INT. J. PROD. RES., 1993, VOL. 31, NO. 4, 753–769

Efficient organizing of design activities

A. KUSIAK† and J. WANG‡

Concurrent design should result in reduction of the duration of a design project, cost reduction, and better quality of the final design; however, it may increase the complexity of the design process and make it more difficult to manage. In this paper, an algorithm is developed for organizing design activities in order to effectively produce an acceptable design. The relationship among design activities is represented with an incidence matrix and the corresponding directed graph. The design process is simplified by identifying and analysing design activities that are coupled. The algorithm presented in the paper generates a sequence of design activities such that the number of cycles is minimized, i.e. the product development time is reduced. The concepts presented are illustrated with examples.

1. Introduction

Product design has been an important task in business for many years. Due to decreasing product life cycles, it is important to reduce the time and cost of product development. Thus, product design has become a focus of competition in many industries. In recent years, a concept of concurrent design has emerged. It attempts to incorporate various constraints related to the product life cycle, i.e. manufacturability, quality, reliability, and so on, in the early design stages. Concurrent design aims at improvement of the product quality, reducing the development time and cost. However, due to interactions between the various facets of the design process, concurrent design may increase the complexity of design process and make it more difficult to manage. In this paper, design process is described as a sequence of interrelated activities that begins with an initial set of customer requirements and ends with a complete description of a product that satisfies these requirements. To simplify and increase the efficiency of the design process, one needs to analyse the information flow between design activities and organize them, accordingly.

In this paper, an activity–activity incidence matrix and the corresponding directed graph (digraph) are used to represent design activities and the relationship among them (Warfield 1973, Steward 1981 a, Kusiak 1993). A non-empty element in an incidence matrix represents a relationship between the corresponding (row and column) activities. The objective is to organize the incidence matrix so that the overall design task is simplified. The organized matrices can be categorized as follows (see Fig. 1): uncoupled matrix, decoupled matrix, and coupled matrix. An incidence matrix is uncoupled if its rows and columns can be ordered in such a way that the matrix separates into mutually exclusive submatrices (see the matrix in Fig. 1 (*a*)). Analogously, an incidence matrix is decoupled if it can be rearranged in a triangular form (see the matrix in Fig. 1 (*b*)). Otherwise, the incidence matrix is coupled (see the matrix in Fig. 1 (*c*)).

Received May 1992.

† To whom correspondence should be addressed.

‡ Intelligent Systems Laboratory, Department of Industrial Engineering, The University of Iowa, Iowa City, IA 52242, USA.

0020–7543/93 $10·00

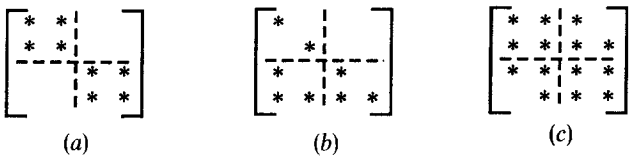


Figure 1. Three types of matrices: (*a*) uncoupled matrix; (*b*) decoupled matrix; (*c*) coupled matrix.

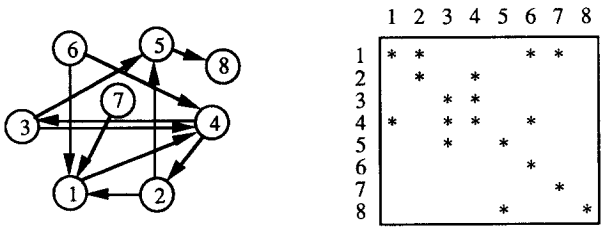


Figure 2. Digraph of activities and the corresponding incidence matrix.

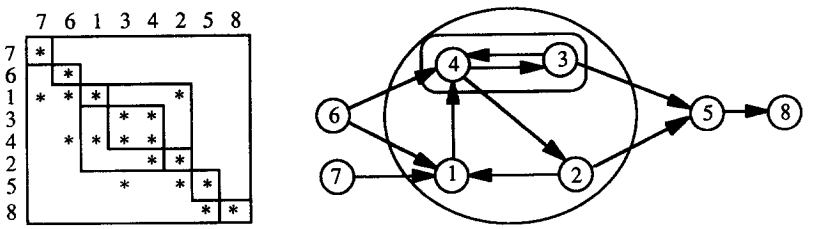


Figure 3. Ordered incidence matrix and the corresponding digraph.

For example, consider six hypothetical design activities represented with a digraph and the corresponding activity–activity incidence matrix (see Fig. 2). An entry a_{ij} = '*' in the incidence matrix means that activity (column) j precedes activity (row) i.

No special structure is visible in the incidence matrix in Fig. 1. The problem considered in this paper is to order rows and columns of the incidence matrix into a lower triangular form with the minimum number of non-empty elements in the upper triangular matrix. Transforming the matrix in Fig. 2 results in the matrix and the corresponding digraph shown in Fig. 3.

The following two facts can be easily observed in the matrix (or digraph) in Fig. 3:

(1) the precedence relationship between activities;
(2) the coupled activities (called strongly connected components in graph theory).

Activities 6 and 7 are independent and they can be performed simultaneously. Activity 5 has to be performed before activity 8. Activities 1, 2, 3, and 4 are coupled and might be performed iteratively. For a better visual effect, the coupled activities 1, 2, 3 and 4 have been circled in the digraph in Fig. 3.

