Modular Design of Mechatronic Systems with Function Modeling

Thom J. van Beek, Mustafa S. Erden, Tetsuo Tomiyama

Intelligent Mechanical Systems Group, BioMechanical Engineering,
Delft University of Technology, the Netherlands

E-mails:
{ t.j.vanbeek, m.s.erden, t.tomiyama}@tudelft.nl

Abstract: This paper develops a modularization scheme based on the functional model of a system. The modularization approach makes use of the function-behavior-state (FBS) model of the system to derive the entity relations. The design structure matrix (DSM) is automatically constructed based on the FBS model. In this way the tedious work of filling the DSM entries based on expert knowledge is avoided. The approach makes use of k-means clustering algorithm to allow the user to try different number of clusters in a fast way. The k-means clustering is adopted for DSM based modularization by defining a proper entity representation, relation measure, and objective function. Two modularization schemes are performed, one based on the immediate relations and one on the deeper behavioral relations between the components. Considering the application on the shifting system of the Delft University of Technology (DUT) Formula Student car, the latter modularization resulted in more mechatronic behavior based modules, while the former resulted in modules based on mere disciplinary and spatial closeness.

Keywords: Modularization, design structure matrix, function-behavior-state modeling, k-means clustering, mechatronics.

1. INTRODUCTION

Modularity provides desirable features for design and development of complex systems. The collaboration of engineers from different domains and integration of different components from various fields are easier with a modular design. Modularity facilitates managing large number of interfaces, which is important for structuring design knowledge, complexity management, upgrading, evolvability, parallel working of teams, and replacement of parts of the system (Whitfield et al., 2002; Baldwin and Clark, 2006; Holmqvist and Persson, 2003). Although integral design might be advantageous from high performance, spatial and material efficiency point of views (Hölttä et al., 2005), the flexibility provided with modular design remains an advantage from technology development, product variations, large scale and multi-scope management point of views. The characteristic of a modular product is identification of separate groups of components within the system in such a way that intra-group relations within the components are maximized and the inter-group relations are minimized (Browning, 2001). Minimizing the component and subsystem dependencies is also in accordance with the famous axiomatic design approach (Suh, 2001).

Mechatronic products are complex systems concurrently related to the disciplines of mechanical, electrical, and computer sciences. Therefore, the designers and developers of mechatronic systems would benefit from modular design. A recent survey demonstrates that companies developing mechatronic products favor “breaking the product up into specific systems, subsystems, assemblies, and components and to allocate requirements to the individual subsystems and components” (Boucher et al., 2008).

A good modularization necessitates performing a modular design process from the very start of the product development. The general steps of a modular design process can be cited as decomposing the system into elements, documenting the interactions between the elements, and grouping the elements into modules (Pimmler and Eppinger, 1994). The challenge in this scheme is the management of a large number and variety of components, as it is the case in typical mechatronic systems. Recording the components one by one, identifying the interactions between the components, and distinguishing the relational groupings is most of the time beyond the capability of a single engineer and even of a group (Holmqvist and Persson, 2003).

In literature there are various approaches for modular design of products (Pimmler and Eppinger, 1994; Fernandez, 1996; Thebeau, 2001; Holmqvist and Persson, 2003; Hung and Kusiak, 1998; Browning, 2001; Whitfield et al., 2002; Xiagong et al., 2006; Chmarra et al. 2008). There are two issues still not addressed in any of these cited work. The information of the relations between the entities is assumed to be given and manually entered into a relational matrix. In reality, this information is not readily available, difficult to extract from any sort of product description and even from the experts and very tedious to manually place in a matrix without making any mistakes. What is missing related to this issue is an automatic derivation of the relations between the entities based on a model of the overall system. Such a model is usually prepared – and in fact very much useful and desirable – during the development of the product. The functional model of the system is very suitable for derivation of the relations between the components (Erden et al. 2008).

The second issue still not solved by the design community is making use of the system knowledge for automatic determination of the modules once the relational matrix is obtained. The current state of art suggests manual determination of the modules based on a diagonalized design structure matrix (DSM) (Whitfield et al., 2002; Browning, 2001; Pimmler and Eppinger, 1994; Hung and Kusiak, 1998; Xiagong et al., 2006; Holmqvist and Persson, 2003). This method necessitates an intensive visual inspection of a diagonalized DSM by the engineer. Moreover, the outcome is quite subjective, because every engineer decides for different modularization looking to the same relational matrix. There have also been attempts to automatically cluster the entities using the DSM. These approaches make use of heuristic clustering algorithms which determine the number of clusters automatically, based on the relation information in the DSM (Fernandez, 1996; Thebeau, 2001, Yu et al., 2003, 2007). However, in these works the size and number of the clusters depend largely on the values of the algorithm parameters. These parameters need to be adjusted by trial and error to get an appropriate level of clustering, a task which is noted to be frustrating by Thebeau (2001).

The contribution of this paper is, on the one hand, developing a modularization scheme based on the functional model information of the system, on the other hand, making use of a conventional clustering algorithm, which allows the user to try different number of clusters in a fast way. The approach addresses aforementioned two issues by making use of the function-behavior-state (FBS) model of the system to derive the entity relations (DSM), and the k-means clustering to group the components into modules based on the DSM. FBS is a particular type of function modeling that relates the subjective level functional descriptions to objective level entities and components through decomposition and instantiation (Umeda et al., 2005, 1995; Tomiyama et al., 1993). An FBS model of the system provides the facility to follow how a function is realized by the structural elements. Therefore the relations between the components of a system are already recorded in the FBS model during the design phase.

The content of the paper is as follows. Section 2 gives a brief review of DSM based modularization attempts and points out the drawbacks of not using the model knowledge. Section 3 gives a brief review of FBS modeling and explains how the model is used to derive information for clustering. Section 4 explains using the k-means clustering algorithm for modularization with the DSM. Section 5 presents embedding the knowledge of modularity back into the FBS model by means of colored graphs. Section 6 presents the application of the method to the case of DUT Formula

Student car designed at Delft University of Technology as an extra-curricular activity for bachelor and master students. Section 7 concludes the paper by mentioning the advantages of the method and points to the future work.

2. MODULARIZATION AND DSM APPLICATIONS

Design Structure Matrix (DSM) is a commonly used method for recording and managing the relations of entities in a complex system. It is a convenient tool for modularization as well. In the following, first the basics of the DSM approach are reviewed. Then the modularization process is explained by elaborating on the contributions of this paper with respect to the conventional DSM modularization approach.

2.1. DSM and Applications in Brief

DSM is first introduced by Steward (1981) to manage the parameter dependencies in the design of a complex system. It is widely used for managing the complexity of components and interfacing in design (Whitfield et al., 2002; Browning, 2001; Pimmler and Eppinger, 1994; Huang and Kusiak, 1998; Xiagong et al., 2006; Holmqvist and Persson, 2003; Baldwin and Clark, 2006; Fernandez, 1998; Thebeau, 2001; Otto and Weck, 2007; Chmarra et al., 2008; Yu et al., 2003, 2007; Sharman and Yassine, 2004; Helmer et al., 2009). A DSM is a relational matrix that constitutes a framework for documenting and evaluation of the interface architecture. The DSM is usually created following the functional decomposition of the system (Thebeau, 2001). DSM can in fact be used in any domain where entities are related with each other on a varying relational basis. For example, it can be used for analyzing the dependencies between marketing, operations management, and engineering decisions for product development (Krishnan and Ulrich, 2001).

Figure 1 (a), shows a sample DSM documenting the relations between six entities. Entity one (*E*1) provides inputs to *E*2 and *E*4, and gets input from *E*4. *E*4 gets inputs from *E*1 and *E*6. The diagonal of the matrix is redundant. The weight of the entries signifies the degree of the relation between two entities. Usually a larger value signifies higher degree of relation. Accordingly, the input relation from *E*6 to *E*3 is higher valued compared to the input relation from E6 to E4.

	*E*1	*E*2	*E*3	*E*4	*E*5	*E*6
*E*1	×	1		1		
*E*2		×				
*E*3			×			1
*E*4	0.5			×		
*E*5					×	
*E*6			1	0.5		×

(a)

	*E*5	*E*4	*E*1	*E*2	*E*3	*E*6
*E*5	×					
*E*4		×	0.5			
*E*1		1	×	1		
*E*2				×		
*E*3					×	1
*E*6		0.5			1	×

(b)

Figure 1: A sample DSM *(a)* and its diagonalized form *(b)*.

For modularizaztion it is usually the case that the relational matrix is “diagonalized” in the sense of bringing the larger weighted relations close to the diagonal. Diagonalization corresponds to finding the optimal permutation of the components that minimizes a cost function. This cost function decreases when the larger weighted relations are placed close to the diagonal. For example the DSM in figure 1(a) can be diagonalized as in figure 1(b). Finding the optimum permutation is an NP-hard problem (there is no proof of the existence of a polynomial order solution algorithm). A complete enumeration of the possible solutions gets computationally too costly for large number of entities. Therefore, some heuristic search methods (genetic algorithms, simulated annealing) are used to find near-optimal solutions.

Diagonalization is obtained by minimizing a cost function which delineates the distance of the entries from the diagonal. Once the diagonalization is performed the engineer can visually determine the modules identifying the groupings in the DSM along the diagonal. In Figure 1(b) the elements {*E*4, *E*1, *E*2}, {*E*3, *E*6}, and {*E*5} constitute three groups that can be named as modules. However, when the size of the matrix is large and the grouping is not clear, visual determination of the modules is difficult and tedious.

Different metrics have been proposed to evaluate the modularity level of a system architecture based on its DSM description. Fernandez (1996) proposed the index of *total coordination cost*. This index separately calculates and adds the coordination cost of intra-group and inter-group relations. Thebeau (2001) also used the total coordination cost and further introduced a *likeness* measure to compare the modularization results obtained by different runs of the algorithm. Yu et al. proposed the index of *minimum description length* (Yu et al., 2003, 2007; Helmer et al., 2009). This index is based on coding the modular structure in a binary string format; the more modular the system, the shorter the string. Whitfield et al. (2002) proposed the *module strength indicator* (*MSI*). This index sums the intra-group and inter-group connections by dividing them with the number of entities inside and outside the group, respectively. In all these indexes, the intra-group connections are rewarded and inter-group connections are punished. Except for the *MSI*, the indexes make use of parameters that should be tuned in advance. The *MSI* has no parameters to tune and produces the same value for a given clustering regardless of the subjective choices. The *MSI* index is simpler in formulation and more intuitive to reveal its purpose. The *singular value modularity index* proposed by Hölttä et al. (2005) evaluates the potential modularity of an architecture by performing a singular value decomposition of the DSM. The index evaluates the overall connection scheme between the components, rather than how they are grouped into modules. It does not differentiate between different modularization of the same component connection scheme.

