AN INFORMATION THEORETIC METHOD FOR DEVELOPING MODULAR ARCHITECTURES USING GENETIC ALGORITHMS

Tian-Li Yu
Ali Yassine
David E. Goldberg

IlliGAL Report No. 2005014
April, 2005

Illinois Genetic Algorithms Laboratory (IlliGAL)
Department of General Engineering
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue, Urbana, IL 61801
`http://www-illigal.ge.uiuc.edu`

AN INFORMATION THEORETIC METHOD FOR DEVELOPING MODULAR ARCHITECTURES USING GENETIC ALGORITHMS

Tian-Li Yu[+], Ali A. Yassine, David E. Goldberg
University of Illinois at Urbana-Champaign
Department of General Engineering, Urbana, IL 61801

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 (if not better) clustering arrangement.

Keywords: Design Structure Matrix (DSM), Product Architecture, Modular, Integral, Genetic Algorithm (GA), Minimum Description Length (MDL).

[+] Corresponding author: Illinois Genetic Algorithm lab, University of Illinois at Urbana-Champaign, Tel. (217) 3332346, email: tianliyu@illigal.ge.uiuc.edu

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). 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. 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 circumstances, it allows increased product or organizational variety (Ulrich and Eppinger 2000), increased rates of technological innovation (Baldwin and Clark 2000), increased opportunities for market dominance through interface capture (Moore 1999), and increased specialization at 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, Stpne 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 2000, 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 design structure matrix - DSM) to describe and visualize a product architecture (Yassine and Braha 2003).[2] Based on this representation, we develop the basic building blocks required for the identification of product 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 (Rissinen 1978 and 1999, Barron et al. 1998, Lutz 2002) and a simple genetic algorithm (GA) (Goldberg 1989). The method is capable of partitioning the product architecture into an "optimal" set of modules or subsystems and can be fine tuned to mimic clustering arrangements proposed by human experts.

The rest of the paper proceeds as follows. The next section provides a brief review of existing clustering methods from graph theory and engineering design literature. Then, in Section 3, we provide a quick 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 Section 4, we propose a metric to evaluate different architectural arrangements based on the MDL principle. Section 5 introduces the genetic algorithm used to search for the optimal product architecture. A method for tuning important model parameters is described in Section 6. In Section 7, we demonstrate this new clustering method using three examples of complex product architectures, and finally, Section 8 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 (Anderberg 1973), which include soft computing

[1] Webster's dictionary defines the noun "module" as "***a compact assembly functioning as a component of a larger unit***".
[2] In a graph, nodes represent product components and edges represent relationships between these components.

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 (Alexander 1964, Rechtin and Maier 1997). 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. In Alexander's (1964) work, the objective function was implemented with a simple hill climbing search strategy. 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.[3] 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/objective, recently, Yu et al. (2003a) and Wang & Antonsson (2004) proposed an information theoretic measure of modularity. These measures were used in conjunction with a genetic algorithm 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 et al. (2002) 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.[4] 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 (Shraman 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

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

al. 2003). The arrows (which represent relationships between components) correspond to the "X" marks inside the matrix.[5] 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). As an example, consider the DSM in Figure 1. As can be seen in Figure 1(b), 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.[6] An alterative arrangement is suggested in Figure 1(c). This arrangement suggests the forming of two overlapping modules (*i.e.*, *AFE* and *EDBCG*).

The DSM representation of a system/product architecture have 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,

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

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 Figure 2(a). It shows five related (*i.e.*, connected) parts or components.[7] 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 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 *SS*" and "Module *TT*".

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 *TT* can be sandwiched between module *SS* (comprising *A*, *B*) and module *UU* (comprising *E*, *F*), or simply described as being comprised of the primitives *A* through *F* with the bus module (comprising *G*, *H*). This example illustrates the notions of

[6] These marks may constitute the interface between the two product modules.
[7] The relationship can be representing a flow of mass, energy, information, or force/geometrical constraint between the parts.

"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 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 they find it hard to extract the relevant information from the data, and then to convey it 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 Figure 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 Figure 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.

3.2.2 Dimensions & Topology

Consider a simple three-dimensional structure such as a tetrahedron or three-dimensional pyramid. This is depicted in Figure 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 objective based on information theoretic measures. 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 three dimensional structures.

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

4.1 Model Description

Suppose we have a model which describes a given product structure or a data set.[8] 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 whole 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, the sender first sends the model (*i.e.*, model description) to the receiver. To ensure that the receiver gets exactly the same data set, the sender is also required to send the data which are mis-described (*i.e.*, mismatched data description) by the model sent earlier. If the model is too simple, the model description is short; but many data mismatches exist and the mismatched data description becomes longer. On the other hand, a complicated model reduces mismatched data, but the model description is longer.

The minimum description length (MDL) principle (Rissinen 1978 and 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).* 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 Encoding

The way we encode 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 7 shows a DSM clustering arrangement and the corresponding model description. It is easily seen that the length of this model description is as follows:

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

where n_c is the number of clusters in the DSM, n_n is the number of nodes, cl_i is the number of nodes in the i^{th} cluster, and the logarithm base is 2. In the example of Figure 7, $n_c = 2$ clusters, n_n =8 nodes, $cl_1 = 3$, 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 $n_c \le n_n$, 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.

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

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: $S_1 = \{(i,j) \mid d_{ij} = 0, d_{ij}' = 1\}$ and $S_2 = \{(i,j) \mid d_{ij} = 1, d_{ij}' = 0\}$. We call the mismatch that contributes to S_1 the *type 1 mismatch*, and the mismatch that contributes to S_2 the *type 2 mismatch*. The mismatched data description length is given by:

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

The first $\log n_n$ in the bracket indicates *i*, the second one indicates *j*, and the additional one bit indicates the type of mismatch.

4.4 MDL Clustering Metric

The MDL clustering metric is given by the weighted summation of the model description length given in Section 6.1 and the mismatched data description given in Section 6.2. With some arithmetic manipulations, the metric can be written as follows:

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

where α and β are weights between 0 and 1.[9] The α and β setting is not a simple task. A naïve setting is $\alpha = \beta = 1/3$. In Section 7, we adjust α and β to mimic the behavior of a manual clustering arrangement.

