Regular Paper

A Method for System Interface Reduction Using N^2 Charts

Ofri Becker,[1]* Joseph Ben-Asher,[2] and Ilya Ackerman[1]

[1]WALES Ltd., 11 Tuval Street, Ramat-Gan 52522, Israel

[2]Department of Aerospace Engineering, Technion IIT, Haifa 32000, Israel

Received July 28, 1999; accepted November 16, 1999

ABSTRACT

This work focuses on system interface reduction using N^2 *charts*. The N^2 *charts* method, which has been first introduced by NASA, and the tools it provides for the systems engineer are explained and demonstrated here in detail. The system interface reduction problem is defined and introduced as finding a reduced order partitioning of the system elements that satisfies either of three different criteria. In order to apply optimization techniques to the problem of system interface reduction, it is first needed to translate it into a quantitative form. A transformation method of the problem is developed after which the problem can be solved by a suitable algorithm. An interesting and useful equivalent approach of the problem in terms of graph theory is provided. A computerized algorithm is used to demonstrate the solutions of two different examples. An analysis of this algorithm's computation complexity and its runtime as a function of the N^2 *chart*'s order is presented. This tool is planned to assist the designer of the system in the partitioning and representation of the tasks to be performed.
© 2000 John Wiley & Sons, Inc. Syst Eng 3: 27–37, 2000

1. INTRODUCTION

The method of N^2 *charts* is a modern tool that offers the systems engineer a useful and efficient mapping of a system and its interfaces. When using this method, one defines an N-by-N square matrix, namely a N^2 chart, in which the N different system elements are on the main diagonal and the interfaces between these elements are outside the main diagonal and flow clockwise, i.e., first horizontally in the row and then vertically in the column as demonstrated in Figure 1. In many cases it is convenient to expand this method and characterize a system by a hierarchy of N^2 charts, dividing the system into different levels of resolution needed by the designer, where every element of the higher levels is also a system and hence may be described by its own N^2 chart. For example, an aircraft (as a system) may consist of

*Author to whom all correspondence should be addressed (e-amil: wales@isdn.net.il)

© 2000 John Wiley & Sons, Inc.

Element #1		Interfaces from element #1 to #3	...	
	Element #2			
		Element #3		
⋮			⋱	
	Interfaces from element #N to #2			Element #N

Figure 1 A general N^2 chart.

the following elements: the fuselage, the engine, the navigation and control, the landing gear, and the fuel systems, each one of which may be referred to as a standalone system consisting of its own elements; e.g., the fuel system may consist of the tank, the pumps, the pipes and the fuel etc.

The latest trends of modern design and manufacturing such as modularity, COTS (Commercial off the Shelf), and outsourcing emphasize the need for efficient interface design and development. A design process well planned ahead is a main cause of success, as is said: "The last of days, for which the first was made" (in the Jewish Sidur in *"Lecha Dodi"* by Alkabetz). In a multielement system design there are usually a number of vendors involved with many interfaces between different elements and vendors. System interface reduction, i.e., the minimization of the number of interfaces and sometimes the number of vendors given designer defined constraints, can make the design process more efficient and cost-effective. Interface reduction is valuable also to the reliability of a system. Suppose the system elements are of much higher reliability and maturity than its interfaces, such as in situations where COTS is emphasized, it follows usually that the fewer interfaces that exist in the operating system, the lower the probability for its failure. A good example for the efficiency of interface minimization has been given by NASA's APOLLO space project. In Shishko [1995] it is noted that only 100 wires were needed to link the APOLLO spacecraft to the Saturn launch vehicle, and it is emphasized that a single person could fully understand the interface and cope with all the effects of a change on either side of the interface, this fact being one of the main reasons for the success of the whole project.

A method for system interface reduction using N^2 charts has been developed and is explained in this paper.

2. THE METHOD OF N^2 CHARTS

The method of N^2 charts is used in order to map the different elements and interfaces that exist in a system. This is done by:

1. Defining the different elements and placing them on the main diagonal of an N-by-N matrix, where N is the number of elements defined. The baseline matrix should separate the different elements up to the highest possible resolution in order to

enable the mapping of all the interfaces between them.

2. Defining the interfaces that flow from each element to another, and placing them in the other cells of the matrix so that they flow clockwise, i.e., first horizontally in the row and then vertically in the column.

This matrix is the N^2 chart of the system. A general N^2 chart is demonstrated in Figure 1.

