Dynamic Modularization throughout System Lifecycle Using Multilayer Design Structure Matrices

Shraga Shoval

Capability Systems Centre, School of Engineering and Information Technology (SEIT)
Australian Defence Force Academy, University of New South Wales Canberra
CANBERRA 2610, Australia
Corresponding author. Tel.: +61 (2) 6268 9566; E-mail address: s.shoval@adfa.edu.au

Abstract

System lifecycle consists of several phases, each entailing different activities and objectives. Modularization is commonly performed during the design stage, and is usually maintained throughout the system’s lifecycle. However, systems may require different modularization architectures in different lifecycle phases. This paper presents a conceptual model for dynamic system modularization based on the different phases in a system’s lifecycle. The model is based on multilayer Design Structure Matrices (DSMs) that construct a 3D representation of the system’s elements and lifecycle phases. A clustering algorithm examines the possible modularization architectures at each phase, and identifies critical dependencies between the various elements of the system.

The proposed model provides an overall evaluation of dynamic system modularization given the variation in the requirements throughout the different phases of the system lifecycle. The model is implemented in a design tool that assists systems planners during the design stages.

Keywords: Modularization; System lifecycle; Design Structure Matrix; Clustering.

1. Introduction

System decomposition is the process of defining the architecture of a complex system in terms of lower-level structures that are separated from each other. While there are some differences between a system’s decomposition and modularization [Kusiak A., and Huan, 1996], this paper uses the two terms interchangeably. System’s decomposition refers mainly to engineering requirements, while modularization refers to system’s abilities. The goal modularization is to group the system components and/or functions into clusters in which the elements within clusters are closely related, while retaining weaker or no relations between elements from different clusters. System modularization is common in the design stage of complex systems (Zakarian and Rushton, 2001; Pinnlner and Eppinger, 1994; Kusiak and Huang, 1996, 1998), aiming to construct a modular system that can be efficiently manufactured, assembled, maintained, and upgraded.

Researchers suggest various methods for system modularization. Group Technology (GT) methodologies (McCormick, 1972; King, 1980; Heragu, 1994; and Shu, 1991) aim at grouping products, machines, tools, and manufacturing processes into manufacturing families (cells) based on their common geometries, physical properties, and manufacturing functions. This cellular manufacturing system is characterized by optimal flow and increased efficiency of the manufacturing process.

Many tools in the design and characterization of the architecture of complex systems use network representations (e.g., Rivkin and Siggelkow, 2007; and Barabasi, 2009) to identify relations between system components. These tools are designed to capture the level of coupling between the various components of the system and to define the structural properties of the system’s functionality (Braha and Bar-Yam, 2007). One limitation of these tools is their inadequate...
capability to identify indirect relations of complex system components that may have a significant effect on the overall system’s performance [Baldwin et al., 2014].

An alternative technique is the use of the Design Structure Matrix (DSM) and the Multiple Domain Matrix (MDM) to identify the structure of complex systems. In DSM, the system is represented as a square matrix in which the cells of the matrix represent the relations between elements (components or functions) of the system [MacCormack et al., 2012]. In Binary DSM, the non-diagonal cells indicate if there is interaction between two elements, while in Numeric DSM, the non-diagonal cells indicate the level or cost of these interactions. The MDM represents hidden interrelations of the system’s elements between various domains, and consists of several DSMs as well as their multiplications. The MDM can identify direct and indirect relations between functions and components, and can clearly and intuitively present these relations as clusters in the matrix.

Clustering algorithms are often used in DSMs and DMMs in order to group elements into modules. The outcomes of these algorithms are groups of elements (modules) that are tightly related to each other within the modules, with loose or no relations between the modules. The clustering of the elements is based on the system’s objectives. For example, system modularization for optimal reliability may result in a specific architecture, while cost-effective maintenance modularization may result in a different one.

System lifecycle consists of several phases in a system’s existence, starting with users’ needs and requirements analysis, high level conceptual design, detailed design, production, construction or acquisition, test and evaluation, verification and validation, integration and operation, maintenance and support, retirement, phase-out, and disposal. Each phase in a system’s lifecycle consists of different activities, defined in the system’s lifecycle model. Modularization is commonly performed during the early stages of the high level conceptual design, and is maintained throughout the system’s lifecycle. However, systems may require different modularization architectures in different lifecycle phases, and interdependencies between its elements should therefore be identified and determined in order to fully comprehend the system throughout its entire lifecycle. Fig. 1 illustrates this idea for three phases in a simplified system lifecycle. The system consists of 9 elements that are divided into modules. Here, element 1 has a strong link to element 2 during the construction and maintenance phases (as they may have similar geometric, physical, or manufacturing properties suitable for cellular manufacturing processes). However, during the disposal phase, element 2 has a strong link to element 3 and a loose link to element 1 (possibly due to disposal or recycling considerations).

