertise, and consequently hard to automate or replicate (McCord and Eppinger 1993; Pimmler and Eppinger 1994; Stone et al. 2000; Gonzalez-Zugasti et al. 2000). Another significant problem deals with the use of simple mathematical constructs to discriminate between product modules and consequently these algorithms collapse when confronted with complex product architectures (Fernandez 1998; Thebeau 2001; Whitfield et al. 2002). In this paper we propose a new clustering method that can be easily automated, yet sophisticated enough to handle complex architectures.

This paper uses a graph and its matrix representation (referred to as the DSM) to describe and visualize a systems architecture (Yassine and Braha 2003).[2] Based on this representation, we develop the basic building blocks required for the identification of subsystems or modules (Sharman and Yassine 2004). These building blocks are then used as guiding principles for the development of a new clustering method. The proposed method is developed based on the minimum description length (MDL) principle (Rissanen 1978, 1999; Barron et al. 1998; Lutz 2002) and a simple genetic algorithm (GA) (Goldberg 1989). The method is capable of partitioning the system architecture into an "optimal" set of modules or subsystems and can be fine-tuned to mimic clustering arrangements proposed by human experts.

The proposed clustering approach can be used mainly in two areas affecting engineering design: (a) development team organization for identifying optimal team arrangements, and (b) product architecture decisions for developing modular products and product families. In the first area, the assignment of team members to development teams has been very informal, which usually rely on either functional basis or physical subsystem decomposition to determine team membership. Our proposed approach, on the other hand, relies on the actual interactions between development members (i.e., documented through interviews) in order to arrive at a more sensible team arrangement taking into account who is talking to whom, about what, and how frequently. In the latter area, our proposed method could be very useful at the system design phase of product development, where appropriate decomposition of the product into subsystems would significantly reduce rework and iteration among subsystems due to integration errors. Also, at this stage, decisions regarding a platform strategy are made: what is the platform and how are we going to achieve variety (Simpson et al. 2006; Jiao et al. 2006)? To answer these questions, we need a method to allow for the identification of the most appropriate subset of physical elements to include in the platform. We also need to determine the most appropriate modules to use for achieving a certain level of product variety. These potential modules can also be identified with our proposed approach.

The rest of the paper proceeds as follows. A brief review of existing clustering methods from graph theory and engineering design literature is presented next. Then, in Sect. 3, we provide an overview of the DSM method, its basic building blocks for the identification of modules, and discuss some of the clustering difficulties encountered when using DSMs. In Sect. 4, we propose a metric to evaluate different architectural arrangements based on the MDL principle. Section 5 introduces the GA used to search for the optimal product architecture. The proposed clustering algorithm, along with few simple examples, is presented in Sect. 6. A method for tuning important model parameters is described in Sect. 7. In Sect. 8, we demonstrate this new clustering method using three examples of complex product architectures, and finally, Sect. 9 concludes the paper.

2 Literature review

Formally, a clustering of a graph $G = (N,E)$ (where N is a set of nodes and E is a set of weighted edges) is a partition of N into k disjoint subsets (called clusters) $N_1, N_2, ..., N_k$ (Hartigan 1975). Many constraints can be specified which can limit the size of each cluster or the number of clusters, among others. Every imaginable variation of the graph clustering problem is known to be NP-Hard (Garey et al. 1976), and therefore, numerous heuristics have been used for providing fast solutions. These heuristics vary depending on the specific application and objective function chosen

[2] In a graph, nodes represent product components and edges represent relationships between these components.

(Anderberg 1973), which include soft computing methods [such as simulated annealing (Kirkpatric et al. 1983); tabu search (Glover 1989, 1990), and genetic algorithms (Bui and Moon 1996)], spectral methods (Gaertler 2002), and operations research methods (Hansen 1997). A lengthy review on this large stream of graph theory literature is outside the scope of this paper; however, the reader is referred to Jian et al. (1999) for a comprehensive review.

Tools for identifying product modules are scarce in the engineering design literature; however, the few ones found are limited to guidelines and heuristics (Rechtin and Maier 1997; Gershenson et al. 2004).[3] Early clustering algorithms for engineering design and underlying principles are described in (Alexander 1964). The relevant clustering metric proposed is some function of the number of linkages that cross a partition boundary. More recently, Stone et al. (2000) proposed the use of Function Diagrams for identifying product modules. This technique starts with creating a function structure for the product under consideration, which is then followed by grouping the sub-functions into modular chunks.[4] Other techniques such as Hatley–Pirbhai method (Zakarian and Rushton 2001), and interaction graphs (Kusiak and Huang 1996) have also been used for the development of modular products. All these methods rely on undirected search for modules in the structure and are likely to result in non-optimal and non-unique modular architectures. Furthermore, these methods rely on mapping the functional decomposition of the product to the physical architecture. A module in this way becomes the physical realization of a function, and the whole interface problem (between physical elements) is not properly addressed. On the other hand, this paper does not address itself to the function/form relationship (i.e., does not require a functional decomposition as a prerequisite for modularity analysis) and rely merely on the physical components composing the product and the interaction amongst them.

In addition to graph theoretic measures used for a clustering metric, recently, Yu et al. (2003a) and Wang and Antonsson (2004) proposed an information theoretic measure of modularity. These measures were used in conjunction with a GA to search for optimal modular configurations. Both proposed algorithms are closely related to this paper. In particular, this paper is an improvement over the Yu et al. (2003a) approach, which allows for weighted edges in the graph and for better tuning of clustering parameters to mimic human expertise. Also, our method differs form Wang's algorithm by its capability to detect and propose overlapping clusters and bus structures in the product architecture. Another clustering measure was proposed by Baldwin and Clark (2000) using real options theory where they calculated the net options value of modularity for various product architectures. Sharman and Yassine (2006) have experimented with such a clustering metric and reported many difficulties encountered by this metric when dealing with complex product architectures that do not possess a pure hierarchical structure.

Another research stream that is related to this paper focuses on the development of product families based on modular product platforms that allow sharing of core modules among different products (Meyer and Lehnerd 1997; Robertson and Ulrich 1998). There are several approaches to designing different kinds of product platforms. Gonzalez-Zugasti et al. (2000) formulated the design of a platform-based product family as a general optimization problem in which the advantages of designing a common platform must be balanced against the constraints of the individual product variants. Their approach allows the identification of modules that could be made common to several product variants. Simpson et al. (2001) used a Decision Support Problem formulation to design families of products based on *scalable* platforms.[5] Martin and Ishii (2002) proposed two indices to measure a product architecture, which were used by design teams to develop a decoupled architecture that requires less design effort for follow-on products. This research stream is different form our work in that we are investigating the "optimal" module architecture for a single product without any commonality considerations that arise in family approaches. However, this paper complements the platform development literature by providing a preprocessing phase to facilitate the identification of core platform modules (that are shared by all variants of a family) and individual (i.e., private) modules that create differentiation among family members.

Finally, DSM models have been used to represent product architecture and both manual and automated algorithms to uncover product modules have been reported (Sharman and Yassine 2004). Details regarding the DSM approach are discussed next.

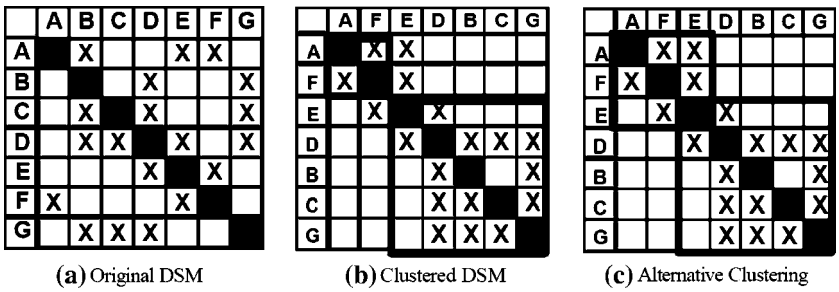
3 The design structure matrix (DSM) method

A DSM is a matrix representation of a graph. The nodes of the graph (which represent components of a product or system) correspond to the column and row headings in the matrix (Eppinger et al. 1994; Yassine et al. 2003). The

[3] For a comprehensive review of product modularity methods, we refer the refer the reader to Gershenson et al. (2004).

[4] A function structure is an input-output diagram of what a product does.

[5] In contrast to the standard approach of adding/substituting modules to produce product variants, this approach to product family design calls for the development of common product platforms that can be "stretched" or "scaled" in one more dimension to satisfy a variety of market requirement (Simpson et al. 2001).



(a) Original DSM (b) Clustered DSM (c) Alternative Clustering

Fig. 1 DSM clustering examples. **a** Original DSM. **b** Clustered DSM. **c** Alternative clustering

arrows (which represent relationships between components) correspond to the "X" marks inside the matrix.[6] For example, if there is an arrow from node C to node A, then a 'X' mark is placed in row A and column C. Diagonal elements have no significance and are normally blacked-out or used to store some element-specific attribute(s). Alternatively, number "one" can be placed instead of an "X" and "zero" instead of a blank. This makes the DSM a binary matrix with entries $d_{ij} = 0$ or 1.

Once the DSM for a product is constructed, it can be analyzed for the identification of modules; a process referred to as clustering. The goal of DSM clustering is to find subsets of DSM elements (i.e., clusters or modules) that are mutually exclusive or minimally interacting. In other words, clusters contain most, if not all, of the interactions (i.e., DSM marks) internally, and the interactions or links between separate clusters is eliminated or minimized (Fernandez 1998).[7] As an example, consider the DSM in Fig. 1. As can be seen in Fig. 1b, the original DSM was rearranged (by simply swapping the position of rows and columns) to contain most of the interactions within two separate blocks or modules: AF and EDBCG. However, three interactions are still outside any block.[8] An alterative arrangement is suggested in Fig. 1c. This arrangement suggests the forming of two overlapping modules (i.e., AFE and EDBCG).

