How Many Systems Are There? – Using the N^2 Method for Systems Partitioning

Tamir Bustnay and Joseph Z. Ben-Asher
Technion, Israel Institute of Technology, Haifa, Israel

ABSTRACT

An algorithm for partitioning a complex system into its independent constituents is developed. The N^2 method is used for the problem formulation whereby the objective is to obtain a system presentation in a simple flow. The development of the algorithm is based on graph theory techniques. The algorithm is formulated using graph theory and N^2 charts. Several examples are presented to demonstrate the relative ease of the algorithm employment.

How Many Systems Are There? – Using the N^2 Method for Systems Partitioning

Tamir Bustnay and Joseph Z. Ben-Asher
Technion, Israel Institute of Technology, Haifa, Israel

ABSTRACT

An algorithm for partitioning a complex system into its independent constituents is developed. The N^2 method is used for the problem formulation whereby the objective is to obtain a system presentation in a simple flow. The development of the algorithm is based on graph theory techniques. The algorithm is formulated using graph theory and N^2 charts. Several examples are presented to demonstrate the relative ease of the algorithm employment.

1. INTRODUCTION

A system is defined in Ref. [1] as: "An interacting combination of elements to accomplish a defined objective. These include hardware, software, firmware, people, information, techniques, facilities, services, and other support elements".

Ref. [2] refines this definition considerably by requiring that the "The set of components has the following properties:

a. The properties and behavior of each component of the set has an effect on the properties and behavior of the set as a whole

b. The properties and behavior of each component of the set depends on the properties and behavior of at least one other component in the set

c. Each possible sub-set of components has the two properties listed previously; the components cannot be divided into *independent* subsets"

Notice that in a. we simply translate the previous definition of Ref. [1] into a verifiable fact; in b. we distinguish between the system and its environment (which affects the system mostly in a unilateral way); and finally in c. we distinguish between a single holistic system that cannot be partitioned further, and a coalition of smaller systems.

The notion of *independency* will be discussed in Section 2 of this article.

For complex systems, breaking down a system into its constituents to determine the maximal partitioning, or in other words, answering the question: "how many systems are there?' is not a simple task.

The importance of this problem is mainly related to integrations issues. "Big-bang" or non-incremental integration of complex systems is a formidable task. Incremental integration of sub-systems is therefore, almost always, the preferred strategy (Ref. [3]). Having the maximal number of independent sub-systems is therefore invaluable for design and integration purposes.

Ref. [4] employs the N^2 method to solve a different problem of partitioning with the maximal interface reduction subject to external constraints. Since elements can affect each other only via interfaces (direct or indirect) we can formulate our problem along similar lines.

In the N^2 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. Fig. 2 is a N^2 example (Ref. [2]) for a spacecraft system.
Figure 3 (Ref. [5]) demonstrates some of the features that can be observed by the N^2 chart. Firstly we can isolate "critical functions" as functions or elements that are interfaced with critically many other elements. We further distinguish between "simple flow" as shown for F5->F6->F7 where the interfaces are flowing down to the same direction (as would still be the case if we add more interfaces from F5 directly to F7), and "control loop" e.g. the connections of F3 and F4 that are in a "closed loop".
Finally, we are interested in the "nodal point" and the "tightly related functional group" concepts. Notice that the "tightly related functional group" F8-F11 can be viewed as a separate system. The only connection through the "nodal point" is a unilateral interface from F7 to F8. Hence the set F8-F11 does not affect the set F1-F7, and by part c. of the above definition, we can divide the system into two separate systems. Notice that although F1-F4 and F4-F7 are also "tightly related functional groups" the separation between them is not possible because of the critical node F4.
Consider now an arbitrary permutation of the system, i.e. a different N^2 chart presenting the same system but in a different order of appearances for the elements. Fig. 4 presents such a permutation by switching between F3 and F9. The problem of breaking down the system into its constituents can now be significantly more difficult. The main contribution of this short paper is the employment of graph theory tools in order to address this problem.
The paper is organized as follows. The problem will be formulated using graph theory tools in the next section. An algorithm to handle it will be developed in Section 3. Section 4 transforms the algorithm from graph theoretical formulation into the N^2 formulation. Several examples will be given and discussed in Section 5. Section 6 concludes the paper.

