JOURNAL OF AIRCRAFT
Vol. 36, No. 1, January–February 1999

Tools and Techniques for Decomposing and Managing Complex Design Projects

James L. Rogers*
NASA Langley Research Center, Hampton, Virginia 23681

Many companies are looking for new tools and techniques to aid a design manager in making decisions that can reduce the time and cost of a design cycle. One tool that is available to aid in this decision-making process is the design manager's aid for intelligent decomposition (DeMAID). Since the initial public release of DeMAID in 1989, much research has been done in the areas of decomposition, concurrent engineering, and process management, and many new tools and techniques have been developed. Based on this research and development, numerous enhancements have been added to DeMAID to aid the design manager in saving both cost and time in a design cycle. The key enhancement, a genetic algorithm (GA), is available in the most recent public release called DeMAID/GA. The GA orders the sequence of design processes to minimize the cost and time to converge to a solution. Several of the most recent advances are discussed and compared with the enhancements in DeMAID/GA. An example of an aircraft conceptual design project is used to show how these enhancements can be applied to improve the design cycle.

Introduction

IN today's competitive environment, companies are under enormous pressure to reduce the time and cost of their design cycle. One way to reduce the time and cost is to develop an understanding of the flow of the design processes and the effects of the iterative subcycles often found in complex design projects. Once these are understood, the design manager can make decisions that take advantage of decomposition and concurrent engineering techniques to reduce the total time and cost of the design cycle. One software tool that is available to aid in this decision-making process is the design manager's aid for intelligent decomposition (DeMAID).[1,2]

Since the initial public release of DeMAID in 1989, much research has been done in the areas of decomposition, concurrent engineering, and process management, and new tools and techniques have been developed. Based on this research and development, numerous enhancements have been added to DeMAID to aid the design manager in saving both cost and time in a design cycle. These features, including a genetic algorithm (GA), are available in the 1997 public release, DeMAID/GA.[3,4] This paper traces the evolution of process-management tools from its beginnings in graph theory through the early tools such as program evaluation and review technique (PERT) charts. The paper then presents an overview of the more recent tools and techniques used in decomposition, concurrent engineering, and process management against the background of graph theory, and shows how they relate to the enhancements found in DeMAID/GA.

Graph Theory

Much of the work in process management can be traced back to graph theory.[5] A directed graph consists of vertices (nodes or points) and arcs (edges or lines). An arc connects two vertices. The direction of the arc is from the *tail* vertex to the *head* vertex, as shown in Fig. 1. If the arcs in a directed graph have weights (based on cost or time, for example) associated with them, then the graph is referred to as a network.

A path in a directed graph is a sequence of vertices connected by arcs. The length of the path is the number of arcs on the path. A path is considered simple if all vertices on the path are distinct. A cyclic graph contains a path of length at least one (arc) that begins and ends at the same vertex. For example, the graph in Fig. 2 is cyclic because it contains cycles such as (1, 3, 2, 1); (1, 3, 2, 4, 3, 2, 1); and (1, 3, 2, 4, 3, 2, 4, 3, 2, 1). In the last example, the subcycle (3, 2, 4, 3) is repeated. These subcycles can be repeated many times, which leads to the concept of iteration.

Numerous methods are available for representing the graphs. One method of representation is the adjacency matrix. The adjacency matrix A, which represents a graph, is a square matrix where $A[i, j] = 1$, if an arc exists from vertex i to vertex j, and $A[i, j] = 0$ if no arc exists. The adjacency matrix for the graph in Fig. 2 is

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 & 1 \\ 3 & 0 & 1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 0 \end{array}$$

Another method, similar to the adjacency matrix, is the precedence matrix. A precedence matrix P for a graph with n vertices is an n by n square matrix, where $P[i, i]$ has a mark (*) for elements on the diagonal; $P[i, j]$ has a mark (X) if an arc exists from vertex i to vertex j; otherwise, $P[i, j]$ is blank. The precedence matrix for the graph in Fig. 2 is

$$\begin{array}{ccccc} & 1 & 2 & 3 & 4 \\ 1 & * & & X & \\ 2 & X & * & & X \\ 3 & & X & * & \\ 4 & & & X & * \end{array}$$

Received Dec. 7, 1996; revision received June 23, 1997; accepted for publication Jan. 1, 1998.

*Senior Computer Scientist, M/S 159. E-mail: j.l.rogers@larc.nasa.gov.

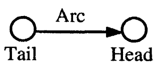

Fig. 1 Vertices and arc of a directed graph.

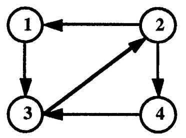

Fig. 2 Cyclic, directed graph.

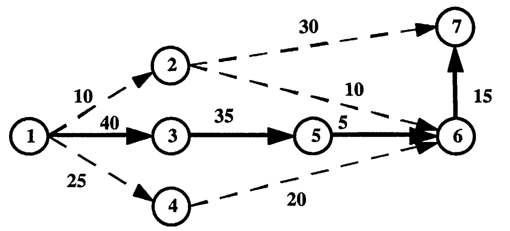

Fig. 3 Sample PERT chart (times on arcs are given in minutes).

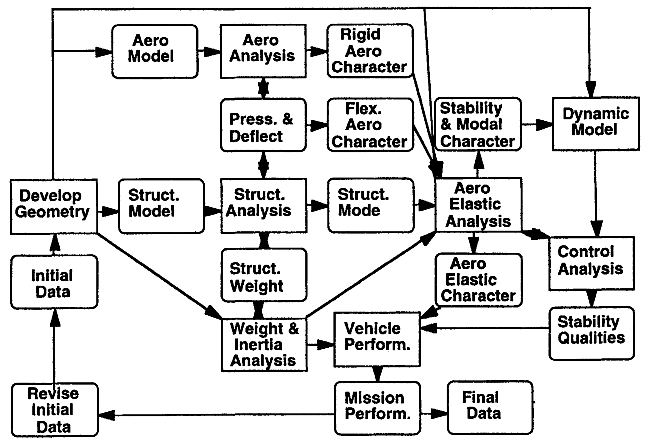

Fig. 4 Process flow chart of an example aircraft conceptual design project.