The algorithm developed in this paper organizes an overall design task into groups of activities, simplifies the entire design process, and generates a sequence of design activities that minimizes the product development time. To reduce the number of cycles

in the design process, the algorithm minimizes the number of non-empty elements in the upper triangular incidence matrix.

The remainder of this paper is organized as follows. Section 2 provides a review of the related literature. Section 3 presents a triangularization algorithm that transforms the activity–activity incidence matrix into the triangular form. The algorithm is illustrated with three examples. Example 1 shows the decomposition of a hypothetical system of design activities. Analysis of the transformed incidence matrix is provided. Example 2 considers the activities involved in design of a passenger vehicle. The triangularization algorithm applied to constraint management in conceptual design (Steward 1965, 1981 a) is illustrated in Example 3. Section 4 concludes the paper.

2. Literature review

The literature on decomposition of graphs and matrices is rather extensive. Graph theory text books present frequently the 'power method' for finding the strongly connected components in digraph (Deo 1974). The advantage of the 'power method' is its simplicity; however, the method is not efficient. Tarjan (1972) developed an algorithm for identification of strongly connected components in a digraph in a way that each arc is traversed exactly once and each vertex is visited at least once. Mathematical programming researchers have been also working with the decomposition problem. For example, Weil and Kettler (1971) presented a systematic method for organizing matrices into a block-diagonal form applied for large linear programs.

Steward (1965) developed techniques for partitioning a system of equations into subsystems that need to be solved simultaneously. In the case when the matrix is not decomposable into mutually separable submatrices, further decomposition might be possible by removing one or more elements from the matrix. One of the frequently used approaches was to minimize the number of elements to be removed. Steward (1981 b) applied this approach for ordering activities involved in the design process. He also proposed a shunt diagram to analyse cycles in a digraph of design activities. Another method presented in the literature (Deo 1974) uses the absorption laws of Boolean algebra (such as $a \cdot a = a$, $a + a = a$, and $a + ab \doteq a$) to recognize a set of paths whose removal destroys cycles in a digraph. The methods discussed are not suitable for decomposition of large digraphs. If the digraph is large, the shunt diagram becomes complex and it is difficult to analyse. Similarly, the Boolean approach is not suitable for analysis of large systems.

More recently, Eppinger *et al.* (1990) used a variation of Steward's (1981 b) design matrix to represent both the sequence and technical relationships among design activities.

The heuristic presented next transforms an incidence matrix into a triangular form with the minimum number of non-empty elements in the upper triangular matrix.

3. Triangularization algorithm

The triangularization algorithm begins with elimination of rows and columns of the incidence matrix that include only one non-empty element, i.e. removal of vertices without any preceding arcs, and removal of arcs originating from the vertices removed in the corresponding digraph. This process reduces the size of the incidence matrix (digraph).

Next, the algorithm identifies all simple cyles (directed cycles) in the digraph using the Weinblatt's algorithm (Weinblatt 1972). A number of algorithms for finding simple cycles exists in the literature, for example Roberts and Flores (1966) and Tiernan (1970).

The Weinblatt's algorithm appears to be more efficient than any other existing algorithm. The efficiency comes from the fact that it determines cycles in such a way that each arc of the digraph is examined only once. Cycles are discovered either when the path being followed is found to be cyclic or by combining previously discovered cycles with the cycle under consideration.

After all the simple cycles have been identified in a digraph, strongly connected components are determined by fusing the simple cycles that have at least one vertex in common. Each strongly connected component can be represented with a vertex and all directed edges from one strongly connected component to another are represented with a single directed edge, so that the digraph becomes a condensation digraph. A digraph with all strongly connected components condensed does not include cycles. Readers interested in the formal definitions of a simple cycle, strongly connected component, and condensation digraph may refer to Deo (1974).

Next, the topological sorting algorithm (Horowitz and Sahni 1983) is used to rearrange the rows and columns of the incidence matrix into a lower triangular form. If each vertex represents a design activity, the lower triangular matrix implies that there are no cycles among activities.

Finally, the algorithm attempts to order the vertices in each strongly connected component in such a way that the number of cycles is minimized. This goal is accomplished by minimizing the number of non-empty elements in the upper triangular matrix. An ideal lower triangular matrix corresponds to the digraph without cycles (Deo 1974). The algorithm attempts to find the minimum number of non-empty elements to be removed to break all cycles in a strongly connected component. The element being removed is determined by selecting an edge that appears most frequently in all the cycles of the digraph. After all the cycles in the strongly connected component have been broken, the topological sorting algorithm is used for ordering the vertices. For example, the digraph in Fig. 4 is strongly connected. Edges (4, 3) and (5, 6) are selected to break the cycles in this strongly connected component. The result is shown in Fig. 5. There are only two non-empty elements in the upper triangular matrix in Fig. 5. This means that only two cycles are left, if the activities are performed in the sequence (6, 1, 3, 4, 6, 5).

The steps of the triangularization algorithm are presented next.

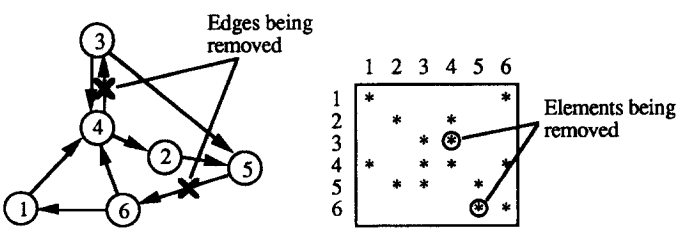


Figure 4. Edges are removed to break cycles in the strongly connected component.