The DSM representation of a system/product architecture has proved useful because of its visual appeal and simplicity, and numerous researchers have used it to propose architectural improvements by simple manipulation of the order of rows and columns in the matrix (McCord and Eppinger 1993; Pimmler and Eppinger 1994). In an attempt to automate this manual process of DSM inspection and manipulation, Fernandez (1998) used a DSM model with a simulated annealing search techniques in order to find "good" DSM clustering arrangements. In his approach, each element is placed in an individual set and bids evaluated from all the other sets (clusters). If any cluster is able to make a bid that is better than the current base case then the element is moved inside the cluster. The objective function is therefore a trade-off between the costs of being inside a cluster and the overall system benefit. Sharman (2002) attempted using the clustering algorithm described in Fernandez (1998) on an industrial gas turbine. However, he showed that this algorithm is incapable of predicting the formation of "good" clustering arrangements for complex product architectures due to the oversimplification of the objective function utilized and the frequent susceptibility of the search algorithm used to be trapped in local optimal solutions. In a similar venue, Whitfield et al. (2002) used genetic algorithms to form product modules. Their algorithm is also built upon the same concepts introduced by Fernandez and as such suffers from similar problems.

3.1 DSM building blocks: product architecture terminology

In this section, we review the basic syntax and semantics for analyzing DSMs in order to characterize the architecture of a complex product and identify its modules. The analysis proceeds using simplified DSMs and product architectures. These simple constructs can be used as building blocks for analyzing more complex architectures (Sharman and Yassine 2004).

Consider the simple product architecture depicted by the physical schematic of Fig. 2a. It shows five related (i.e., connected) parts or components.[9] The situation may be visualized as part E being some form of system level integrating component, or bus, which is connected to parts A, B, C, and D. In this instance, the relationships between the parts are symmetrical, i.e., part A depends on part E in

[6] There are different ways of building a DSM. For a full description, please refer to the DSM website at http://www.DSMweb.org.

[7] It worth noting that "partitioning" of DSMs is different than "clustering" (Yassine and Braha 2003). Although both methods rely on re-ordering DSM elements, the objectives are different. In partitioning, the objective is to remove feedback marks from the DSM by making the DSM lower triangular (McCulley and Bloebaum 1996). In clustering, we do not care about feedbacks and DSM marks' location above or below the matrix diagonal is immaterial.

[8] These marks may constitute the interface between the two product modules.

[9] The relationship can be representing a flow of mass, energy, information, or force/geometrical constraint between the parts.

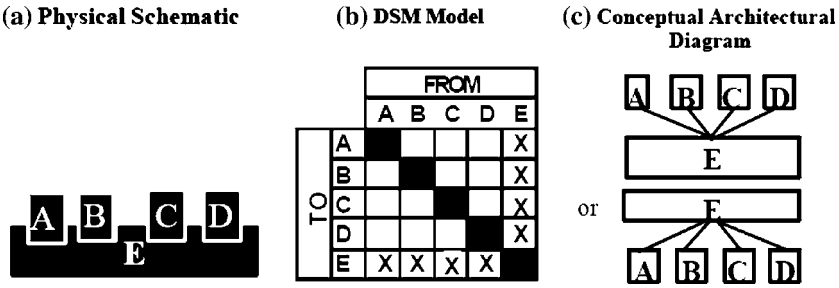


Fig. 2 A simple bus and no modules. **a** Physical schematic. **b** DSM model. **c** Conceptual architecture design

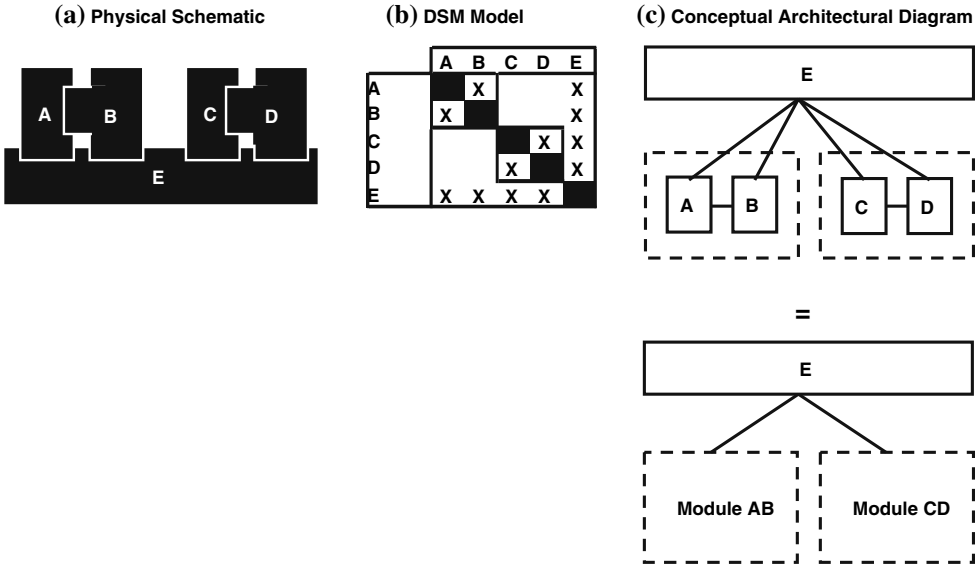


Fig. 3 A simple bus and two modules. **a** Physical schematic. **b** DSM model. **c** Conceptual architecture design

the same way that part E does on A. To the right of the schematic the DSM for this system is drawn. To the right of the DSM a more conceptual architectural diagram that represents the same situation is shown. This third diagram can be drawn in any orientation as the relationships are symmetrical, thus at this stage no value should be assigned to any convention that chooses to draw part E above rather than below or to the side.

A part is the smallest possible decomposition of something while a chunk or element could be assemblies in their own right. For this reason strict terminology should differentiate between primitive elements (i.e., parts) or higher-level elements. We term a bus as being "simple" when it is a primitive element (buses may also be comprised of multiple primitive elements, in which case they may be referred to as a bus module). Figure 3 introduces more relationships between the parts. In this instance, the pair of parts A and B is related symmetrically to each other as well as to part E. A similar structure can be seen in the pair of parts C and D. In the DSM, this pairing is defined as being a modular cluster and denoted by "Module AB" and "Module CD".

Many product architectures are more complicated than the examples inspected above. Figure 4 shows a number of possibilities. Either (A, B) and (C, D) can be concatenated into modules, the sequence A through F can be thought of as one super module, an intermediate module CD (comprising C, D) can be sandwiched between module AB and module EF, or simply described as being comprised of the primitives A through F with the bus module GH. This example illustrates the notions of "pinning" and "holding away." Here part B is pinned in place between A and C by its relationship with A and C. This is a common and easily appreciated situation in much physical architecture. A less

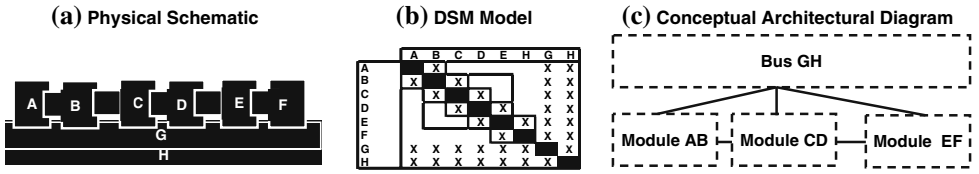


Fig. 4 Imperfection, pinning, and holding away. **a** Physical schematic. **b** DSM model. **c** Conceptual architecture design

easily appreciated situation is that of "holding away," which is seen here in that element A is held away from element C by element B. The combination of these two notions assists in describing the drawbacks of various clustering arrangements and clustering heuristics. Pinning can only occur to compound elements (modules) that have relationships with two modules, while any primitive element can be held away.

3.2 Dsm clustering complexities

The problem with applying automated clustering algorithms to complex DSMs is that it is difficult for these algorithms to extract the relevant information from the data, and then to convey the information to the user. This is most noticeable in the poor handling of pinned modules, buses, and path-dependent situations. We investigate this further in the rest of this section.

3.2.1 Path dependency in clustering

Consider a triangular arrangement of symmetrical relationships that can be loosely clustered into three similar modules AA, BB, and CC, as shown in Fig. 5. The DSM for this physical arrangement can only show two of the real clusters and must break up the third. The way in which the clustering algorithm operates will be path dependent inasmuch as once it has started to cluster on any two nodes it is unlikely to reverse into a different configuration. This situation may occur because of the way in which raw data is presented to the clustering algorithm (for example branch and bound algorithms that are presented with a partially clustered starting point may never branch widely enough to evaluate alternative solutions) or may arise through chance if a perfectly random starting point is presented to an algorithm that makes an initial random guess. In the example of Fig. 5, cluster CC has been broken up even though it is identical in all respects to the other two clusters. It could as easily have been AA or BB clusters that were broken up by being positioned where CC is.

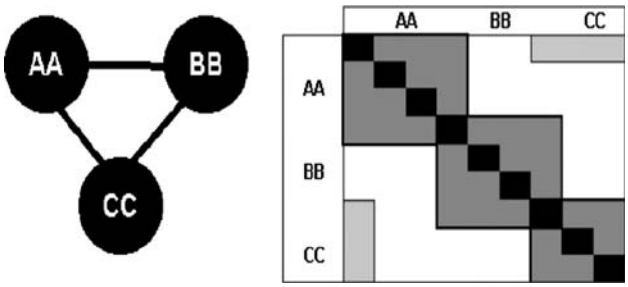


Fig. 5 Planar triangular clusters

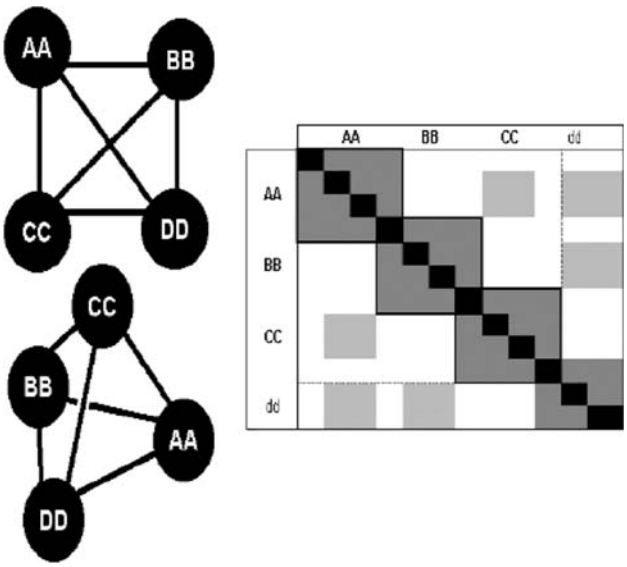


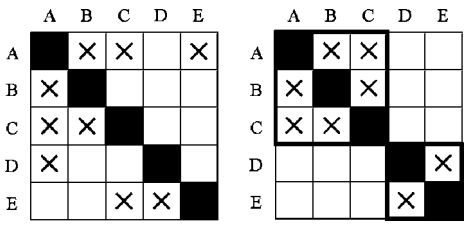
Fig. 6 Tetrahedron of clusters

3.2.2 Dimensions and topology

Consider a simple three-dimensional (3D) structure such as a tetrahedron or 3D pyramid. This is depicted in Fig. 6 showing four equal clusters, each with dense internal relationships and weaker (or sparser) external relationships. As before, if all the clusters are perfectly equal it is purely a matter of chance how any clustering algorithm would present an answer. In this example, cluster DD is the one that is visually disrupted most by being presented last in the sequence. This has the effect of spreading its inter-cluster relationships over a wider spatial area, which is depicted in the DSM as lower density blocks of grey. To an untrained observer this might be thought to be a bus structure where cluster DD is the unique possessor of system wide integrating functions and some semi-random cross-linking occurs in the zone AA-CC.