It should be noted that N^2 gives a pairwise-only interrelationship prospective of the system elements and fails to recognize relations between three and more components (e.g., modern aircraft data-bus with multiple terminals).

There are several different ways of mapping a system as an N^2 chart, noting that sometimes some elements can be grouped into one from the designer's or the systems engineer's point of view. For example, let the elements be the different components of a system to be designed, and let the interfaces be the hardware, software, and information needed to be transported between the different teams or vendors that design and manufacture different elements; thus grouping the elements into one could mean that these two or more components of the system are to be designed and manufactured by the same vendor, hence removing the need for hardware, software, and information transport between the design and manufacturing processes of these grouped elements. Moreover, if grouped elements have a same kind of interface with another element of the system, then it is usually practical to consider only one interface of that kind between the group and the other element. For example, if the control system (e.g., *element #1*) and the internal electricity system (*element #2*) of an aircraft (the *system*) are manufactured by the same vendor (*grouped elements)*, then the information about the aircraft power distribution (an *interface)* has not to be transported between vendors, and the company (another *group of elements*) that manufactures electrical cables for the aircraft has to send only one truck (an *interface)* containing cables to one vendor instead of two different trucks to different vendors. Grouping elements creates a partitioning of the system into Blocks such that the N^2 chart is changed and its order is reduced. The new order is the number of Blocks of grouped elements defined. Note that if an element is not grouped with others, then it is considered as a single element Block. The interfaces that appear in the new chart are only the ones that flow between the Blocks. For example, let us suppose that elements #2 and #3 are grouped into one block and elements #4 and #5, up to #*N* are grouped into another block, while element #1 is left as a single element block; then the N^2 chart of order *N* from Figure 1 collapses into a new chart of order 3 as shown in Figure 2.

Block 1 (Element #1 only)	Interfaces from element #1 to elements #2 and #3	Interfaces from element #1 to elements #4,#5,…, #N
Interfaces from elements #2 and #3 to element #1	**Block 2** (Elements #2 and #3)	Interfaces from elements #2 and #3 to elements #4,#5,…, #N
Interfaces from elements #4,#5,…, #N to element #1	Interfaces from elements #4, #5,…, #N to elements #2 and #3	**Block 3** (Elements 4,5,…,N)

Figure 2 A reduced N^2 chart.

Were grouping all elements into one block possible, such that they can be designed and/or manufactured by the same vendor, then the N^2 chart could collapse into a single block matrix; but in most real cases the possibility of designing or manufacturing elements by the same vendor is subject to constraints. Some combinations of elements can be under the same vendor's responsibility while some others cannot, due to various considerations such as vendors' expertise, manpower availability, schedule, cost, etc. As an immediate result, not all the grouping manipulations of the N^2 chart are valid, and thus the interface reduction problem is not one of a trivial solution.

3. THE PROBLEM OF SYSTEM INTERFACE REDUCTION

3.1. Problem Definition

Suppose one has defined a baseline N^2 chart of a system including all elements as separated items, all interfaces between them, and all grouping constraints considering whether each pair of elements can or cannot be grouped together. Let us search for an optimal valid partitioning of the system by means of interface reduction. Three optimization criteria can be formulated:

1. Lowest order partitioning (disregarding the number of interfaces).
2. Minimal number of interfaces for a given order of partitioning.
3. Overall minimal number of interfaces.

3.2. Transformation of the Problem into a Quantitative Form

The nature of optimization is to find a minimum or a maximum value of a function calculated over a field of variables, which may reflect to constraints. In order to apply optimization techniques to the problem of system interface reduction, it is first necessary to translate the problem into a quantitative form. This is done as follows:

1. First define serial numbers 1, 2, …, N for each element, where N is the baseline order of the system dealt with.
2. Define one binary bit for each one of the different interfaces existing in the system; thus a binary *word* can describe a checklist of the existing interfaces flowing from one element to another. The order of the different bits within the binary *word* is arbitrary but must be used consistently. Let R be the number of different interfaces existing in the system; then the interface binary *word* is of length R.
3. Given the N^2 chart and the partitioning constraints, define two numerical matrices, namely, the Interface Binary Matrix (IBM), describing the interfaces by the binary representation, and the Grouping Constraints Matrix (GCM), describing the partitioning constraints. In the Interface Binary Matrix the diagonal elements are zeros, while the general element in row i and column j for $i \neq j$ is a binary *word* of length R containing a one–zero checklist of the different interfaces that flow from element i to element j, namely, 1's at all bits that represent the existing interfaces and 0's at bits representing those that do not exist there.