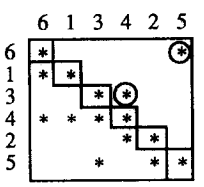


Figure 5. Ordered incidence matrix from Fig. 4.

3.1. *Steps of the triangularization algorithm*

Step 1. Elimination of rows and columns in the incidence matrix:

(*a*) Delete from incidence matrix $[a_{ij}]$ each row and column with only one non-empty element (corresponding to a vertex with no arcs terminating or arcs originating) in the row or column. Indicate the deletion by drawing a broken horizontal and vertical line.

(*b*) If each of the remaining rows and columns have more than one non-empty element besides the diagonal elements, go to Step 2; otherwise, repeat from (*a*).

Step 2. Find all simple cycles with the Weinblatt's algorithm (Weinblatt 1972).

Step 3. Condensation:

Merge the simple cycles that have at least one vertex in common in a condensation graph G_i, $i=1,\ldots,n$.

Step 4. Sorting of the incidence matrix $[a_{ij}]$:

Use the topological sorting algorithm to order vertices of the incidence matrix (Horowitz and Sahni 1983).

Step 5. Triangularization of elements of condensation digraphs G_i, $i=1,\ldots,n$:

(*a*) For each edge in each cycle, determine the frequency of occurrence in all the cycles, i.e. the number of edges identical with the edge considered.

(*b*) Select an edge with the maximum frequency of occurrence for removal.

(*c*) If no cycle is left, go to (*d*); otherwise go to (*a*).

(*d*) Use the topological sorting algorithm to order the vertices in each G_i.

The triangularization algorithm is illustrated in Example 1.

Example 1

Consider the digraph of design activities and the corresponding activity–activity incidence matrix presented in Fig. 6.

Step 1. The vertices that do not form cycles are eliminated (Fig. 7):

It is observed from the matrix in Fig. 6 that row 1 has only one non-empty element. Thus, removing row 1 and column 1 from the matrix in Fig. 6 results in matrix (1). Row

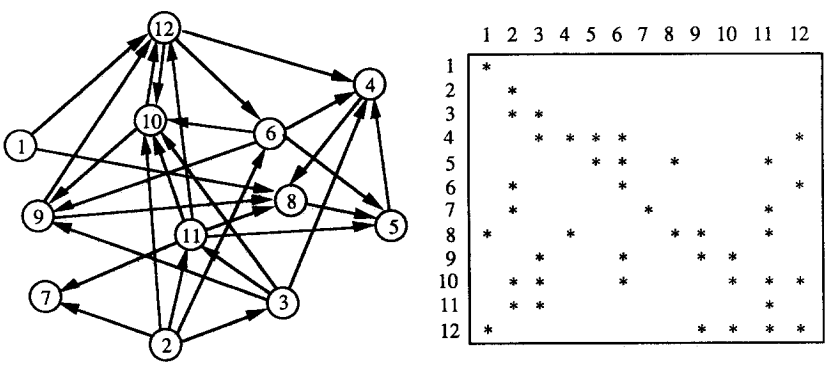
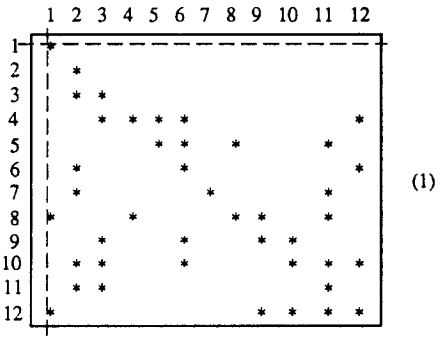


Figure 6. A digraph and the corresponding incidence matrix with 12 activities.



(1)

(2)

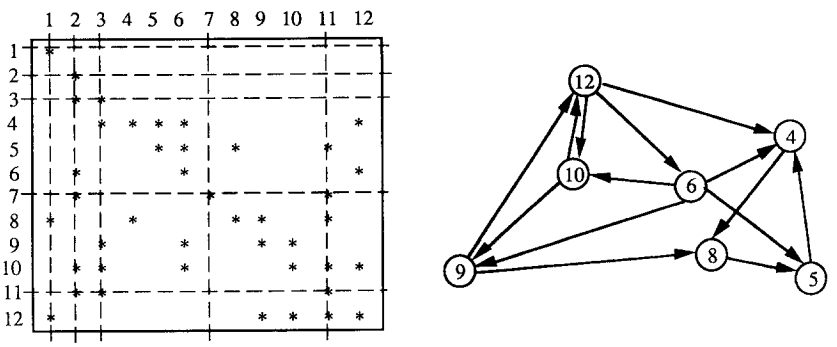


Figure 7. A simplified incidence matrix and the corresponding digraph.

2 has only one non-empty element. Removing row 2 and column 2 from the matrix (1) results in matrix (2). Similarly, rows and columns 3, 7, and 11 can be removed from matrix (2) which results in the matrix shown in Fig. 7.

Step 2. Find all simple cycles:

Weinblatt's algorithm (Weinblatt 1972) is used to find all simple cycles in the directed graph in Fig. 7, and are listed in Table 1.

Step 3. Condensation:

The cycles C_1, C_2, C_3, C_4, and C_5 are merged into G_1 because they share common vertices. Also, the vertices in cycle C_6 are merged into G_2. The digraph in Fig. 7 with merged cycles is shown in Fig. 8.