The *MSI* index is chosen for its simplicity and intuitiveness, compared to the other mentioned indexes (Whitfield et al., 2002). This index enables comparing different modularizations. The *MSI* is based on the internal and external connections of a grouping within the system. The value of the internal connections of the group is denoted by MSI_i, and the external connections by MSI_e, as given in (1).

$$
\begin{aligned}
MSI_i &= \frac{\sum_{i=n_1}^{n_2}\sum_{j=n_1}^{n_2} w_{ij}}{(n_2-n_1+1)^2-(n_2-n_1+1)} \\
MSI_e &= \frac{\sum_{i=0}^{n_1}\sum_{j=n_1}^{n_2}(w_{ij}+w_{ji})}{2\times n_1\times(n_2-n_1)} + \frac{\sum_{i=n_2}^{N}\sum_{j=n_1}^{n_2}(w_{ij}+w_{ji})}{2\times(N-n_2)\times(n_2-n_1)} \\
MSI &= MSI_i - MSI_e
\end{aligned}
\tag{1}
$$

N: The number of elemnts in the DSM

n_1: Index of the first component in the group.

n_2: Index of the last component in the group.

Whitfield et al. (2002), color all the groupings in the DSM based on their *MSI* values. The larger the *MSI* the darker the group color. This helps the engineer to visually distinguish the groupings which have a large *MSI*, hence which grouping is a good candidate to be a module. This approach can be considered only as an aid for determination of the modules, rather than automatic module detection.

Another approach of using the DSM for modularization is using clustering algorithms based on the relational values recorded in the DSM matrix (Fernandez, 1996; Thebeau, 2001; Yu et al., 2003, 2007; Helmer et al., 2009). In this approach there is no need of diagonalization of the DSM, therefore a computationally complex procedure is avoided. Fernandez (1996) and Thebau (2001) applied clustering algorithms for which the number of clusters was not provided in advance. However, these algorithms are still not computationally efficient in comparison to k-means clustering. The number and size of the clusters depend on the values of the parameters set by the user. The algorithm of Thebau (2001), as the author states, produces too many clusters which do not make sense to call a module. The user needs to tune the parameters to get a satisfactory result. All these mean a tedious work to come up with a satisfactory modular architecture; the satisfaction being still a subjective feeling of the designer. Yu et al. (2003, 2007) and Helmer et al. (2009) used genetic algorithms to cluster the components based on the DSM. Their approach is noted to be advantageous over diagonalization based modularizations, as it overcomes path dependency and limitations of two dimensional representation of connections in a DSM. Their algorithm is also capable of detecting overlapping clusters and bus connections. The clustering of 32 components is noted to take around five minutes with an AMD Athlon XP 2000 machine. This rather long computation time is due to the fact that genetic algorithms perform various iterations to reach the optimum solution. The algorithm also necessitates tuning two important parameters that influence the number and size of the resultant clusters. The authors mention that these parameters can be tuned to mimic the modularization preferences of human experts.

In this paper we use the conventional k-means clustering algorithm by adapting it to the problem of component clustering with DSM (Alsabti et al. 1998; Tian et al., 2005). In this way the advantages of clustering over diagonalization based modularization are preserved and the computational efficiency of k-means clustering is utilized to get fast results. The purpose here is to demonstrate the adaptation of a conventional clustering algorithm to the modularization problem, rather than to search for the most efficient clustering algorithm. The adaptation in this paper can be applied also with more developed clustering techniques, such as modified k-means algorithms (Charalampidis, 2005), unsupervised clustering (Montgomery et al., 2005), fuzzy-clustering (Gaths and Geva, 1989), and neural network based clustering (Xu and Wunsch, 2005).

2.2. Modularization using FBS Modeling and K-means Clustering

Modularization generally follows the three steps of d*ecomposition into elements, identification of the relations between the elements*, and *clustering the elements into modules* (Primmler and Epinger, 1994). *Decomposition into elements* corresponds to describing the product usually in terms of functionalities in a hierarchical way. The lowest level functions are associated with the physical elements that realize them. The functional decomposition of a product is usually and preferably constructed in the conceptual design phase. In fact, the decomposition corresponds to building the functional model of the product that will guide the physical implementation. There are various approaches for functional decomposition of systems (Erden et al., 2008). The approach adopted in this paper is FBS modeling, developed by Umeda and Tomiyama (Umeda et al., 1990, 1995a, 1996, 2005; Umeda and Tomiyama, 1995, 1997).

After decomposition, the next step is *identifying the relations* between the elements that realize the lowest level functions. They are recorded into a relational matrix. That is a tedious task for the product architects (Thebeau, 2001). Usually such relational information is gathered by consulting to experts of different domains. The information is collected on paper and then recorded in a matrix form. In this conventional way it is very probable that the architects skip some of the relations and make mistakes while manually filling the relational matrix. Moreover, it is not always clear what weight to assign for a relation between any two components. The approach in this paper is to use the FBS model of the system in order to avoid one by one consultancy, manual filling, and arbitrary weighting of relations. The development of the FBS model already necessitates linking the components based on their behavior relations. The FBS modeling process guides the designer to be

systematic and consistent by demonstrating the behaviors that the elements realize. Integration of function, behavior, and physical entity in the same model makes it less likely that the engineer misses any relation or makes a mistake. Once the FBS model is developed, the links between the components can be used to derive the relations between the entities. This way the entities are related to each other in a weighted way depending on the distance between them in the model. There is no need for extra manual work to identify, document and weight the relations.

Clustering the elements into groups is the last step of a modular design. The elements are clustered into groups based on the weight of their relations. The more related elements are brought together. It is usually the case that the DSM is diagonalized with a near-optimal heuristic algorithm and modularization is performed by visual inspection. The computational cost of the heuristic algorithms is still high. Especially in the cases that the user likes to perform some trial and error by inspecting the results (e.g. using different ranges for the weights), the user needs to wait for minutes for each trial.

The diagonalization approach is computationally costly, because it aims at finding an optimal sequencing of all elements in the relational matrix. In fact, an optimal modular grouping does not necessitate finding an optimal permutation of the elements. The sequence of the elements within the modular groups and the sequence of the modules are irrelevant for the sake of optimal modularity. Therefore, modularization is in fact a clustering problem, rather than a combinatorial optimization problem. The clustering algorithms, which have a lot less computational complexity than heuristic sequential optimization techniques, can be used for this purpose. The approach in this paper is to adopt k-means algorithm for grouping the components based on their weight of relations. K-means clustering algorithm has a linear complexity with respect to the number of elements and number of clusters, namely the computational cost is linearly dependent on the number of entities and clusters (Alsabti et al. 1998; Tian et al., 2005).

Another problem with the DSM based clustering is that the modules are usually determined manually after the DSM is diagonalized. The designer makes a decision of the number and size of the clusters based on a visual inspection of the DSM matrix. There is not yet an effective automated way of determining the modules based on a diagonalized DSM. Visual inspection is tedious and the result is subjective. The approach in this paper is to make use of the knowledge of the designer to provide a range for the number of clusters based on experience or desired level of granularity. The algorithm performs k-means clustering with the numbers in the given range, and suggests the one with the number of clusters that results in the optimal modularization with respect to *MSI*.

Figure 2 shows a schematic description of the steps in the overall modular design approach presented in this paper. The steps related to the modularization are explained next in the following sections. According to this approach the designer starts with developing the FBS model of the system. Developing the FBS model is not a subject of this paper. Preparation of such a functional description model in the conceptual design phase is desirable for an effective design. The FBS conceptual design paradigm is based on this understanding. The modularization approach of this paper should be considered as developing on and a part of this paradigm. The DSM of the system is derived automatically by making use of the component, behavior, and function connections within the FBS model. Then the user determines a range for the number of clusters; the default range covers all possible numbers, namely from one to the number of components. The k-means clustering is applied on the DSM for all the numbers in the given range. The *MSI* index is used to evaluate the modularizations for all these numbers. The one with the minimum *MSI* value is presented to the designer as the best modular structure. Lastly, the FBS diagram is colored based on the resultant modularization.

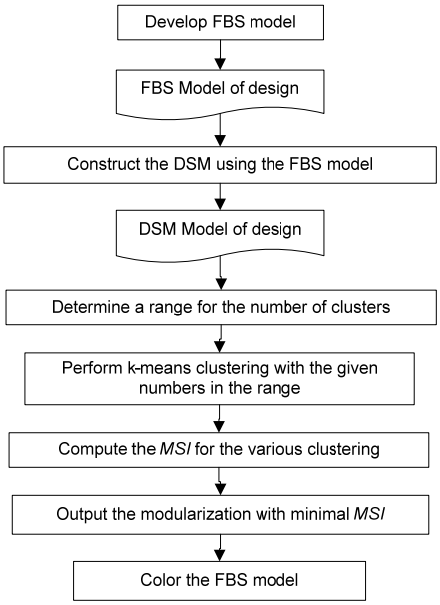


Figure 2: Schematic description of the steps of modularization using the FBS model.

3. AUTOMATED CONSTRUCTION OF THE DSM FROM THE FBS MODEL

FBS is a modeling approach based on the functional and behavioral descriptions of the structural elements (Umeda et al., 1990, 1995a, 1996, 2005; Umeda and Tomiyama, 1995, 1997). The advantage of FBS over the other function modeling techniques is that it associates the functional descriptions with the structural elements via a behavioral level in between (Erden et al., 2008). In this way developing the functional model of the system goes hand in hand with consideration of the real physical world. Moreover, FBS modeling is implemented in the computer environment as FBS Modeler within the Knowledge Intensive Engineering Framework (KIEF) (Yoshioka et al., 2004). This tool supports the designer to develop the FBS model of a system by suggesting functional decompositions, association of lowest level functions with physical structures, and by checking the consistency of the model. Any unrealizable function with the instantiated physical structures is detected and brought to the attention of the user. All these support activities make use of the knowledge base and reasoning algorithms of the FBS Modeler within the system. The study of this paper aims at enriching this FBS Modeler environment by equipping it with an automated modularization algorithm.