This paper presents a model for system modularization given the different phases in the system’s lifecycle. The model is based on DSM tools for integrating the various lifecycle phases. Heppele et al. [2010] present a methodology for determining the interdependencies of a system’s elements between the different lifecycle elements. They use MDM to identify these dependencies based on Eco-Design strategies. Bartolomei et al. [2012] suggest the use of MDM as a framework for modelling large-scale complex systems. Their model consists of social, functional, and technical domains that are represented in Engineering Systems Multiple Domain Matrix (ES-MDM). Schoettl and Lendemann [2014] suggest a generic approach for system modularization based on its lifecycle properties. Their model considers the system’s lifities, and by matrix multiplications they detect the indirect dependencies of the various system’s elements. Once the MDM is constructed, a visual clustering determines the possible modularizations of the system. Newcomb et al. [1998] present two modularity measures for analysis of modularity architectures concerned with, what they call, “lifecycle viewpoints.” Their measures are the Correspondence Ratio (CR) that measures the correspondence between the components of different modules, and the Cluster Independence (CI), which measures the interdependencies between the clusters. Their overall modularity measure is a weighted sum of the two measures.

A new modularization concept that considers the system’s lifecycle phases and lifities is proposed. The concept is based on multilayer DSMs that construct a 3D representation of the system’s elements and lifecycle phases. A clustering algorithm examines the possible modularization architectures of the system at each phase of the lifecycle, and identifies the critical direct and indirect dependencies between the various elements. 3D DSM has already been used by Alizon et al. [2007] in an analysis of product family design to identify uniqueness, varieties, and commonalities of modules in different family products (they used a series of Kodak cameras as a case study). Their goal was to design common modules that can be used in all products from the same family, and to identify the interfaces to these modules across all family products. The analysis of their model is represented by a coloured 3D matrix that is analysed visually by the system’s designers. Although the concept proposed in this paper is similar to the methodology used by Alizon et al. in the sense of using a 3D representation of the system, there are several unique features in the proposed model. First, the proposed model considers the variance in system modularization through the lifecycle phases, while Alizon’s model considers the variance in modularization of a family of products at a single phase of the lifecycle. As a result, the 3D model is a sequence dependent model (along the “vertical” dimension), and the different layers in the model cannot be replaced by each other. Also, the proposed model considers numerical DSM, compared with the binary DSM used in Alizon’s model. Finally, the outcome of the proposed model is a numerical measure that indicates the cost of a particular modularization, given the interdependencies of the elements in the different phases of the system’s lifecycle.

Section 2 formulates the problem, and Section 3 describes the construction of the 3D structure. Section 4 provides a cost analysis of the proposed concept, and Section 5 provides a discussion.
2. Problem formulation

Consider a system that consists of \( n \) elements. The system exists through a regular lifecycle that consists of \( m \) phases. For each phase there is a square matrix \( M_p \in \{0,1\}^{n \times n} \) that maps the interactions of the system’s elements at phase \( p \). A numerical interaction method is used (rather than the binary interaction), where the \( M_p(i,j) \) element represents the level of interaction between element \( i \) and element \( j \) (larger values represent stronger interaction or dependency between the elements). The construction of each \( M_p \) is performed by the standard procedure of constructing a DSM (decomposing the system into its subsystems and elements, and identifying the interactions and interdependencies between the elements at each phase of the system’s lifecycle).

As mentioned, often there are significant differences in the interactions between elements at different phases of the lifecycle, and therefore there might be different matrices for the different phases.

Once all the DSMs are constructed for all lifecycle phases, clustering each DSM into modules is performed. The objective is to generate modules that have strong interactions of the elements within each module and minimal or no interactions and dependencies between the modules. While a few “ideal” systems may be decomposed into totally independent and separable modules, in most systems, especially large and complex ones, there are dependencies and interfaces between elements from different modules. Some algorithms can provide an optimal clustering solution for relatively small systems, however, the problem has been proven to be quadratic, and therefore heuristic solutions are required for large systems. As mentioned in the previous section, many clustering algorithms have been proposed over the past 50 years. Early clustering algorithms originally developed for Group Technology use matrix permutations of rows and columns (e.g., King, 1980; and McMormick et al. 1972).