Cycle number	Cycle vertices
C_1	(6, 9, 12, 6)
C_2	(6, 10, 12, 6)
C_3	(10, 12, 10)
C_4	(12, 10, 9, 12)
C_5	(6, 10, 9, 12, 6)
C_6	(8, 5, 4, 8)

Table 1. Simple cycles in the digraph in Fig. 7.

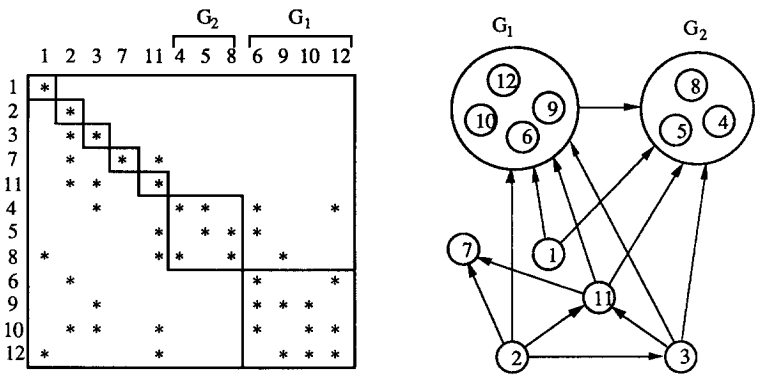


Figure 8. Incidence matrix and the corresponding condensation digraph of design activities.

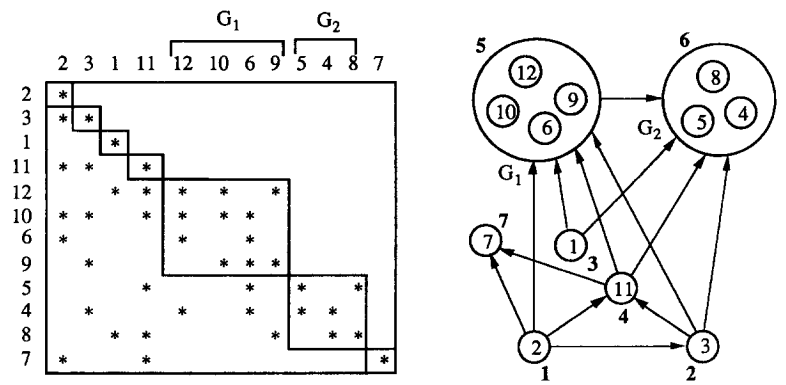


Figure 9. Ordered matrix and the corresponding condensation digraph.

Step 4. Sorting:

Using the topological sorting algorithm (Horowitz and Sahni 1983) results in the ordered matrix and digraph shown in Fig. 9. The bold numbers in Fig. 9 denote the sequence of vertices.

Step 5. Subdigraphs G_1 and G_2 are considered.

(1) Subdigraph G_1

(*a*) Five cycles C_1, C_2, C_3, C_4, and C_5 shown in Table 1 are included in subdigraphy G_1. The frequency of occurrence of each edge of the cycles in subdigraph G_1 is shown in Table 2.
(*b*) Edges (9, 12) or (12, 6) can be selected because they have the same frequency of occurrence.
(*c*) If edge (9, 12) is selected for removal, cycles C_2 and C_3 are still left. Go to (*a*).
(*a*) The edges included in the existing cycles C_2 and C_3 are (6, 10), (12, 6), (12, 10), and (10, 12). The frequency of occurrence of each edge is shown in Table 3.
(*b*) Edge (10, 12) is selected for removal.
(*c*) Since there is no cycle left, go to (*d*).
(*d*) Using the topological sorting algorithm to order elements of the submatrix, results in the matrix in Fig. 10. The bold numbers denote the sequence of vertices.
(*c*) If edge (12, 6) is selected for removal, cycles C_3 and C_4 are still left. Go to (*a*).
(*a*) The edges included in the existing cycles C_3 and C_4 are (9, 12), (10, 9), (12, 10), and (10, 12). The frequency of occurrence of each edge is shown in Table 4.
(*b*) Edge (12, 10) is selected for removal.
(*c*) Since there is no cycle left, go to (*d*).
(*d*) Ordering the submatrix with the topological sorting algorithm produces the result shown in Fig. 11.

From the above analysis, we know that selecting for removal of either edge (9, 12) or (12, 6) produces the same number of non-empty elements in the upper triangular matrix.

(2) Subdigraph G_2

(*a*) Subdigraph G_2 has the cycle C_6 shown in Table 1. The frequency of occurrence of each edge is shown in Table 5.
(*b*) Edge (8, 5) is randomly selected for removal.
(*c*) Since there are no more cycles in the subdigraph, go to (*d*).
(*d*) The resultant matrix and subdigraph are shown in Fig. 12.

Edges	(6, 9)	(9, 12)	(10, 9)	(6, 10)	(12, 6)	(12, 10)	(10, 12)
Frequency of occurrence	1	3	2	2	3	2	2

Table 2. Frequency of occurrence of edges in the cycles included in subdigraph G_1.

Edges	(6, 10)	(12, 6)	(12, 10)	(10, 12)
Frequency of occurrence	1	1	1	2

Table 3. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_1.