Based on this terminology and representation, graph theory can be applied to process-management problems.

Early Tools for Process Management

Any nontrivial project consists of numerous processes that are dependent on one another. This interdependency can become quite complicated. To reduce this complexity, numerous approaches were developed to aid in understanding and managing these processes. These network-based approaches (directed graphs with weights associated with their arcs) include the PERT and the critical path method.[6]

A PERT network (Fig. 3) is a weighted, acyclic, directed graph. Each arc on the PERT network represents a process, and the weight of the arc represents the time needed to complete that process. For example, the time needed to complete the process represented by arc ⟨3, 5⟩ is 35 min. Each vertex on the network represents a time by which all processes that terminate at that vertex can be completed. In any given project, some processes must be completed before other processes can begin. For example, vertex 6 represents 80 min. Even though paths (1, 4, 6) and (1, 2, 6) only take 45 and 20 min, respectively, to complete, process ⟨6, 7⟩ cannot begin until process ⟨5, 6⟩ is complete, and path (1, 3, 5, 6) requires 80 min. Another important aspect of process management is that some processes can be executed concurrently. For example, processes ⟨1, 2⟩, ⟨1, 3⟩, and ⟨1, 4⟩ can execute concurrently.

A project's critical path consists of those processes that will cause the project end date to be delayed if they are delayed.[7] In the example in Fig. 3, the critical path (with solid arrows) is (1, 3, 5, 6, 7), and requires 95 min for completion. If the time required for any process along this path is increased, then the time to complete this project increases accordingly. However, if the time for process ⟨1, 2⟩ is increased from 10 to 20 min, for example, no effect is realized on the scheduled completion of this project. Strategies for reducing the critical path should focus on reducing process duration. PERT charts can serve as an aid in this reduction.

Unfortunately, these tools are only applicable to sequential activities and cannot handle the cyclic graphs (implying iteration) that are often found in complex design problems. A PERT-related tool, called the general evaluation review technique,[8] that can handle cycles, is only effective for simple networks.

Many design projects are based on a process flow chart similar to the one for this example of an aircraft conceptual design project shown in Fig. 4. This chart is easily recognizable as a directed graph with vertices and arcs. The problems with this representation are that it is difficult to determine the order in which to execute the processes, where to take advantage of decomposition and concurrent processing, and where the iterative subcycles exist.

New tools are needed to decompose, manage, and display the processes in a complex design project and to handle the iterative subcycles that are commonly found in these types of projects.

Design Structure Matrix

The design structure matrix (DSM), originally formulated by Steward,[9] is another tool for displaying the sequence of processes. A sample DSM is shown in Fig. 5.

The DSM is derived from graph theory (graph terms are shown in parentheses), and is similar to the adjacency–matrix representation. In the DSM in Fig. 5, the processes are shown as numbered boxes (vertices) on the diagonal. Output from a process is shown as a horizontal line (arc) that exits a numbered box, and input to a process is shown as a vertical line (arc) that enters a box. The off-diagonal squares that connect the horizontal and vertical lines represent couplings between two processes. Output data may be fed forward in the design cycle, which means that output data are being computed by one process before they are needed as inputs to another process. Squares in the upper triangle of the DSM represent feed-forward couplings. In some instances, data may be required as input before it has actually been computed. In these instances, the first time the process is executed, the input data not previously computed must be estimated. After the actual data are computed downstream in the design cycle, those data are fed back to the process to obtain more accurate results. Squares in the lower triangle of the matrix represent this type of data as feedback couplings. If feedback couplings exist, then iterations may be required to obtain optimal results. Therefore, feedback couplings should be eliminated if possible. However, in many cases, not all feedback couplings can be eliminated. If certain feedback couplings cannot be eliminated, the processes coupled by the feedback coupling are grouped into iterative subcycles, called circuits (cyclic graphs). In Fig. 5, processes 1–3, 5–19, 21–25, and 26–29 are grouped into iterative subcycles.

The primary advantage of the DSM format over other display tools, such as PERT, is the capability to group and display the iterative subcycles commonly found in a design project. After the iterative subcycles have been determined, their processes should be ordered in such a way to produce the best design in the least time at a minimum cost. The total cost and time to complete a design project are dependent on the ordering of the processes (paths) in the iterative subcycles.[3] A large, iterative subcycle such as the one in Fig. 5 that contains processes 5–19 can be very expensive to converge because the iterative subcycles defined by the feedback couplings are

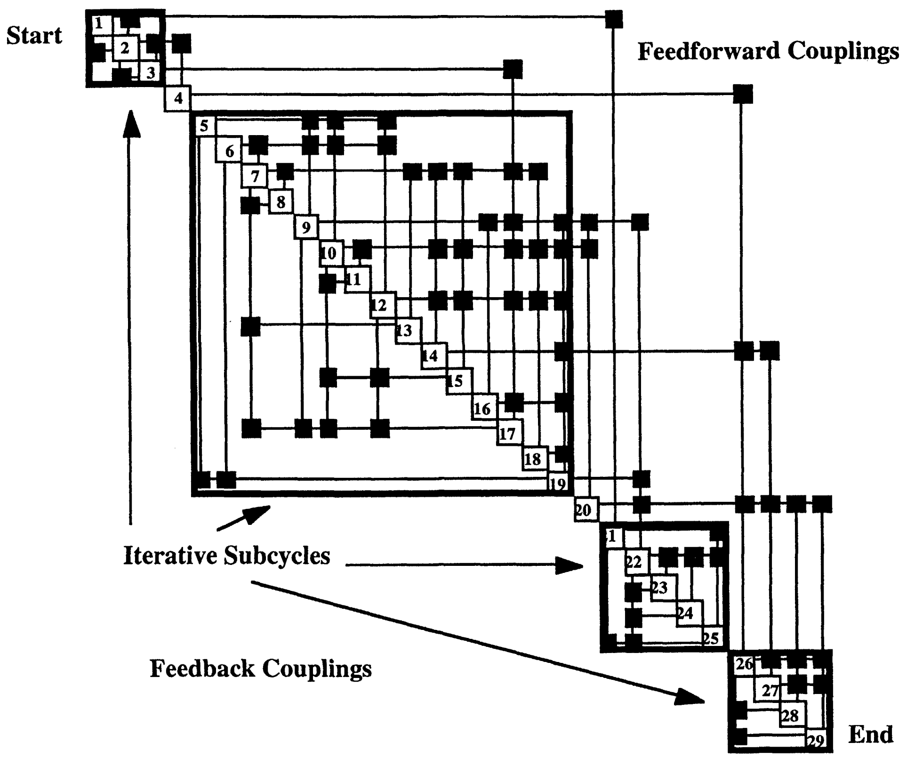

Fig. 5 Design structure matrix.