2. PROBLEM FORMULATION

We are interested in the partitioning that breaks down a system to the *maximal* number of independent sub-systems. We need to elaborate on the definition of sub-system independency.

Definition I:
Two sub-systems will be mutually independent if their interfaces structure is restricted to a simple flow.
Thus the three sub-systems presented in Fig 5a are mutually independent, but those of fig. 5b are not.
Our problem can be formulated by graph theory. A system may be described as a directional graph G(V:E) (Ref. 6), i.e. a set of vertices V (the components) connected by directed arcs or edges E (interfaces). For example: the spacecraft system of Fig. 2 is presented by the graph given in Fig. 6a.
We further define a "circle" as follows:
Definition II:
A "Circle" is a subset M of vertices in a graph with the following attribute:

There exists a path (made by sequential edges) from every vertex in M to every other vertex in M (this path is, at most 2*(|M|-1) in length).

Remark:

From Definitions I and II we conclude that two sub-systems (two disjoint sets of systems components) are independent if and only if they do not belong to the same circle.

We use Definition II to define an "isolated vertex" as follows:

Definition III:

A vertex will be called "An Isolated Vertex" if it does not belong to a circle.

Having this definition we define *four mutually exclusive* types of sub-systems, namely: "an isolated system"; "a source system"; "a destination system"; and "a connector".

Definition IV:

"An Isolated System" is a subset L of vertices in a graph with the following attributes:

1. There are no edges, E, connecting L to the rest of the vertices in the graph.
2. L is either a circle or an isolated vertex.

Definition V:

"A Source System" – Is a subset S of vertices in a graph with the following attributes:

1. Any connections between S and the rest of the vertices in the graph exist only on edges going from S.
2. S is either a circle or an isolated vertex.

Definition VI:

"A Destination System" is a subset D of vertices in a graph with the following attributes:

1. Any edges connecting D and the rest of the vertices in the graph exist only on edges going into D.
2. D is either a circle or an isolated vertex.

Definition VII:

"A Connector" is a subset C of vertices in a graph with the following attributes:

1. Edges going into C only from subsets of type S or C.
2. Edges going out of C only to subsets of type C or D.
3. C is either a circle or an isolated vertex.

The problem is to partition a given system (graph) into the maximal number of L, D, S and C sub-systems thus obtaining a simple-flow structure representation.

3. ALGORITHM FOR SYSTEM PARTITIONING

Given a directional graph G(V:E). Clearly, we can identify vertices that are L, D, S, or C type *single-element* sub-systems. By eliminating them from the original graph, we get a new graph, G'(V':E'), which contains a collection of vertices that are joined together by circles. Our goal is to identify such circles in the graph.

Every time a circle M will be identified, we will replace *all* the vertices within that circle with a single vertex, m, and we will re-iterate the process.

The algorithm has 6 steps:

Step 1:
Go over the list of edges and eliminate all the "self pointing edges", i.e. eliminate all the edges that connect a vertex to it self.
For each vertex vi in the graph, check if it qualifies as a vertex of type L, S, or D.
If the vertex qualifies, this vertex is a "sub-system" – mark it as such.
Once the list has been exhausted, take every marked vertex off the graph; put the vertex into the corresponding list of "sub-systems" (list of type L sub-systems, list of type S sub-systems or list of type D sub-systems), and delete the edges going to or from each of these vertices.