4 Minimum description length (MDL) based objective function

As mentioned earlier, DSM clustering algorithms can be found in (McCord and Eppinger 1993; Fernandez 1998; Thebeau 2001; Whitfield et al. 2002). Experiments with these algorithms showed that the clustering metric (and thus the objective function) used was insufficient for accurately predicting "good" clustering arrangements (Sharman 2002; Sharman and Yassine 2004). The lack of an efficient clustering method, particularly suited for analyzing product architectures, motivated us to seek a better clustering metric based on information theoretic measures.



(a) (b)

	Hypothesized marks that do not exist in the real data set (Type I mismatch)	Real marks that the hypothesized model misses (Type II mismatch)
Inside clusters	1) Row B, Column C 2) Row D, Column E	ϕ (always)
Outside clusters	ϕ (always)	1) Row A, Column E 2) Row D, Column A 3) Row E, Column C

(c)

Fig. 7 **a** Given data set. **b** DSM created based on the model description. **c** Mismatched data description

The previous section outlined the requirements necessary for the development of a new clustering metric and its corresponding algorithm:

1. The algorithm should be able to suggest the optimal number of clusters.
2. The algorithm should be able to detect the existence of bus modules.
3. The algorithm should be able to detect overlapping clusters and 3D structures.

While the above two requirements can be addressed by the appropriate choice of a clustering objective function (i.e., clustering metric), as discussed next, the third requirement is directly related to the proposed search strategy, discussed in Sect. 5.

4.1 Minimal description length principle (MDL)

Suppose we have a model which describes a given product structure or a data set.[10] Usually, the model does not completely describe the given data; otherwise, the model would become too complicated. Therefore, the description length needed to describe the given data consists of two parts: the *model description* and the *mismatched data description*. This scheme may be easier to understand in light of the following sender–receiver example.

Assume a sender has a given data set which is needed by the receiver. Given a model that approximately (i.e., not exactly) describes the given data set, as described in the DSM of Fig. 7a, the sender first sends the model (i.e., model description) to the receiver. The model description is a hypothesized description of the data set, which assumes that elements within a cluster are fully connected to each other and the clusters are completely disjoint. For example, a model description could simply provide the number of clusters and the names of elements contained in each cluster. For the data set of Fig. 7a, the sender would send the following model description: [Cluster 1: A, B, C; Cluster 2: D, E]. Then, the receiver understands that the data is as in Fig. 7b. Note that the model description does not completely represent the given data set. Therefore, to ensure that the receiver gets the correct data set, the sender also sends the data which are mis-described by the model (i.e., mismatched data description), shown in Fig. 7c. Now, given both the model description and the mismatched data description, the receiver can completely recreate the original data set. Note that if the model (the hypothesized description) is simple, the model description is short; however, many data mismatches would exist, and the mismatched data description would become longer. On the other hand, a complicated model reduces the description of mismatched data, but the model description would be longer.

The MDL principle (Rissanen 1978, 1999; Barron et al. 1998; Lutz 2002) satisfies our needs for dealing with the above tradeoff. The MDL can be interpreted as follows: *among all possible models, choose the model that uses the minimal length for describing a given data set (that is, model description length plus mismatched data description length)*. In other words, MDL tries to find a simple model (short model description length) that describes the given data well (short mismatch description length). There are two key points that should be noted when MDL is used: (1) the encoding should be uniquely decodable, and (2) the length of encoding should reflect the complexity. For example, the encoding of a complicated model should be longer than that of a simple model. Next, we define the MDL clustering metric in detail.

4.2 Model description

The way we describe the model is naïve. The description of each cluster starts with a number which is sequentially assigned to each cluster, and then this is followed by a sequence of nodes in the cluster. Figure 8 shows a DSM clustering arrangement and the corresponding model description. It is easily seen that the length of this model description is as follows:

[10] Here, the model means a description that specifies which node belongs to which cluster. This can be a graph or its DSM representation.

$$\sum_{i=1}^{n_c} (\log n_n + cl_i \cdot \log n_n), \quad (1)$$

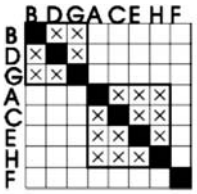
where n_c is the number of clusters in the DSM, n_n is the number of rows or columns in the DSM (i.e. DSM elements), cl_i is the number of nodes in the ith cluster, and the logarithm base is 2. In the example of Fig. 8, n_c = 2 clusters, n_n = 8 DSM elements, cl_1 = 3, and cl_2 = 4. The table in the figure reads as follows: " *cluster 1 has 3 nodes: B, D, and G; cluster 2 has 4 nodes: A, C, E, and H.*"

If n_n and n_c are known, it is not difficult to see that the above model description is uniquely decodable. When n_n is given, and assuming $n_c \leq n_n$, then log n_n bits are needed to describe n_c. The log n_n bits are fixed for all models, and therefore they are omitted without loss of accuracy. Finally, if we allow for bus detection, the bus is treated exactly as another cluster (in our implementation, the bus is the last cluster) in the model description. Therefore, the description length is also captured by Eq. 1.

4.3 Mismatched data description

Based on the model, we first construct another matrix (call it *DSM'*), where each entry d'_{ij} is "1" *if and only if*: (1) some cluster contains *both* node i and node j simultaneously, or (2) the bus contains *either* node i or node j. Then, we compare d'_{ij} with the given d_{ij}. For every mismatched entry, where $d'_{ij} \neq d_{ij}$, we need a description to indicate where the mismatch occurred (i and j) and one additional bit to indicate whether the mismatch is zero-to-one or one-to-zero. Define the following two mismatch sets: $S_1 = \{(i,j) | d_{ij} = 0, d'_{ij} = 1\}$ and $S_2 = \{(i,j) | d_{ij} = 1, d'_{ij} = 0\}$. We call the mismatch that contributes to S_1 the *type-I mismatch*, and the mismatch that contributes to S_2 the *type-II mismatch*. The mismatched data description length is given by:

$$\sum_{(i,j)\in S_1} (\log n_n + \log n_n + 1) + \sum_{(i,j)\in S_2} (\log n_n + \log n_n + 1). \quad (2)$$



Length	log n_n	3 log n_n	log n_n	4 log n_n
Description	3	B,D,G	4	A,C,E,H

Fig. 8 Model description length. The above is the DSM created from the model description. In this example, the model description length is 9log n_n, where n_n is the number of elements (=8)

The first log n_n in the bracket indicates i, the second one indicates j, and the additional one bit indicates if the mismatch is of type-I or type-II. The two summations are the description lengths needed for all type-I mismatches and type-II mismatches respectively. For example, if there is a type-I mismatch in row 3, column 6, it can be represented as (3, 6, 0). Likewise, if there is a type-II mismatch in row 4, column 5, it can be represented as (4, 5, 1).

4.4 MDL clustering metric

The MDL clustering metric is given by the weighted summation of the model description length given in Sect. 4.2 and the mismatched data description given in Sect. 4.3. With some arithmetic manipulations, the metric can be written as follows:

$$f_{\mathrm{DSM}}(M) = (1-\alpha-\beta)\left(n_c \log n_n + \log n_n \sum_{i=1}^{n_c} cl_i\right) + \alpha[|S_1|(2\log n_n + 1)] + \beta[|S_2|(2\log n_n + 1)], \quad (3)$$

where α and β are weights between 0 and 1.[11] The α and β setting is not a simple task. A naïve setting is $\alpha = \beta = 1/3$. In Sect. 7, we adjust α and β to mimic the behavior of a manual clustering arrangement. The objective is to find a model M that minimizes f_{DSM}. In other words, f_{DSM} is the length (in bits, when the logarithm is taken in base 2) that model M needs to describe the given data.

Figure 9 illustrates an example how the metric choose between different models. Figure 9a is the original DSM before clustering. Figures 9b and c are two possible models describing the given DSM. Figure 9b has four type-I mismatches, and Fig. 9c has four type-II mismatches (shadowed cells). If the weight of type-I mismatch are the same as that of type-II mismatch, our metric favors the model in Fig. 9c since it is a simpler model (i.e., has a shorter model description).

4.5 From binary DSM to weighted DSM

Most real-world DSMs are real valued or contain information of different levels of interaction strength. Therefore, it is necessary to extend our algorithm to be capable of clustering weighted DSMs. If we normalize a weighted DSM so that every entry in the DSM is between 0 and 1, the value of each entry can be considered as the probability of communication. This is consistent with binary DSMs. In a binary DSM, d_{ij} = 1 can be thought as that node i communicates with node j with probability 1. The same

[11] Here we use weighting in order to match the preference of human experts. This becomes more evidential in the case study.