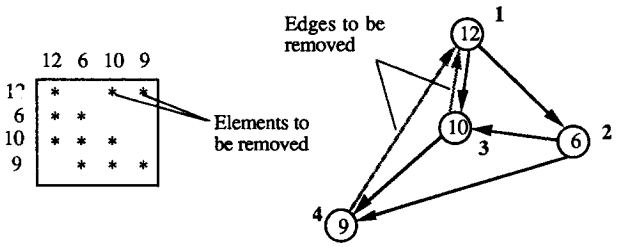


Figure 10. Ordered submatrix and the corresponding digraph G_1.

Edges	(9, 12)	(10, 9)	(12, 10)	(10, 12)
Frequency of occurrence	1	1	2	1

Table 4. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_1.

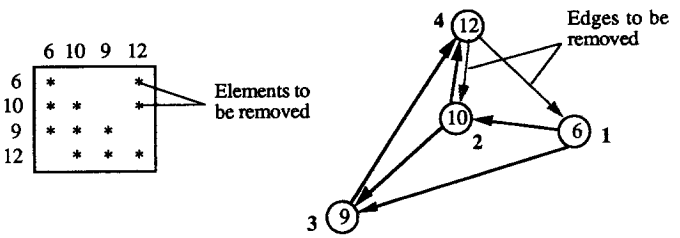


Figure 11. Ordered submatrix and the corresponding digraph G_1.

Edge	(8, 5)	(5, 4)	(4, 8)
Frequency of occurrence	1	1	1

Table 5. Frequency of occurrence of edges in the cycles included in the subdigraph G_2.

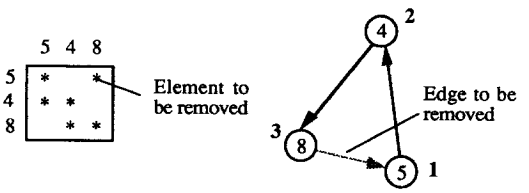


Figure 12. Ordered matrix and the corresponding subdigraph G_2.

Thus the result is shown in Fig. 12. There are only three non-empty elements in the upper triangular incidence matrix. The triangularization algorithm produces the following sequence of activities $(2, 3, 1, 11, G_1, G_2, 7)$ of activities. The matrix in Fig. 13 shows that activity 2 has to be performed before activity 3. Activities 1 and 3 are independent so that they can be accomplished in parallel. Also, there are two groups of coupled activities G_1 and G_2. To minimize the number of cycles, the activities in group G_2 should be performed in the sequence $(12, 6, 10, 9)$.

There are a number of ways of decoupling design activities. For example, to decouple the activities included in group G_2, one could introduce a new activity 13. The new activity might remove the precedence relationship (8, 5), i.e. element (8, 5) in the incidence matrix. The resulting matrix is shown in Fig. 14. The decoupling process may decrease the duration of the design project; however, it requires more information about the activities.

Application of the triangularization algorithm for arrangement of activities involved in design of a passenger vehicle is illustrated in Example 2.

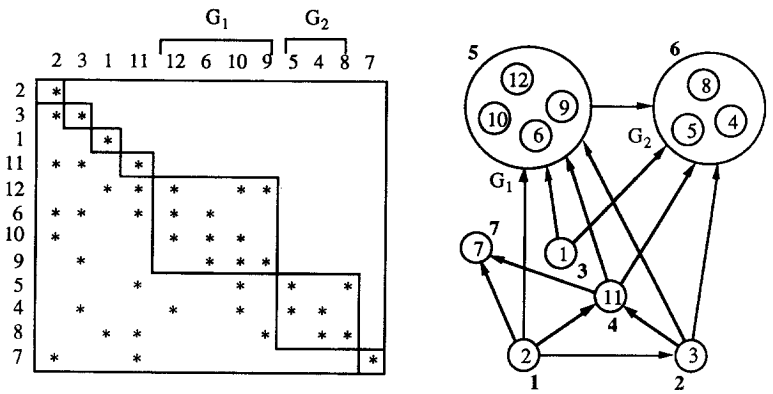


Figure 13. Final result of Example 1.

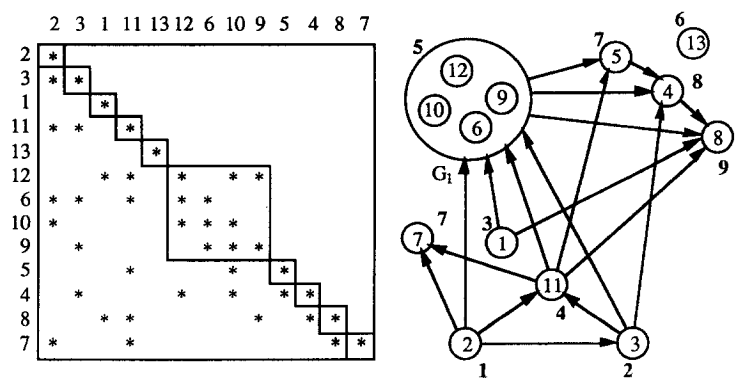


Figure 14. Resulting matrix and the corresponding digraph after decoupling.

Example 2

Consider design of a vehicle for which the number of passengers (capacity), cruising speed, acceleration, and range have been specified. The major subsystems of the vehicle are shown in Fig. 15. Each subsystem corresponds to a set of design activities. Table 6 shows precedence relationships among design activities of a vehicle. For example, to begin activity 6, activities 2, 3, 7, and 11 have to be completed. The steps of the algorithm are presented next.

Step 1. The vertices that do not form cycles are eliminated (see Fig. 16).