Step 2:
Go over the list of the remaining vertices and check if it qualifies as a vertex of type L, S, or D. If a vertex vi qualifies, vi is a "sub- system" of type C – mark it as such.
Once the list has been exhausted, take every marked vertex off the graph; put the vertex into the corresponding list of "sub-systems" (list of type C sub-systems), and delete the edges going to or from each of these vertices.
Repeat this step until either all the vertices in the graph have been marked or you haven't managed to mark any vertex.
If all the vertices are marked off – go to *step 6*; otherwise go to the *next step*.
Step 3:
Choose a vertex – v. This is your *current vertex*
Step 4:
Choose an edge, e, which reaches the current vertex.
Mark the vertex v as "been there" and back propagate to the source of the edge, v', with a list including the identity of v and the identities of all previously marked vertices.
If v' is already "marked", you've found a circle, M, which includes all the vertices which are on the new list and weren't on any previously obtained list. Go to the *next step*.
If v' is not yet marked, *repeat this step* (with v' being the current vertex).
Step 5:
Replace all the vertices in M with a single vertex, m.
For every vertex, v, in the graph, that has an edge going from it and connecting it to M, mark an edge going from v to m.
For every vertex, v, in the graph, that has an edge going from a vertex in M to him mark an edge going from m to it.
Once the replacement is done, go back to *step one*, with a new graph, containing m instead of M.
Step 6:
Replace every vertex of type m found by the algorithm, with the list of vertices M corresponding to it (which makes out a circle) and arrange the groups in the N^2 graph.
Place all the vertices of type S at the top of the graph, place all the vertices of type D at the end of the graph, and place all the vertices of type C in the middle of the graph in a simple-flow fashion. *Stop* the algorithm.

Remarks:

1. In essence, what we are "looking after" are groups of vertices which no circle can be found connecting them.

2. Any iteration in the fifth step causes our graph to get smaller. Because of the fact that the number of circles in a given graph is finite, and we perform a reduction of a group of vertices into a single vertex, the algorithm converges and will finally stop.

4. N^2 MATRIX FORMULATION

Let Mat be an N^2 matrix. Thus, Mat[i,i]=0 and for i≠j, Mat[i,j] = 1 if and only if there exists an arc going from vertex i to vertex j in G(V:E), otherwise, Mat[i,j] = 0. For convenience, we will continue to refer to the diagonal elements as vertices, thus vi is the element corresponding to Mat[i,i].
If the matrix Mat is triangular the system is in *simple flow*. Our objective is to transform any non-triangular matrix into a triangular equivalent matrix by combining elements that are in *closed-loop* into a single element and reorganizing the chart.
The various definitions of Section 2 take now the following form:

Definition II':
A circle – two vertices, v1, v2 belong to a circle iff Mat contains a series of the form: Mat[v1, vn_1], Mat[vn_1, vn_2], …Mat[vn_i, vn_j], Mat[vn_j, vn_k], …, Mat[vn_M, v1], such that the product of the series element equals 1.
Definition III':
A vertex, vi, will be called "an isolated vertex" iff, for every i≠j vi and vj do not belong to the same circle.
Definition IV':
"An Isolated System" L has the following attributes:

1. If vi belongs to L, and vj does not belong to L, then Mat[i,j] + Mat[j,i] = 0.
2. L is either a circle or an isolated vertex

Definition V':
"A Source System" S has the following attributes:

1. If vi belongs to S and vj does not belong to S, then Mat[j,i] = 0.
2. If vi belongs to a source system, then vi does not belong to an isolated system. i.e. there exist vertices vi and vj such that Mat[i,j] = 1.
3. S is either a circle or an isolated vertex.

Definition VI':
"A Destination System" D has the following attributes:

1. If vi belongs to D and vj does not belong to D, then Mat[i,j] = 0.
2. If vi belongs to a source system, then vi does not belong to an isolated system. i.e. there exist vertices vi and vj such that Mat[j,i] = 1.
3. D is either a circle or an isolated vertex.

Definition VII':
"A Connector" C has the following attributes:

1. If vi belongs to C, and vj does not belong to C, then

Mat[j,i] = 1 implies that vj belongs to subsets of type S or C,
and, similarly
Mat[i,j] = 1 implies that vj belongs to subsets of type C or D