Finally, 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.

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

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

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 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 is 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.[10] 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 genetic algorithm used in this paper: a genetic algorithm with ($\lambda + \mu$) *selection, uniform crossover*, and *bit-wise mutation*. An outline of how the specialized genetic algorithm uses these operators to search for optimal solutions is described below.

1. <u>Create an initial population of chromosomes</u>. 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.

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

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 Section 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 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 Section 6, the encoding should deal with overlapping clusters and three dimensional structures. As long as the encoding allows that a node belongs to several different clusters, the GA is able to detect overlapping clusters and three dimensional 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 Figure 7, n_n =8, and given c is 3, then the model can be described by the following chromosome shown in Figure 8. When manipulated, the chromosome is transformed into a binary string which is a concatenation of all rows.

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.

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 9 shows the GA results of some illustrative examples.

Figure 9(a) is the simplest case, two non-overlapping clusters. Figure 9(b) demonstrates the ability of identifying overlapping clusters. In Figure 9(c), a bus is introduced, and the genetic algorithm is able to identify it. One thing worth to note is that the DSM in Figure 9(d) resembles the DSM in Figure 9(c); however, the result is totally different. The genetic algorithm recognized the DSM in Figure 9(d) as three overlapping clusters instead of a bus. Figure 9(d) also demonstrates the ability of identifying three dimensional 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 three dimensional 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 for the description length. Recalling that the GA objective is to find model *M* (clustering arrangement) that minimizes $f_{DSM}(M)$ for a given *DSM*. For simplicity, we rewrite Equation 3 as:

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

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 word, we assume that the following Jacobian matrix is close to zero.

$$\begin{bmatrix} \frac{\partial c_1}{\partial w_1} & \frac{\partial c_2}{\partial w_1} \\ \frac{\partial c_1}{\partial w_2} & \frac{\partial c_2}{\partial w_2} \end{bmatrix} \approx O, \tag{6}$$

where O 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:

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

Hence we can calculate the Jacobian matrix as follows:

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

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.

In the next section, several real-world DSMs are investigated to retrieve the human preference of DSM clustering arrangements.

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 1993Eppinger 2001), (2) a DSM for a 10 MWe gas turbine (GAS) (Sharman 2002, Sharman and Yassine 2003), and (3) a DSM for a turbofan engine at Pratt & Whitney (P&W) (Rowles 1999, Sosa et el. 2004). A short description of each case is as follows:

a) 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. This team arrangement was improved by suggesting a different arrangement through manual clustering of the Team DSM as shown in Figure 12 (McCord and Eppinger 1993, Eppinger 2001).

b) 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 2003). 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.

c) Case 3: A DSM for a commercial high by-pass ratio turbofan engine at Pratt & 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 Figure 16. The interactions shown in the figure represent weak (*e.g.*, 1) and strong (*e.g.*, 2) levels of dependency between various engine components.

8.1 Automated Clustering Using the Proposed MDL-GA

By inspecting the three expert-clustered DSMs, GMPT (Figure 12), GAS (Figure 14), and P&W (Figure 16), the average ratios of the description lengths of the model, type 1 mismatch, and type 2 mismatch is set as the average of 0.0784: 0.8116:0.1102 as shown in the right of Figure 10. 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 $30 \times 60 = 1800$ 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 fifty generations. A set of experiments showed that (3000+3000) 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 (4250+4250) selection and a (8500+8500) selection are used for GAS and P&W respectively. Since the number of Widrow-Hoff iterations is limited to 10, then after 10 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 10 Widrow-Hoff iterations are shown in Figure 11 for all three DSMs. Finally, applying the proposed MDL-GA clustering algorithm resulted in Figures 13, 15, and 17 for the GMPT, GAS, and P&W cases, respectively.

8.2 Discussion of Results

Figures 18 and 19 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 Figure 10). According to Figure 19, 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 individual case (*i.e.*, according to the weights listed in the left side of the table in Figure 10), 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 Figure 20. In this case, the MDL-GA gave shorter description lengths in all three categories: model, type 1 mismatch, and type 2 mismatch.

Furthermore, according to Table 10 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 1 mismatch (*i.e.*, inside clusters) more than type 2 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 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 Section 7 (Figure 10). 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 three dimensional 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 and 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 in 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, MA.

Anderberg, M., Cluster Analysis for Applications. Academic Press, new York, 1973.

Baldwin, C. and Clark, K., 2000, Design Rules: The Power of Modularity, MIT Press, Cambridge.

Barron, A., Rissenen, J., and Yu, B., 1998. “The MDL principle in Coding and Modeling,” IEEE Transactions Info. Theory, 44(1998). 2743-2760.

Broyden, C.G., 1965, Mathematics of Computation, 19, pp. 577-593.

Bui, T., and Moon, B. (1996) Genetic Algorithm and Graph Partitioning. *IEEE Transactions on Computers* 45(7) 841-855.

Eppinger, S. (2001) Innovation at the Speed of Information. Harvard Business Review 79(1)149-158.

Eppinger, S. D., Whitney, D. E., Smith, R., D. Gebala (1994) A Model-based Method for Organizing Tasks in Product Development. *Research in Engineering Design* 6(1) 1-13.

Fernandez, C., 1998, Integration Analysis of Product Architecture to Support Effective Team Co-location, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.

Gaertler, M. (2002) Clustering with Spectral Methods. Unpublished Masters Thesis, Universitat Konstanz, Demark.

Garey, M., Johnson, D., and Stockmeyer, L. (1976) Some Simplified NP-complete graph problems. *Theoretical Computer Science* 1, 237-267.

Glover, F. (1989) Tabu Search – Part I. *ORSA Journal of Computing* 1, 190-206.

Glover, F. (1990) Tabu Search – Part II. *ORSA Journal of Computing* 2, 4-32.

Goldberg, D. E., 1989, Genetic Algorithms in Search, optimization, and Machine Learning, Addison-Wesley, New York, NY.