The FBS modeler contains two types of knowledge. The first type is about the physical features – namely, physical phenomena (processes), entities, and spatial relationships of entities – corresponding to the knowledge about the objective behavior of the system. The second type of knowledge is about the subjective level functionalities. They are stored in two forms, as decomposition knowledge (how functions are decomposed into sub-functions) and behavioral knowledge (which physical features realize which functions). In designing a product with the FBS Modeler, the designer first defines and decomposes the required functions. Then physical features are instantiated in order to realize the functions. In this paper we use the very fundamental form of the FBS modeling. Many of the concepts – like physical phenomena, physical features, function prototype, attributes of entities, and physical laws – of FBS modeling are not necessary for the modularization purposes. The paper uses only the concepts of *function, behavior, state, entity,* and *relation* as they are defined in the FBS modeling framework:

Function: A subjective description of the behaviors of physical structures. Functions can be considered as a bridge between human intention and physical behavior of artifacts. Functions can be hierarchically decomposed into lower level sub-functions.

Behavior: A behavior is an objective category defined by sequential changes of states of a physical structure over time. This change of states has an influence on the environment and it is perceived as the impact of the behavior.

State: States are the different modes of a physical system or entity. Changes of these modes are the underlying cause of behaviors.

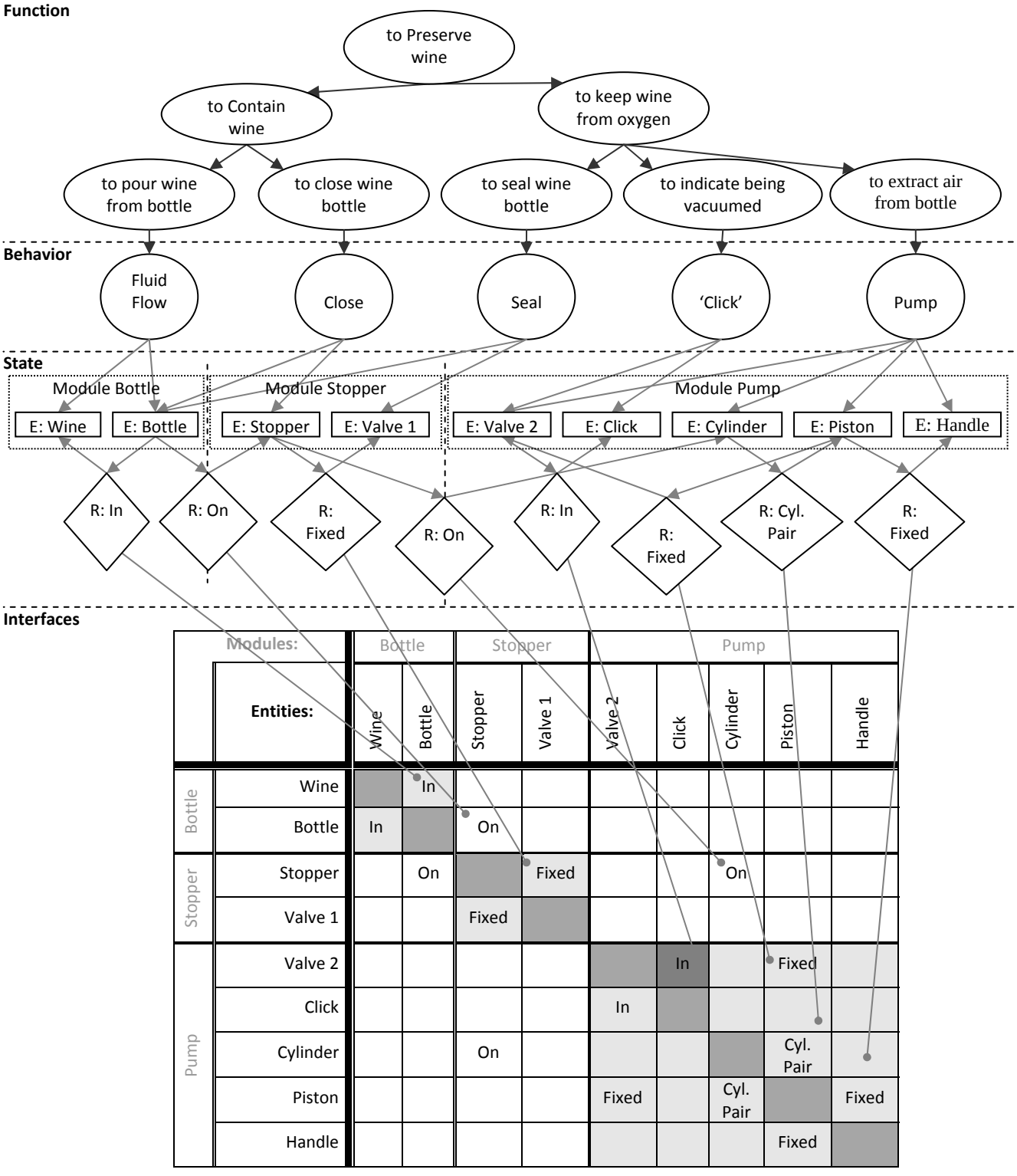
Entity (Component): An atomic physical object that has different states, hence the capability to generate behavior.

Relation: Association between entities that describe preconditions for their state transitions and thus behavior.

In Figure 3, a sample FBS model diagram is given for a Vacuvin Wine Saver. This simple system is chosen to describe the FBS modeling and the modularization to be performed. The figure consists of four layers. In the *functional* layer the functional decomposition of the system is given in a hierarchical way. The lowest level functions need to be associated with physical behaviors. This is performed in the *behavior* layer. The entities in this layer are the physical behaviors that can realize the associated functions. In the third layer the behaviors are associated with physical entities. The behaviors are the result of the change of state of the entities; hence this is the *state* level. It can be the case that a behavior is realized by more than one entity. Therefore there is no one-to-one correspondence between the functions and physical entities. This way of modeling lets disassociation of the functional definitions from the physical entities and allows more intuitive way of modeling in the functional realm. In the state level, the relations between the entities are shown. These relations are required for the entities to be operational within the system and to realize the behaviors. For example, a fluid flow behavior cannot be realized if the wine is not in the bottle. The fourth level shows the relations between the entities in a DSM form. Constructing this matrix, namely determining the weights of the entries of the matrix, by making use of the FBS model is one of the contributions of this paper built on the FBS modeling paradigm.

The DSM matrix gives the relations between the components of the system. The components can be related to each other for various reasons. Spatial placement of the components, energy, information, and material transfer between the components can be such reasons (Pimmler and Eppinger, 1994). These dependencies are delineated in the FBS model as the *relations* between the components in the state level. In the FBS model the relations are named in an intuitive way, such as *in*, *on*, *fixed*, *connected*, *signal*, etc. For simplicity, the modularization in this paper does not make a distinction for the type of relation between the components. However, this is always possible by giving different values to different type of connections.

Two different DSM are derived based on the FBS model. The first DSM is based on the explicit relations in the FBS model; it is named as *DSM_r*. The modularization algorithm directly extracts the relational information from the FBS model, namely it places a *1* in the entry related to two components if they are connected via a relation (direction of the relation is also considered). *DSM_r* is a binary valued matrix, either there is a relation from one component to the other (*1*) or not (*0*). *DSM_r* corresponds to the conventional DSM constructed by consulting the experts and collecting information about the mutual component relations. The difference in the approach here is that the process makes use of an available model instead of consulting the experts; therefore we designate it as automatic. The information, whether it is collected from the experts or automatically derived from the FBS model is usually based on the immediate relation between the components, in the form of spatial placement or disciplinary closeness.



Modules:		Bottle		Stopper		Pump				
	Entities:	Wine	Bottle	Stopper	Valve 1	Valve 2	Click	Cylinder	Piston	Handle
Bottle	Wine		In							
	Bottle	In		On						
Stopper	Stopper		On		Fixed			On		
	Valve 1			Fixed						
Pump	Valve 2						In		Fixed	
	Click					In				
	Cylinder			On					Cyl. Pair	
	Piston					Fixed		Cyl. Pair		Fixed
	Handle								Fixed	

Figure 3: The FBS model of a Vacuvin Wine Saver, and the DSM matrix for the relations of its entities.

The second DSM is constructed based on the connections of the components through the behavioral level of the FBS model; it is named as *DSM_b*. The relational information derived from the behavioral model is more implicit than the one used for the *DSM_r*. The behavior based relations of the components reveal the more indirect dependency of the components which are not visible in the first sight, even to the experts.

The FBS model provides a graphical overview of the function-behavior-structure relations. It is composed of nodes, and edges as described in graph theory (Meng et al., 2007; Witzke and Frese, 1988). To construct the *DSM_b* the knowledge of paths between the components through the behavioral level is required. The number of paths determines the weight of the relation between components. The adjacency matrix of a directed graph gives the number of one-step paths between the nodes (Witzke and Frese, 1988; Agnarsson and Greenlaw, 2007). In deriving the relations the connections of the components to the behavior level of the FBS model is considered (the functional connections and direct relation connections are not used). The adjacency matrix constructed in this way gives the number of one-step paths between all the component and behavior nodes through the behavioral level of the FBS diagram. Taking the *nth* power of the adjacency matrix reveals the number of *n-step paths* between the nodes (Agnarsson and Greenlaw, 2007). A two step path between two components corresponds to a connection of component-behavior-component; a three step path between two components corresponds to a connection of component-behavior-behavior-component. In the application of this paper the connections of at most four steps are considered. Namely, the search for connectedness is stopped after the fourth power of the adjacency matrix. The numbers of two, three, and four step paths are summed after weighting by *3*, *2*, and *1*, respectively. In this way the closer connectedness is given more importance. The sum gives the strength of the connectedness between the nodes.