Other algorithms (Arabie et al. 1996) use various clustering techniques with versatile objective functions. The model in this paper uses an algorithm suggested by Zakarian (2008) for the initial clustering of \( M_p \). The algorithm, designed for non-binary as well as binary matrices, considers interactions that are categorized as “bottlenecks”. Bottleneck interactions are those that are outside the modules, and therefore represent interactions or dependencies between elements that belong to different modules. The objective function of Zakarian’s algorithm is to minimize the weights of the bottlenecked interactions by identifying the critical bottleneck interactions, and to reorganize the elements related to these bottlenecks. The efficiency of the clustering is measured by summation of all the bottleneck interactions, with no consideration to the number of clusters and their dimension. For a system that consists of \( n \) elements, the algorithm starts with \( n \) clusters, each containing a single element (obviously with maximal bottleneck interactions) resulting in the worst efficiency value. As the algorithm proceeds, the number of clusters is reduced and the dimensions of the modules (number of elements within the modules) increase. The algorithm continues to construct the elements until all elements are organized within clusters, and there no bottleneck interactions remain. If no “ideal” clustering is found, the algorithm continues, and eventually it constructs one large cluster that consists of all elements in the system. While mathematically the latter case is acceptable, this is usually not a feasible solution. As a result, there is a need to define a modified objective function that considers, in addition to the weight of the bottleneck interfaces, the number of the modules and their size. Such an objective function is given in Eq. 1:

\[
\Phi = \alpha \sum b_{ij} + \beta \sum d_c^2
\]  

(1)

where

- \( b_{ij} \) – bottleneck interface between element \( i \) and \( j \)
- \( d_c \) – dimension of cluster \( C \) (elements in the cluster)

The coefficients \( \alpha \) and \( \beta \) are determined by the system’s goals and constraints. The system’s designers may imply some constraints regarding the total number of clusters, the clusters’ size, and/or the weight of the bottleneck interfaces. For example, an organization that consists of several production plants may require the division of its manufacturing process among part or all of its plants (determining the minimal number of modules) in order to allow parallel manufacturing and/or utilize its facilities. On the other hand, the system’s designers may determine the maximal number of modules or the maximal number of components in each module based on operational constraints.

The outcome of the clustering algorithm with the modified objective function is a list of possible modular architectures, each with a numerical value of the objective function. The different solutions are then ranked according to their objective function values from best to worst. For example, consider the DSM shown in Fig. 2a. For simplicity, the system consists of only 6 elements and has binary interactions between the elements. Although this example is purely numeric, it can represent a practical engineering procedure such as an assembly process. In this example \( \alpha = 4, \beta = 2 \) for the coefficients of the objective function (commonly \( \alpha = \beta^2 \) in order to maintain a homogeneous objective function). Starting with 6 clusters, each cluster consists of one element (Fig. 2b).

The initial weight of all bottleneck interactions is 12, and the efficiency value according to Eq. 1 is therefore \( 4 \times 12 + 2 \times (1 + 1 + 1 + 1) = 56 \). The second step considers bottleneck interface (1,4) that interacts with cluster 1, and cluster 4, and bottleneck interface (2,3) that interacts with clusters 2 and 3. The algorithm assesses if adding element 4 to cluster 1 and element 3 to cluster 2 improves the objective function, and thus get the matrix shown in Fig. 2c. The new matrix consists of two modules of 2X2 elements and two modules with a single element. The value of the objective function is now \( 4 \times 7 + 2 \times (2^2 + 2^2 + 1) = 48 \), an improvement over the previous matrix. Next, bottleneck interface (3,5) that interacts with module 2-3 and module 5 is considered. Adding element 5 to module 2-3 results in Fig. 2d, with an objective function of \( 4 \times 6 + 2 \times (2^2 + 3^2 + 1) = 52 \). Since there is no improvement, adding the next element – (3,6) to module 2-3 is considered, resulting in the configuration shown in Fig. 2e, with an objective function of \( 4 \times 3 + 2 \times (2^2 + 3^2 + 1) = 40 \). Next step considers adding element 5 either to module 1-4 (Fig. 2f) with an objective function of \( 4 \times 1 + 2 \times (3^2 + 3^2) = 40 \), or to module 2-3-5 (Fig. 2g) with an objective function of \( 4 \times 2 + 2 \times (2^2 + 4^2) = 48 \). Finally the case where all elements are within one module (Fig. 2h) with an objective function of \( 2 \times (6^2) = 72 \) is examined.
Fig. 2. Applying the clustering algorithm to the original DSM (a).