Gonzalez-Zugasti, J., Otto, K. and Baker, J., “A Method for Architecting product Platforms,” Research in Engineering Design (2000) 12:61-72.

Hansen, P., and Jaumard, B., “Cluster Analysis and Mathematical Programming,” Mathematical Programming, 79, 191-215, 1997.

Harik, G., Cant_u-Paz, E., Goldberg, D. E., Miller, B. L. (1997) The gambler's ruin problem, genetic algorithms, and the sizing of populations." Proceedings, 1997 IEEE International Conference on Evolutionary Computation, pp. 7-12.

Hartigan, J., 1975, Clustering Algorithms, Wiley & Sons, New York, NY.

Jian, A., Murty, M., and Flynn, P., "Data Clustering: A Review," ACM Computing Survey, Vol. 31, No. 3, Sept 1999. pp. 264-323.

Kirkpatric, S., Gelatt, C., Vecchi, M. (1983) Optimization by Simulated Annealing. Science 220, 671-680.

Kusiak, A. and Huang, C., 1996, "Development of modular products," IEEE Transactions on Components, Packaging and Manufacturing technology- Part A, 19(4) 523-538.

Lutz, R., 2002, "Recovering High-Level Structure of Software Systems Using a Minimum Description Length Principle," Artificial Intelligence and Cognitive Science, M. O'Neill, R.F.E. Sutcliffe, C. Ryan, M. Eaton, N.J.L. Griffith (Eds.): Proceedings of the 13th Irish International Conference, AICS 2002, Limerick, Ireland, Sept. 12, 02.

Martin, M., Ishii, K. (2002) Design for variety: developing standardized and modularized product platform architectures. Research in Engineering Design 1, 213-235.

McCord, K., Eppinger, S., 1993 "Managing the Integration Problem in Concurrent Engineering," Working Paper 3594, MIT Sloan School of Management, Cambridge, MA.

Meyer, M.H. and Lehnerd, A.P., 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, New York, HarperBusiness.

Nascimento, H. A. D. & Eades, P. (2001) Interactive graph clustering based upon user hints. Paper presented at the Proceedings of the Second International Workshop on Soft Computing Applied to Software Engineering, Enschede, The Netherlands.

Pahl, G. and Beitz, W. (1996) Engineering Design: A Systematic Approach. Springer.

Pimmler, T., Eppinger, S.D., 1994, "Integration Analysis of Product Decompositions," Proceedings, ASME Design Theory and Methodology, DE-Vol. 68.

Press, W. H., Teukolsky, S. A., and Vetterling, W. T., Flannery, B. P., 1992, §10.6 in Numerical Recipes in C: The Art of Scientific Computing, 2nd ed. Cambridge, England: Cambridge University Press.

Rechtin, E. and Maier, M., 1997, The Art of Systems Architecting, Boca Raton, CRC Press.

Rissinen, J., 1978, “Modeling by Shortest Data Description,” Automatica, 14, pp. 465-471.

Rissinen, J., 1999. “Hypothesis Selection and Testing by the MDL Principle,” Computer Journal, 42, 260-269.

Robertson, D. and Ulrich, K., 1998. “Planning for Product Platforms,” Sloan Management Review, 39(4): 19-31.

Rowles, C. (1999) System Integration Analysis of a Large Commercial Aircraft Engine. Masters Thesis, Massachusetts Institute of Technology, SDM program.

Sanchez, R., Mahoney, J., 1996, “Modularity, flexibility, and knowledge management in product and organization design,” Strategic Management Journal, 17, 63-76.

Schilling, 2000.Towards a General Modular System and its Application to Inter-firm Product Modularity,” Academy of Management Review, 25(2) 2000. 312-334.

Schloegel, K., Karypis, G., Kumar, V. (1999) A New Algorithm for Multi-objective Graph partitioning. Proceedings of the European Conference on Parallel Processing, Toulouse, France, August/September 1999.

Sharman, D., 2002, Valuing Architecture for Strategic Purposes, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.

Sharman, D., Yassine, A., Carlile, P., 2002, “Architectural Optimization using real Options Theory and Dependency Structure Matrices,” 28th Design Automation Conference, Montreal, Canada, Sept. 29-Oct. 2, 2002.

Sharman, D., Yassine, A., 2004, “Characterizing Complex Product Architectures,” Systems Engineering Journal, 7(1).

Simpson, T., Maier, J. and Mistree, F., 2001, “Product Platform Design: Method and Application,” Research in Engineering Design 13: 2-22.

Sosa, M., Eppinger, S., Rowles, C. (2004) The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. *Management Science,* 50(12) 1674-1689.

Stone, R., Wood, K. and Crawford, R., 2000, “A Heuristic Method for Identifying Modules for Product Architectures,” Design Studies 21(1):5-31.

Thebeau, R., 2001, Knowledge Management of System Interfaces and Interactions for Product Development Processes, SM thesis, Massachusetts Institute of Technology, Cambridge, MA.

Ulrich, K. and Eppinger, S., 2000, Product Design and Development, McGraw Hill, New York.

Ulrich, K., 1995, "The Role of Product Architecture in the Manufacturing Firm," Research Policy, 24, pp. 419-440.

Wang, B., and Antonsson, E. (2004) Information Measure for Modularity in Engineering Design. Proceedings of the ASME 2004 Design Engineering technical Conferences, Sept. 28-Oct. 2, 2004. Salt lake City, Utah, USA.

Whitfield, R., Smith, J., Duffy, A., 2002, “Identifying Component Modules,” Seventh International Conference on Artificial Intelligence in Design AID’02, Cambridge, UK, 15-17 July 2002.

Widrow, B., Hoff, M. E., 1960, “Adaptive Switching Circuits,” 1960 IRE WESCON Convention Record, 4, pp. 96-104. New York: IRE. Reprinted in Anderson and Rosenfeld, 1988.

Yassine, A., Braha, D. (2003) Four Complex Problems in Concurrent Engineering and the Design Structure Matrix Method, Concurrent Engineering Research & Applications 11(3).

Yassine, A., Jogleker, N., Braha, D., Eppingher, S., Whitney, D. (2003) Information Hiding in Product Development: The Design Churn Effect, *Research in Engineering Design* 14(3), 145-161.