   In the Grouping Constraints Matrix, the diagonal is zeros and the general element in row i and column j is "1" if the designer determines that elements i and j can be grouped together, and "0" if they cannot. For example, if there is no vendor that is able to design both the aircraft's jet engine (say element i), and the air inlet (say element j), then these two elements cannot be grouped together and should provide a zero in the ij and ji cells of GCM. Note that GCM is necessarily symmetric while IBM is not. An example for each one of the matrices is given in Figures 3(a) and 3(b).

$$\begin{bmatrix} 0 & 1001 & 0001 & 0101 & 1000 \\ 1010 & 0 & 0000 & 0001 & 0101 \\ 1000 & 0000 & 0 & 0000 & 1010 \\ 0010 & 0000 & 0111 & 0 & 1000 \\ 0000 & 0000 & 0000 & 0100 & 0 \end{bmatrix}$$

Figure 3(a) An example for an Interface Binary Matrix for a system of order 5 with 4 different interfaces.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 3(b) An example for a Grouping Constraints Matrix of order 5.

This transformation of the problem into a numerical form is a major stage in the method, after which the final solution is within reach by the operation of an optimization algorithm.

4. AN EQUIVALENT APPROACH OF THE PROBLEM IN TERMS OF GRAPH THEORY

The Grouping Constraints Matrix is equivalent to the Adjacency Matrix known in graph theory [Harary, 1972; Deo, 1974], indicating the existence of arcs between the different vertices of a graph with the vertices representing the elements and the existing arcs representing the pairs of elements that can be grouped together. The graph that is represented by the GCM from Figure 3(b) is shown in Figure 4.

Figure 4 The graph represented by the GCM in Figure 3(b).

Figure 5 The labeled and directed multigraph represented by the IBM from Figure 3(a).

In the same manner most of the concepts of the defined quantitative form have an equivalent in graph theory, as is also for the optimization criteria.

- A group of elements is equivalent to the subgraph including all vertices that represent this group's elements and all the existing arcs between them.
- A *valid block* is equivalent to a complete subgraph, i.e., a subgraph in which an arc exists between each pair of vertices. (e.g., the triangle formed by the vertices 2, 3, and 5 and by the arcs connecting between them in Figure 4).
- The minimal order partitioning criterion is equivalent to finding a minimal number of disjoint subgraphs that combine together all the vertices of the GCM's equivalent graph.

In Harary [1972] it is shown that the nondiagonal entries of a matrix which is the power n of the Adjacency Matrix, where n is an integer, represent the number of paths of length n existing between each two vertices. Using this theorem, one can indicate whether a graph is connected or not by a look at the powers 1, 2, …, N-1 of the adjacency matrix. If a specified non-

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Figure 6(a) The Grouping Constraints Matrix for problem #1.

$$\begin{bmatrix} 0 & 10 & 00 \\ 00 & 0 & 01 \\ 00 & 00 & 0 \end{bmatrix}$$

Figure 6(b) The Interface Binary Matrix for problem #1.

diagonal entry (i, j) is zero in all of these matrices, then there exists no path between vertices i and j, which means that the graph is not connected. Obviously, when this occurs, there will be more such entries, and the graph can then be separated into its disconnected blocks being subgraphs, each of which is a connected one. This realization of separation of the problem, if it exists, is valuable for the system interface reduction process. The solution of one problem of a high order is much more complicated and time-consuming than the solution of the separate lower order problems all together as is analyzed further in this work.

In order to realize the equivalence of the interfaces minimization criteria to a graph theory problem, it is necessary to define a new labeled and directed multigraph, i.e., a graph in which more than one arc may exist between a pair of vertices and every arc is labeled and directed. The problem is then to divide the GCM's graph into complete subgraphs (the number of which being the order of partitioning) such that the same partitioning of the IBM's graph minimizes a suitable functional that counts the number of interfaces. The graph representing the IBM from Figure 3(a) is shown in Figure 5. It should be remarked that for generality we

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 7(a) The Grouping Constraints Matrix for problem #2.

allow for the same interface to flow both ways (e.g., the leftmost bit from vertex 1 to 2 and back) as happens sometimes in real systems, e.g., a launch signal returning from a smart bomb back to the communications and control center.