4. K-MEANS CLUSTERING FOR DSM BASED MODULARIZATION

K-means is the most popular algorithm for clustering a given number of elements (*N*) into a given number of groups (*K*). The algorithm assumes that the elements to be classified form a vector space in which a distance relation can be defined. The relations between the elements are defined in terms of this distance. The *jth* cluster is represented by its center (*Cj*); each element is associated with the cluster of the closest center. The algorithm tries to minimize the total intra-cluster variance by shifting the centers of the clusters in the vector space. In order to apply k-means clustering the elements, vector space, and distance measure should be defined. The steps of the algorithm can be given as follows:

Step 1: Start with k initial random cluster centers

Step 2: Construct the initial clusters by associating each element with the closest center

Step 3: Calculate the new centers of the clusters

Step 4: Construct the new clusters by associating each element with the closest center

Step 5: Repeat Steps 3 and 4 until the centers are no longer changed.

The elements of the clustering in this paper are naturally the components represented in the DSM. However, the representation of the components need some caution. Each line of the DSM in Figure 1 might be considered as a vector representing the relation of the entity to the others. This vector has the dimension of the number of components (number of columns and rows) in the system. The entries which are empty in the DSM matrix can be filled with zeros, in order to delineate that the entity has no relation with the one corresponding to that field. The diagonal entries should also be filled in order to obtain a proper vector representation. The intuitive way is filling the diagonal entries with the largest possible weight, signifying the entity is related to itself in the highest degree.

The term *distance* in the k-means clustering has a different meaning than the term *relation* in the DSM matrix. The conventional application of k-means aims at grouping the elements with the minimum distance under the same cluster. The most common distance measure used with k-means clustering is the Euclidian distance. The application of the k-means in this paper aims at grouping the components that are most related. Let us consider the vector couples representing the entities (components) *E*1-*E*2 and *E*5-*E*2 in figure 1 (a), obtained by filling the empty slots properly:

$$E1 = [\ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ ] \quad E5 = [\ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ ]$$
$$E2 = [\ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ ] \quad E2 = [\ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ ] \tag{2}$$

If the Euclidian distance were applied for the measure the following would be calculated as the distances between the vectors:

$$d(E1,E2) = \sqrt{1+0+0+1+0+0} = \sqrt{2}$$
$$d(E5,E2) = \sqrt{0+1+0+0+1+0} = \sqrt{2} \tag{3}$$

The Euclidian distance measure would indicate that *E*2 is equally distanced from *E*1 and *E*5. However, by inspection one sees that *E*2 has a relation with *E*1. This observation renders the Euclidian distance measure irrelevant for the clustering application with DSM. Instead of a distance measure, what is needed here is a similarity measure (Pedrycz, 2005). Jacard index is a measure of similarity for binary valued vectors. It gives the ratio of the number of common relations to the number of all non-zero relations in both vectors. In (4), the contingency table of two generic vectors and the corresponding *Jacard index* is given (e.g. *a* corresponds to the number of common 1s in both vectors).

$$\begin{array}{|ccc|}\hline & 1 & 0 \\ 1 & a & b \\ 0 & c & d \\ \hline\end{array} \rightarrow Jacard\_index = \frac{a}{a+b+c} \tag{4}$$

In the index the connections of the system which are irrelevant to both of the entities (*d*) are ignored. In this paper the *Jacard index* is generalized to the continuous weight values in the range [0,1] with using the *min*, and *max* operations. We name this modified form *relation index*. The numerator and denominator of the *relation index* are calculated as in (5), for the generic entities *Em* and *En*:

$$a = \sum_{i=1}^{N} \min(Em(i), En(i))$$
$$a+b+c = \sum_{i=1}^{N} \max(Em(i), En(i)) \tag{5}$$

The *relation index* reduces to the *Jacard index* for the binary valued vectors. The *relation index* for the couples *E*1-*E*2 and *E*1-*E*6 is calculated as follows:

$$r(E1,E2) = \frac{0+1+0+0+0+0}{1+1+0+1+0+0} = \frac{1}{3}$$
$$r(E5,E2) = \frac{0+0+0+0+0+0}{0+1+0+0+1+0} = 0 \tag{6}$$

This relation measure says that *E*1 and *E*2 are related to each other to the degree *~0.3*, and *E*5 and *E*2 are not related at all. The result is consistent with the inspection that *E*1 and *E*2 have common components, but *E*5 and *E*2 have nothing in common. This measure gives a value of relation between zero and one for each couple of elements.

Using a relation measure instead of distance measure necessitates changing the objective function of the conventional k-means clustering. The more related two elements in the DSM, the closer they should be in the clusters. The objective should be maximizing the relation of the elements to the cluster centers, as in (7), instead of minimizing the distance. These definitions of the measure of relation and objective function make it possible to apply k-means algorithm for DSM based modularization.

$$\text{maximize} \sum_{j=1}^{K} \sum_{i=1}^{N} r(En, Cj) \tag{7}$$

5. EMBEDDING THE MODULARITY IN THE FBS MODEL THROUGH COLORED GRAPHS

Up to this point the FBS modeling and the modularization based on that modeling is presented; component modularity and the FBS model are not yet integrated. In this section the created knowledge of component modularity is fed back into the FBS model by means of colored graphs. This provides a visual feedback for the roots of the modularization in the functional and behavioral model. Such a visual feedback is useful to follow the correspondence between the functional decomposition and the modularity. This information can be used for organization of the design and development process, such as deciding on the composition of development teams. Visual feedback allows quick understanding of the architecture. A benchmark report on mechatronics system design delineates the change of design and visualization techniques as a direction to improve mechatronic system development (Jackson, 2006). The modules are usually shown on the DSM by drawing rectangles around the entries of the components in a module. The graph coloring introduced in this section is the equivalent of that for the FBS model, to show how functions and behaviors relate to the clustered components. Sharman and Yassine propose three visualization techniques (micro level DSM, intermediate level molecular diagrams, and macro level visibility-dependence signature diagrams) based on DSM (Sharman and Yassine, 2004). Each technique visualizes the architecture on a different level of abstraction. The visualization in this paper differs from theirs as we use not only the components, but also the functions and behaviors in the model.

The modularization process creates new information, namely a mapping of design entities to modules. To visually combine the modularization information with the FBS model, the graph needs to be altered in an intuitive way for human perception. Several options can be considered to associate the modules with the graph of the FBS model. First, the nodes can be annotated with text. This requires a detailed inspection and deep concentration of the viewer in order to associate the nodes with the modules. Second, different shapes can be used for the nodes associated with different modules. In this case it is not clear how to handle the functional and behavioral nodes that are connected to more than one module. Moreover, a graph legend is needed to explain the meaning of the shapes. The third option, which is followed in this paper, is coloring the nodes depending on the degree of connectedness to different modules. Colors are easy to distinguish with a bird's-eye view and provide an intuitive and fast way of grasping the association between the functional decomposition and component modules.

In the node coloring approach, the colors of each function and behavior node should express the weighted sum of the connection of the node to all the modules that take part in its realization. A connection between a function/behavior and a module corresponds to a path from the function/behavior node to any component within the module. The number of such paths determines the weight of the connection between the two. The component nodes belong to one and only one module; therefore they are given a single color representing their module. The color of each function and behavior node is constructed by weighing the color values of the modules by the level of their connection

The number of paths between the function/behavior and component nodes is determined again by using the adjacency matrix. As mentioned in Section 3, taking the *nth* power of the adjacency matrix reveals the number of *n-step paths* between the nodes (Figure 4). At one level, the power of the adjacency matrix results in the null-matrix, designating there are no more paths from the component to the function/behavior nodes. (It should be noted that this only holds for acyclic-directed-graphs.) The coloring scheme follows the direction from the modules towards the functions/behaviors. Summing up all the powers of the adjacency matrix gives the number of paths between the nodes of the graph. The domain mapping matrix (DMM) is used to map the components to the modules (Danilovic and Browning, 2006). The weight of the connections between the modules and function/behavior nodes are determined by multiplying the DMM with the sum of powers of the adjacency matrix.

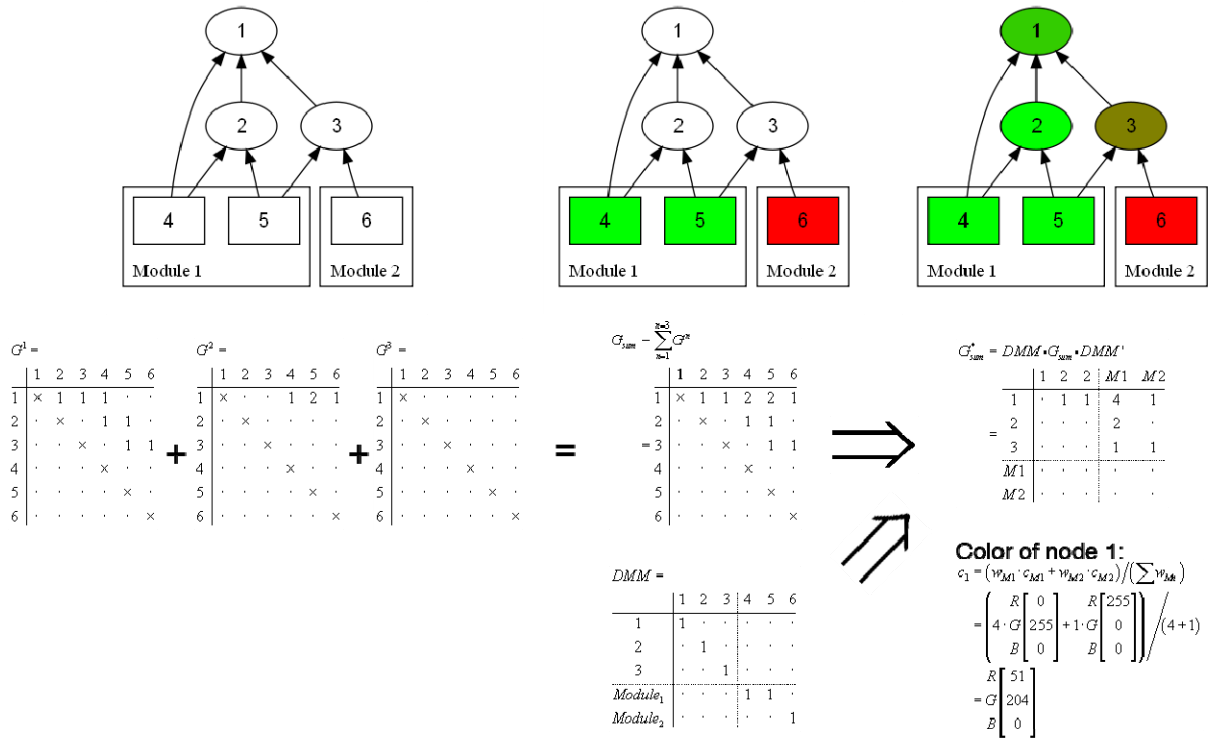


Figure 4: An example of constructing the colors of the function/behavior nodes based on their connection to the component modules. The oval nodes represent the functions/behavior; the rectangular nodes represent the components. *G* represents the one level acyclic-directed-connections from the component to function/behavior nodes. The *DMM* is the domain mapping matrix associating the components with the modules.