nested. These nested couplings may require numerous executions of potentially expensive processes. Any software tool based on the DSM should have the capability to examine many different sequences of processes within an iterative subcycle and select the optimal sequence based on cost, time, and iteration requirements.

DSM-Based Software Tools for Process Management and Decomposition

Over the years, numerous tools have been developed to aid the project manager in decomposing a design project, managing the design processes, and providing valuable insight into the iteration process. Smith and Eppinger[10] provide a comprehensive overview with references of some of the more recently developed models of design iteration. These models include addressing the frequency of design reviews, deciding the composition and relation of design teams, showing the effects on product development time and quality based on managerial inputs, and minimizing total lead time by finding the proper timing of information transfers. This section focuses on two DSM-based tools, the problem-solving matrix (PSM) and the work transformation matrix (WTF) for decomposing and managing the design process. Some tools display the DSM with feedback couplings in the upper triangle of the matrix, whereas others have the feedback couplings in the lower triangle. For consistency, this paper will always have the feedback couplings in the lower portion of the triangle.

Problem-Solving Matrix

In Steward's problem-solving enterprise (PSE), problems are solved by forming teams around the structure of the problem.[11] This concept enables the solution of the problem to rely on the flow of information inherent in the problem rather than the flow of information inherent in the organization. The PSE contains a software tool, the PSM, which examines the couplings among the various processes of a project, and aids in ordering the processes to solve the problem.

Steward's[11] method for decomposing and ordering the design project is a two-step process. The steps are partitioning and tearing. Partitioning is the decomposing of the design project into the iterative subcycles (circuits), and tearing is the removal of certain couplings (arcs) and testing to see if the iterative subcycle remains. Consider the digraph from Steward's book,[9] shown in Fig. 6.

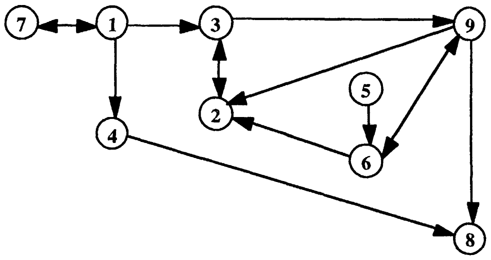

Fig. 6 Digraph.

	1	2	3	4	5	6	7	8	9
1	*		X	X			X		
2		*	X						
3		X	*						X
4				*				X	
5					*	X			
6		X				*			X
7	X						*		
8								*	
9		X				X		X	*

Fig. 7 DSM from the PSM.

The PSM displays a DSM in a format similar to the precedence matrix. This display of the DSM has * on the diagonal elements to represent the processes, and X in the off-diagonal elements to represent the couplings among the processes. PSM allows for numbers (0–9), representing coupling strengths in the off-diagonal elements, which serve as an aid in partitioning and tearing. The higher the number, the easier it is to estimate the data represented by the coupling. The digraph in Fig. 6 is transformed into the DSM shown in Fig. 7.

PSM manipulates the rows and columns of the matrix to partition the processes into iterative subcycles. PSM partitions the matrix based on a procedure that checks the predecessors of each vertex in the graph. This procedure chooses a vertex and traces the path backward until re-encountering that vertex. The vertices contained in that trace form the iterative subcycle. For example, begin with vertex 3 and trace the arcs back through vertices 9, 6, 2, and 3. After partitioning, the digraph in Fig. 6 and the resulting DSM would appear as shown in Figs. 8 and 9, respectively.

The iterative subcycles are unique when considering the processes contained within them. The ordering of the processes within the iterative subcycles is not unique. After the iterative subcycles have been determined, the design manager can make choices about where to tear the iterative subcycle to divide it into smaller iterative subcycles. A tear is made by estimating the data contained in a feedback coupling. Tearing can be repeated until only couplings remain in the upper triangle of the matrix, leaving no iterative subcycles. Where to make the tears involves engineering judgment based on knowledge of the project and the structure of the project displayed in the DSM. Steward[9] presents several techniques to aid in the tearing process.

Work Transformation Matrix

The faculty and students at the Massachusetts Institute of Technology, Sloan School of Management, have done substantial work in the application of the DSM to real-world design problems.[12] One of their major contributions to this area of research has been the extension of the DSM to the WTM model.[10] The WTM can predict slow and rapid convergence of iteration within a design project. It also predicts those coupling features of the design project that will require many iterations to converge to a technical solution.

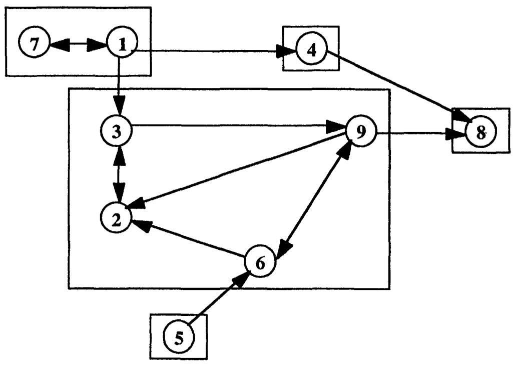

Fig. 8 Digraph after partitioning.