5. A COMPUTERIZED OPTIMIZATION ALGORITHM

Having translated the problem into a numerical form, it can be solved by a suitable algorithm. In simple cases, e.g., when the baseline N^2 chart is of very low order (up to 3 or 4) or when there exist very few interfaces in the system, the solution can be found intuitively without using any software; but, as it appears in many cases, a system may contain a much larger number of elements, many interfaces, and a complicated combination of grouping constraints. In the latter case, a computerized algorithm is needed in order to solve the problem within a reasonable time. The GCM and IBM of a relatively simple case (problem #1) are demonstrated in Figures 6(a) and 6(b), while those of a much more complicated case (problem #2) are shown in Figures 7(a) and 7(b).

In order to get the feeling of the different complexities of interface reduction problems, the reader is invited to try and solve the problems of which numerical matrices are presented. Obviously, problem #1, which is of order 3, and involves only two different interfaces, does not necessarily require the use of a computer in order to be solved in a short time, while problem #2, which is of order 9 and involves five different interfaces (still a relatively small number for this order), could surely be solved faster using a computer rather than the human brain only.

A computer algorithm that solves the interface reduction problem, given the Grouping Constraints Matrix, and the Interface Binary Matrix has been developed. It consists of the following steps:

1. Read the GCM and the IBM.
2. Input the type of optimization criterion (minimal order/minimum interfaces for a given order/global minimum interfaces).
3. According to the Grouping Constraints Matrix, create a pool of all *valid blocks*, i.e., groups of elements, which can be grouped together (pairwise), including all single element groups. For efficiency purposes, it is recommended to sort this pool by size (number of elements in a *valid block*). An identification whether a group of two or more elements is a *valid block* is made by looking at the minor that refers to all of its elements in the Grouping Constraints Matrix. Only if apart from its main diagonal (which is all zeros), this minor contains 1's only (equivalently equal to $1_{k,k}$-$I_{k,k}$, where k is the number of elements in the group and $1_{k,k}$ is a k-by-k matrix of 1's), then the group it refers to is a *valid block.* The pool of *valid blocks* for the Grouping Constraints Matrix presented in problem #1 is shown in Figure 8.
4. Find the largest *independent* group, namely, the group containing a maximum number of elements possible in which no pair can be grouped together; this group is obviously not a *valid block.* A group of elements is identified as independent by its minor being all zeros. It is easily understood that the size of the largest independent group is a lower bound for the possible partitioning order of the system, although it is not necessary that this order of partitioning is achievable. In the example of problem #1, the largest independent group is {1,2} the size of which is

$$\begin{bmatrix} 0 & 00000 & 10101 & 11100 & 00000 & 00000 & 01000 & 00000 & 00000 \\ 00001 & 0 & 10000 & 11100 & 00000 & 01000 & 00000 & 00000 & 00100 \\ 00001 & 00000 & 0 & 01000 & 10000 & 10000 & 00000 & 00000 & 11000 \\ 00000 & 00000 & 01010 & 0 & 10100 & 00000 & 01100 & 01010 & 10001 \\ 00100 & 01011 & 11101 & 01011 & 0 & 10110 & 11001 & 01000 & 00000 \\ 00000 & 00000 & 01010 & 01100 & 10001 & 0 & 01100 & 01101 & 10101 \\ 01000 & 10000 & 10000 & 10001 & 00000 & 00000 & 0 & 10000 & 11100 \\ 01111 & 01101 & 01000 & 00000 & 00001 & 10000 & 00000 & 0 & 01111 \\ 10000 & 00000 & 00000 & 00000 & 01100 & 00000 & 01110 & 00000 & 0 \end{bmatrix}$$

Figure 7(b) The Interface Binary Matrix for problem #2.

Serial no.	*Valid block*
1	{1,3}
2	{2,3}
3	{1}
4	{2}
5	{3}

Figure 8 The pool of *valid blocks* for problem #1.

2, and thus there does not exist in this case a partitioning of an order less than 2.