2. C is either a circle or an isolated vertex.

We can now reformulate the partitioning algorithm:
Step 1:
Construct an N dimensional qualification vector q.
Go over the list of vertices in Mat, and check if it qualifies as a vertex of type L, S, or D. If a vertex, vi qualifies, this vertex is a "sub-system" – mark it as such in the corresponding place in the qualification vector.
Once the list has been exhausted, for every marked vi do the following:
for every j, Mat[i,j] = 0 and Mat[j,i] = 0 (i.e. eliminating the edges in the row and column where vi appears).
Step 2:
For each vertex vi in Mat, check if it qualifies as a vertex of type L, S, or D. If a vertex vi qualifies, vi is a "Connector" – mark it as type C in the qualification vector. Once the list has been exhausted, for every marked vertex vi do the following: for every j, Mat[i,j] = 0 and Mat[j,i] = 0.
Repeat this step until all the vertices in the graph have been marked or you haven't managed to mark any vertex.
If all the vertices are marked – go to *Sixth step*; otherwise go to the next step.
Step 3:
Choose a vertex – vi.
Build an auxiliary vector named Vector, of length N. Initialize: for i=1 to N, Vector[i] = {Ø}. Each cell in Vector will hold the set of cells we have "traveled through" in order to get to this cell.
Step 4:
For each j, go over the i-th column of Mat until the first j that satisfies Mat[j,i] = 1.
Check Vector[j]. If Vector[j] ≠ {Ø}, you've found a circle, M, which includes all the vertices which are on the newly arrived list and weren't on the previously arrived list: M ={Vector[i] ∩ Vector[j] , j}. Go to the *next step.*

If Vector[j] = {Ø}, we haven't passed through this vertex yet, fill this cell with the list of the cells which is found in Vector[i] and add j to that list (Vector[j] = Vector[i],j). Make vj your new current vertex and *repeat the step*
Step 5:
Reorganize Mat by uniting rows and columns of members of M into a single row and column, and *go to step 1*.

5. EXAMPLES

Example I
We consider first the spacecraft system example of Fig. 6b. Fig. 6b describes the steps leading to the conclusion that this system cannot be divided into two systems (or more). The sign '*' is used for each backward walk. As seen, the grouping does not allow for any separation of the system as the components are accumulating together. Notice that had the primary user been passive (not transmitting data request, etc.) then the system would be divided into two. This is the case for the GPS systems where the users are not part of the system (under our definition).

Example II

Next consider the system of Fig 3 and we start the algorithm with the permutation version of Fig 4. The process is described in Fig. 7. Each step starts at the first entry (1,1) of the N^2 matrix.

The first two steps combine elements 1,2 and 4. Elements 8 and 9 are combined in the following step. The algorithm then combines the block {4,1,2} with elements 5, 6 and 7, sequentially. At this point we can identify the block {7,6,5,4,1,2} as S-type node. Finally elements 10 and 11 are added to the block {9,8}, rendering the block {10,11,9,8} D-type node, and the algorithm stops. As a result we have two separate systems as was expected from the original partitioning.

Example III

The last example is presented in Fig. 8. We have 8 functions with interfaces as shown. First we identify F1 as S-type node, and we eliminate it from the N^2 chart. We then follow the circle 2->5->4->3->2 to get the block {5,4,3,2} as a single node. The circle {5,4,3,2}->8->{5,4,3,2} adds F8 to the block, and turns it into it a connector, i.e. C-type node.

Combining elements 6 and 7 into a D-type node concludes the analysis.

6. CONCLUSIONS

High-level complex systems design can be greatly facilitated by systems partitioning into constituents such that independent sub-systems will be identified. In case of a single vendor it will simplify the system definition, the interfaces and modules development and finally the integration and verification processes. In case of multiple vendors it may additionally help to define the responsibly and work sharing in an efficient way.

Independency of sub-systems in this paper is defined pairwise. The partitioning sought for is such that all sub-systems are mutually independent. Consequently, sub-systems are classified into four classes: isolated, source, connector and destination sub-systems.

The problem of systems partitioning was formulated by the systems engineering approach of N^2. Graph theory provided the basis for the algorithm to resolve the problem. The algorithm is based on simple operations performed in a carefully defined procedure by which sub-systems are sequentially eliminated until the process is completed. The algorithm was transformed back to the N^2 matrix formulations by which several examples were studied. The algorithm is easy to implement and can be easily computerized using MATLAB or any other Matrix library.