Yu, T., Yassine, A., Goldberg, D. (2003) A Genetic Algorithm for Developing Modular Product Architectures. Proceedings ASME DETC DTM 2003.

Yu, T.-L., Goldberg, D. E., Yassine, A., Chen, Y.-P., 2003b, “Genetic Algorithm Design Inspired by Organizational Theory: Pilot Study of a Design Structure Matrix Driven Genetic Algorithm,” Artificial Neural Networks in Engineering 2003 (ANNIE 2003), pp. 327-332. Also IlliGAL Technical Reports, Publication No. 2003007.

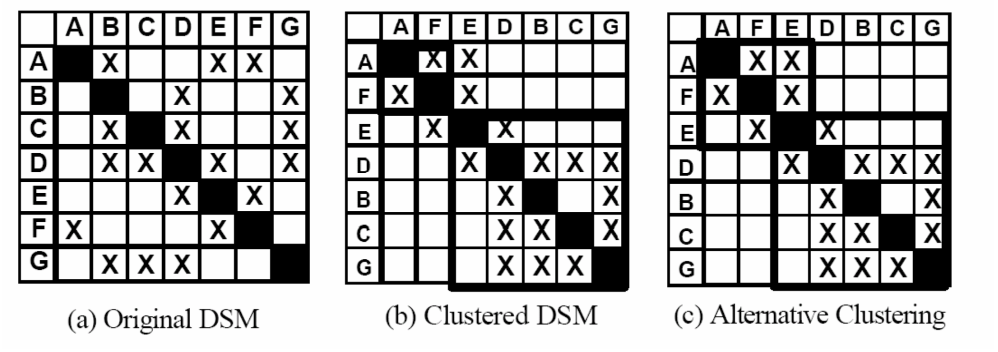
Zakarian, A. and Rushton, G., 2001, “Development of modular electrical systems,” IEEE Transactions on Mechatronics 6(4) 507-520.

List of Figures

Figure 18: Comparisons of the clustering arrangements given by human experts and the proposed GA using the averaged weights.

Figure 19: Description length and mismatches of the DSM clustering arrangement done by human experts versus by GAs.

Figure 20: 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.

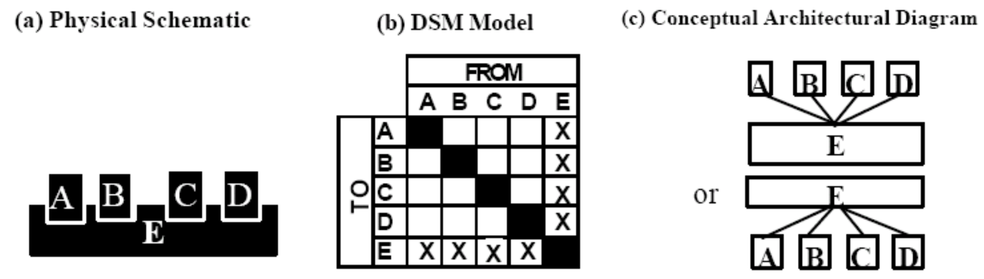

(a) Original DSM
(b) Clustered DSM
(c) Alternative Clustering

	A	B	C	D	E	F	G
A	■	X			X	X	
B		■		X			X
C		X	■	X			X
D		X	X	■	X		X
E				X	■	X	
F	X				X	■	
G		X	X	X			■

	A	F	E	D	B	C	G
A	■	X	X				
F	X	■	X				
E		X	■	X			
D			X	■	X	X	X
B				X	■		X
C				X	X	■	X
G				X	X	X	■

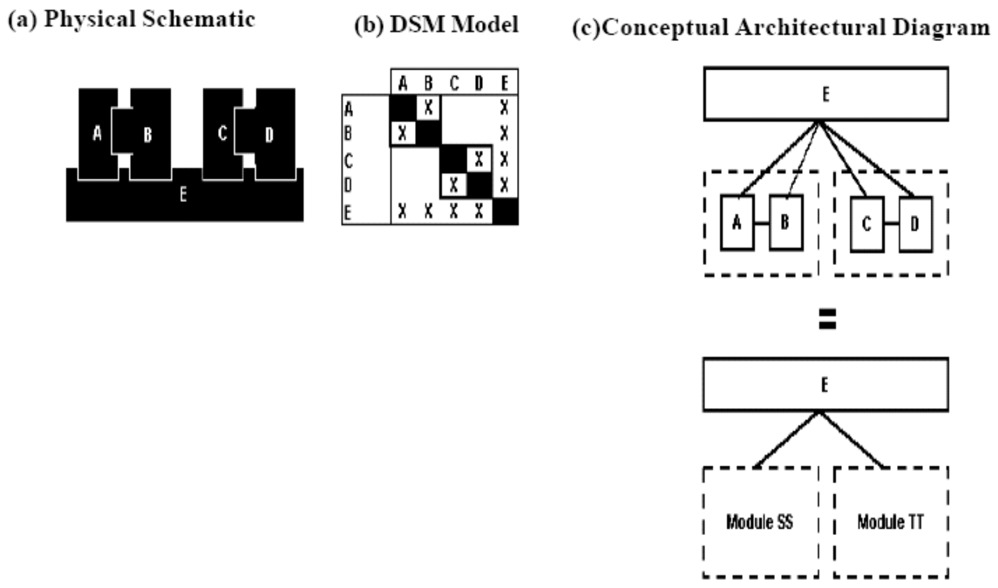
	A	F	E	D	B	C	G
A	■	X	X				
F	X	■	X				
E		X	■	X			
D			X	■	X	X	X
B				X	■		X
C				X	X	■	X
G				X	X	X	■

Figure 1: DSM clustering examples.



		FROM				
		A	B	C	D	E
TO	A	■				X
	B		■			X
	C			■		X
	D				■	X
	E	X	X	X	X	■

Figure 2: A simple bus and no modules.



	A	B	C	D	E
A	■	X			X
B	X	■			X
C			■	X	X
D			X	■	X
E	X	X	X	X	■

Figure 3: A simple bus and two modules.