5. Define O_{min} as the lower bound order found in step 4, and let $ORDER = O_{min}$.
6. Begin a search throughout the pool created in step 3, for *ORDER* different *valid blocks* that include exactly all the different elements (i.e., with a single representation of each and every element). The first combination found will represent a minimal order partitioning of the system. If all the combinations of *ORDER valid blocks* do not meet the mentioned criteria for a partitioning, increase *ORDER* by 1 and search again. Two different partitionings of order 2 are found in the example problem, {{1,3}, {2}} and {{2,3}, {1}}, the first of which is found depending on the sequence of the search.
7. After the minimal order partitioning is found, it matters whether the optimization is to be made by means of minimal order or by means of interface reduction. If the former is the case, then output the partitioning that was found and stop. Otherwise, it is necessary to count the number of interfaces in this partitioning and continue to search throughout all the relevant values of *ORDER*, for all valid partitionings. Note that only interfaces that flow between different blocks of the partitioning are counted, with attention to the fact that if between two blocks of a partitioning flows the same interface more than once in the same direction, then it is counted only once; thus the new binary *word* flowing from new block *i* to new block *j* is the output of a logical inclusive OR, operating on all the old binary *words* from elements in block *i* to those in block *j* according to the partitioning. This is the main justification for using the binary system to represent the interfaces. In the example problem #1 the number of interfaces for the partitioning {{1,3}, {2}} is two, one of which flows from element #1 (in new block #1) to element #2 (in block #2 as a single element) and the other flows from element #2 to element #3 (in new block #1).
8. Continue the search up to the maximal order of partitioning needed by the designer (obviously, if the order was given then the search is restricted to it).

The optimization algorithm is described by the flowchart given in Figure 9.

6. AN ANALYSIS OF THE OPTIMIZATION ALGORITHM COMPLEXITY

A practical question about the system interface reduction process is how long does it take by means of computational operations. The elementary operation that is the major time consumer is the check whether a combination of valid blocks from the pool is a partitioning of the system. This check can be done by a checksum comparison or by elimination of disqualifying criteria using IF statements. Either way the number of combinations to be checked strongly depends on the number of valid blocks in the pool and on the order of partitioning searched for. Apparently C_O^P which equals to $P!/(P-O)!O!$ different combinations of valid blocks need to be checked, where P is the total number of valid blocks in the pool and O is the order of partitioning. Note that the maximal possible number of valid blocks in the pool of a Grouping Constraints Matrix of order K, $P_{max}(K)$ is $C_K^K + C_{K-1}^K + \cdots + C_1^K = 2^K - 1$ (when all pairs of elements can be grouped and thus all groups of size $K, K-1, K-2, \ldots, 1$ are valid blocks). The efficiency of the algorithm can be improved significantly by realizing disqualification of a combination as a partitioning during its creation and jumping directly to the next qualified partial combination. For example: During the creation of the combination {{1,2,3}, {2,3,4}, {4,5}}, one can realize that its first two valid blocks {1,2,3} and {2,3,4} can never sit together in a partitioning due to duplication of elements. Moreover, if, for example, the baseline system is of order 6, then the sum of the sizes of these two mentioned valid blocks, being already 6, is too large for two blocks of a partitioning of order 3, regardless of the third valid block; therefore, it is reasonable to seek for the next qualified second valid block of the combination (instead of the valid block {2,3,4}), skipping many disqualified combinations without further checks. It is easily understood that, as the sum of the sizes of valid blocks in a partial combination can be too large, it can also be too small given that the pool is sorted by size. Determination of the exact number of

Figure 9 The optimization algorithm.

Figure 10 Optimization algorithm average runtime for minimal order partitioning vs. baseline order.

checks that are saved due to the disqualification of combinations during their creation is not within the scope of this work. A time consumption evaluation of the optimization algorithm operation as a function of the baseline order of the system has been investigated by running the optimization algorithm (written in MATLAB) many times for different random GCMs. The results of this evaluation shown in Figures 10 and 11 show a nearly exponential dependence of the runtime on the baseline N^2 chart's order. For minimal order partitioning criterion only, the runtime is multiplied by a factor of about 2 for every increase of the baseline N^2 chart's order by 1, while, for minimum interfaces partitioning for all possible orders, the nearly exponential behavior remains but the runtime growth factor with the baseline order is about 5.

Figure 11 Optimization algorithm average runtime for minimum interfaces partitioning for all possible partitioning orders vs. baseline order