Figure 4 describes the process of coloring the FBS model based on the modularization information. A simplified graph of an FBS model is displayed with three oval nodes, representing the functional/behavioral elements, and three rectangular nodes, representing the components. Let's assume that *node 4* and *5* belong to *module 1* and *node 6* belongs to *module 2*. *G* represents the adjacency matrix of the graph, where each arrow in the graph is represented by an entry. To find all the paths between the nodes the consecutive powers of *G* are taken and summed in G_{sum}. A transformation matrix, *DMM*, holds the component-to-module mapping information. In Figure 4 the upper-left part of the DMM is filled with diagonal *1*s to show a one-to-one mapping for the function/behavios nodes, which are not a part of the modules (*nodes 1* to *3*). The component *nodes 4* and *5* are mapped to *module 1*; and the component *node 6* is mapped to *module 2*. Multiplying G_{sum} by the *DMM* and the transpose of *DMM* from left and right, respectively, gives G'_{sum}. This matrix shows the number of paths from *nodes 1, 2,* and *3* to the two modules. For example, *node 1* has four diferent paths to the components in *module 1* and one path to the component in *module 2*. The numbers of paths are used to determine the weights of colors of the nodes. Colors are expressed

in the red-green-blue (RGB) code as three-tuple of values. The color of *node 1* is set to four times the base color of *module 1* (green) plus one time the base color of *module 2* (red). A bird's-eye view reveals that the function of the green node (*node 1*) is mainly realized by *module 1*. Especially for FBS models with a large number of nodes, such coloring makes the association of functions/behaviors with the modules explicit.

6. CASE STUDY: SHIFTING SYSTEM OF FORMULA STUDENT CAR

The Formula Student is a student design and race competition for small formula style, single-seater racecars (http://www.formulastudent.com). The participating teams compete in different areas such as design, manufacturing, management and marketing. The DUT racing team has been taking part in these competitions in Great Britain and Germany since 2001. The DUT team designs and produces a new car each year. Participation in the DUT team is voluntary and open to students of all university departments. This activity provides the students with experience in multi disciplinary design. The DUT team consists of approximately 60 people. The team members are divided into sub-teams to handle various technical, operational, and management tasks. One of the authors of this paper, T.J. van Beek, was an active member of the team in 2002 and 2003, and is a member of the technical advice committee since 2006.

Over the years the DUT team has built up a reputation of lightweight design. Since the 2003 car (DUT'03) the weight of the cars is around 140 kg (Figure 5 (a)). Lightweight design requires many design iterations. For example, if the weight of the engine is reduced, this results that the size of the engine mounts can also be reduced. Such a change has an impact on the overall design; therefore having a modular design is appreciated, not only for team organization, but also to manage such design changes.

(a) *(b)*

Figure 5: One of the DUT Formula Student Cars *(a)* and its shifting system *(b)* [Pictures by Jorrit Lousberg].

The design of the shifting system is chosen as the case study for this paper (Figure 5 (b)). The DUT Cars use single cylinder motorcycle engines that originate from motocross. On the motorcycle the rider shifts gears by using his hand, to operate the clutch, and his foot, for the gear lever. In the DUT Car the driver sits in front of the engine. A system has to be designed to interface the driver and the shifting system. To improve the performance of the car during acceleration, a launch control system is designed. This means that a mechatronic system interfaces the driver and the clutch-gear lever operation. The case study is about how to integrate a 'shift by wire' launch control shifting system in the DUT Car.

In Figure 7, the FBS model of the shifting system is given. The model is reduced as much as possible in order to fit to the margins of the paper. It is clear that the approach of the paper is applicable to any detailed level of the model, because everything that operates on the model is computer based, rather than manual work or visual inspection. The three levels of function,

behavior, and state (structure) are easily distinguished in the model. The two modularizations based on the direct relations (*DSM_r*) and the relations through the behavior level of the model (*DSM_b*) are performed on this case study. The two modularizations are compared.

6.1. Modularization Based on the Direct Entity Relations (*DSM_r*)

In Figure 6, the initial *DSM_r* of the shifting system is given. The entries of this matrix take binary values; either there is a relation (*1*) or no relation (*0*) from one component to another. The diagonal entries are also filled with *1*, designating that a component is related to itself. For the clustering algorithm it is mentioned that the user would provide a range for the number of clusters depending on the desired detail of modeling. In the application here the ranges is given as [1,18], covering all possible number of modules. Since the model used here is quite small, searching for the optimum number of clusters over all possible is no problem. It takes only 0.72 seconds in the MATLAB environment on a conventional laptop computer with 1.86 GHz Intel Pentium-M processor and 1.25 GByte RAM, to perform clustering for 1 to 18 clusters, finding out the number of clusters that result in the minimum value of *MSI*, and showing the modules with the optimum result.

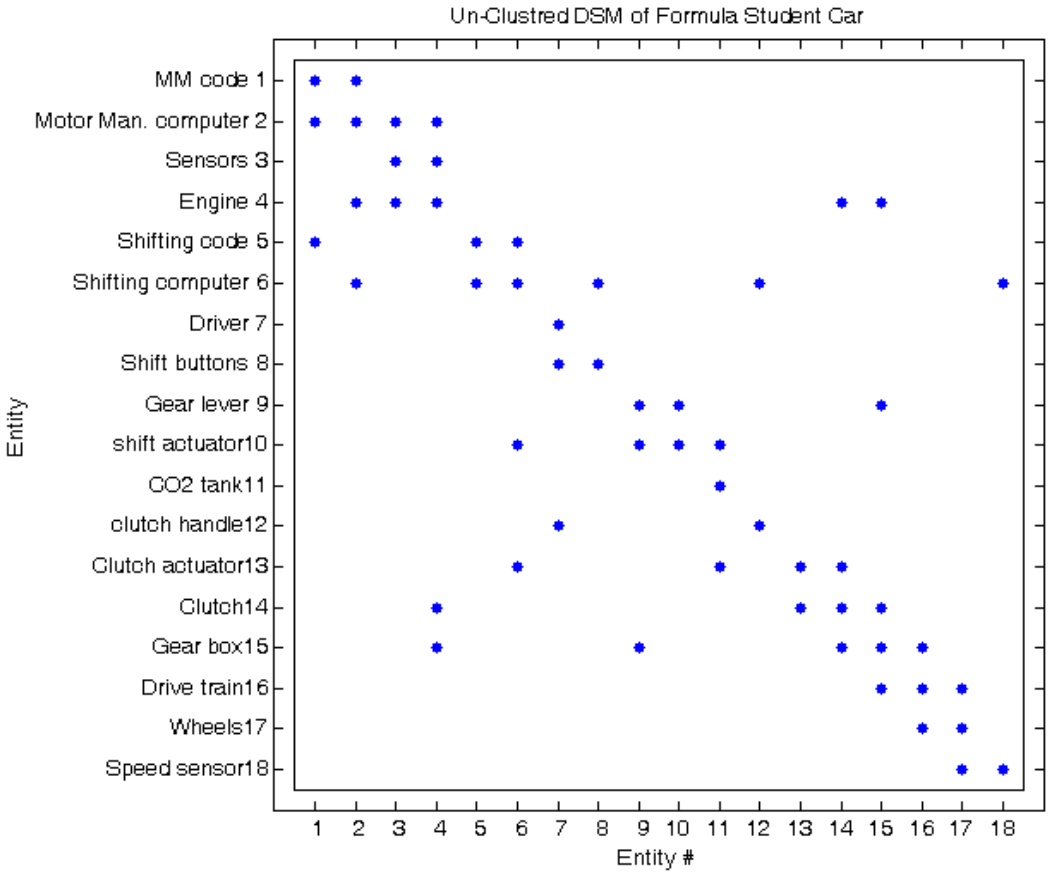


Figure 6: The initial *DSM_r* constructed based on the direct relations between the components.