Fig 1: N^2 Chart

Element # 1		Interfaces from #1 to #3	• • •	
	Element # 2			
		Element # 3		
• • •			• • •	
	Interfaces from #N to #2			Element # N

Fig 2: N^2 Example

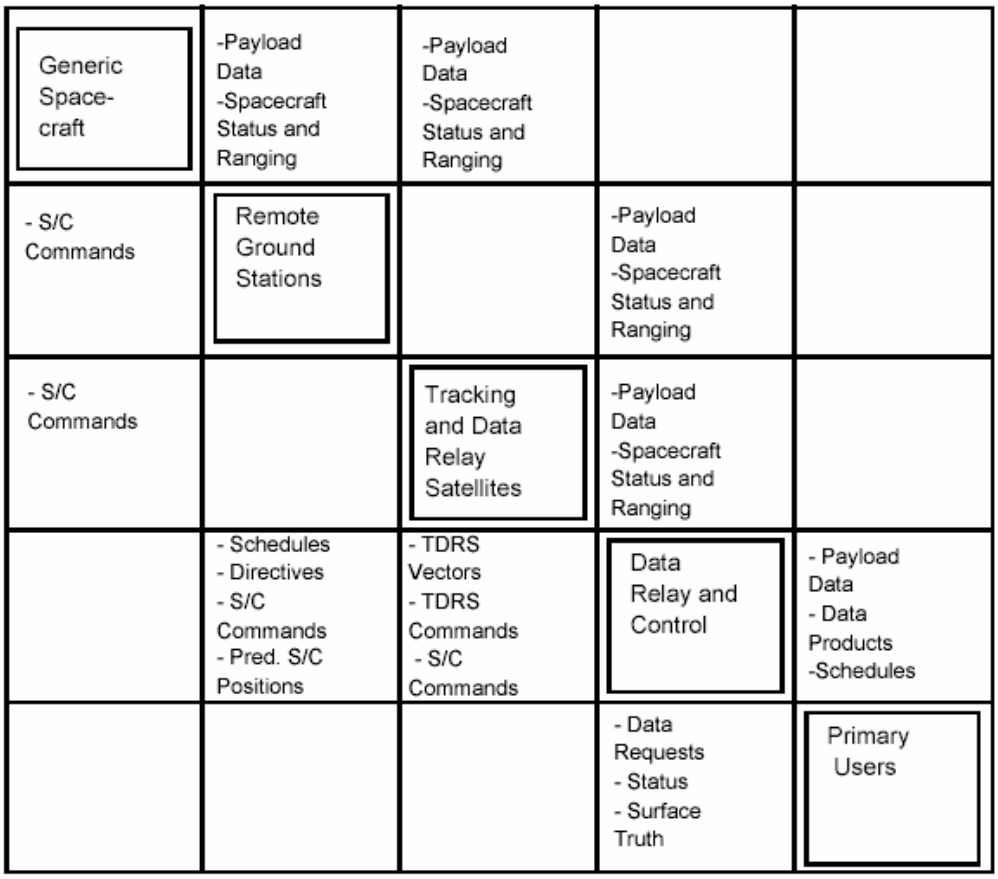