7. RESULTS

An output of the solutions of problems #1 and #2 by the computerized optimization algorithm is shown below:

```
Problem #1
Grouping constraints matrix =
      0     0     1
      0     0     1
      1     1     0

Interface matrix (in decimal values) =
      0     2     0
      0     0     1
      0     0     0


Run #1: Minimal order partitioning


Solution:

Order =
     2

Partitioning (blocks in rows):
   1      3

   2

---------------------------


Run #2: Minimum interfaces for all
possible orders


Solution:

Order =
     2

Partitioning (blocks in rows):
     2     3

     1

Number of interfaces:
     1
**********************************
**********************************

Problem #2
```

```
Grouping constraints matrix =
 0   0   1   0   1   1   1   0   1
 0   0   1   1   0   0   1   1   0
 1   1   0   0   1   0   1   0   0
 0   1   0   0   0   0   0   1   1
 1   0   1   0   0   1   1   0   1
 1   0   0   0   1   0   1   1   0
 1   1   1   0   1   1   0   1   0
 0   1   0   1   0   1   1   0   1
 1   0   0   1   1   0   0   1   0

Interface matrix (in decimal values) =
 0   0  21  28   0   0   8   0   0
 1   0  16  28   0   8   0   0   4
 1   0   0   8  16  16   0   0  24
 0   0  10   0  20   0  12  10  17
 4  11  29  11   0  22  25   8   0
 0   0  10  12  17   0  12  13  21
 8  16  16  17   0   0   0  16  28
 15 13   8   0   1  16   0   0  15
 16  0   0   0  12   0  14  0   0

Run #1: Minimal order partitioning

Solution:

Order =
  3

Partitioning (blocks in rows):
     1     5     6     7

     4     8     9

     2     3

---------------------------

Run #2: Minimum interfaces for orders
3 and 4

Solution:

Order =
     3

Partitioning (blocks in rows):
     1     5     6     7

     4     8     9

     2     3

Number of interfaces:
    25

***********************************

Order =
     4

Partitioning (blocks in rows):
     1     5     6     7

     4     8     9

     2

     3

Number of interfaces:
   34

***********************************
***********************************
```

8. CONCLUSIONS

The different elements and interfaces of a system can be mapped efficiently using the method of N^2 charts. The system interface reduction problem is to find a reduced order partitioning of the system elements such that it satisfies any of the optimization criteria that have been formulated, subject to designer defined grouping constraints. This problem is translated into a quantitative form by defining two numerical matrices, the GCM (Grouping Constraints Matrix) and the IBM (Interface Binary Matrix) and solved by a suitable computer algorithm which creates a pool of all valid grouping blocks of elements and searches throughout it for the optimal partitioning(s). In order for the algorithm to be as efficient as possible, a great deal of attention must be paid to the avoidance of redundant operations. This is important especially during the creation of combinations of *valid blocks*, where a partial combination can be indicated as disqualified from being a part of a partitioning regardless of its remaining *valid blocks*. An analysis shows an approximately exponential growth of the algorithm's runtime with the increase of the baseline order.

The solutions of the problem by means of each one of the three different optimization criteria has a practical use in system design. The minimal order partitioning, for which only the GCM is needed, can determine the minimal number of vendors that should take part in a process. The minimum interfaces partitioning for a

given order, and the global minimum interface partitioning can be used:

1. In outsourcing the system elements among vendors in cases where interfaces between vendors are costly.
2. For efficient representation of the system during its development.
3. For improving the reliability of a system especially in cases where the maturity and reliability of the system elements is much higher than the maturity and reliability of its interfaces, such as situations where COTS is emphasized.

REFERENCES

B.S. Blanchard and W.J. Fabrycky, Systems engineering and analysis, 3rd ed., Prentice Hall, Englewood Cliffs, NJ, 1998.

N. Deo, Graph theory with applications to engineering and computer science, Prentice Hall, Englewood Cliffs, NJ, 1974.

F. Harary, Graph theory, Addison-Wesley Series in Mathematics, Addison-Wesley, Reading, MA, October 1972.

T.C. Robertson (Editor), INCOSE systems engineering handbook, January 1998.

R.S. Shishko (Editor), NASA systems engineering handbook, SP-6105, June 1995.

Ofri Becker has been an aerospace engineer at WALES Ltd. in Ramat-Gan, Israel since 1999. He earned his B.Sc. and M.Sc. at Technion IIT in Haifa, Israel in 1997 and 1999, respectively.

Joseph Ben-Asher has been an associate professor at Technion IIT in Haifa, Israel since 1998. He earned his B.Sc. at Technion ITT in 1978 and his M.Sc. and Ph.D. at Virginia Polytechnic Institute and State University in 1986 and 1988, respectively. He was formerly employed at IAF (1978–1984) and IMI (1984–1997).

Ilya Ackerman has been a Senior Consultant in Systems Engineering and Marketing at WALES Ltd. in Ramat-Gan, Israel since 1997. He earned his B.Sc. at Technion IIT in Haifa, Israel in 1977 and his MBA at Ben-Gurion University in Israel in 1997. He was employed with IAF from 1977 to 1994. He was the Assistant to the R&D Attaché at the Embassy of Israel in Washington, DC in 1994–1996.