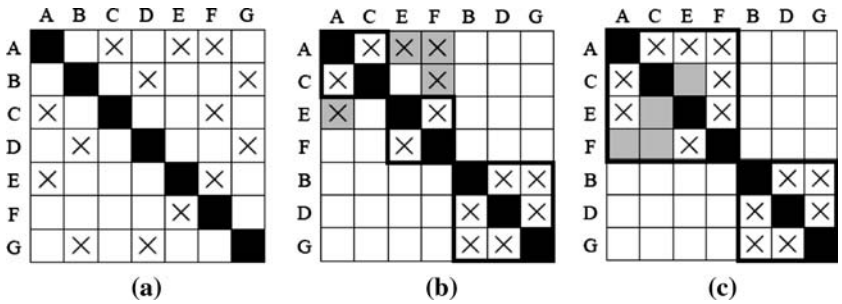


Fig. 9 How does the metric choose between different models. **a** Original DSM before clustering. **b** One possible model for describing the given DSM in (**a**). **c** Another possible model for describing the given DSM

interpretation is also valid for $d_{ij} = 0$. Based on the interpretation, we can modify the MDL scoring metric as follows. First, the entries d_{ij} in the DSM are normalized to $p_{ij} = (d_{ij} - d_{\min})/(d_{\max} - d_{\min})$, where $d_{\max} = \max_{i,j} d_{ij}$ and $d_{\min} = \min_{i,j} d_{ij}$. The formula of the description length of model complexity remains the same. The mismatch sets for type 1 and type 2 are modified as $S_1 = \sum_{d'_{ij}=1} (1 - p_{ij})$ and $S_2 = \sum_{d'_{ij}=0} p_{ij}$, respectively. The modification is so because entry ij has a probability $(1 - p_{ij})$ to be a type 1 mismatch if it is inside a cluster, and a probability p_{ij} to be a type 2 mismatch if it is outside clusters.

5 Search strategy: a simple genetic algorithm

Exhaustive search is the most basic search method. It generates all possible architectures for a given product and exhaustively evaluates each one to determine the best one. This search method is only possible for small DSM sizes, as the number of possible product architectures grows large, very fast, as the DSM size increases.[12] Even with today's high-powered computers, using an exhaustive search to find the optimal solution for even relatively small problems can be prohibitively expensive.

Genetic algorithms are general-purpose search algorithms based upon the principles of evolution observed in nature (Goldberg 1989). Genetic algorithms are able to find optimal or near optimal solutions by using natural mechanisms, such as selection, crossover, and mutation. This section provides an introduction to the GA used in this paper: a *simple* GA with $(\lambda + \mu)$ *selection, uniform crossover*, and *bit-wise mutation*. An outline of how the specialized GA uses these operators to search for optimal solutions is described below.

1. *Create an initial population of chromosomes*. Each chromosome is made up of a collection of genes and represents a complete solution to the problem at hand. The gene values are usually initialized to random values within user-specified boundaries. Initially, there are λ chromosomes.
2. *Crossover chromosomes to produce new offspring chromosomes*. This crossover occurs according to some user-defined probability p_c(usually high) and results in new chromosomes having characteristics taken from both of the parent chromosomes. In this paper, *uniform crossover* is adopted. Uniform crossover operator randomly switches each gene of the two parent chromosomes with a certain probability (0.5, in this paper) to produce two new offspring. If an offspring takes the best parts from each of its parents, the result will likely be a better solution. The two parents are randomly picked up (without replacement) from the λ chromosomes, and the reproduction is continued until μ offspring chromosomes are produced. Note that no selection is performed here yet.
3. *Mutate the genes of the offspring chromosomes*. This mutation occurs according to some user-defined probability p_m (usually low) and serves to introduce some variability into the gene pool. For a binary encoding chromosome, the bit-wise mutation inverts the value of genes (from 0 to 1, or from 1 to 0) with the mutation probability p_m. Without mutation, offspring chromosomes would be limited to only the genes available within the initial population.
4. *Evaluate each of the chromosomes in both parent chromosomes and offspring chromosomes*. Each chromosome is evaluated by a fitness function (i.e., objective function as described in Sect. 6) to determine the quality of the solution. Note that only μ chromosomes need to be evaluated except for the first generation.
5. $(\lambda + \mu)$ *selection is performed to select chromosomes that will have their information passed on to the next generation*. In step 5, totally $(\lambda + \mu)$ chromosomes are evaluated. $(\lambda + \mu)$ selection chooses λ best chromosomes from the $(\lambda + \mu)$ chromosomes and passes them to the next generation. Note that elitism is embedded in $(\lambda + \mu)$ selection.
6. *Repeat steps 2 through 5 until some termination condition has been met*. This termination condition can be based simply on the number of generations, or it can

[12] The number of possible product architectures for a system with n elements without cluster overlaps is: $\sum_{i=1}^{n} C_i^n = 2^n - 1$

be based upon more complex criteria such as the fitness convergence.

6 The proposed clustering algorithm and illustrative examples

Previous sections have described the proposed MDL clustering metric and a simple GA. This section first gives the encoding method, and the proposed DSM clustering algorithm is then described and tested on several manually designed examples.

6.1 Encoding

To use a GA with the MDL clustering metric, an encoding method that encodes a clustering arrangement into a chromosome is needed. As indicated in Sect. 4, the encoding should deal with overlapping clusters and 3D structures. As long as the encoding allows that a node belongs to several different clusters, the GA is able to detect overlapping clusters and 3D structures.

The chromosome is a binary string of $(c \cdot n_n)$ bits, where c is predefined maximal number of clusters, and n_n is the number of nodes. The $(x + n_n \cdot y)$-th bit represents that node $(x + 1)$ belongs to cluster $(y + 1)$. The last cluster is treated as a bus. For example, in Fig. 8, $n_n = 8$, and given c is 3, then the model can be described by the following chromosome shown in Fig. 10. When manipulated, the chromosome is transformed into a binary string which is a concatenation of all rows. At the end of the GA run, the result would a binary string similar to the one in Fig. 10. The binary string tells which nodes should form which clusters as well the bus, and it should correspond to a very short description length. In other words, the GA tries to find a model M that minimizes Eq. 3 for a given DSM.

6.2 Put it all together

With the encoding method, now it is sufficient to apply the proposed MDL clustering metric to a GA to create a DSM clustering algorithm. At the beginning, a population of (encodings of) DSM clustering arrangements is randomly initialized. The MDL clustering metric is then applied to each DSM clustering arrangement, and the description length of each DSM clustering arrangement is obtained. With the evaluations, a GA with $(\lambda + \mu)$ selection, uniform crossover, and simple mutation searches the DSM clustering arrangement with a minimal description length.

Node	A	B	C	D	E	F	G	H
Cluster 1	0	1	0	1	0	0	1	0
Cluster 2	1	0	1	0	1	0	0	1
Bus	0	0	0	0	0	0	0	0

Fig. 10 A chromosome that represents the model shown in the *lower part* of Fig. 8. The binary string manipulated is 010100101010100100000000

6.3 Illustrative examples

Before applying the proposed DSM clustering algorithm on a real-world problem, we tested it on some manually designed problem to illustrate the abilities of the proposed DSM clustering algorithm.

In the following experiments, the number of nodes is 9, the maximal number of clusters is set to 4, and hence the chromosome length is $4 \times 9 = 36$. The crossover probability p_c is 1, mutation probability p_m is 1/36, and a (50 + 50) selection is adopted. The GA is terminated if no improvement happens for five generations. The weights in the MDL clustering metric are set equally to 1/3. Figure 11 shows the GA results of some illustrative examples. Those DSMs are designed in such a way that we know what the optimal clustering should be. Then we randomly reordered the DSMs and handed them to the GA as inputs to test if the GA is capable of rediscovering the correct clusters.

Figure 11a is the simplest case, two non-overlapping clusters. Figure 11b demonstrates the ability of identifying overlapping clusters. In Fig. 11c, a bus is introduced, and the GA is able to identify it. One thing worth to note is that the DSM in Fig. 11d resembles the DSM in Fig. 11c; however, the result is totally different. The GA recognized the DSM in Fig. 11d as three overlapping clusters instead of a bus. Figure 11(d) also demonstrates the ability of identifying 3D structures. To sum up, these results show that the GA with the MDL clustering metric is able to solve complex problems with overlapping clusters, a bus, or 3D structures.

7 Adjusting weights in the scoring metric

In this section, we demonstrate the use of Widrow–Hoff iteration (or Delta rule) (Widrow and Hoff 1960) for tuning the weights of the proposed MDL–GA algorithm, so that the GA results would have similar ratios of the description lengths preferred by human experts. Recalling that the GA objective is to find model M (clustering arrangement) that minimizes $f_{\mathrm{DSM}}(M)$ for a given DSM. For simplicity, we rewrite Eq. 3 as:

$$f_{\mathrm{DSM}}(M) = w_1 f_1 + w_2 f_2 + w_3 f_3. \quad (4)$$

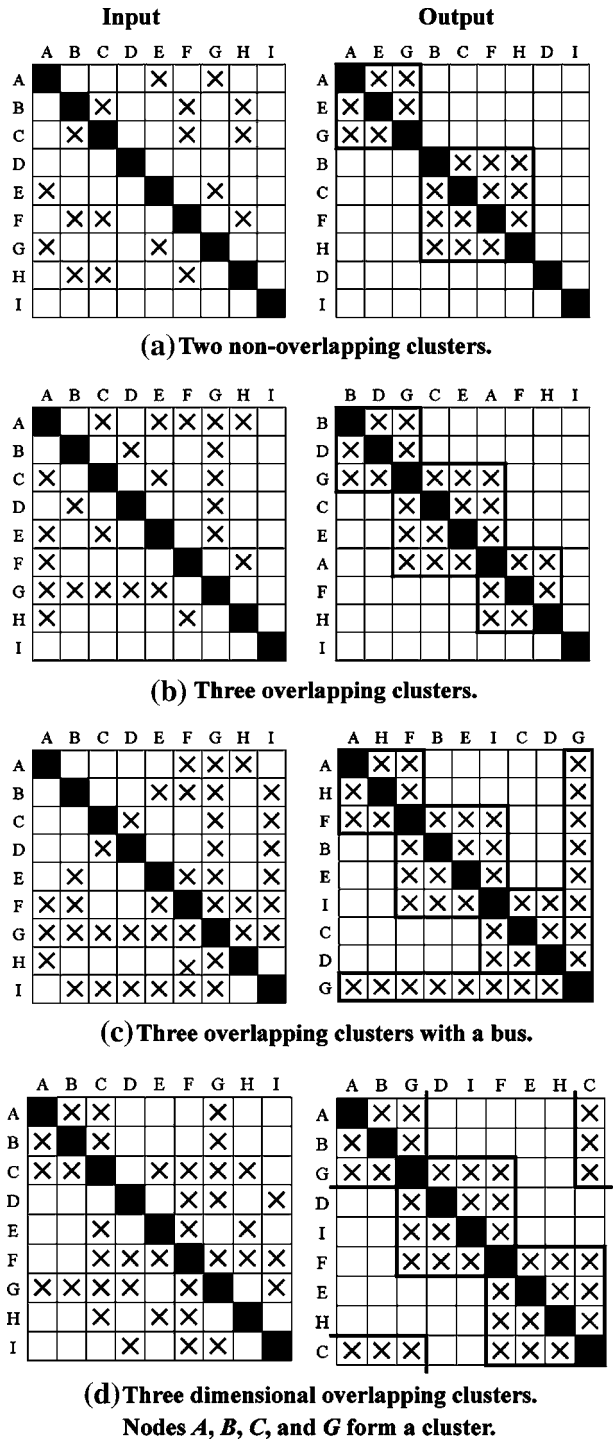


Fig. 11 Illustrative examples. The *left column* is the manually designed DSMs given as inputs to the GA. The *right column* shows how the GA clusters them. **a** Two non-overlapping clusters. **b** Three overlapping clusters. **c** Three overlapping clusters with a bus. **d** Three-dimensional overlapping clusters. Nodes *A*, *B*, *C*, and *G* form a cluster