Fig 3: N^2 Features

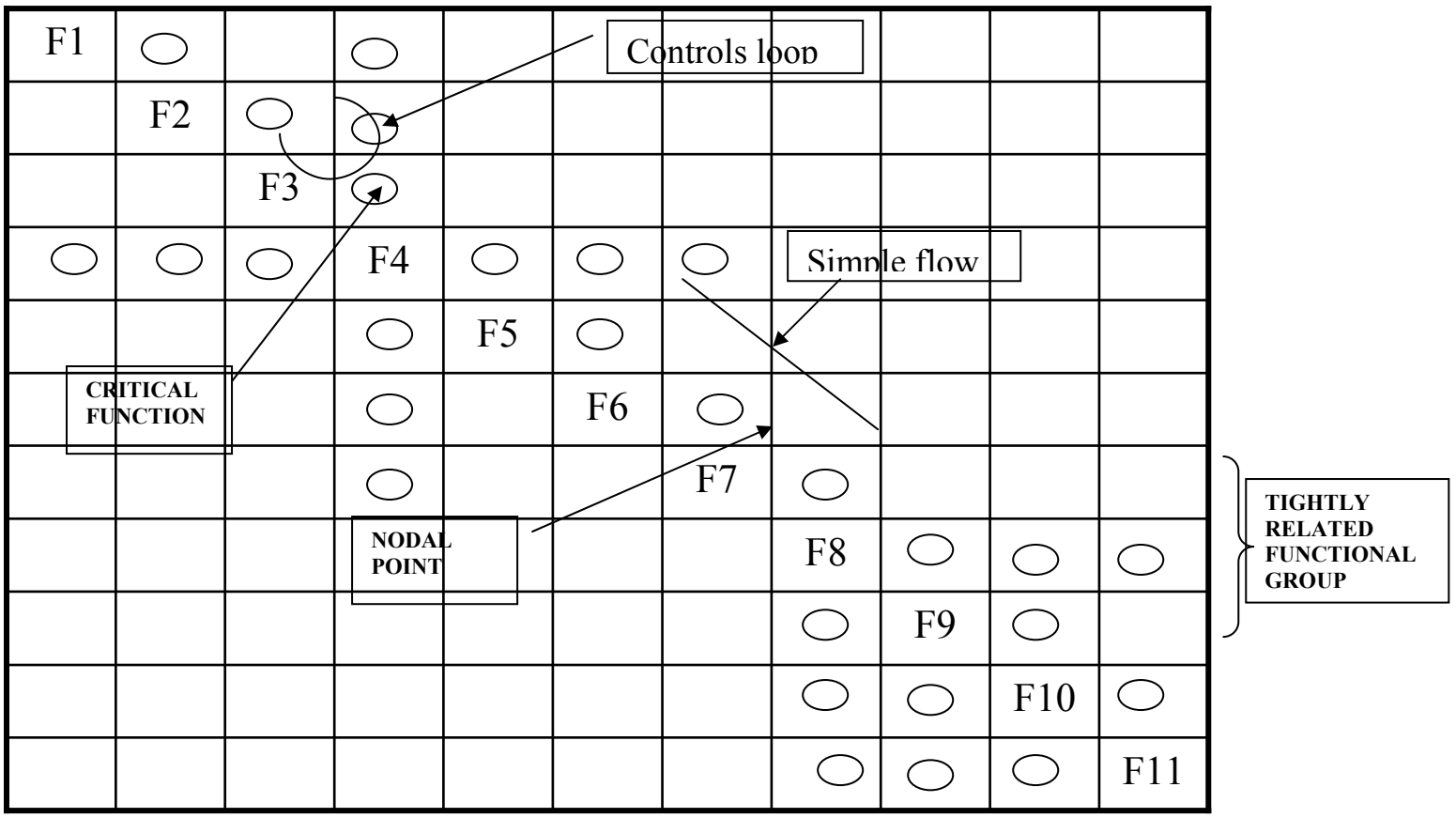
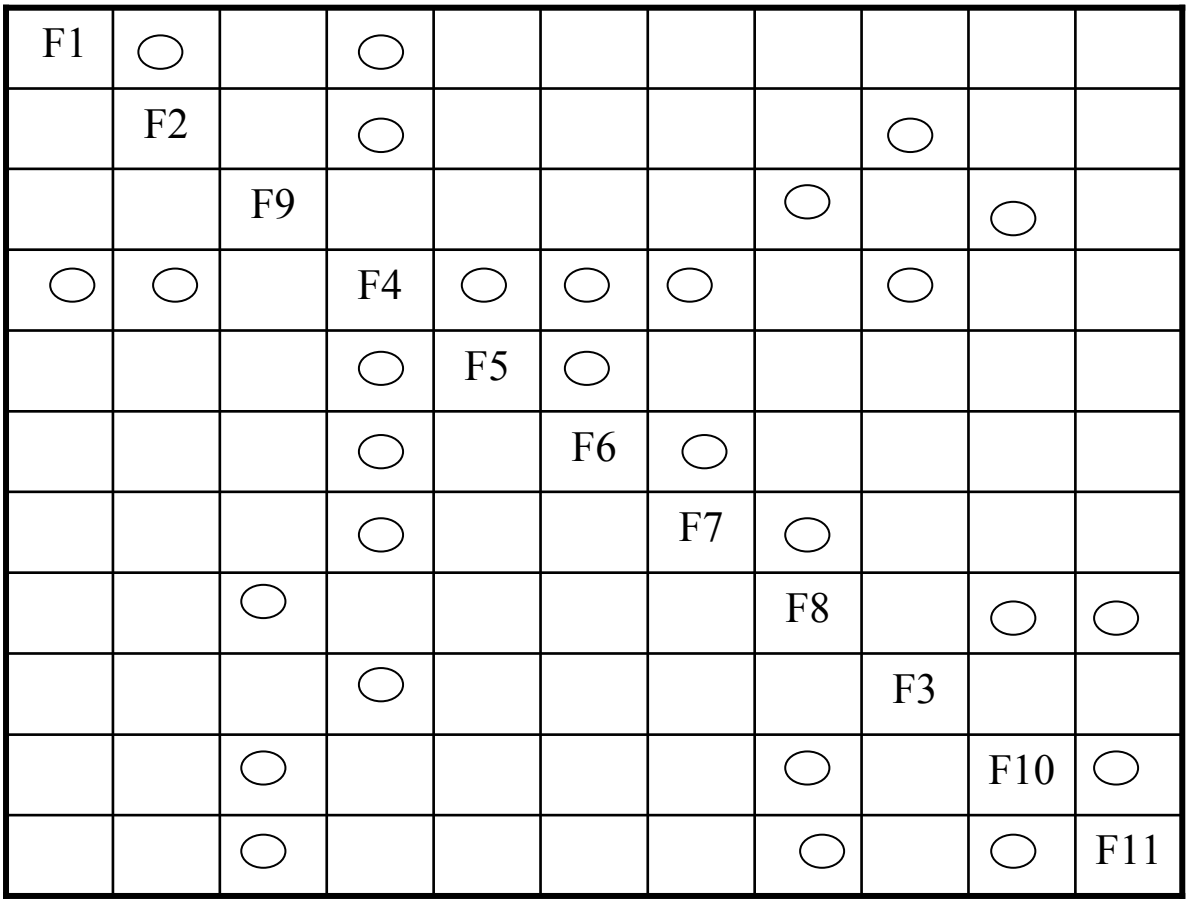