The above procedure is performed for each phase of the lifecycle, with an updated objective function for each phase (the objective function might change significantly at each phase). The possible modularization architectures at each phase are ranked according to the values of the objective function from the best to worst, as illustrated in Fig. 3. The list of possible modular architectures and their corresponding objective function values construct the central database for the subsequent stages in the process, as described in the following sections.

Fig. 3. Data structure of the system’s lifecycle phases, modules and elements

3. Construction of DSM3

Once the clustering algorithm is completed for all phases, a 3D DSM (called DSM3) is constructed. Each layer in DSM3 is a regular DSM representing one phase in the system lifecycle after being processed by the clustering algorithm. The optimal modular architecture for each lifecycle phase (the one with the minimal objective function value at each phase) is chosen for the initial DSM3. Fig. 4 illustrates this concept. Each DSM represents different system architectures (in terms of the system’s modularization), in which clusters are bounded by the solid red line. The different colours outside the clusters represent the different interaction levels between the elements. Initially, the best DSM of each phase is chosen for the DSM3.

The goal of DSM3 is to find an optimal or close-to-optimal modularization throughout the system’s lifecycle. In an “ideal” system, the DSM3 consists of identical layers of DSMs and therefore the optimal modularization is determined by that single DSM. However, different modularizations at different lifecycle phases may create a compound problem when determining the system architecture, as re-modularization of the system may be exhaustive in terms of the system’s resources, and in some cases may be impossible due to operational constraints.

There are three possible types of modules in the DSM3:

1. **Class I modules** – these are the “ideal” modules that are identical in all layers of the DSM3, as they do not change throughout the system’s lifecycle (in terms of the elements within the module).

2. **Class II modules** – modules in this class consist of elements that belong to different modules at different lifecycle phases.

3. **Class III modules** – this type of module exists in only one phase of the lifecycle, as all its comprising elements are not clustered together in any other module during other phases the system’s lifecycle.

For the classification of the modules through the different lifecycle phases, the Jaccard similarity index, also known as the Jaccard commonality coefficient (Jaccard, 1901) is used. In general, the Jaccard similarity index $J_{A,B}$ quantifies the similarity between two sets based on the logical intersection and union of the sets as given by:

$$J_{A,B} = \frac{|A \cap B|}{|A \cup B|}$$  \hspace{1cm} (2)

Fig. 4. DSM3 representing modularization in different phases of the system lifecycle
Other measures of similarity, such as the Sorenesen Similarity Index (Sorenesen, 1948) and the Tanimoto Measure (Rogers and Tanimoto, 1960) use similar techniques, and are different only when measuring the distance between two sets. The data structure shown in Fig. 3 is used in order to measure the similarity between two modular architectures in two lifecycle phases. A set of similarity matrices \( S(m \times n) \) is constructed, where \( m \) and \( n \) are the number of modules in the two lifecycle phases. The \( S(k,l) \) element in the similarity matrix is the Jaccard similarity coefficient between module \( k \) of phase \( i \), to module \( l \) in phase \( (i+1) \). The calculation of \( S(k,l) \) is

\[
S(k,l) = \frac{|M^i_k \cap M^{i+1}_l|}{|M^i_k \cup M^{i+1}_l|} = \frac{p}{p+q+r}
\]

where \( M^b_a \) - is the set of all elements that belong to module \( a \) at lifecycle phase \( b \), \( p \) - the number of common elements in the two modules, \( q \) - the number of elements in \( M^i_k \) and not in \( M^{i+1}_l \), \( r \) - the number of elements in \( M^{i+1}_l \) and not in \( M^i_k \).

The similarity matrix - \( S \) determines the similarities between two modular architectures from two adjacent phases only (phases \( i \) and \( i+1 \)). As mentioned, the initial similarity matrix is constructed for the optimal modular architecture of each phase. Theoretically, there are \( n^2 \times n \) possible similarity matrices for \( u \) and \( v \) possible modular architectures at phases \( i \) and \( i+1 \) respectively. This is due to the fact that similarities between non-adjacent modules are irrelevant. Fig. 5 shows two DSMs for two adjacent phases, while Fig. 6 shows the similarity matrix for these DSMs.