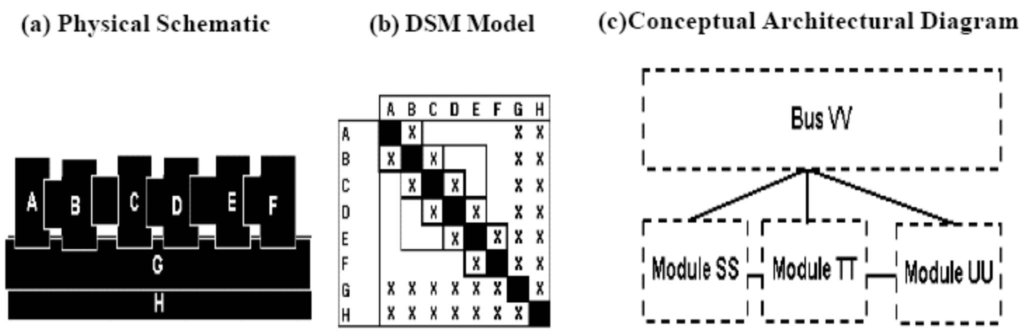


Figure 4: Imperfection, pinning, and holding away.

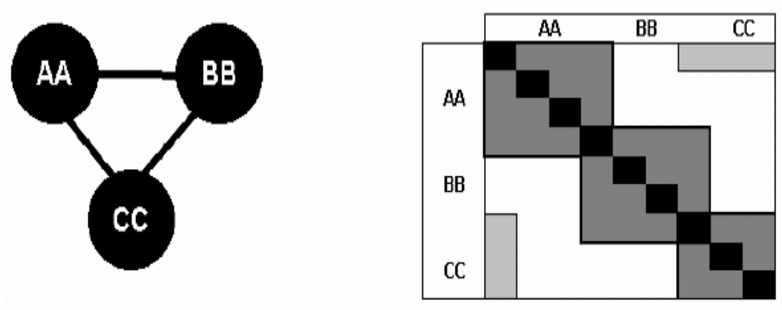


Figure 5: Planar triangular clusters.

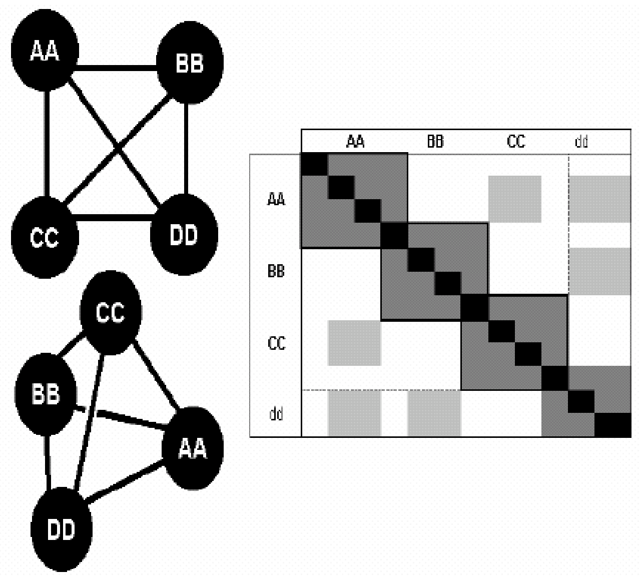
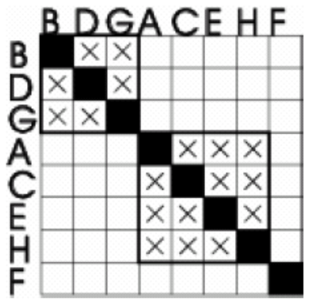


Figure 6: Tetrahedron of clusters.



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

Figure 7: Above is a clustering arrangement of a DSM. Below is the associated model description.

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

Figure 8: A chromosome that represents the model shown in the lower part of Figure 7. The binary string manipulated is 010100101010100100000000.

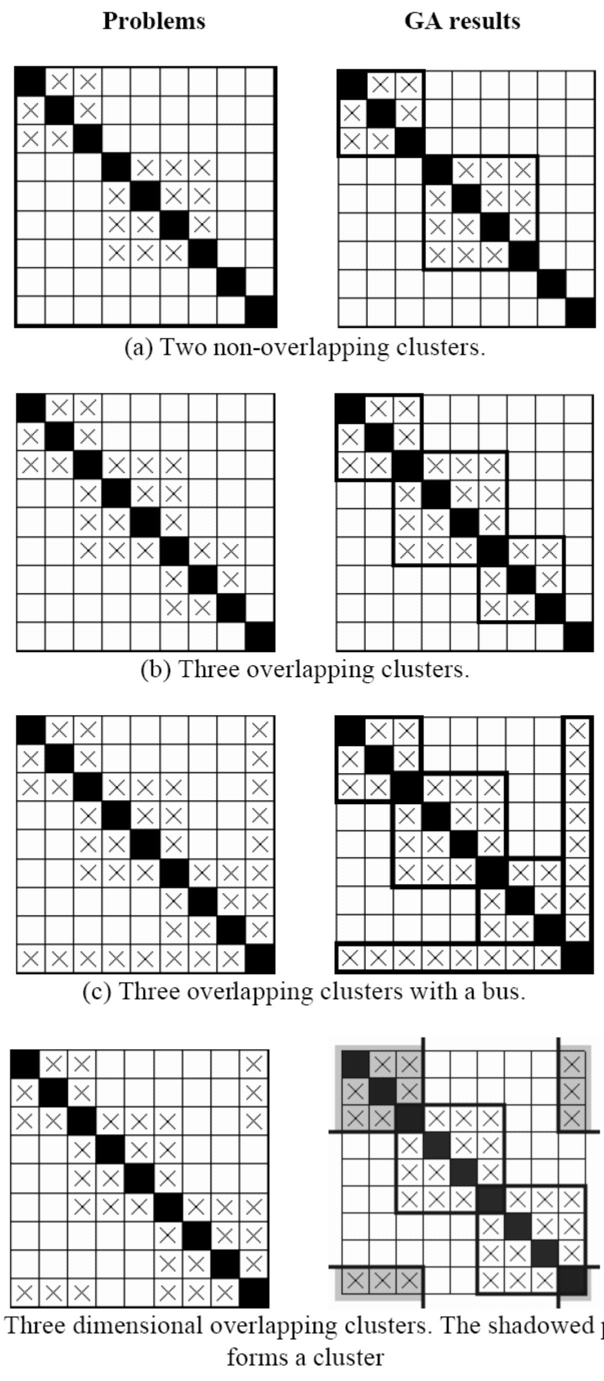


Figure 9: Illustrative examples. The left column is the given DSM, and the right column shows how the GA clusters it.