The objective of adjusting the weights is to find w_i such that after GA runs, the ratios $\frac{f_1}{f_1+f_2+f_3} : \frac{f_2}{f_1+f_2+f_3} : \frac{f_3}{f_1+f_2+f_3} \approx r_1 : r_2 : r_3$, where r_i reflects the human preference of clustering arrangements (in the next section, we survey several real-world DSMs to retrieve these ratios). Since the summation of the weights and ratios are ones, we are actually facing a nonlinear system with two degrees of freedom. Therefore, we define two nonlinear functions as follows:

$$\begin{cases} R_1(w_1, w_2) = \dfrac{f_1(w_1, w_2)}{f_1(w_1, w_2) + f_2(w_1, w_2) + f_3(w_1, w_2)} - r_1 \\ R_2(w_1, w_2) = \dfrac{f_2(w_1, w_2)}{f_1(w_1, w_2) + f_2(w_1, w_2) + f_3(w_1, w_2)} - r_2 \end{cases} \tag{5}$$

Now, the objective of adjusting weights is to find a pair (w_1, w_2) such that $(R_1, R_2) \approx (0,0)$. There have been many numeric methods trying to conquer nonlinear systems such as Newton–Raphson method (Press et al. 1992), Broyden method (Broyden 1965), and the Steepest Descent method (Press et al. 1992). However, none of the above is suitable for our situation. Newton–Raphson method requires the information of first derivatives, which we do not know; Broyden method utilize difference to retrieve the gradient information on the fly, which in our case, increases the noise produced by the GA; the Steepest Desent method converges slowly and requires more GA runs. Therefore, we make a reasonable assumption and derive the gradient information which guides the Widrow–Hoff iteration.

Define $c_1 = w_1 f_1$ and $c_2 = w_2 f_2$. The assumption we make is that for small changes of w_i, the c_i do not vary much. In other words, we assume that the following Jacobian matrix is close to zero:

$$\begin{bmatrix} \dfrac{\partial c_1}{\partial w_1} \dfrac{\partial c_2}{\partial w_1} \\ \dfrac{\partial c_1}{\partial w_2} \dfrac{\partial c_2}{\partial w_2} \end{bmatrix} \approx 0, \tag{6}$$

where 0 is a zero metric. Based on this assumption, we are able to express R_1 and R_2 as explicit functions of w_1 and w_2 as follows:

$$\begin{cases} R_1 = \dfrac{f_1}{f_1+f_2+f_3} - r_1 = \dfrac{c_1}{w_1\left(\frac{c_1}{w_1} + \frac{c_2}{w_2} + \frac{c_3}{1-w_1-w_2}\right)} - r_1 \\ R_2 = \dfrac{f_2}{f_1+f_2+f_3} - r_2 = \dfrac{c_2}{w_2\left(\frac{c_1}{w_1} + \frac{c_2}{w_2} + \frac{c_3}{1-w_1-w_2}\right)} - r_2 \end{cases} \tag{7}$$

Hence we can calculate the Jacobian matrix as follows:

$$J = \begin{bmatrix} \dfrac{\partial R_1}{\partial w_1} \dfrac{\partial R_2}{\partial w_1} \\ \dfrac{\partial R_1}{\partial w_2} \dfrac{\partial R_2}{\partial w_2} \end{bmatrix} = \begin{bmatrix} \frac{-c_1\Delta - c_1 w_1 \Delta_1}{w_1^2 \Delta^2} & \frac{-c_2 \Delta_1}{w_2 \Delta^2} \\ \frac{-c_1 \Delta_2}{w_1 \Delta^2} & \frac{-c_2\Delta - c_2 w_2 \Delta_2}{w_2^2 \Delta^2} \end{bmatrix}, \tag{8}$$

where $\Delta = \frac{c_1}{w_1} + \frac{c_2}{w_2} + \frac{c_3}{w_3}$, $\Delta_1 = \frac{-c_1}{w_1^2} + \frac{c_3}{w_3^2}$, $\Delta_2 = \frac{-c_2}{w_2^2} + \frac{c_3}{w_3^2}$, and $w_3 = 1 - w_1 - w_2$, $c_3 = w_3 f_3$. It is easily seen that $[\partial w_1 \quad \partial w_2] \cdot J = [\partial R_1 \quad \partial R_2]$. Hence the Widrow–Hoff iteration is given as follows:

$$[w_1 \quad w_2]_{(k+1)} = [w_1 \quad w_2]_{(k)} - \eta \cdot [R_1(w_1, w_2) \quad R_2(w_1, w_2)]_{(k)} \cdot J_{(k)}^{-1}, \tag{9}$$

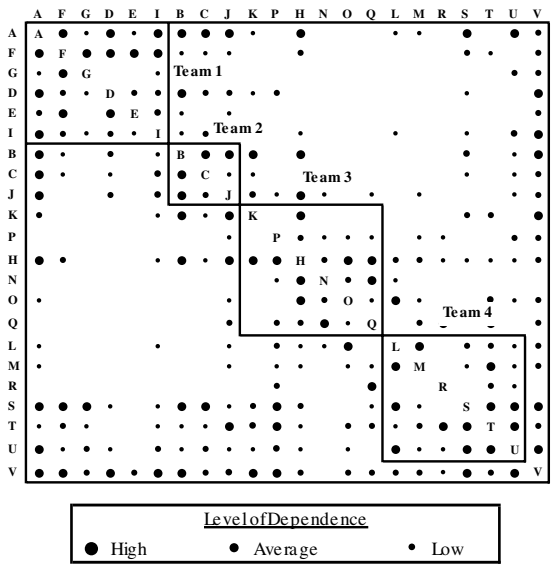
where k is the number of iteration, and η is the learning rate. A higher learning rate enables a faster learning; however, a too high learning rate causes the iteration to diverge. With Eq. 9, we can find the appropriate weights that yield $\frac{f_1}{f_1+f_2+f_3} : \frac{f_2}{f_1+f_2+f_3} : \frac{f_3}{f_1+f_2+f_3} \approx r_1 : r_2 : r_3$. Given a starting point $[w_1 \quad w_2]_{(0)}$, we execute the GA which gives f_1, f_2, f_3. Then we calculate R_1, R_2, and the Jacobian matrix by Eqs. 7 and 8. After that, Eq. 9 tells us the next weights that we should use. The more iterations we execute, the closer we approach the objective ratios (human experts' preference) if the Widrow–Hoff iteration converges.

8 Real-world case studies

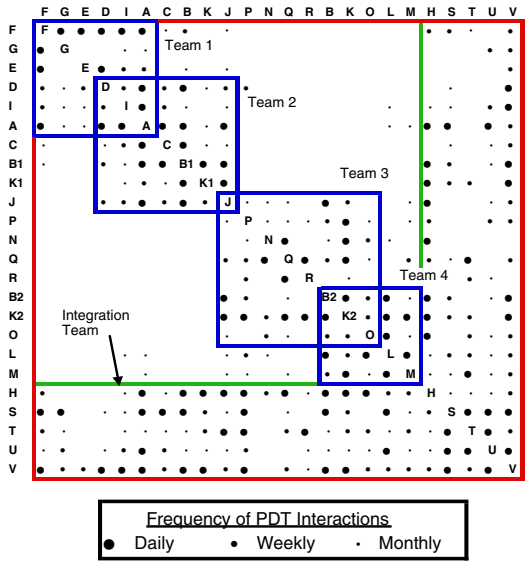
In this section, the proposed DSM clustering algorithm is tested on three real-world DSMs: (1) a DSM for the GM powertrain development teams (GMPT) (McCord and Eppinger 1993; Eppinger 2001), (2) a DSM for a 10 MWe gas turbine (GAS) (Sharman 2002; Sharman and Yassine 2004), and (3) a DSM for a turbofan engine at Pratt and Whitney (P&W) (Rowles 1999, Sosa et al. 2004). A short description of each case is as follows:

8.1 Case 1

A DSM for the communication patterns between 22 engine product development teams at GM (GMPT) was constructed by circulating a survey instrument asking each team about their communication frequency (i.e., daily, weekly or monthly) with other teams. These 22 teams were originally grouped into 4 system teams, named: short block, valve train, induction, system, and electrical systems, as shown in Fig. 12a. This team arrangement was improved by suggesting a different arrangement through manual clustering of the Team DSM as shown in Fig. 12b (McCord and Eppinger 1993; Eppinger 2001). Alternatively, the arrangement proposed by the GA for the GMPT DSM is shown in Fig. 13. Although the manually rearranged DSM (in Fig. 12b) provides a better team structure for this organization (since more of the interactions are now contained within teams and the overall coordination was assigned to a system team), the clustered DSM in Fig. 13 (which was performed using the proposed clustering algorithm) performs comparably well if not better. An elaborate discussion and comparison of both results is explored in Sect. 8.2, next.