Fig. 4: Permutation Example



E1	1	
	E2	1
		E3

Fig. 5a : Independent Sub-systems

E1	1	
	E2	1
1		E3

Fig. 5b : Dependent Sub-systems

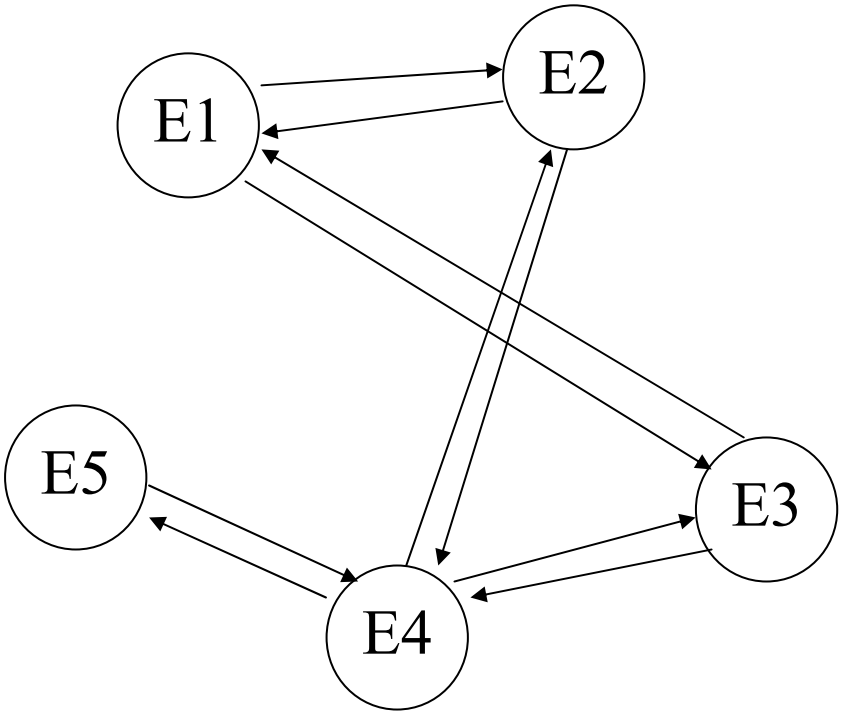


Fig 6a. Spacecraft Example

F1*	1	1		
1	F2*		1	
1		F3	1	
	1	1	F4	1
			1	F5

F12 *	1	1	
1	F3*	1	
1	1	F4	1
		1	F5

F321*	1	
1	F4*	1
	1	F5

F4321 *	1
1	F5*

F54321

Fig 6b. Spacecraft Example analysis (Example I)

F1 *	1		1							
	F2		1					1		
		F9					1		1	
1	1		F4 *	1	1	1		1		
			1	F5	1					
			1		F6	1				
			1			F7	1			
		1					F8		1	1
			1					F3		
		1					1		F10	1
		1					1		1	F11

F2*		1					1		
	F9					1		1	
1		F41*	1	1	1		1		
		1	F5	1					
		1		F6	1				
		1			F7	1			
	1					F8		1	1
		1					F3		
	1					1		F10	1
	1					1		1	F11

Fig 7. Analysis of Example II

F9*					1		1	
	F412	1	1	1		1		
	1	F5	1					
	1		F6	1				
	1			F7	1			
1					F8*		1	1
	1					F3		
1					1		F10	1
1					1		1	F11

F412*	1	1	1		1		
1	F5*	1					
1		F6	1				
1			F7	1			
				F89		1	1
1					F3		
				1		F10	1
				1		1	F11

Fig 7. Analysis of Example II (cont'd)

F5412 *	1	1		1		
1	F6*	1				
1		F7	1			
			F89		1	1
1				F3		
			1		F10	1
			1		1	F11

F65412 *	1		1		
1	F7*	1			
		F89		1	1
1			F3		
		1		F10	1
		1		1	F11

Fig 7. Analysis of Example II (cont'd)

F765412 *	1	1		
	F89		1	1
1		F3*		
	1		F10	1
	1		1	F11

F89*		1	1
1	F3765412 S		
1		F10*	1
1		1	F11

F3765412 S	1	
	F10 89*	1
	1	F11 *

F3765412 S	1
	F11 10 8 9 D

Fig 7. Analysis of Example II (cont'd)

F1 -**S**	1	1	1	1	1		
	F2	1			1		1
		F3	1				
		1	F4	1	1	1	
	1	1	1	F5	1		
					F6	1	
					1	F7	
		1			1		F8

F2*	1		1	1		1
	F3*	1				
	1	F4*	1	1	1	
1	1	1	F5*	1		
				F6	1	
				1	F7	
	1			1		F8

Fig 8. Analysis of Example III

F5432 *	1	1	1
	F6	1	
	1	F7	
1	1		F8*

F54328 **C**	1	1
	F6*	1
	1	F7*

F1 **S**	1	1
	F654328 **C**	1
		F67 **D**

Fig 8. Analysis of Example III (cont'd)

REFERENCES

[1] T.C. Robertson (Editor), INCOSE Systems Engineering Handbook, Version 2.0, July 2000, p. 10
[2] Blanchard, B. S. and Fabrycky, W. J., Systems Engineering and Analysis (2nd ed.).Prentice-Hall, Englewood Cliffs, N.J.,1990, pp. xx
[3] Ian Sommerville, Software Engineering, , 6th edition, Chapter 20
[4] Becher, O., Ben-Asher, J. Z. and Ackerman I, "A Method for System Interface Reduction Using N^2 Charts," Systems Engineering – The J. of The Int. Council on Systems Engineering, Jan., 2000
[5] R.S. Shishko (Editor), National Astronautics and Space Administration (NASA), Systems Engineering Handbook, SP-6105, June 1995.
[6] F. Harary, Graph theory, Addison-Wesley Series in Mathematics, Addison-Wesley, Reading, MA, October 1972.