Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 48 (2016) 271 – 276

www.elsevier.com/locate/procedia

23rd CIRP Conference on Life Cycle Engineering

Dynamic modular architecture for product lifecycle

Shraga Shoval, Li Qiao, Mahmoud Efatmaneshnik and Michael Ryan

Capability Systems Centre, School of Engineering and Information Technology (SEIT)
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

A module is a set of components with interfaces selected in order to help designers address ilities or non-functional system requirements. Consequently, the boundaries of a module do not necessarily coincide with those dictated by functional decomposition. Modularization usually makes the architecture more complex due to additional interfaces and redundancies that have negative consequences on system performance. As a result, modularization is accompanied by a trade-off between non-functional and functional requirements. Additionally the system lifecycle consists of several phases, each characterized by different activities and goals. Systems may benefit from different modular architectures in the different lifecycle phases. This paper presents a dynamic modular architecture methodology, where the modular architecture changes over the different product lifecycle phases. An example of a relatively simple mechanical system - a bicycle – is presented to illustrate the implementation of the methodology.

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

1. Introduction

Decomposition of a system into lower-level structures based on its functional requirements is a common tool in systems design and development, especially for complex systems. Decomposition and modularization are not necessarily identical [1], and the major difference between the two terms is that a system's functional decomposition is often related to engineering requirements, while modularization is undertaken in order to accommodate the desired system ilities, which are defined as non-functional requirements. Ilities are used for appraising the entire system at specific phases and from a specific viewpoint over the system's lifecycle, rather than satisfying the functional requirements of specific elements in the system. Ilities usually add beneficial or even luxurious features to the essential functional requirement of the system. As a result, the functional requirements affect the system's functional structure, and ilities are accommodated by the system's modular architecture. Modular design is commonly implemented as part of the system design [2,3,4] for creating architecture that exhibits ilities, in addition to the required operational features.

Many methods have been suggested for system modularization. The Group Technology (GT) methods [5,6,7,8] group products, machines, tools, and manufacturing processes into manufacturing families (cells). The GT methods are based on common features of the products (e.g. geometries, materials, shapes) and their manufacturing processes. The outcome of the GT methods is a *cellular manufacturing system* that results in optimal flow and increased efficiency. A liaison network that represents the functional, as well as non-functional relations between system's components, is another common tool in the design of modular architectures [9, 10]. Liaison networks are based on nodes that represent the system's elements and arcs that represent the level of coupling and the structural properties between the system's elements [11]. While liaison networks have limited capabilities in identifying indirect relations of complex system elements [12], Design Structure Matrices (DSMs) and Multiple Domain Matrices (MDMs) can identify these hidden structures in complex systems. Various clustering algorithms are used in DSMs and MDMs in order to group elements into modules, using cost functions which are based on the system's objectives.

Each phase in the system's lifecycle is characterized by different functional requirements, as well as non-functional ilities [13,14]. Unfortunately, modularity and functionality don't always have a positive relationship. In fact, modularity may have adverse effects on functionality as illustrated in Fig. 1. The system's elements are initially divided into three clusters (subsystems) according to functional and logical decomposition at the operation phase. The elements within each cluster have strong connections between them (shown by the solid connection lines) with looser or no connections with elements in the other subsystems (shown by the thinner connecting lines). Ideally, the connections between clusters are weak such that they can be easily disconnected from other clusters. However, the modular architecture that is based on the system's ilities may constitute a different structure that, in some instances, may contradict the functional structure during the operation phase.

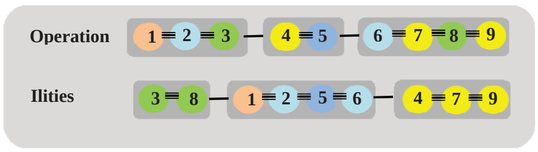

Fig. 1. Changes of modular configuration in system's lifecycle

In this paper we discuss the design of system modularization given the differences between the system's functional structure and the architecture driven by the system's ilities. A dynamic modularization concept is proposed. The dynamic modularization considers the system's structure based on functional requirements and non-functional illities. Newcomb et al. [15] present two measures for analysing modular architectures in what they call "lifecycle viewpoints." The measures consider the differences between the components of various modules and the interdependencies between the clusters. Modularity is determined by the weighted sum of the two measures. Alizon et al. [16] analyse product family design in terms of the uniqueness, varieties, and commonalities of modules in family products. The objective is to determine common features among all modules. These features can then be used in a large range of products from the same family.

The concept proposed in this paper considers the variance between the modular structures resulted from functional and non-functional considerations through the lifecycle phases. The outcome of the proposed model is a numerical value that indicates the clustering cost of a particular architecture, given the interdependencies of the elements in the functional structure and the modularization requirements as derived from the system's ilities.

2. Problem formulation

Consider a system that consists of n elements. A square matrix $M_F \in (n \times n)$ also known as the functional DSM (we refer to this matrix as DSM^F), that maps the interactions of the system's elements according to the functional requirements. A numerical value $m_F(i,j)$ represents the level of interaction between element i and element j (larger values represent stronger interaction or dependency between the elements). A similar procedure is performed for the construction of the system's ilities DSM – $M_I \in (n \times n)$ (we refer to this matrix as DSM^I) that consists of the same n elements as in the functional DSM^F. Once the two DSMs are constructed, a clustering procedure is performed independently on each matrix to generate groups of elements that have strong internal interactions, and minimal or no interactions and dependencies between the groups. As mentioned, 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. Other algorithms use techniques based on versatile objective functions. Zakarian [17] proposes a model for non-binary as well as binary matrices that considers interactions that are categorized as "bottlenecks". Bottleneck interactions are outside the clusters, representing interactions or dependencies between elements that belong to different clusters. The objective is to minimize the weights of the bottlenecked interactions by reorganizing the elements related to these bottlenecks. The efficiency of the clustering process is measured by the summation of all the bottleneck interactions. The algorithm starts with n clusters, each containing a single element (with maximal bottleneck interactions) resulting in the worst clustering cost. As the algorithm proceeds, the number of clusters is reduced and the dimension of the clusters (number of elements within the clusters) increases. The algorithm continues to extend the clusters until no bottleneck interactions remain. In an "ideal" system, the outcomes are clusters with closely related elements and no bottleneck connections. If no "ideal" clustering is found, the algorithm continues until eventually it constructs one large cluster that consists of all elements in the system. While mathematically the latter case is acceptable (as there are no bottleneck interactions and the cost is zero), this is usually not a feasible solution. There is a need to define a modified objective function that considers, in addition to the the bottleneck interfaces, the number of the clusters and their size. For example, consider the following objective function:

$$C_{tot} = \sum_{i=1}^{k} C_{int}(i) + C_{ext} \quad (1)$$
$$C_{int}(i) = (W(u,v) + W(v,u)) * d_i^p \quad (2)$$
$$C_{ext} = (\sum(B(u,v) + B(v,u))) * n^p \quad (3)$$

where C_{tot} - total clustering cost
$W(u,v)$ - internal connection within a cluster
$B(u,v)$ - external connection (bottleneck connections)
$C_{int}(i)$ - internal cost of cluster i
C_{ext} - external cost
d_i - dimension (size) of cluster i
n - number of elements in the entire system
p - penalty factor