(a) Original teaming structure for GMPT DSM



(b) Manually clustered GMPT DSM

Fig. 12 Original and improved clustering arrangements for the GMPT DSM system teams. **a** Original teaming structure for GMPT DSM. **b** Manually clustered GMPT DSM

Fig. 13 Clustering arrangement by the proposed GA for the GMPT DSM

		P	N	Q	R	B	C	J	K	A	D	I	F	G	E	O	L	M	H	S	T	U	V
Fuel System	P	P	1	1	1			1								1		1	2			2	2
Air Cleaner	N	1	N	3												2	1		3				
Throttle Body	Q	2	3	Q	3			2								1		2	2	1	3		2
EVAP	R	2		3	R																2	1	
Cylinder Heads	B					B	3	3	3	3	2	1	1						3	2		1	3
Camshaft/Valve Train	C					3	C	1	1	3	1	2	1							1		1	2
Water Pump/Cooling	J	1	1	1		3	2	J	2	3	2	2						1	3			1	2
Intake Manifold	K					3	1	3	K	2		1							3	2	2		3
Engine Block	A					3	3	3	1	A	3	3	3	1	1		1	1	3	3		3	2
Pistons	D	2				3	2	2	1	3	D	2	2	1	2					1			3
Lubrication	I					1	2	1		3	2	I	2	1	1		1		1	1		2	3
Crankshaft	F					1	1	1		3	3	3	F	3	3				2	2	1		2
Flywheel	G									1		1	3	G								2	2
Connecting Rods	E					1		1		2	3	2	3		E								2
A.I.R.	O		2	1				1		1						O	3	1	3		2	1	2
Exhaust	L	2	1					1		1		1				3	L	3	1	2	2	1	2
E.G.R.	M	1		1				1		1						1	3	M	1	1	3	1	2
Accessory Drive	H	3	2	3	1	3	1	3	3	3		1	2			3	2	2	H	1	1	1	2
Ignition	S	3		1		3	3	1	2	3	1	1	3	3			3	1	2	S	3	3	3
E.C.M.	T	3		2	3	1	1	3	2	1		1	2	1		2	1	2	1	3	T	3	2
Electrical System	U	2			1	1	2	1	1	3	1	2	1	2		1	3	1	1	3	3	U	3
Engine Assembly	V	3		2	1	3	2	2	3	3	3	3	3	2	2	2	2	2	2	3	2	3	V

8.2 Case 2

A DSM for a generic 10 MWe gas turbine driven electrical generator set (GAS) was constructed by decomposing it into 31 sub-systems. The sub-systems initially were listed randomly in the DSM and then tick marks denoting material relationship from one sub-system to another were inserted (Sharman 2002). Figure 14 shows an attempt at manually clustering the DSM (Sharman and Yassine 2004). This took few manual changes to the order of elements in the initial DSM, which revealed the clusters marked in the figure. After inspection of the clusters, they were given names to identify them. The corresponding GA result is shown in Fig. 15.

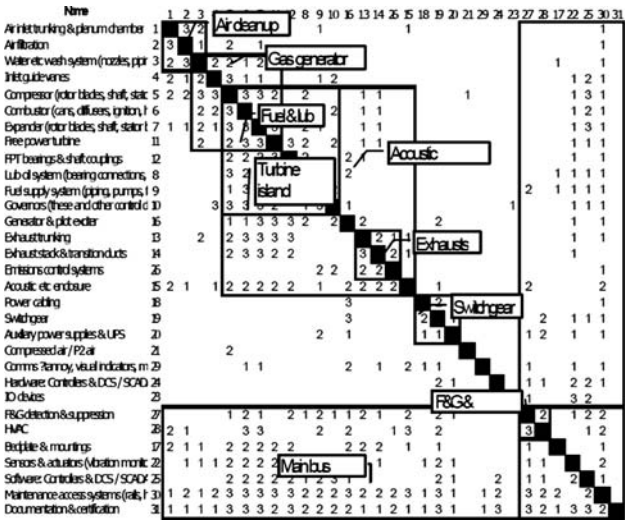


Fig. 14 Manual clustering of the gas turbine with named clusters (The numbers (1, 2, and 3) in the figure represent varying dependency strengths)

8.3 Case 3

A DSM for a commercial high by-pass ratio turbofan engine at Pratt and Whitney (P&W) was constructed by circulating a survey instrument to 60 development teams involved in product development (Rowles 1999; Sosa et al. 2004). The existing product architecture for this engine is shown in the DSM of Fig. 16. The interactions shown in the figure represent weak (e.g., 1) and strong (e.g., 2) levels of dependency between various engine components. The corresponding GA result is shown in Fig. 17.

8.4 Automated clustering using the proposed MDL-GA

By inspecting the three expert-clustered DSMs, GMPT (Fig. 12), GAS (Fig. 14), and P&W (Fig. 16), the average ratios of the description lengths of the model, type-I mismatch, and type-II mismatch is set as the average of (0.0784; 0.8116; 0.1102) as shown in the right of Fig. 18. The maximal number of clusters is set to half of the number of nodes. Hence the length of the chromosomes are $11 \times 22 = 132$, $15 \times 31 = 465$, and $30 \times 60 = 1{,}800$ for the GMPT, GAS, and P&W case studies, respectively. Crossover probability and mutation probability are set to one over the chromosome length. The GA is terminated if there is no improvement in 50 generations. A set of experiments showed that

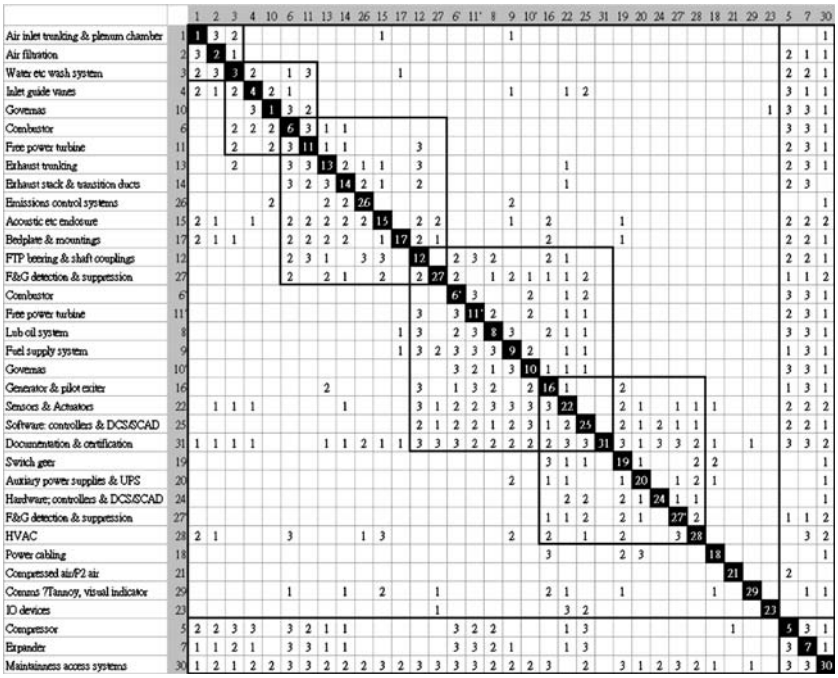


Fig. 15 Clustering arrangement by the proposed GA for the gas turbine DSM

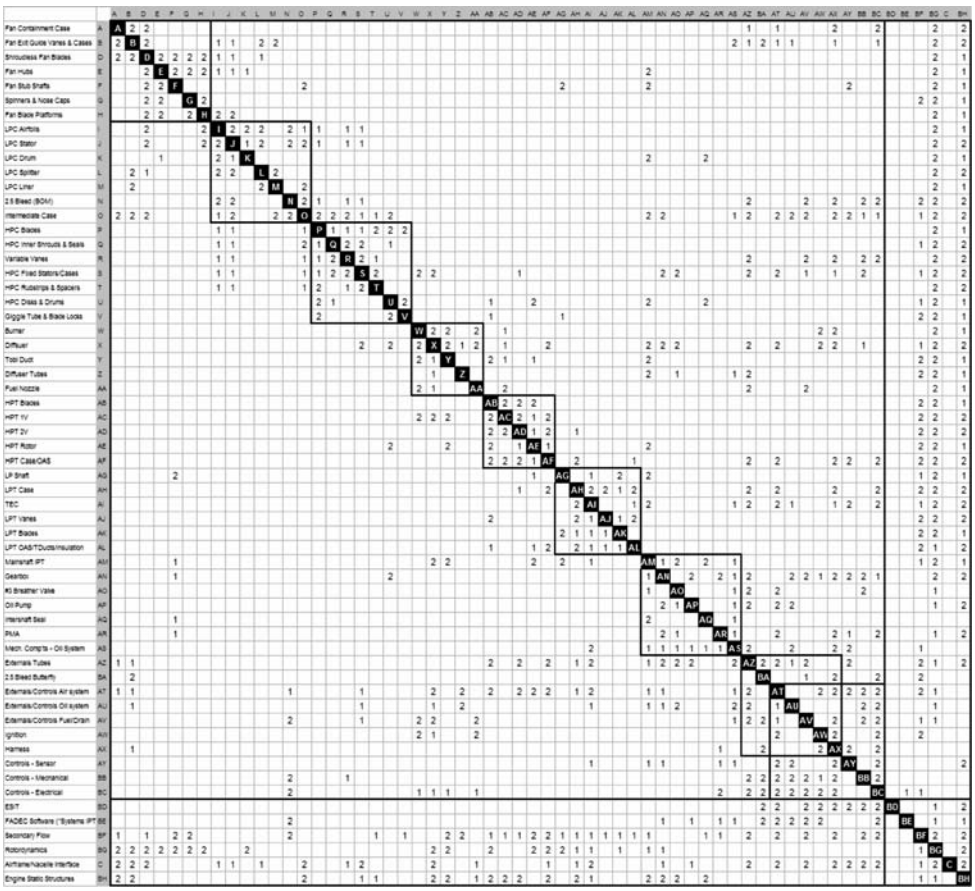


Fig. 16 Clustering arrangement done by human expert for the P&W DSM

(3,000 + 3,000) selection produces satisfactory results for the GMPT case. By assuming the same order of salience of the problem, the population size grows proportional to the square root of the problem size (Harik et al. 1997). Therefore, a (4,250 + 4,250) selection and a (8,500 + 8,500) selection are used for GAS and P&W respectively. Since the