		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

Figure 10: The description length for three different categories and their ratios for clustering arrangement of the DSM of GMPT, GAS, and P&W (Figure 12, 14, and 16) by human experts. The right is the average ratios.

	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

Figure 11: Results of Widrow-Hoff iteration: weights and the ratios of description lengths

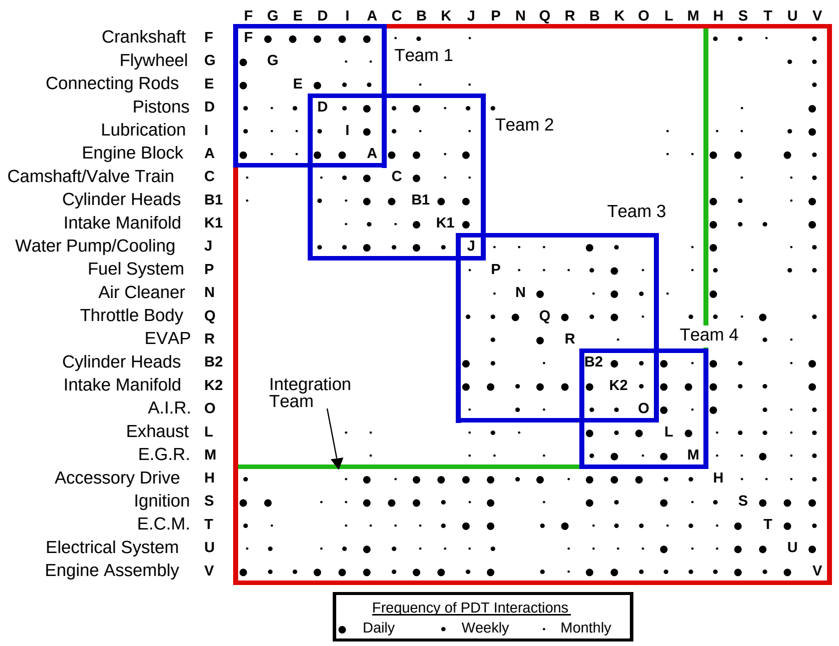
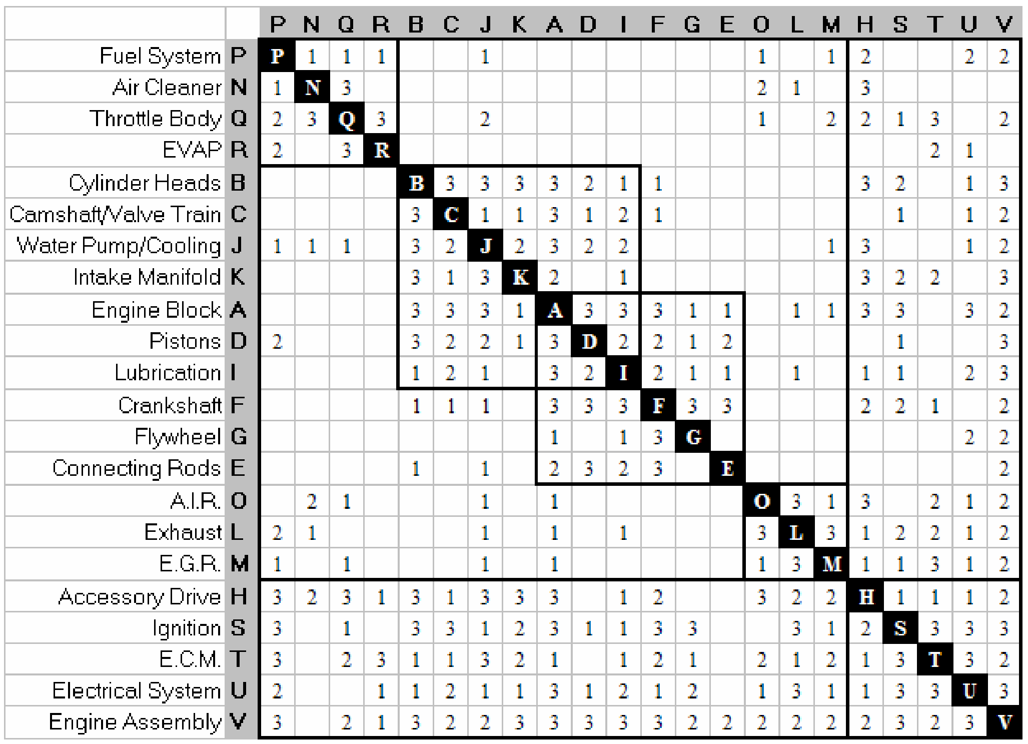