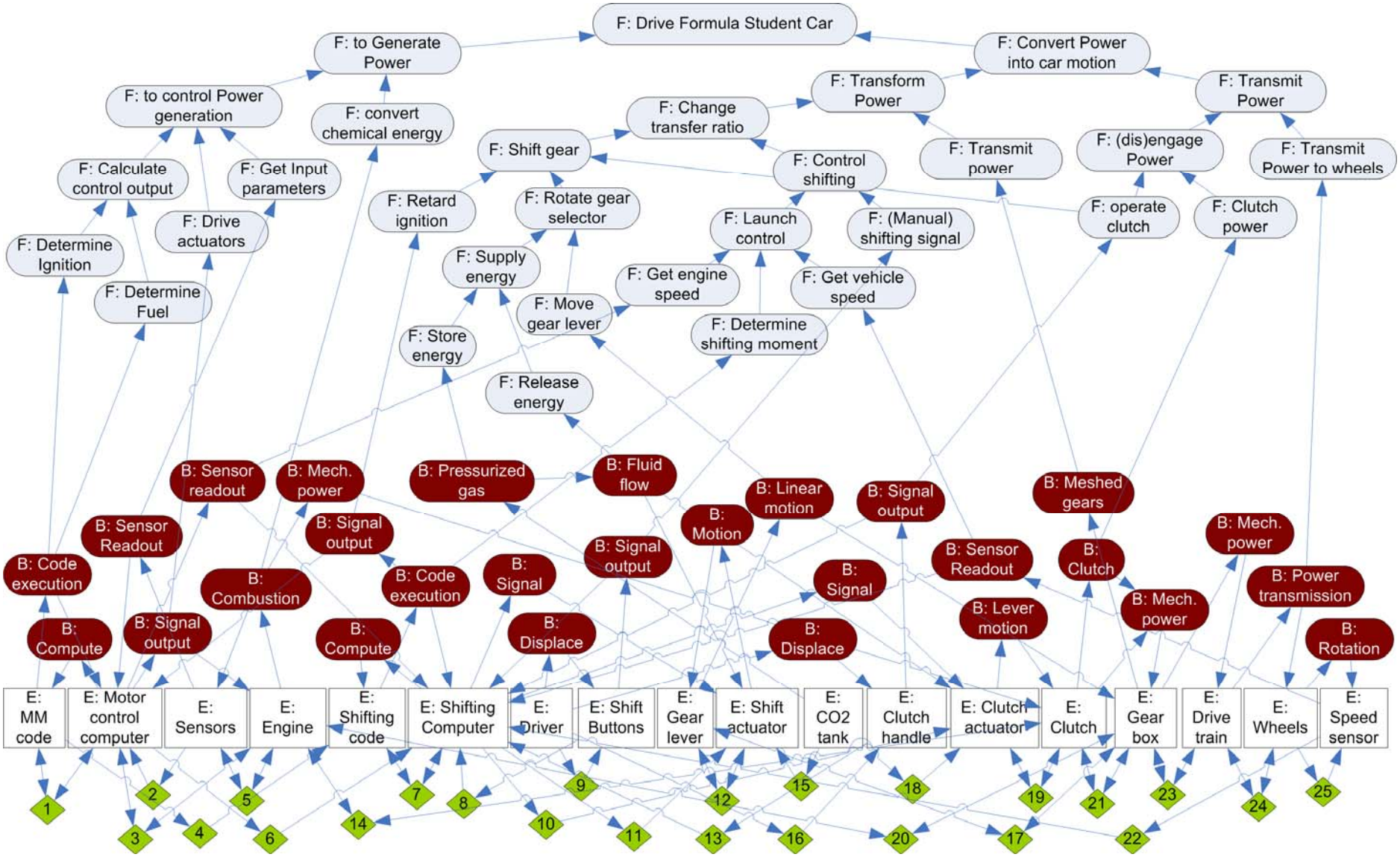


Figure 7: The FBS model of the shifting system of the DUT Formula Student Car. The functions are shown as rounded rectangles, behaviors as red rounded rectangles, components as white rectangles, and the direct relations between the components as green diamonds. The names of the relations are excluded due to limitations on the space; instead numbers are given sequentially.

In Figure 8, the optimal modularization output based on the direct relations is given. The algorithm generated the minimum *MSI* value for the case of three clusters. The corresponding three modules are shown in the figure as three separate squares. Module one contains the entities related to the motor management, shift management and the interface to the driver. The combination of these three is expected from the viewpoint of information flow and is quite mono-disciplinary in that sense. The driver determines what actions should take place; this information is processed by the shifting computer; and the shifting computer and motor management perform the required actions. Module two and three contain mostly mechanical items. Module two contains the items located around the engine; module three contains the drive train components. The modular structure generated in this way reflects the disciplinary and spatial closeness of the components, which are observable at the very first sight. Namely, the immediate relations of the components are based on disciplinary and spatial closeness and ignore the behavioral connections.

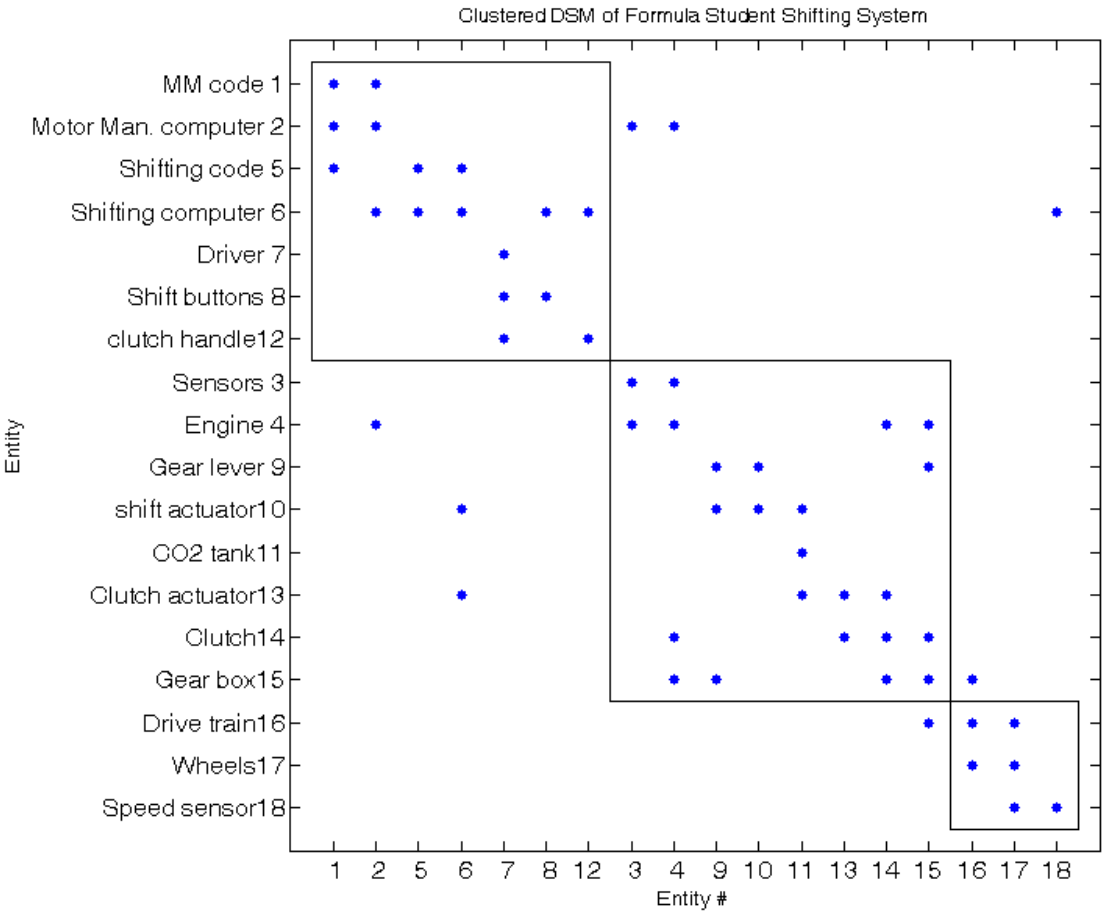


Figure 8: The *DSM_r* after modularization. The three modules that are generated by the clustering algorithm are shown in rectangles.

Figure 9, shows the colored graph of the FBS model of the shifting system after embedding the modularity knowledge on the model. In this graph, the viewer easily distinguishes the functions and behaviors related to each of the modules. The functions and behaviors that are related to more than one module take an intermediary color between the colors of the modules. These intermediary colors provide an intuitive way of following to which modules the nodes are related. It is observed that the three high level functions highlighted in the figure are realized together by the three modules. Therefore, these three high level functions are coupled within this modularization scheme.

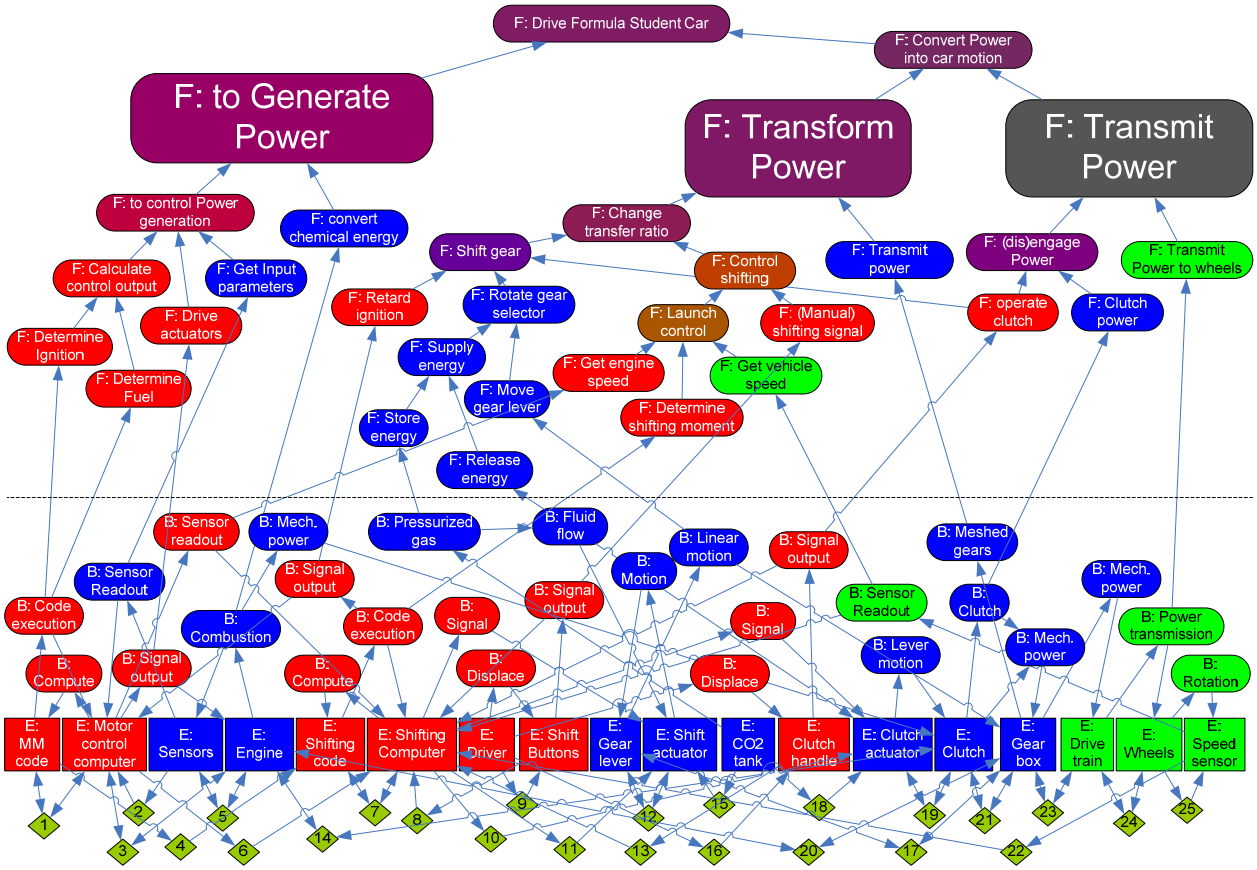


Figure 9: The embedding of the *DSM_r* based clustering on the FBS model by graph coloring.