	5	7	1	6	2	3	9	4	8
5	*			X					
7		*	X						
1		X	*			X		X	
6				*	X		X		
2					*	X			
3					X	*	X		
9				X	X		*		X
4								*	X
8									*

Fig. 9 DSM after partitioning.

a)

	1	2
1	3	.2
2	.4	5

b)

	1	2
1	0	.2
2	.4	0

c)

	1	2
1	3	0
2	0	5

Fig. 10 Work transformation matrices: a) spreadsheet format, b) coupling strengths, and c) process times.

The typical WTM display is in a spreadsheet format (Fig. 10a), where the diagonal elements represent the time to complete a process during the first iteration; and the off-diagonal elements represent the strength of dependence between two processes. The coupling strengths are used to form a rework matrix. For example, in Fig. 10, every time process 1 is done, then process 2 must have 40% of its work redone; and when process 2 is done, then process 1 must have 20% of its work redone. In this example, because process 1 requires 3 units of time to complete execution, the rework for process 1 is 1.2 (0.4×3) units of time, and 1.0 (0.2×5) units of time for process 2.

Work on a particular process will cause some rework to be created for all processes that are dependent on the output of that process. The WTM documents this dependence. A matrix that contains only the off-diagonal coupling data is obtained from the WTM (Fig. 10b). Zeros are placed on the diagonal. The eigenvalues and eigenvectors of this matrix, along with a diagonal matrix containing the process times (Fig. 10c), determine the rate and nature of the convergence of the design project. Much of the analysis of the WTM is accomplished through spreadsheet macros. This method gives explicit representation to the difficult problem of deciding when to do a design sequentially rather than parallel.

Additional Process Management and Decomposition Tools

The original approach to managing and decomposing complex multidisciplinary design problems was based on a hierarchical decomposition scheme.[13] This approach has been advanced over the past few years with the development of new tools and techniques that do not rely on the DSM. The major focus of these tools is to decompose a large design project to exploit any available parallelism to reduce the total computational time and cost.

Incidence Matrix

Kusiak and Wang[14] represent the relationship between design parameters (variables) and processes (constraints) with an incidence matrix. In the incidence matrix, the parameters are represented in the columns; and the processes in the rows with an * to indicate a relationship. The purpose of this tool is to minimize the interdependency among the subprocesses that enhances the concurrency of the design project. A design project can be decomposed into subprocesses if the rows and columns of its incidence matrix can be grouped so that the matrix separates into mutually exclusive submatrices (Fig. 11). If the

	Parameters						
Processes	a	b	c	d	e	f	g
1	*	*	*				
2	*	*	*				
3				*	*		
4				*	*		
5						*	*
6						*	*

Fig. 11 Decomposable incidence matrix.

	Parameters						
Processes	a	b	c	d	e	f	g
1	*	*	*			*	*
2	*	*	*			*	*
3	*	*	*			*	*
4				*	*	*	*
5				*	*	*	*
6				*	*	*	*

Fig. 12 Nondecomposable matrix with overlapping parameters.

submatrices are not mutually exclusive, then there are the overlapping parameters (parameters *f* and *g* in Fig. 12) or overlapping processes (processes 5 and 6 in Fig. 13). A branch-and-bound technique is applied to resolve the overlapping problems shown in Figs. 12 and 13.

Functional Dependence Table

Michelena and Papalambros[15] represent the functional dependence of constraints and objective functions (in the rows) on design variables (in the columns) by a Boolean matrix called a functional dependence table (FDT). In an FDT, a shaded box implies a Boolean *true*, that the function in the row is dependent on the variable in the column. This approach is based on an undirected graph representation called a hypergraph. Linking variables that link independent subprocesses are identified and deleted. Heuristics are used to determine the appropriate linking variables. A mathematical programming strategy is then used to solve the original problem as a set of smaller, independent subproblems coordinated by a master problem. The remaining variables in each subgraph are local variables to the corresponding subprocess. In Fig. 14, design variable $x2$ has been selected as a linking variable, the rows and columns have been reordered as two subprocesses. The first subprocess contains the functions $\{g2, f1, g3, \text{and } f3\}$ and the local variables $\{x6, x3, x7, \text{and } x1\}$. The second subprocess contains the functions $\{f4, g4, \text{and } g1\}$ and the local variables $\{x4, x8, \text{and } x5\}$.

In hierarchically decomposed projects, the main project is solved for the linking variables, which are then input as parameters to the subprocesses. Information on the dependence of the local variables with respect to the linking variables is fed back to the main problem.

Decomposition with Compatibility Constraints

Kroo et al.[16] advocates a representation where all of the iterative subprocesses are removed. This approach is based on directed acyclic graphs where an auxiliary variable, which represents an initial guess to begin the iterative subprocess, is added to the list of design variables with an additional compatibility constraint that represents the convergence criterion. This approach is applied to both feedforward and feedback couplings. The representation in Fig. 15 shows the changes this approach makes in the process flow. The result is a simple decomposition of the design project that is free of iterative subprocesses and parallel.

Knowledge-Based Decomposition

Liu and Brown[17] propose a knowledge-based approach to decompose parametric design problems. This procedure works top down through the hierarchy. Coupling strength is used to partition components and attributes. In this approach, knowledge is drawn from three sources: design object knowledge, design cases, and functional knowledge. Design object knowledge describes the objects structure, its components, and how they are related. A design case records a design problem and how it was decomposed into subproblems. Functional knowledge indicates the connections between functional descriptions and the design objects. This approach is viewed as a type of knowledge compilation where existing knowledge is converted into new forms, in hopes of improving problem-solving efficiency.

Liu and Brown[17] test their approach on the design of a force transducer. The force transducer decomposes into several parts, including a cantilever bending beam and a strain gauge on the beam. The knowledge about the transducer includes the attributes of the components and the relationships among them. The requirements specify restrictions on force, weight, length, and displacement. By applying general decomposition heuristics to their knowledge about the structure, they are able to decompose the transducer as shown in Fig. 16. Their experiments also show that different requirements often result in different decompositions.