Step 2. All the simple cycles in the digraph in Fig. 16 are listed in Table 7.

Step 3. The cycles C_1, C_2, C_3, C_4 and C_5 are merged into G_1 because they share common vertices. Also, the vertices in cycle C_6 are merged into G_2 (Fig. 17).

Step 4. Applying the topological sorting algorithm to the matrix in Fig. 17 produces the ordered matrix shown in Fig. 18.

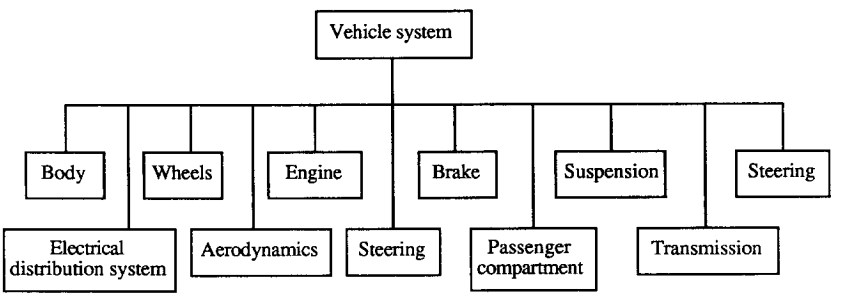


Figure 15. Subsystems of a vehicle.

Activity	Activity name	Predecessors
1	Capacity specification	None
2	Cruising speed specification	None
3	Acceleration specification	None
4	Range specification	None
5	Engine system design	2, 3, 4, 6, 7, 8, 11
6	Drive motor system design	2, 3, 7, 11
7	Passenger compartment design	1
8	Body design	5, 6, 7, 10, 11, 12
9	Electrical distribution system design	5, 6
10	Suspension system design	8, 13
11	Transmission system design	2, 3, 6
12	Steering mechanism design	6, 8
13	Wheels design	5, 6, 7, 10
14	Brake system design	1, 5, 6, 8, 10, 11, 15
15	Aerodynamics design	1, 2, 3, 6, 5, 10

Table 6. Precedence constraints among design activities.

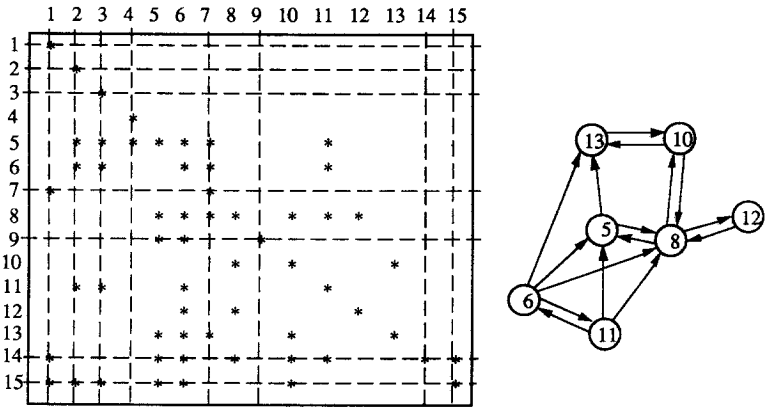


Figure 16. A simplified incidence matrix and the corresponding digraph.

Cycle number	Cycle vertices
C_1	(8, 5, 8)
C_2	(8, 5, 13, 10, 8)
C_3	(13, 10, 13)
C_4	(8, 10, 8)
C_5	(8, 12, 8)
C_6	(6, 11, 6)

Table 7. Simple cycles for the digraph in Fig. 16.

Step 5. Subdigraphs G_1 and G_2 are considered.

(1) Subdigraph G_1

(*a*) Subdigraph G_1 includes five existing cycles C_1, C_2, C_3, C_4 and C_5, shown in Table 7. The frequency of occurrence of edges for the cycles included in subdigraph G_1 is shown in Table 8.

(*b*) Edges (8, 5), (13, 10) or (10, 8) can be selected because they have the same frequency of occurrence.

(*c*) If edge (8, 5) is selected for removal, cycles C_3, C_4 and C_5 are still left. Go to (*a*).

(*a*) The edges included in the existing cycles C_3, C_4 and C_5 are (10, 8), (13, 10), (10, 13), (8, 10), (12, 8) and (8, 12). The frequency of occurrence for each edge is shown in Table 9.

(*b*) Edges (8, 10), (10, 13), (13, 10), (10, 8), (8, 12), or (12, 8) can be selected because they have the same frequency of occurrence.

(*c*) If edge (10, 8) is selected for removal, cycles C_3 and C_5 are still left. Go to (*a*).

(*a*) The edges included in the existing cycles C_3 are (10, 13), (13, 10), (8, 12), and (12, 8). The frequency of occurrence of each edge is shown in Table 10.

(*b*) Edges (10, 13), (13, 10), (12, 8), or (12, 8) can be selected because they have the same frequency of occurrence.

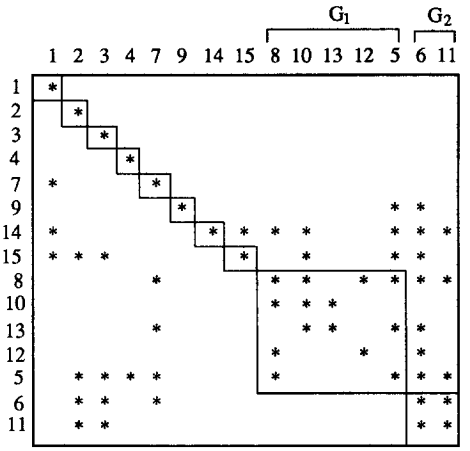


Figure 17. Condensation incidence matrix.