4. Cost of modularization changes

This section presents the estimate of the costs associated with changes in the system's modularization through different lifecycle phases. The estimate is based on the similarity matrices presented in Section 3, as well as on the DSM. Let us first analyse the process of updating the modularization architecture according to the modular classification.

4.1. Class I modules

The class I modules do not change when being transferred from one lifecycle phase to another. These modules are characterized by a value of ‘1’ in the Jaccard similarity coefficient as there are no changes made to the content of the modules in terms of their elements. Modules with a similarity coefficient of 1 throughout the entire lifecycle are considered to be "closed" modules with fixed internal (and most likely external) interactions. There is no cost associated with these modules, in terms of alterations during transitions between phases, and if all modules in the system are class I modules, then the system does not need to be reconfigured during its lifecycle. Such systems are characterized by similarity matrices with values of ‘0’ except the diagonal cells.

4.2 Class II modules

Class II modules have similarity index in the range of 0-1, and as additional analysis is required to determine the transition cost from one phase to the next. Commonly there are two costs associated with the transition of an element from one module to another: the disconnection cost of leaving the current module, and the connection cost for joining the new module. However, it might be that an element is not required for the next phase and therefore only the disconnection cost is considered. Similarly, if a new element is introduced at a new phase, the connection cost is considered.

Consider module \( j \) at phase \( i+1 \) (\( M^{i+1} \)). Cell \((k,j)\) in the similarity matrix \( S \) represents the Jaccard coefficient between \( M^i_k \) and \( M^j \). As shown in Eq. 3, this coefficient consists of \( p \), \( q \), and \( r \). In terms of the costs associated with the construction of \( M^{i+1} \), there is only one relevant parameter - \( r \); the cost of connecting the new elements that were not in \( M^r_i \). These costs are given in the DSM of phase \( i+1 \) (the connection cost of the
elements between sets $k$ and $j$. Module $M_{ij}^k$ may have several similarity coefficients with modules from the previous lifecycle phase (e.g., column 3 in Fig. 6), and all are shown in column $j$ of the similarity matrix. Each cell in column $j$ determines the cost of connecting the new elements in the new module relative to the previous module. However, in order to avoid redundant calculation, it is sufficient to determine the cost with the highest similarity between the new and the previous modules. The rationale is that the new module receives several elements from the previous modules, some of which are already connected to each other, and only the costs related to the new elements should be considered. In terms of the similarity matrix, there is a need to identify the maximal cell in the terms of the similarity matrix, and connecting them to new modules in the lifecycle phase (e.g., column 3 in Fig. 6), and all are shown in several equal cells in the $k$th row, then any of them can be selected randomly.

In order to calculate the costs associated with disconnecting elements from module $k$ at phase $i$, the $k$th row in the similarity matrix associated with module $M_i$ is scanned. Going through all the cells of the $k$th row, the elements that belong to $M_i$ and don’t belong to $M_{ij}$ ($g$ in Eq. 3) are considered. As with the connection cost, the cell with the maximal similarity coefficient is determined, and the cost for that cell is calculated (in the case of several equal cells in the $k$th row, then any cell can be selected).

4.3 Class III modules

Since all elements of the modules in this class leave the system as one unit, there are no costs associated with disconnection, and therefore this type of module does not affect the costs associated with the phase transfer.

4.4 Optimal modularization cost

As stated, the cost estimates presented above are based on the similarity matrices determined by the DSM. The initial DSM consists of the best DSM of each phase. However, given a system with $p$ phases and $a_i$ possible modular architectures for each phase $i$, there are $\prod_{i=1}^{p} a_i$ possible permutations for the DSM, each with a different modularization cost. Since there is a finite number of permutations for DSM, it is feasible to estimate all possible costs, and determine the lowest one. However, this must be considered against the modular operational requirements at each phase.

5. Conclusions

This paper analyses the effect of changes in the system’s modularization during its lifecycle. These changes may be due to operational or other requirements, and introduce additional costs throughout the system’s lifecycle. The paper proposes a model for determining these additional costs using a multi-layer DSM, called DSM. The additional costs are associated with reconfiguring modules by detaching elements from their initial module, and connecting them to new modules in the subsequent lifecycle phases. The analysis is based on similarity matrices that determine the differences between the modular configurations of the system through its lifecycle phases. A design tool assists the system planners to examine the effect of different modular configurations at different phases in the lifecycle, and to determine the additional costs associated with these modifications in the system configuration. The system planner can then determine the modular architecture at each phase, given the operational requirements, as well as the additional costs of changes in the modular architecture at the different lifecycle phases.

References