Genetic Algorithm

The preceding sections discuss several approaches to decomposing and managing complex design projects. None of these approaches has become a standard for multidisciplinary applications. Altus et al.[18] proposes a new decomposition tool based on a GA that can incorporate various approaches tailored for specific applications. This approach is incorporated into a software tool called a GA for decomposition of analysis.

The use of GAs has been instrumental in achieving acceptable solutions to discrete optimization problems that have not been satisfactorily addressed by other methods.[19] GAs have proven useful in solving the discrete problems such as the sequencing problem.[20] A population of design points that are coded as finite length, finite alphabet strings is searched by the GA. Successive populations are produced primarily by the operations of selection, crossover, and mutation. The selection operator determines those members of the population that survive to participate in the production of members of the next population. Selection is based on the value of the fitness func-

	Parameters						
Processes	a	b	c	d	e	f	g
1	*	*	*				
2	*	*	*				
3				*	*	*	*
4				*	*	*	*
5	*	*	*	*	*	*	*
6	*	*	*	*	*	*	*

Fig. 13 Nondecomposable matrix with overlapping processes.

	x2	x6	x3	x7	x1	x4	x8	x5
f2	■							
g2		■			■			
f1		■						
g3	■	■		■	■			
f3			■	■	■			
f4	■							■
g4	■						■	■
g1						■	■	■

Fig. 14 Decomposed functional dependence table.

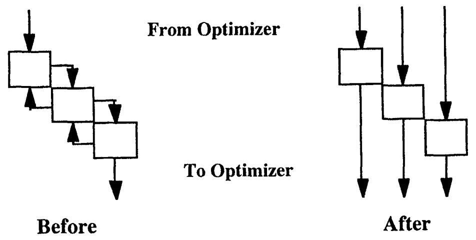

Fig. 15 Process flow after applying compatibility constraints.

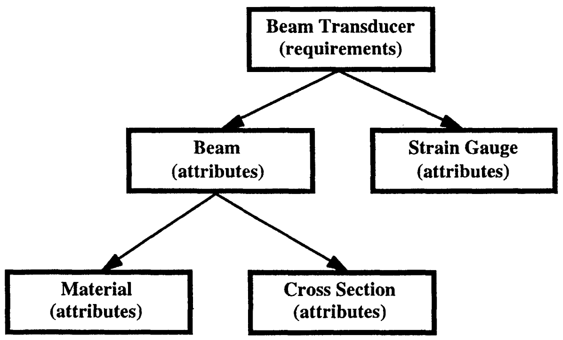
Beam Transducer (requirements)
Beam (attributes)
Strain Gauge (attributes)
Material (attributes)
Cross Section (attributes)

Fig. 16 Decomposition of a force transducer.

tion, or the fitness of the individual members, such that members with greater fitness levels tend to survive. Crossover is the recombination of traits of the selected members, called the mating pool, in the hope of producing a child with better fitness levels than its parents. Crossover is accomplished by swapping parts of the string into which these design points have been coded. The final operation, mutation, prevents the search of the space from becoming too narrow. After the production of a child population, this operator randomizes small parts of the resulting strings, with a very low probability that any given string position will be affected.

Design Manager's Aid for Intelligent Decomposition

The DeMAID[1-4,21-23] uses a knowledge-based approach based on Steward's partitioning procedure[9] to group the processes into iterative subcycles and display the processes with the DSM format similar to the one shown in Fig. 5. Once this step has been completed, DeMAID offers the design manager several methods for decomposing and ordering the design processes. Other features based on the methods described earlier have been added to aid the design manager in reducing the time and cost of the design cycle. For example, the DSM has been applied to a system-oriented model of aircraft design; and shown the potential of a 50% reduction in cycle time, resulting in re-engineering the design approach around the flow of information.[24]

Hierarchical Decomposition

Once the iterative subcycles have been grouped, there are no feedback couplings outside the iterative subcycles. In this format, a nonhierarchical problem can be decomposed into a hierarchy of iterative subcycles, as shown in Fig. 17. In the decomposed hierarchy, iterative subcycles on the same level, e.g., blocks 2, 4, and 8 in the figure, can be executed concurrently.

Decomposing an Iterative Subcycle

Typically, a large design project will have several large iterative subcycles as shown in Figs. 5 and 17. The DSM example in Fig. 18 is limited to a single iterative subcycle for illustrative purposes. The DSM in Fig. 18 shows the process flow for the example aircraft conceptual design project shown