6.2. Modularization Based on the Entity Relations through the Behavioral Level (*DSM_b*)

In Figure 10, the initial *DSM_b* of the shifting system is given. The entries of this matrix take continues values in the range [0,1]. The values are derived from the behavior level connections, based on the strength of the relation from one component to the other. The diagonal entries are again filled with *1*, designating that a component is related with itself in the highest degree. Most of the high valued relations (0.6) correspond to some direct relations given in the *DSM_r*. Most of the lower valued relations (0.2) provide extra information which is not visible in *DSM_r*. In Figure 11, the optimal modularization output based on the behavior relations is given. The algorithm generated the minimum *MSI* value again for three clusters. The corresponding three modules are shown in the figure in three separate rectangles. One can observe that most of the large valued relations (0.6, 0.4) are situated within the modules and most of the small valued relations are scattered outside the modules.

Module one contains all entities related to the shifting system. It contains electrical, mechanical and computer engineering items that together perform the shifting operation; therefore they can be labeled as a mechatronic whole. Module two contains the mostly mechanical drive train components. In the third module all components related to running the engine are clustered. It is again a functionally coherent mechatronic cluster.

The behavior based clustering reflects the mechatronic character of the shifting system. It does not separate disciplines as in the relation based clustering. This is because the behavior based modularization considers the deeper behavior relations besides the disciplinary and spatial closeness. An organization of the sub-teams based on the behavior based modularization would bring together engineers from different disciplines under the same team dedicated to a part of the system. Figure 12, shows the colored graph of the FBS model after embedding the *DSM_b* based modularity knowledge.

In Figure 12, one can distinguish the functions/behaviors that are related solely to one module, and that are related to more than one. According to the axiomatic design theory of Suh, the design parameters associated with functional requirements should be decoupled (Suh, 2001). This is a consequence of the independence axiom of the theory. The design parameters can be associated with physical parameters, components, or assemblies of components. The modules in Figure 11 can be considered as the component assemblies mentioned in axiomatic design theory. According to axiomatic design we should be able to identify a set of high level functions that almost independently map to these modules. It can be seen Figure 12 that the function “generate power” is solely realized by the green colored module (RGB: 100%, 0%, 0%); the function “transform power” is mostly but not solely realized by the red colored module (RGB: 60%, 10%, 30%); finally the function “transmit power” is mostly realized by the blue colored module (RGB: 33%, 0%, 67%). The nodes of these three functions are highlighted in the figure. In accordance with the axiomatic theory, it is possible to associate the modules to high level functions with a minimized coupling. This is due to the fact that the relations between the components are derived using a functional model, the FBS.

		MM code	Motor Man. computer	Sensors	Engine	Shifting code	Shifting computer	Driver	Shift buttons	Gear lever	shift actuator	CO2 tank	clutch handle	Clutch actuator	Clutch	Gear box	Drive train	Wheels	Speed sensor
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
MM code	1	1	0,6	0,2															
Motor Man. computer	2	0,6	1	0,6	0,2	0,4													
Sensors	3		0,2	1	0,6														
Engine	4	0,2	0,6	0,2	1														
Shifting code	5		0,2			1	0,6		0,2				0,2						0,2
Shifting computer	6	0,2	0,6	0,2		0,6	1	0,4	0,6				0,6					0,2	0,6
Driver	7							1											
Shift buttons	8							0,6	1										
Gear lever	9						0,2			1	0,6								
shift actuator	10		0,2			0,2	0,6		0,2		1	0,4	0,2						0,2
CO2 tank	11											1							
clutch handle	12							0,6					1						
Clutch actuator	13		0,2			0,2	0,6		0,2			0,4	0,2	1					0,2
Clutch	14				0,4		0,2							0,6	1				
Gear box	15									0,6	0,2			0,2	1	1			
Drive train	16									0,2					0,2	0,6	1		
Wheels	17															0,2	0,6	1	
Speed sensor	18																0,2	0,6	1

Figure 10: The initial *DSM_r* constructed based on the relations between the components through the behavioral level.

		Shifting code	Shifting computer	Driver	Shift buttons	Shift actuator	CO2 tank	Clutch handle	Clutch actuator	Gear lever	Clutch	Gear box	Drive train	Wheels	Speed sensor	MM code	Motor Man. computer	Sensors	Engine
		5	6	7	8	10	11	12	13	9	14	15	16	17	18	1	2	3	4
Shifting code	5	1	0.6		0.2			0.2							0.2		0.2		
Shifting computer	6	0.6	1	0.4	0.6			0.6						0.2	0.6	0.2	0.6	0.2	
Driver	7			1															
Shift buttons	8			0.6	1														
Shift actuator	10	0.2	0.6		0.2	1	0.4	0.2							0.2		0.2		
CO2 tank	11						1												
Clutch handle	12			0.6				1											
Clutch actuator	13	0.2	0.6		0.2		0.4	0.2	1						0.2		0.2		
Gear lever	9		0.2			0.6				1									
Clutch	14		0.2						0.6		1								0.4
Gear box	15					0.2			0.2	0.6	1	1							
Drive train	16									0.2	0.2	0.6	1						
Wheels	17											0.2	0.6	1					
Speed sensor	18												0.2	0.6	1				
MM code	1															1	0.6	0.2	
Motor Man. computer	2	0.4														0.6	1	0.6	0.2
Sensors	3																0.2	1	0.6
Engine	4															0.2	0.6	0.2	1

Figure 11: The *DSM_r* after modularization. The three modules that are generated by the clustering algorithm are shown in rectangles.

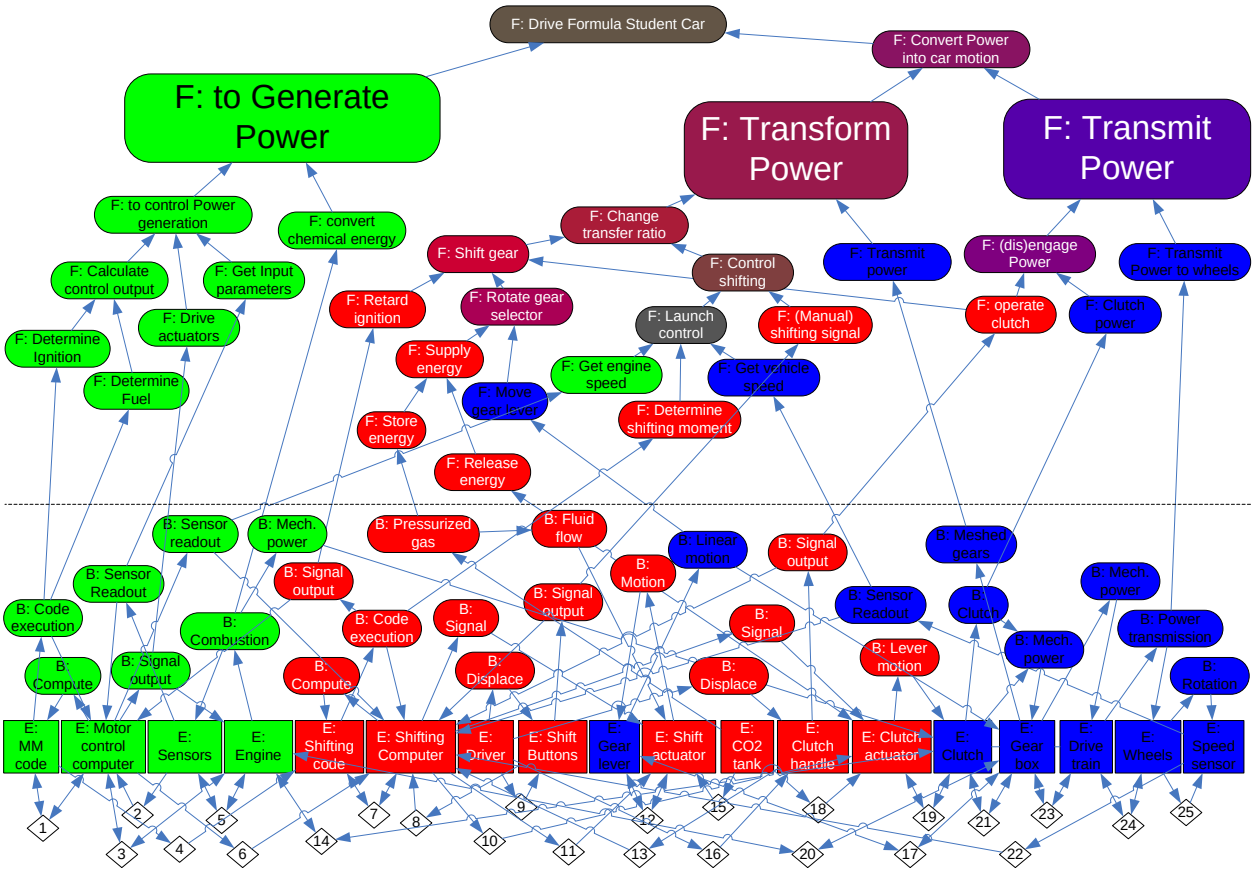


Figure 12: The embedding of the *DSM_b* based clustering on the FBS model by graph coloring.