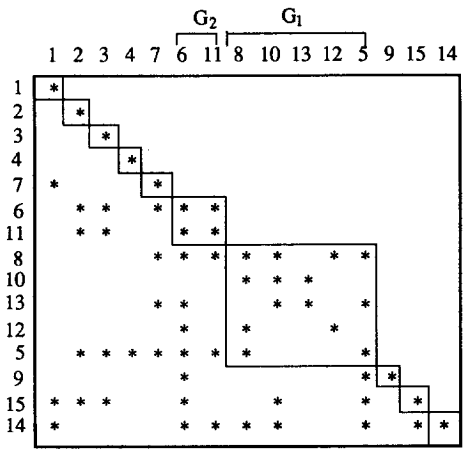


Figure 18. Ordered activity–activity incidence matrix.

Edge	(8, 5)	(5, 13)	(13, 10)	(10, 8)	(5, 8)	(10, 13)	(8, 12)	(12, 8)	(8, 10)
Frequency of occurrence	2	1	2	2	1	1	1	1	1

Table 8. Frequency of occurrence of edges in the cycles included in subdigraph G_1.

Edge	(13, 10)	(10, 8)	(10, 13)	(8, 12)	(12, 8)	(8, 10)
Frequency of occurrence	1	1	1	1	1	1

Table 9. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_1.

Edge	(13, 10)	(10, 13)	(8, 12)	(12, 8)
Frequency of occurrence	1	1	1	1

Table 10. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_1.

Edge	(8, 12)	(12, 8)
Frequency of occurrence	1	1

Table 11. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_1.

Edge	(8, 12)	(12, 8)
Frequency of occurrence	1	1

Table 12. Frequency of occurrence of edges in the cycles included in the modified subdigraph G_2.

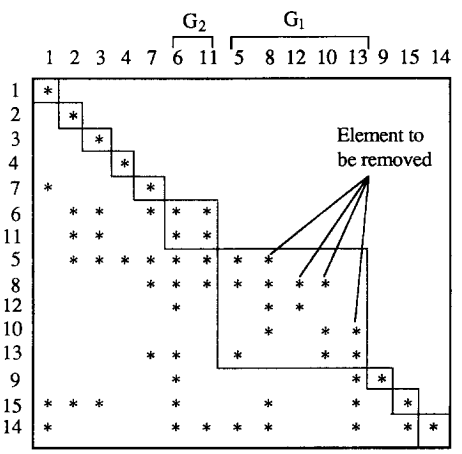


Figure 19. Ordered activity–activity incidence matrix.

(*c*) If edge (10, 13) is selected for removal, cycle C_5 still exists. Go to (*a*).
(*a*) The edges included in the existing cycle C_3 are (8, 12) and (12, 8). The frequency of occurrence of each edge is shown in Table 11.
(*b*) Edges (12, 8) or (12, 8) can be selected because they have the same frequency of occurrence.
(*c*) If edge (12, 8) is selected for removal, no cycle is left. Go to (*a*).
(*d*) The result of topological sorting is shown in Fig. 19.

(2) Subdigraph G_2

Subdigraph G_2 includes cycle C_6 shown in Table 7. The frequency of occurrence of edges in the cycles included in subdigraph G_2 is shown in Table 12. Either edge (6, 11) or (11, 6) can be selected for removal. Ordering matrix in Fig. 19 does not change the sequence of rows (columns).

The triangularization algorithm applied for constraint management (Steward 1965, 1981 a) in conceptual design is illustrated in Example 3.

Example 3

Consider design of the cantilever beam in Fig. 20.

The equations pertaining to design of the cantilever beam are as follows (Shigley and Mischke 1986):

$$f_1 \quad \sigma = \frac{MY}{I} \Rightarrow \sigma = \sigma(M, Y, I)$$

$$f_2 \quad M = FL \Rightarrow M = M(F, L)$$

$$f_3 \quad I = \frac{BH^3}{12} \Rightarrow I = I(B, H)$$

$$f_4 \quad Y = \frac{H}{2} \Rightarrow Y = Y(H)$$

$$f_5 \quad \Delta = \frac{FL^3}{3EI} \Rightarrow \Delta = \Delta(F, E, I, L)$$

where

σ	the maximum bending stress occurring at the section closest to the support
M	bending moment
L	length of beam
H, Y	geometry of beam
F	applied force
Δ	deflection parameter
E	modulus of elasticity
I	second moment of area about the neutral axis

Using the equations above, a designer can define variables of interest to obtain the design required. In this example, variables F, E, H, B, and σ are designed to be known.

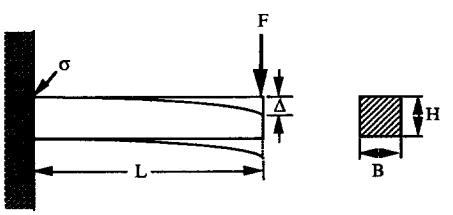


Figure 20. Cantilever beam.