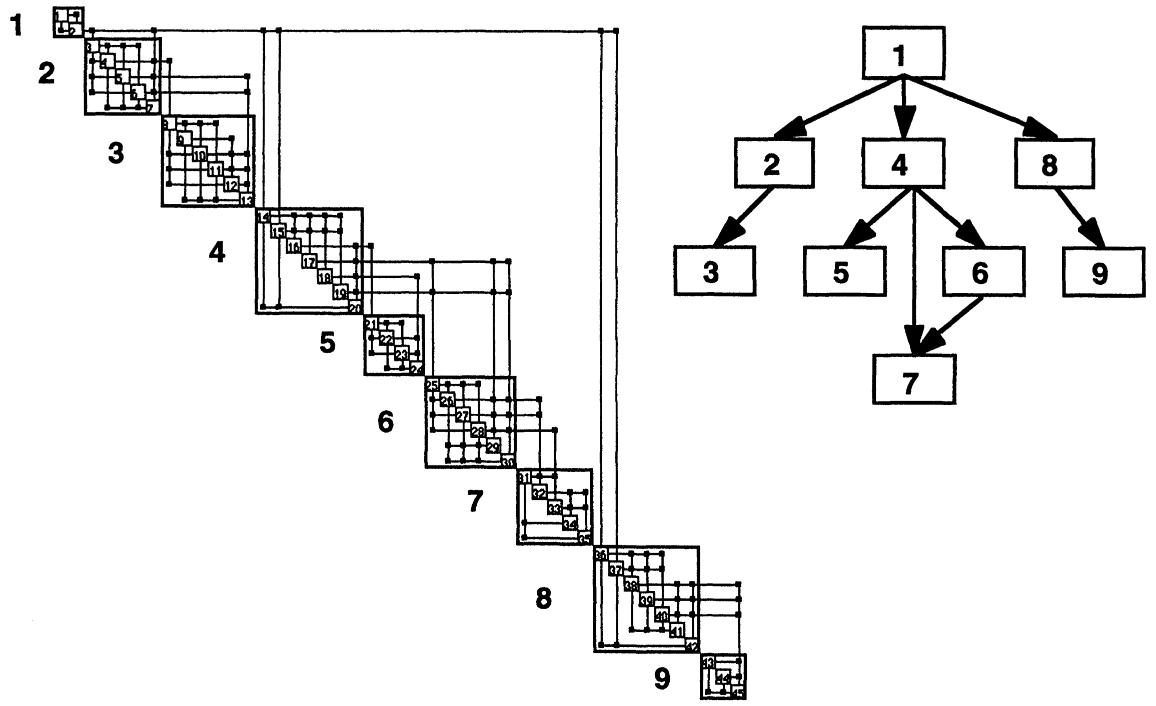

Fig. 17 Hierarchical decomposition.

Label	Time	Cost
DYNMODL	30	30
STDMOCH	40	20
STRMODL	10	50
HANDQUL	10	50
STRMODE	10	50
GEOMDEV	50	10
AROSRYO	40	20
STRDYNA	50	10
CSYSANL	20	40
FAEROCH	20	40
INITDAT	40	20
RYSEDAT	30	30
MISPERF	30	30
VEHPERF	20	40
RAEROCH	30	30
AEROANL	20	40
PRESDEF	30	30
STRANAL	40	20
STRCTWT	50	10
WIANAL	40	20
AEROMDL	20	40
FINLDAT	20	40

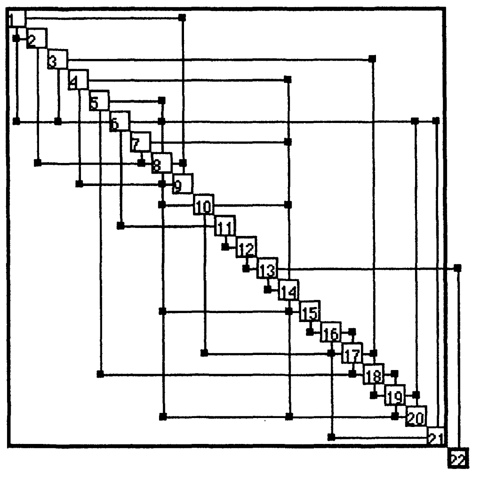

Fig. 18 Unordered DSM for the aircraft conceptual design project.

in Fig. 4, after the iterative subcycle has been determined, but before the processes in the iterative subcycle have been ordered by DeMAID.

The previously discussed hierarchical decomposition focused on determining where concurrency exists, based on iterative subcycles within the design cycle. Many iterative subcycles such as the one in Fig. 18 are large, and significant time savings can be achieved by executing some of the processes in parallel. To determine the parallel processing capabilities within an iterative subcycle, DeMAID treats all feedback coupling data as available. With this assumption, the processes in an iterative subcycle can be divided into a hierarchy where any processes on the same level of the hierarchy can be executed in parallel. For example, prior to reordering, the processes in the iterative subcycle of the conceptual design DSM in Fig. 18 decompose into the following levels:

Level 1	1 2 3 4 5 6 7 10 11 12 13 15 16
Level 2	8 14 17 21
Level 3	9 18
Level 4	19
Level 5	20

Dependency Matrix

DeMAID also forms a dependency matrix that relates the design variables to the constraint and objective functions as shown in Fig. 19.

Tracing the Effects of a Design Change

DeMAID has the capability to use the DSM to trace the effects of changes in the design cycle.[21] If a change is made to one process, this does not mean that all processes must be re-executed. For example, in the DSM shown in Fig. 20, if a change is made in process 5, the design manager can see, that with respect to process 29, this change does not effect processes, 18, 23, 24, and 26. If these processes are very costly, time consuming, or require a critical resource, then there is a possibility of substantial savings.

Interfacing to Other Project Management Tools

DeMAID contains an interface function to take advantage of other process management tools. This interface function allows the user to save a file that can be input to other tools, such as spreadsheets, PERT, and Steward's PSE.[11]

Coupling Strengths

Much of the benefits that can be derived from DeMAID is in its application of the DSM to aid the design manager in reducing the time and cost of the design project. Input to DeMAID allows for a time and a cost (units are determined by the user) to be associated with each process. In addition, coupling strengths can be quantified in seven levels, ranging from extremely weak to extremely strong.[22,25] The coupling strengths can be defined by the user or computed from sensitivity analysis. In DeMAID, the couplings strengths are color coded in the DSM for easier viewing. Arbitrary iteration factors have been assigned to the couplings strengths to aid in the computation of the time and cost of an iterative subcycle. The key to these factors is that the weaker couplings require fewer cycles to converge than stronger couplings; therefore, the weaker couplings tend to be placed in the feedback position, allowing a coupling to be temporarily suspended or removed (similar to Steward's[9] concept of tearing). The knowledge base in DeMAID makes recommendations for removing couplings based on their strengths and the relationship to the processes they connect.

Computing Time and Cost

The steps for computing time and cost of the DSM shown in Fig. 21 are as follows.

1) For each feedback coupling in the DSM, sum the times and costs associated with the processes coupled by the feedback coupling: time = 10 + 30 + 40 + 20 = 100 and cost = 20 + 15 + 40 + 35 = 110.

2) Multiply that sum by the iteration factor associated with the feedback coupling: iterative time = 100 × 5 = 500 and iterative cost = 110 × 5 = 550.

3) To find the total cost and time of the design project, sum all of the times and costs found for the feedback couplings in steps 1 and 2.

The DSM shown in Fig. 18 is most likely a meaningless sequence, but serves as an excellent example when comparing the time and cost differences for various sequences. For the DSM in Fig. 18, the time is 21,340 units, and the cost is 19,640 units to converge the unordered iterative subcycle.