Figure 12: Clustering arrangement by human expert for the GMPT DSM system teams



		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

Figure 13: Clustering arrangement by the proposed GA for the GMPT DSM

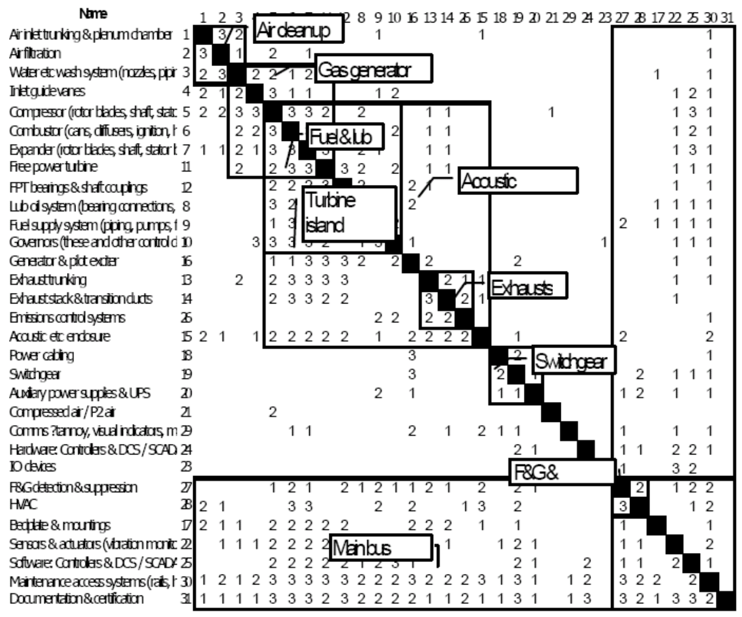
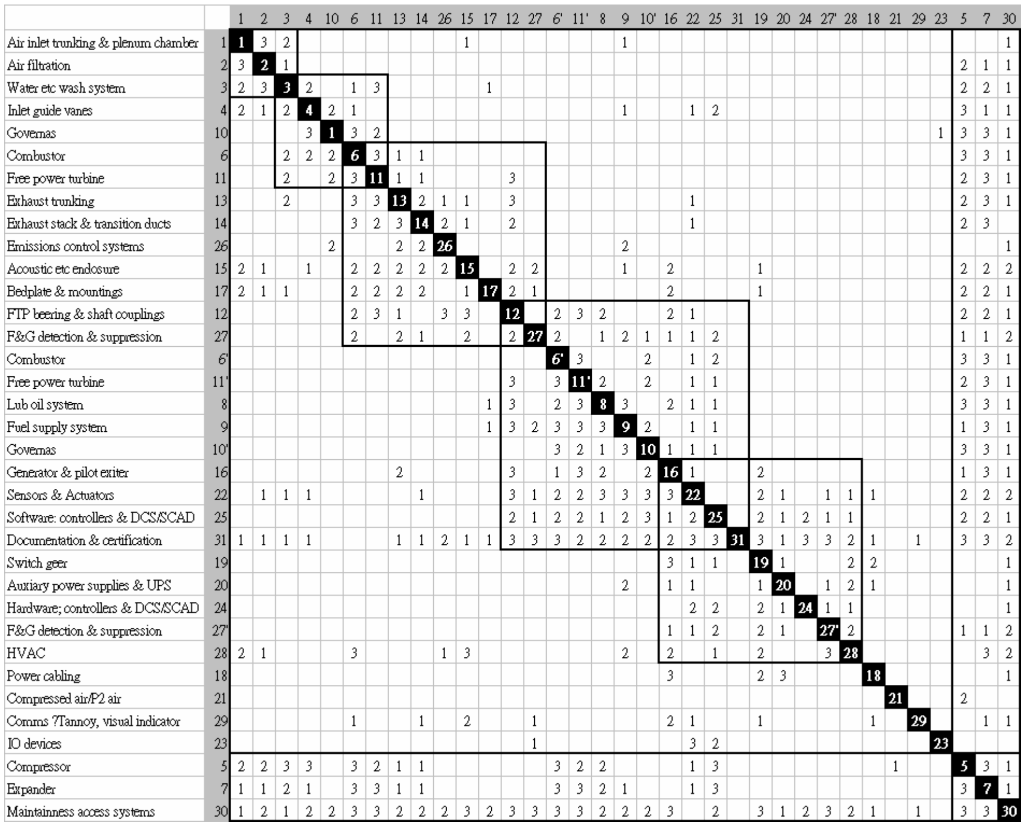


Figure 14: Manual clustering of the gas turbine with named clusters (The numbers (1, 2, and 3) in the figure represent varying dependency strengths)



		1	2	3	4	10	6	11	13	14	26	15	17	12	27	6'	11'	8	9	10'	16	22	25	31	19	20	24	27'	28	18	21	29	23	5	7	30
Air inlet trunking & plenum chamber	1	1	3	2								1							1																	1
Air filtration	2	3	2	1																														2	1	1
Water etc wash system	3	2	3	3	2		1	3					1																					2	2	1
Inlet guide vanes	4	2	1	2	4	2	1												1			1	2											3	1	1
Governas	10				3	1	3	2																									1	3	3	1
Combustor	6			2	2	2	6	3	1	1																								3	3	1
Free power turbine	11			2		2	3	11	1	1				3																				2	3	1
Exhaust trunking	13			2			3	3	13	2	1	1		3								1												2	3	1
Exhaust stack & transition ducts	14						3	2	3	14	2	1		2								1												2	3	
Emissions control systems	26					2			2	2	26								2																	1
Acoustic etc enclosure	15	2	1		1		2	2	2	2	2	15		2	2				1		2				1									2	2	2
Bedplate & mountings	17	2	1	1			2	2	2	2		1	17	2	1						2				1									2	2	1
FTP beering & shaft couplings	12						2	3	1		3	3		12		2	3	2			2	1												2	2	1
F&G detection & suppression	27						2		2	1		2		2	27	2		1	2	1	1	1	2											1	1	2
Combustor	6'															6'	3			2		1	2											3	3	1
Free power turbine	11'													3		3	11'	2		2		1	1											2	3	1
Lub oil system	8												1	3		2	3	8	3		2	1	1											3	3	1
Fuel supply system	9												1	3	2	3	3	3	9	2		1	1											1	3	1
Governas	10'															3	2	1	3	10	1	1	1											3	3	1
Generator & pilot exiter	16								2					3		1	3	2		2	16	1			2									1	3	1
Sensors & Actuators	22		1	1	1					1				3	1	2	2	3	3	3	3	22			2	1		1	1	1				2	2	2
Software: controllers & DCS/SCAD	25													2	1	2	2	1	2	3	1	2	25		2	1	2	1	1					2	2	1
Documentation & certification	31	1	1	1	1				1	1	2	1	1	3	3	3	2	2	2	2	2	3	3	31	3	1	3	3	2	1		1		3	3	2
Switch geer	19																				3	1	1		19	1			2	2						1
Auxiary power supplies & UPS	20																		2		1	1			1	20		1	2	1						1
Hardware; controllers & DCS/SCAD	24																					2	2		2	1	24	1	1							1
F&G detection & suppression	27'																				1	1	2		2	1		27'	2					1	1	2
HVAC	28	2	1				3				1	3							2		2		1		2			3	28						3	2
Power cabling	18																				3				2	3				18						1
Compressed air/P2 air	21																														21			2		
Comms ?Tannoy, visual indicator	29						1			1		2			1						2	1			1					1		29			1	1
IO devices	23														1							3	2										23			
Compressor	5	2	2	3	3		3	2	1	1						3	2	2				1	3								1			5	3	1
Expander	7	1	1	2	1		3	3	1	1						3	3	2	1			1	3											3	7	1
Maintainness access systems	30	1	2	1	2	2	3	3	2	2	2	3	2	3	3	3	3	2	2	2	3		2		3	1	2	3	2	1		1		3	3	30