7. CONCLUSION

This paper dwells upon the modular design of interdisciplinary complex systems, like mechatronics. It integrates the DSM based modularization with the FBS modeling. Integration corresponds to deriving the knowledge of relation between the components from the functional model. This paper assumes that the FBS model is already prepared in the early design phase. Once the FBS model is available, construction of the DSM and modularization is performed automatically. The proposed approach avoids the tedious work of consulting to the experts and manually filling of the DSM matrix. The approach incorporates the k-means clustering algorithm with an appropriate adaptation. The computational efficiency of k-means clustering provides fast modularization compared to the ones based on diagonalization of the DSM matrix or heuristic clustering techniques (genetic algorithms). The contributions of the paper can be cited as follows: 1) Automatic construction of the DSM from the FBS model; 2) relating the components through the deeper functional/behavioral level relations, rather than mere spatial and disciplinary closeness; 3) adapting conventional k-means algorithm for DSM based clustering; 4) embedding the knowledge of modularization onto the FBS model by means of graph coloring.

The DSMs corresponding to the direct relations between the components and the deeper relations through the behavioral level are automatically constructed. The values for the behavior based relations are derived following the paths between the components by making use of operations from the graph theory. The direct relations between the components can perhaps be obtained by consulting the experts, though it is a tedious work, even in the absence of a functional model. However, the relations based on the behaviors provide deeper and more detailed knowledge that is not visible or ignored. Therefore, the behavior based modularization makes use of a deeper knowledge of relations derived from the model. The modularization is performed by using k-means clustering algorithm. Proper vector representation of the entities and the relation measure between the entities are adopted in order to make k-means clustering applicable to DSM based modularization. This is important for fast modularization; it enables examining various number of clusterings in a short time.

The knowledge obtained by modularization is embedded back into the functional model of the system in order to visualize the dependencies of the functions and behaviors on the constructed modules. The visualization algorithm makes use of the operations from graph theory. The color of the function/behavior nodes are determined based on their dependency level on the modules. Embedding the modularity knowledge on the functions and behaviors makes the overall view of the model more intuitive for deciding on the different tasks and constituting different teams to work on the overall system.

The approach is applied to the case of the shifting system of the DUT Formula Student Car. The behavior based modularization of the shifting system produced better results than the modularization based on the direct relations. It was better in the sense of resulting a modularization based on the mechatronically related behaviors, rather than mere disciplinary and spatial closeness. The DUT Formula Student Car, constitutes an example for which division of design teams is of paramount importance. The results of the paper demonstrate that once the modularization is performed based on the deeper functional/behaviour relations, the teams happen to be more “mechatronic”, in the sense that students from different disciplines focus on the same module that perform a set of mechatronic functions. If the components are related based on only the spatial and disciplinary closeness, the teams would be divided according to the disciplines. Mechanical, electrical, and computer science students would be separated. This results in a lack of interaction between the designers of different domains.

The modularization in this paper does consider the degree of relations between the components, but not the type. The future work that evolves on this study will explore the impact of considering different type of the connections between the components. Which types are more important to construct modules, to decide on the sub-teams of design, to separate functional/behavioral

groupings based on the modules will be some of the questions to be answered. The performance of the overall approach will be tested on a more detailed FBS model with deeper interdependencies between the components. Moreover, the algorithms developed here will be integrated with the FBS modeling framework of KIEF, and implemented as a modularization tool within that framework.

8. ACKNOWLEDGEMENT

This work has been carried out as a part of the DARWIN project at Philips Healthcare under the responsibilities of the Embedded Systems Institute. This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

9. REFERENCES

[1] Alsabti, K., Ranka, S., and Singh, V. (1998), "An efficient k-means clustering algorithm". In Proc. of. the IEEE First Workshop on High-Performance Data Mining, Orlondo, FL.

[2] Baldwin, C.Y. and Clark, K.B., (2006), "Modularity in the design of complex engineering systems". In *Understanding Complex Systems*, Springer Berlin, Heidelberg, pp. 175-205.

[3] Boucher, M. and Houlihan, D., (2008), "System Design: New Product Development for Mechatronics". Publication of *Aberdeen Group*, Boston.

[4] Browning, T.R., (2001), "Applying the design structure matrix to system decomposition and integration problems: A review and new directions". *IEEE Transactions on Engineering Management*, 48 (3): 292-306.

[5] Charalampidis, D., (2005), "A Modified K-means Algorithm for Circular Invariant Clustering". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27 (12): 1856-1865

[6] Chmarra, M.K. and Arts, L., and Tomiyama, T, (2008), "Towards adaptable architecture". In *Proc. of the ASME 2008 Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, IDETC/CIE 2008, New York, USA.

[7] Danilovic, M. and Browning, T.R., (2006), "Managing complex product development projects with design structure matrices and domain mapping matrices". *International Journal of Project Management*, 25: 300-314.

[8] Erden, M.S, Komoto, H., van Beek, T.J., D'Amelio, V., Echavarria, E. and Tomiyama, T. (2008). "A review of function modeling: Approaches and applications". *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22: 147–169. DOI: 10.1017/S0890060408000103.

[9] Fernandez, C.I.G., (1996), "Integration Analysis of Product Architecture to Support Effective Team Co-Location". Master Thesis, Massachusetts Institute of Technology.

[10] Gath, I. and Geva, A.B., (1989), "Unsupervised Optimal Fuzzy Clustering". In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11 (7): 773-780

[11] Helmer, R., Yassine, A. and Meier, C., (2009), "Systematic Module and Interface Definition using Component DSM". *Submitted to Journal of Engineering Design*

[12] Holmqvist, T.K.P. and Persson, M.L., (2003), "Analysis and improvement of product modularization methods: Their ability to deal with complex products". *System Engineering*, 6 (3): 195: 209.

[13] Hölttä, K., Suh, E.S. and Weck, O. de, (2005), "Tradeoff between Modularity and Performance for Engineered Systems and Products". *Proceedings of the 15th International Conference on Engineering Design*, p15-18

[14] Huang, C.C. and Kusiak, A., (1998), "Modularity in design of products and systems". *IEEE Transactions on Systems. Man, and Cybernetics – Part A: Systems and Humans*, 28 (1): 66-77.

[15] Jackson, C.K., (2006), "The Mechatronics System Design Benchmark Report; Coordinating Engineering Disciplines", Publication of *Aberdeen Group*, Boston.

[16] Krishnan, V. and Ulrich, K.T., (2001), "Product development decisions: A review of the literature". *Management Science*, 47 (1): 1-21.

[17] Meng, K.K., Fengming, D., and Guan, T.E. (2007). *Introduction to Graph Theory*, World Scientific Publishing Co. Pte. Ltd., Singapore.

[18] Montgomery Jr., E.B., Huang, H. and Assadi, A., (2005), "Unsupervised clustering algorithm for N-dimensional data". In *Journal of Neuroscience Methods*, 144 (1): 19-24.

[19] Otto, K.H. and de Weck, O., (2007), "Metrics for assessing coupling density and modularity in complex products and systems". In *Proc. of the ASME 2007 Int. Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, IDETC/CIE 2007, Las Vegas, Nevada, USA.

[20] Pedrycz, W., (2005), *Knowledge-Based Clustering – From Data to Information Granules*. John Wiley and Sons Inc., New Jersey, pp. 1-26.

[21] Prommler, T.U. and Steven, D.E., (1994), "Integration analysis of product decomposition". In *ASEM Design Theory and Methodology Conference*, Minneapolis.

[22] Sharman, D.M. and Yassine, A.A., (2004), "Characterizing Complex Product Architectures". *Systems Engineering*, 7(1): 35-60.

[23] Steward, D.V., (1981), "The design structure system: A method for managing the design of complex systems". *IEEE Transactions on Engineering Management*, 28: 71-74.

[24] Suh, N.P., (2001), "Axiomatic Design: Advances and Applications", Oxford University Press, USA

[25] Thebeau, R.E., (2001), *Knowledge Management of System Interfaces and Interactions for Product Development Process*. Master Thesis, Massachusetts Institute of Technology.

[26] Tian, J., Zhu, L., Zhang S., and Liu, L. (2004), "Improvement and Parallelism of k-Means Clustering Algorithm". Tsinghua Science &Technology, 10 (3): 277-281.

[27] Umeda, Y. and Tomiyama, T. (1995). FBS modeling: Modeling scheme of function for conceptual design. In *Proc. Of Working Pap. 9th Int. Workshop on Qualitative Reasoning about Phys. Systems*, Amsterdam, pp. 271-278.

[28] Umeda, Y. and Tomiyama, T. (1997). Functional reasoning in design. *IEEE Expert*, March-April: 42-48.

[29] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y. and Tomiyama, T. (1996). Supporting conceptual design based on the function-behavior-state modeller, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 10 (4): 275-88.

[30] Umeda, Y., Kondoh, S., Shimomura, Y. and Tomiyama, T. (2005). Development of design methodology for upgradable products based on function-behavior-state modeling, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, 19 (3): 161-182.

[31] Umeda, Y., Takeda, H. and Tomiyama, T. (1990). Function, Behaviour, and Structure. In *Aplications of Artificial Intelligence in Engineering V*, ed. by J.S. Gero, Computational Mechanics Publications and Springer-Verlag, Berlin, pp. 177-193.

[32] Whitfield, R.I., Smith, J.S. and Duffy, A.H.B., (2002), "Identifying component modules". In *Seventh Int. Conf. on Artificial intelligence in Design*, AID'02, Cambridge, united Kingdom.

[33] Witzke, E.L. and Fres, S.D., (1988), "Some limitations of adjacency matrices in computer network analysis". *ACM SIGCOMM Computer Communication Review*, 18 (5): 43-47.

[34] Xu, R. and Wunsch, D.C., (2005), "Survey of clustering algorithms". In *IEEE Transactions on Neural Networks*, 16 (3): 645-678

[35] Xu, X., Li, C., Yan, J., and Chen, Y., (2006), "An analytical method based on design structure matrix for modular identification". In *Proc. of the 7th Int. Conf. on Computer-Aided Industrial Design and Conceptual Design*, CAIDCD'06.

[36] Yu, T.L., Yassine, A., and Goldberg, E., (2003), "A genetic algorithm for developing modular product architectures". *Proceedings of the ASME International Design Engineering Technical Conferences and computers and Information in Engineering Conference,* Chicago, DTM-48657, pp. 22-6.

[37] Yu, T.L., Yassine, A., and Goldberg, E., (2007), "An information theoretic method for developing modular architectures using genetic algorithms". *Research in Engineering Design*, 18: 91-109. DOI: 10.1007/s00163-007-0030-1.