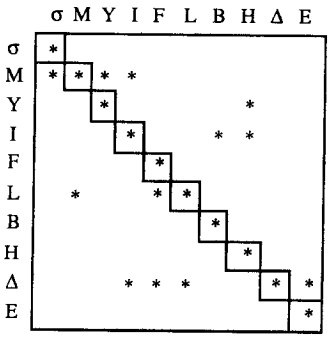


Figure 21. Incidence matrix corresponding to the set of transformed design equations.

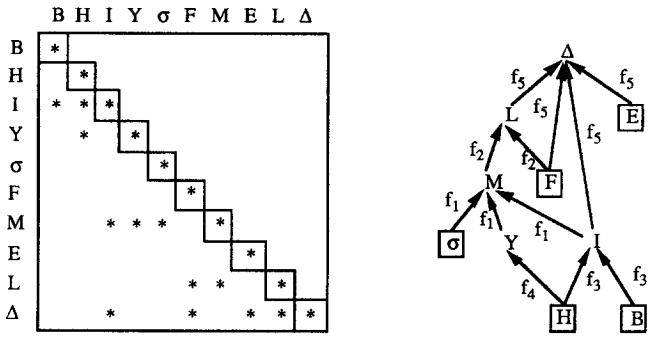


Figure 22. Ordered matrix and the corresponding digraph obtained from the incidence matrix in Fig. 21.

The unknown variables are M, I, L, Y, and Δ. To analyse the set of design equations, each equation should be transformed so that the known variables appear on the right-hand side. The transformed equations are as follows:

$$M = M(\sigma, Y, I) \qquad Y = Y(H) \qquad I = I(B, H)$$

$$L = L(M, F) \qquad \Delta = \Delta(F, L, E, I)$$

A set of equations can be represented with an incidence matrix in which the non-empty elements of the incidence matrix represent the relationship between variables (Fig. 21). For example, to evaluate the bending moment M, variables σ, Y, and I have to be obtained. If the design variables could be sequenced so that each would be able to receive all the information required, then there would be no coupling in the design process. Applying the triangularization algorithm to the incidence matrix in Fig. 21 results in the matrix shown in Fig. 22.

4. Conclusions

In concurrent engineering, an attempt is made to perform the design activities simultaneously rather than in the series as in the case of traditional design. This results in reduction of the duration of the design project, cost savings, and better quality of the final design. However, the concurrent strategy increases the complexity of the design process and makes it more difficult to manage.

In this paper, the triangularization algorithm was developed for organizing design activities to increase the efficiency of the design process. The design process was simplified by identifying and analysing the coupled design activities. Also, the sequence of activities produced by the algorithm reduced the product development time.

Finally, the coupled activities determined by the algorithm may represent an iterative process or negotiation among experts. A way of decoupling design activities is through the redefinition of some of the coupled activities. The decoupling process should decrease the duration of the design project; however, some additional information about the activities is required.

Acknowledgement

This research has been partially supported by the research grant DDM-9215259 from the National Research Foundation and research contracts from Rockwell International and John Deere and Company.

References

DEO, N., 1974, *Graph Theory with Applications to Engineering and Computer Science* (Englewood Cliffs, NJ: Prentice-Hall).

EPPINGER, S. D., WHITNEY, D. E., SMITH, R. P., and GEBALA, D. A., 1990, Organizing the Tasks in Complex Design Projects. J. R. Rinderle (Ed.), *Proceedings of the 1990 ASME Conference—Design Theory and Methodology* (New York: ASME), pp. 39–46.

EVEN, S., 1979, *Graph Algorithms* (Haifa, Israel: Technion Institute Press).

HOROWITZ, E., and SAHNI, S., 1983, *Fundamentals of Data Structures* (Rockville, Md: Computer Science Press), pp. 312–313.

KUSIAK, A., 1993, *Concurrent Engineering: Automation, Tools, and Techniques* (New York: John Wiley).

KUSIAK, A., and PARK, K., 1990, Concurrent engineering: decomposition and scheduling of design activities. *International Journal of Production Research*, **28** (10), 1883–1990.

ROBERTS, S. M., and FLORES, B., 1966, Systematic generation of Hamiltonian circuits. *Communications of ACM*, **9** (9), 690–694.

SHIGLEY, J. E., and MISCHKE, C. R., 1986, *Standard Handbook of Machine Design* (New York: McGraw-Hill).

STEWARD, D. V., 1965, Partitioning and tearing systems of equations. *Journal of the Society for Industrial and Applied Mathematics: Numerical Analysis*, Ser. B, **2** (2), 345–365.

STEWARD, D. V., 1981 a, *Systems Analysis and Management: Structure, Strategy, and Design* (New York: Petrocelli Books).

STEWARD, D. V., 1981 b, The design structure system: a method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, **28** (3), 71–74.

TARJAN, R., 1972, Depth-first search and linear graph algorithms. *SIAM Journal of Computing*, **1** (2), 146–160.

TIERNAN, J. C., 1970, An efficient search algorithm to find the simple cycles of a finite directed graphs. *Communications of ACM*, **13** (12), 722–726.

WARFIELD, J. N., 1973, Binary Matrices in System Modeling. *IEEE Transactions on Systems, Man and Cybernetics*, **SMC-3** (5), 441–449.

WEIL, R. L., and KETTLER, P. C., 1971, Rearranging matrices to block-angular form for decomposition (and other) algorithms. *Management Science*, **18** (1), 98–108.

WEINBLATT, H., 1972, A new search algorithm for finding the simple cycles of finite directed graohs. *Journal of ACM*, **19** (1), 43–56.