Figure 15: Clustering arrangement by the proposed GA for the gas turbine DSM

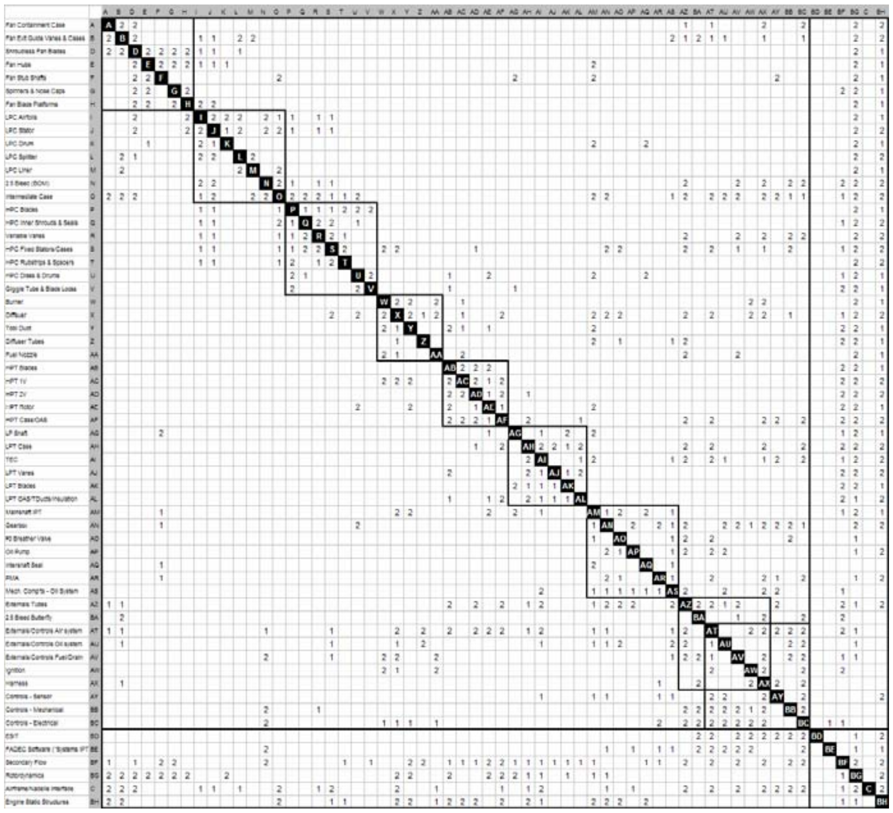


Figure 16: Clustering arrangement done by human expert for the P&W DSM

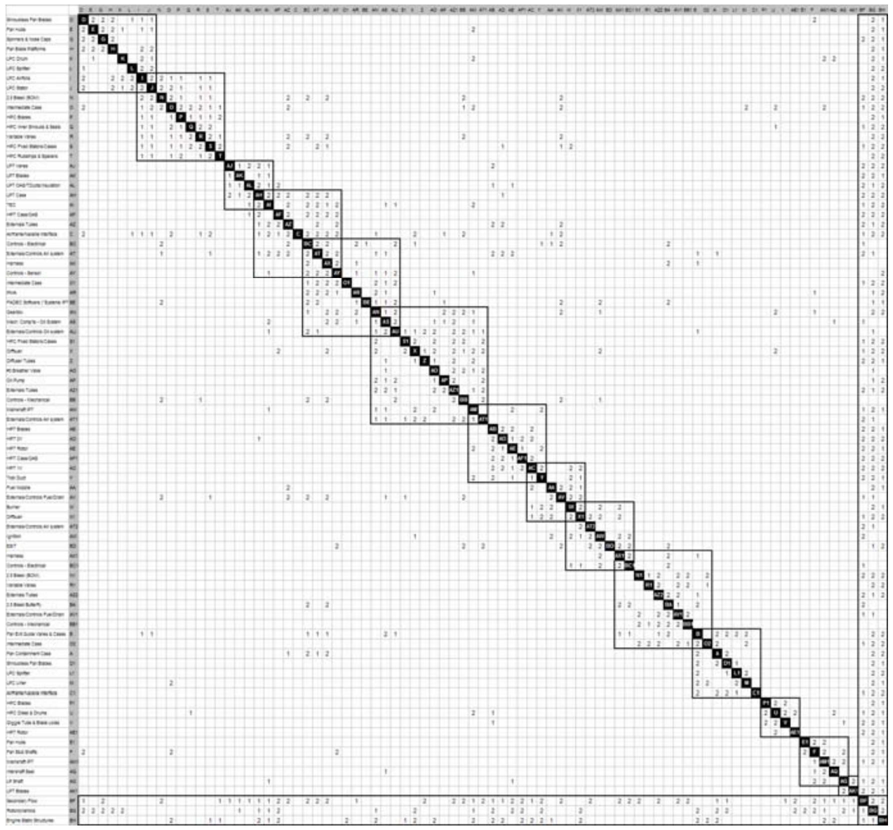


Figure 17: Clustering arrangement by the proposed GA for the P&W DSM

		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, 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

Figure 18: 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

Figure 19: 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

Figure 20: 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.