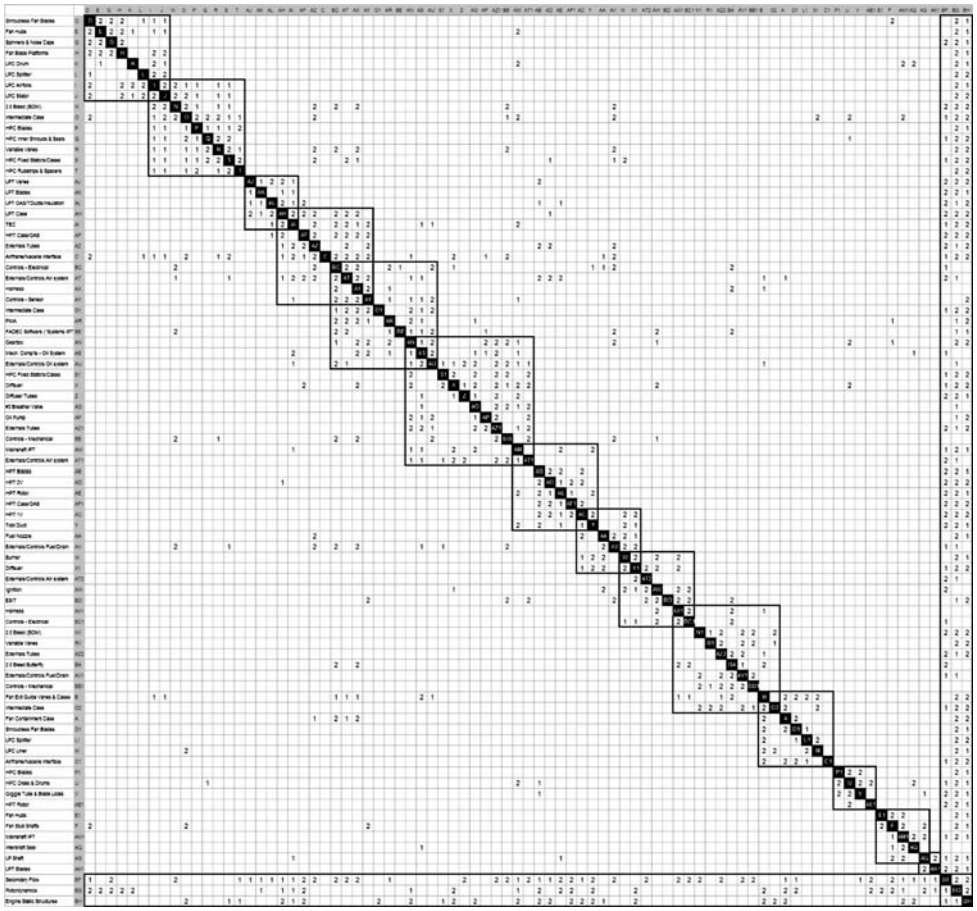


Fig. 17 Clustering arrangement by the proposed GA for the P&W DSM

number of Widrow–Hoff iterations is limited to 10, then after ten iterations, the best run is chosen according to the minimal sum of squared errors ($R_1^2 + R_2^2 + R_3^2$). The objective ratios and the experimental ratios obtained from the ten Widrow–Hoff iterations are shown in Fig. 19 for all three DSMs. Finally, applying the proposed MDL-GA clustering algorithm resulted in Figs. 13, 15 and 17 for the GMPT, GAS, and P&W cases, respectively.[13]

8.5 Discussion of results

Figures 20 and 21 show the clustering arrangements, mismatches, and description length for the three DSMs performed by human experts manually and the proposed MDL-GA algorithm (using average weights from Fig. 18). According to Fig. 21, the proposed MDL-GA clustered both the GMPT and GAS DSMs with less complex models and fewer mismatches. The expert manual clustering for the P&W DSM gave a simpler model than the MDL-GA, but it yields more mismatches. Nevertheless, in all three cases, the MDL-GA gave shorter total description lengths. If the MDL-GA clusters the DSMs using the specific weights tuned for each

		Description length	Ratios
GMPT	Model	160.54	0.0933
	Type 1 mismatch	1467.99	0.8529
	Type 2 mismatch	92.58	0.0538
GAS	Model	262.57	0.0603
	Type 1 mismatch	3897.93	0.8954
	Type 2 mismatch	192.71	0.0443
P&W	Model	443.02	0.0811
	Type 1 mismatch	3747.92	0.6863
	Type 2 mismatch	1269.98	0.2326

Average	
Model	0.0784
Type 1 mismatch	0.8116
Type 2 mismatch	0.1102

Fig. 18 The description length for three different categories and their ratios for clustering arrangement of the DSM of GMPT, GAS, and P&W (Figs. 12, 14 and 16) by human experts. The *right* is the average ratios

[13] The program is written in C++. It takes about a day to obtain results for a (60 × 60) DSM, on an AMD Athlon™ XP 2500+ machine.

	w_1	w_2	w_3	Ratios		
Objective				0.0784	0.8116	0.1102
GMPT	0.5095	0.1543	0.3362	0.0919	0.8128	0.0954
GAS	0.4533	0.1228	0.4239	0.0729	0.8154	0.1117
P&W	0.4730	0.1275	0.3996	0.0888	0.8100	0.1012

Fig. 19 Results of Widrow–Hoff iteration: weights and the ratios of description lengths. The objective is to match the weight ratios (the *right table* of Fig. 18) given by manual clustering

individual case (i.e., according to the weights listed in the left side of the table in Fig. 18), then it will always find a better arrangement (i.e., less complex models and fewer mismatches) in all three cases, as depicted by the results of Fig. 22. In this case, the MDL-GA gave shorter description lengths in all three categories: model, type-I mismatch, and type-II mismatch.

Taking a closer look at the GA results for the GMPT case study, we notice that the bus module (consisting of elements H, S, T, U, and V) detected by the GA is equivalent to that found by the human expert. Furthermore, Team 1 (consisting of elements F, G, E, D, I, and A) and Team 2 (consisting of elements D, I, A, C, B, K, and J) as identified by the human expert were also identified by the GA. Nevertheless, The GA excludes some loosely dependent elements in Teams 3 and 4, and constructs two more solid teams instead. In the GAS care study, only the "air cleanup" group is identical to the human expert's arrangement. The GA result is even more different than the human arrangement in the P&W case study; none of the groups are identical. It is worth noting that the larger the DSM is, the larger the discrepancy between the GA results and the expert's arrangement. We speculate that it is difficult for humans to cluster large DSMs with consistent preference. If we compare the ratios of the model complexity, type-I mismatches, and type-II mismatches for human experts' arrangements (the left of Fig. 18) and the GA results (the right of Fig. 19), we can see that the ratios of GA results are more consistent.

Furthermore, according to Fig. 18 we can see that the magnitude of type 1 mismatch is always larger than type 2 mismatch in all three cases. In other words, human experts tend to endure type-I mismatch (i.e., inside clusters) more than type-II mismatch (i.e., between clusters). This observation is intuitive if we consider that system engineers tend to pay more attention to minimizing interactions between clusters (i.e., product modules or design teams) than interaction patterns inside a cluster. Therefore, an automated algorithm that would mimic human clustering would be biased in the same direction.

These results can be interpreted by two points of view. If the MDL is a more appropriate criterion for the clustering problem, then the GA provides better solutions than humans. On the other hand, if human clustering is more

		n_c	cl_i	$\|S_1\|$	$\|S_2\|$
GMPT	**manual**	5	5, 5, 6, 7, 8	148.00	9.33
	GA	5	3, 4, 5, 6, 7	119.33	14.00
GAS	**manual**	8	2, 3, 3, 3, 6, 7, 8, 13	342.33	17.67
	GA	6	3, 3, 5, 9, 9, 11	233.67	32.00
P&W	**manual**	10	5, 5, 6, 6, 7, 7, 7, 7, 8	634.50	215.00
	GA	15	2, 3, 4, 5, 5, 6, 7, 7, 8, 8, 9, 9, 10, 10, 12	504.50	63.00

Fig. 20 Comparisons of the clustering arrangements given by human experts and the proposed GA using the averaged weights

Description length		Model	Type 1 mismatch	Type 2 mismatch	Total Weighted Description Length
GMPT	**manual**	160.54	1467.99	92.58	1214.21
	GA	133.78	1183.65	138.86	986.44
GAS	**manual**	262.57	3897.93	192.71	3205.38
	GA	227.89	2548.93	349.07	2125.05
P&W	**manual**	443.02	8130.34	2754.96	6936.91
	GA	708.83	6464.55	807.27	5391.12

Fig. 21 Description length and mismatches of the DSM clustering arrangement done by human experts versus by GAs

Description length		Model	Type 1 mismatch	Type 2 mismatch
GMPT	**manual**	160.54	1467.99	92.58
	GA	123.10	1355.58	92.58
GAS	**manual**	262.57	3897.93	192.71
	GA	208.72	3421.60	174.53
P&W	**manual**	443.02	8130.34	2754.96
	GA	324.88	5779.02	2453.85

Fig. 22 Comparisons of the clustering arrangements given by human experts and the proposed GA using the weights according to the respective human clustering arrangement. It is worth noting that the results given by the GA dominate in all three categories

appropriate due to considering several subtle constraints which were not observed by the GA, then the problem is how to "tune" the MDL-metric to mimic human experts' preference. As an initial attempt, we have accomplished that by tuning the weights (w_1, w_2, w_3) according to the method described in Sect. 7 to mimic human experts' preference (Fig. 18). However, our point of view is that the MDL-GA need not provide better results than human expert clustering, but the ability to devise an automated algorithm capable of producing competitive clustering arrangements aligned with human expertise. The proposed method provides a consistent, systematic, and automatic way to cluster DSMs, and the clustering results can be either used directly, or used as an initial clustering arrangement for human experts to tune.

9 Summary and concluding remarks

\This paper started by reviewing the graph/matrix clustering literature related to product architecture and modularity. Then, we presented the MDL concept and used it as a metric for the proposed clustering objective function. The MDL-based objective function was then used with a simple GA to cluster weighted graphs or their corresponding DSMs. Several simplified illustrative examples showed that the proposed MDL-GA is capable of solving complex clustering problems that include overlapping clusters, a bus, and with a 3D structure. Finally, we applied the MDL-GA to three real-world problem. The results were compared to manual clustering to show the promise of automated clustering using GAs, and the parameters can be tuned to mimic expert clustering arrangements.