Genetic Algorithm in DeMAID/GA

The most recent version of DeMAID is called DeMAID/GA, to reflect the addition of a GA to optimize the ordering of the processes within the iterative subcycles.[23] This capability was derived from the work at the State University of New York at Buffalo[26,27] and Stanford University.[18] The fitness function of the GA is based on minimizing the time, cost, feedback couplings, and crossovers of each iterative subcycle. The user can weigh the importance of each of these factors. The previously discussed method for determining the cost and time associated with a feedback coupling is used with the GA.

		Design Variables								
	Process	1	2	3	4	5	6	11	13	14
Constraints	7	X	X	X	X					
	8	X	X	X	X					
	9		X	X	X					
	10		X	X	X	X	X			
	12	X						X	X	X
	15	X						X	X	X

Fig. 19 Dependency matrix.

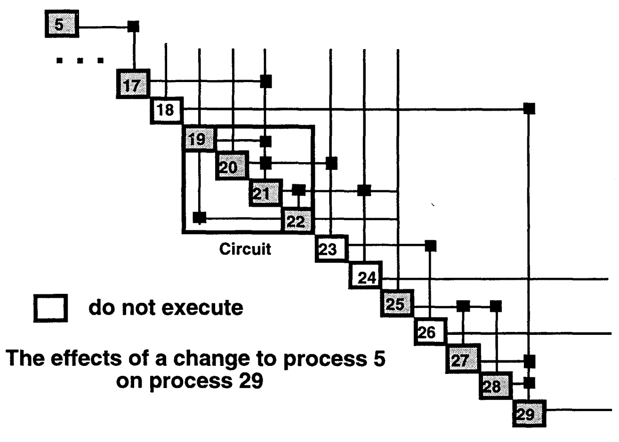

Fig. 20 DSM for tracing the effects of a design change.

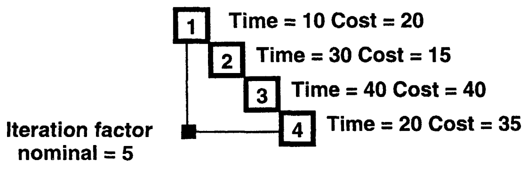

Fig. 21 DSM with process times and costs and an iteration factor.

###	Label	Time	Cost
1	RYSEDAT	30	30
2	INITDAT	40	20
3	GEOMDEY	50	10
4	AEROMDL	20	40
5	STRMODL	10	50
6	AEROANL	20	40
7	PRESDEF	30	30
8	STRANAL	40	20
9	STRCTWT	50	10
10	WIANAL	40	20
11	FAEROCH	20	40
12	RAEROCH	30	30
13	STRMODE	10	50
14	STRDYNA	50	10
15	STDMOCH	40	20
16	DYNMODL	30	30
17	CSYSANL	20	40
18	AROSRYO	40	20
19	HANDQUL	10	50
20	YEHPERF	20	40
21	MISPERF	30	30
22	FINLDAT	20	40

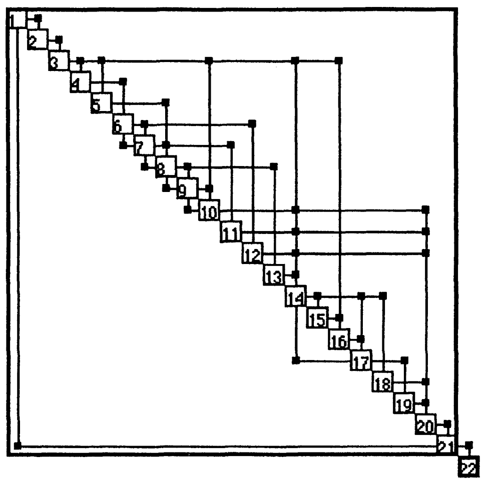

Fig. 22 DSM for the aircraft conceptual design project after GA optimization.

###	Label	Time	Cost
1	RYSEDAT	30	30
2	INITDAT	40	20
3	GEOMDEY	50	10
4	AEROMDL	20	40
5	STRMODL	10	50
6	AEROANL	20	40
7	PRESDEF	30	30
8	STRANAL	40	20
9	STRCTWT	50	10
10	WIANAL	40	20
11	FAEROCH	20	40
12	RAEROCH	30	30
13	STRMODE	10	50
14	STRDYNA	50	10
15	STDMOCH	40	20
16	DYNMODL	30	30
17	CSYSANL	20	40
18	AROSRYO	40	20
19	HANDQUL	10	50
20	YEHPERF	20	40
21	MISPERF	30	30
22	FINLDAT	20	40

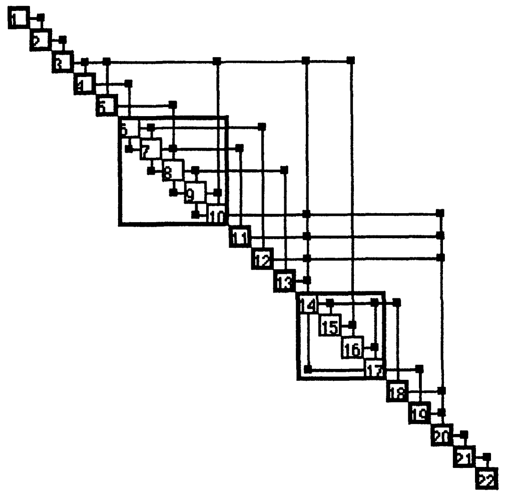

Fig. 23 DSM of modified aircraft conceptual design project.

The sequence of the processes in the iterative subcycle for the DSM shown in Fig. 18 has been optimized with the GA. The new sequence is shown in Fig. 22. With this reordering, the design cycle time is reduced from 21,300 units to 3800 units, whereas the design cycle cost is reduced from 19,640 units to 3220 units.

However, this sequence reduces the amount of concurrent processing available to the design manager, as shown by decomposing the iterative subcycle into the following levels.