The DSM is a powerful tool for representing and visualizing product architectures. This representation allows for the analysis and development of modular products by clustering the DSM. Clustering algorithms are scarce and inefficient especially when confronted with complex product architectures as in the reported case studies. The MDL-GA clustering algorithm presented in this paper was capable to identify complex clustering arrangements and overcome many of the difficulties observed in previous clustering tools. Using MDL as a principle to cluster DSMs is more systematic than manual clustering, and the results are promising. However, more research is still required to refine the proposed algorithm. These efforts include an extension to multi-objective clustering, where the values of entries in the DSM represent different types of dependencies between two nodes (Schloegel et al. 1999). Along similar lines, a more explicit representation of domain specific expert knowledge may allow for better tuning of the weights of the model description, and more experiments (i.e., case studies) might be needed to find the real preference of human experts (Nascimento and Eades 2001). Furthermore, the proposed method is capable of identifying buses and overlapped clusters, but other predominant architectural features may also need to be identified and incorporated into the MDL clustering metric. One such example is the concept of a mini-bus or a floating bus discussed by Sharman and Yassine (2004). Finally, our proposed MDL clustering metric could be used in conjunction with other search algorithms for the different needs of clustering speed and quality.

In order to verify that the proposed clustering metric and the associated GA search algorithm are superior in extracting "optimal" modular arrangements from a graph or its matrix representation, we need to compare its performance to exiting clustering methods using known problem instances. However, our proposed approach is unique in many ways and simple, direct comparisons are unavailable. The unique features of our proposed clustering method lie in the following features: (1) it accounts for bus modules, (2) it allows overlapping modules, (3) it is specifically designed to overcome DSM manual/human clustering problems, (4) it has a unique information theoretic clustering measure, (5) it has the tuning capability to mimic human experts' clustering, (6) it allows tuning of GA parameters for specific types of products by human experts (i.e., different parameters/weights for different products), then the tuned algorithm (i.e., particular weights) can be used repeatedly by others (e.g., non-experts) for similar products, or future generations of the same product.

In closing, we believe that the algorithm presented in this paper serves as a strong basis for a series of rigorous mathematical clustering algorithms. This should lead to improved products, processes, and organizational designs. Ironically, the MDL-DSM combination also lead us to investigate the dual problem of using DSM clustering to design more effective genetic algorithms (Yu et al. 2003b).

References

Alexander C (1964) Notes on the synthesis of form. Harvard Press, Boston

Anderberg M (1973) Cluster analysis for applications. Academic, New York

Baldwin C, Clark K (2000) Design rules: the power of modularity. MIT, Cambridge

Barron A, Rissenen J, Yu B (1998) The MDL principle in coding and modeling. IEEE Trans Info Theory 44(6):2743–2760

Broyden CG (1965) A class of methods for solving nonlinear simultaneous equations. Math Comput 19(92):577–593

Bui T, Moon B (1996) Genetic algorithm and graph partitioning. IEEE Trans Comput 45(7):841–855

Eppinger SD (2001) Innovation at the speed of information. Harv Bus Rev 79(1):149–158

Eppinger SD, Whitney DE, Smith RP, Gebala D. A. (1994) A model-based method for organizing tasks in product development. Res Eng Des 6(1):1–13

Fernandez C (1998) Integration analysis of product architecture to support effective team co-location, Master Thesis, Massachusetts Institute of Technology, Cambridge

Gaertler M (2002) Clustering with spectral methods. Masters Thesis, Universitat Konstanz, Germany

Garey M, Johnson D, Stockmeyer L (1976) Some simplified NP-complete graph problems. Theor Comput Sci 1(3):237–267

Gershenson JK, Prasad GJ (1997) Modularity in product design for manufacturing. Int J Agile Manuf 1(1):99–109

Gershenson JK, Zhang Y, Prasad GJ (2003) Product modularity: definitions and benefits. J Eng Des 14(3):295–313

Gershenson JK, Zhang Y, Prasad GJ (2004) Product modularity: measures and design methods. J Eng Des 15(1):33–51

Glover F (1989) Tabu Search – Part I. Operations Research Society of America (ORSA). J Comput 1(3):190–206

Glover F (1990) Tabu Search – Part II. Operations Research Society of America (ORSA) J Comput 2(1):4–32

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, New York

Gonzalez-Zugasti J, Otto K, Baker J (2000) A method for architecting product platforms. Res Eng Des 12(2):61–72

Hansen P, Jaumard B (1997) Cluster analysis and mathematical programming. Math Program 79:191–215

Harik G, Cantú-Paz E, Goldberg DE, Miller BL (1997) The Gambler's ruin problem, genetic algorithms, and the sizing of populations. In: Proceedings of the 1997 IEEE international conference on evolutionary computation, pp 7–12

Hartigan J (1975) Clustering algorithms. Wiley, New York

Jian A, Murty M, Flynn P (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

Jiao J, Simpson TW and Siddique Z (2006) Product family design and platform-based product development: a state-of-the-art review. J Intell Manuf, Special Issue on Product Family Design and Platform-Based Product Development

Kirkpatric S, Gelatt C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680

Kusiak A, Huang C (1996) Development of modular products. IEEE Trans Compon Packag Manuf Technology-Part A 19(4):523–538

Lutz R (2002) Recovering high-level structure of software systems using a minimum description length principle. In: Proceedings of the 13th Irish international conference on artificial intelligence and cognitive science (AICS), pp 61–69

Marshall R, Leaney P, Botterell P (1999) Modular design. Manuf Eng

Martin M, Ishii K (2002) Design for variety: developing standardized and modularized product platform architectures. Res Eng Des 13(4):213–235

McCord K, Eppinger SD (1993) Managing the integration problem in concurrent engineering. Working Paper 3594, MIT Sloan School of Management, Cambridge

McCulley C, Bloebaum CL (1996) A genetic tool for optimal design sequencing in complex engineering systems. Struct Optim 12(2):186–201

Meyer MH, Lehnerd AP (1997) The power of product platforms. The Free Press, New York

Moore G (1999) Crossing the chasm: marketing and selling high-tech products to mainstream customers. HarperCollins Publishers, New York

Nascimento HAD, Eades P (2001) Interactive graph clustering based upon user hints. In: Paper presented at the proceedings of the 2nd international workshop on soft computing applied to software engineering, Enschede

Pahl G, Beitz W (1996) Engineering design: a systematic approach. Springer, New York

Pimmler T, Eppinger, S.D. (1994) Integration analysis of product decompositions. In: Proceedings of the ASME international conference on design theory and methodology, DE-68, pp 343–351

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (1992) §10.6 in numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, Cambridge

Rechtin E, Maier M (1997) The art of systems architecting. CRC Press, Boca Raton

Rissanen J (1978) Modeling by shortest data description. Automatica 14:465–471

Rissanen J (1999) Hypothesis selection and testing by the MDL principle. Comput J 42:260–269

Robertson D, Ulrich K (1998) Planning for product platforms. Sloan Manage Rev 39(4):19–31

Rowles C (1999) System integration analysis of a large commercial aircraft engine. Master Thesis. Massachusetts Institute of Technology. Cambridge

Sanchez R, Mahoney J (1996) Modularity, flexibility, and knowledge management in product and organization design. Strateg Manage J 17:63–76

Schilling MA (2000) Towards a general modular system and its application to inter-firm product modularity. Acad Manage Rev 25(2):312–334

Schloegel K, Karypis G, Kumar V (1999) A new algorithm for multi-objective graph partitioning. In: Proceedings of the European conference on parallel processing (EuroPar '99), pp 322–331

Sharman D (2002) Valuing architecture for strategic purposes. Master Thesis. Massachusetts Institute of Technology, Cambridge

Sharman D, Yassine A (2004) Characterizing complex product architectures. Syst Eng J 7(1):35–60

Sharman D, Yassine A (2006) Architectural valuation and optimization using the dependency structure matrix and real options. Concurr Eng Res Appl (Forthcoming)

Simpson T, Siddique Z, Jiao J (2006) Platform-based product family development. In: Simpson TW, Siddique Z, Jiao J (eds) Product platform and product family design: methods and applications. Springer, New York, pp 1–15

Simpson T, Maier J, Mistree F (2001) Product platform design: method and application. Res Eng Des 13(1):2–22

Sosa M, Eppinger S, Rowles C (2004) The misalignment of product architecture and organizational structure in complex product development. Manage Sci 50(12):1674–1689

Stone R, Wood K, Crawford R (2000) A heuristic method for identifying modules for product architectures. Des Stud 21(1):5–31

Thebeau R (2001) Knowledge management of system interfaces and interactions for product development processes. Master Thesis. Massachusetts Institute of Technology, Cambridge

Ulrich K (1995) The role of product architecture in the manufacturing firm. Res Policy 24(3):419–440

Wang B, Antonsson E (2004) Information measure for modularity in engineering design. In: Proceedings of the ASME 2004 international design engineering technical conferences, DETC-57515

Whitfield R, Smith J, Duffy A (2002) Identifying component modules. In: Proceedings of the 7th international conference on artificial intelligence in design AID'02, pp 571–592

Widrow B, Hoff ME (1960) Adaptive switching circuits. IRE WESCON (New York. Convention Record), vol 4, pp 96–104 (Reprinted in Anderson and Rosenfeld, 1988)

Yassine A, Braha D (2003) Four complex problems in concurrent engineering and the design structure matrix method. Concurr Eng Res Appl 11(3):165–176

Yassine A, Jogleker N, Braha D, Eppingher S, Whitney D (2003) Information hiding in product development: the design Churn effect. Res Eng Des 14(3):145–161

Yu T-L, Yassine A, Goldberg DE (2003a) A genetic algorithm for developing modular product architectures. In: Proceedings of the ASME 2003 international design engineering technical conferences, DTM-48657

Yu T-L, Goldberg DE, Yassine A, Chen Y-P (2003b) Genetic algorithm design inspired by organizational theory: pilot study of a design structure matrix driven genetic algorithm. In: Proceedings of artificial neural networks in engineering 2003 (ANNIE 2003), pp 327–332

Zakarian A, Rushton G (2001) Development of modular electrical systems. IEEE Trans Mechatron 6(4):507–520