Level 1	1	Level 9	10
Level 2	2	Level 10	14
Level 3	3	Level 11	15 18
Level 4	4 5	Level 12	16
Level 5	6	Level 13	17
Level 6	7 12	Level 14	19
Level 7	8 11	Level 15	20
Level 8	9 13	Level 16	21

This is one of the tradeoffs that must be considered when ordering the processes within a large iterative subcycle.

A decomposition benefit might be realized from this ordering. For example, suppose that the coupling from process 22 to process 1 in the DSM in Fig. 22 is extremely weak, and could be removed without affecting the accuracy of the solution. Then the design project could be decomposed into the DSM shown in Fig. 23 and contain only two relatively small iterative subcycles.

Concluding Remarks

Much of the cost and time involved to complete a complex, multidisciplinary design project is associated with expensive iterative subcycles. New tools and techniques have been researched and developed to aid the design manager in reducing these costs. Significant enhancements based on these developments, including a GA, have been added to DeMAID/GA. After decomposing the design project into iterative subcycles, and, if possible, determining the coupling strengths among the design processes, the GA orders the processes within the subcycle to minimize the time and cost to converge to the design solution. Another enhancement enables the design manager to determine where to take advantage of concurrently executing design processes within a subcycle. An aircraft conceptual design project demonstrates the gains that can be made by optimizing the order of the design processes in a complex design project.

Acknowledgments

The author wishes to acknowledge all researchers who contributed to the development of tools and techniques for decomposing and managing complex design projects. It is not possible within the scope of a single paper to list all contributors, nor is it possible to give an in-depth discussion of all contributions. It is hoped that this paper can serve as a starting point for future research into the topic.

References

1Rogers, J. L., "A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA TP 2903, May 1989.

2Rogers, J. L., "DeMAID—A Design Manager's Aide for Intelligent Decomposition User's Guide," NASA TM-101575, March 1989.

3Rogers, J. L., "DeMAID/GA—An Enhanced Design Manager's Aid for Intelligent Decomposition," AIAA Paper 96-4157, Sept. 1996.

4Rogers, J. L., "DeMAID/GA User's Guide—Design Manager's Aid for Intelligent Decomposition with a Genetic Algorithm," NASA TM-110241, April 1996.

5Aho, A. V., Hopcroft, J. E., and Ullman, J. D., *Data Structures and Algorithms*, Addison–Wesley, Reading, MA, 1983, pp. 198–229.

6Kerzner, H., *Project Management*, Reinhold, New York, 1989.

7Bertziss, A. T., *Data Structures Theory and Practice*, Academic, New York, 1971, pp. 223–230.

8Neumann, K., and Steinhardt, U., *GERT Networks and the Time-Oriented Evaluation of Projects*, Springer–Verlag, New York, 1979.

9Steward, D. V., *Systems Analysis and Management: Structure, Strategy and Design*, Petrocelli Books, Inc., New York, 1981.

10Smith, R. P., and Eppinger, S. D., "Identifying Controlling Features of Engineering Design Iteration," *Management Science*, Vol. 43, No. 3, 1997, pp. 276–293.

11Steward, D. V., "Re-Engineering the Design Process," *Proceedings of the 2nd Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (Morgantown, WV), 1993.

12Eppinger, S. D., Whitney, D. E., Smith, R. P., and Gebala, D. A. "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design*, Vol. 6, 1994, pp. 1–13.

13Sobieszczanski-Sobieski, J., "A Linear Decomposition Method for Large Optimization Problems—A Blueprint for Development," NASA TM-83248, Feb. 1982.

14Kusiak, A., and Wang, J., "Decomposition of the Design Process," *Journal of Mechanical Design*, Vol. 115, No. 4, 1993, pp. 687–695.

15Michelena, N. F., and Papalambros, P. Y., "Optimal Model-Based Decomposition of Powertrain System Design," *Journal of Mechanical Design*, Vol. 117, No. 4, 1995, pp. 499–505.

16Kroo, I. M., Altus, S. S., Braun, R. D., Gage, P. J., and Sobieski, I., "Multidisciplinary Optimization Methods for Aircraft Preliminary Design," AIAA Paper 94-2543, Sept. 1994.

17Liu, J., and Brown, D. C., "Generating Design Decomposition Knowledge for Parametric Design Problems," *Proceedings of the Artificial Intelligence in Design '94 Conference*, edited by J. S. Gero and F. Sudweeks, Kluwer Academic, Norwell, MA, 1994, pp. 661–678.

18Altus, S. S., Kroo, I. M., and Gage, P. J., "A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems," American Society of Mechanical Engineers, Paper 95-141, New York, Sept. 1995.

19Goldberg, D., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley, Reading, MA, 1989.

20Syswerda, G., "Schedule Optimization Using Genetic Algorithms," *Handbook of Genetic Algorithms*, Reinhold, New York, 1990.

21Rogers, J. L., and Barthelemy, J.-F. M., "Enhancements to the Design Manager's Aid for Intelligent Decomposition (DeMAID)," AIAA Paper 92-4809, Sept. 1992.

22Rogers, J. L., and Bloebaum, C. L., "Ordering Design Tasks Based on Coupling Strengths," AIAA Paper 94-4326, Sept. 1994.

23Rogers, J. L., McCulley, C. M., and Bloebaum, C. L., "Integrating a Genetic Algorithm into a Knowledge-Based System for Ordering Complex Design Processes," *Proceedings of the 4th International Conference of Artificial Intelligence in Design*, Stanford Univ., Stanford, CA, 1996, pp. 119–133.

24Grose, D. L., "Re-Engineering the Aircraft Design Process," AIAA Paper 94-4323, Sept. 1994.

25Bloebaum, C. L., "An Intelligent Decomposition Approach for Coupled Engineering Systems," AIAA Paper 92-4821, Sept. 1992.

26McCulley, C. M., and Bloebaum, C. L., "Optimal Sequencing for Complex Engineering Systems Using Genetic Algorithms," AIAA Paper 94-4327, Sept. 1994.

27McCulley, C. M., and Bloebaum, C. L., "A Genetic Tool for Optimal Design Sequencing in Complex Engineering Systems," *Structural Optimization*, Vol. 12, No. 2/3, 1996, pp. 186–201.