ENHANCEMENTS TO THE DESIGN MANAGER'S AIDE FOR INTELLIGENT DECOMPOSITION (DeMAID) *

James L. Rogers ** and Jean-Francois M. Barthelemy +
Interdisciplinary Research Office
NASA Langley Research Center
Hampton, VA

Abstract

This paper discusses the addition of two new enhancements to the program Design Manager's Aide for Intelligent Decomposition (DeMAID). DeMAID is a knowledge-based tool used to aid a design manager in understanding the interactions among the tasks of a complex design problem. This is done by ordering the tasks to minimize feedback, determining the participating subsystems, and displaying them in an easily understood format. The two new enhancements include (1) rules for ordering a complex assembly process and (2) rules for determining which analysis tasks must be re-executed to compute the output of one task based on a change in input to that or another task.

* Copyright @1992 by the American Institute of Aeronautics and Astronautics, Inc. No copyright is asserted in the United States under Title 17, U.S. Code. The U.S. Government has a royalty-free license to exercise all rights under the copyright claimed herein for Governmental purposes. All other rights are reserved by the Copyright Owner.

** Senior Computer Scientist

+ Senior Research Engineer
Senior Member AIAA

Introduction

Many engineering systems are large and multidisciplinary, and before the design of such complex systems can begin, much time and money must be invested in determining the possible interactions among the participating subsystems and their parts. In 1989, a new knowledge-based tool (ref. 1 and 2) was developed to aid the engineer in determining these interactions and displaying them in the format of a design structure matrix (NxN). This tool was called the Design Manager's Aide for Intelligent Decomposition (DeMAID). Since its release to the public, two new enhancements have been incorporated into DeMAID. The first enhancement includes rules to order a complex assembly process. The second enhancement applies to complex analyses processes. It determines which analysis tasks must be re-executed to compute the output of one task based on a change in input to that or another task. This paper gives a brief overview of DeMAID followed by a brief description of the two new enhancements.

Overview of DeMAID

DeMAID was originally developed as a tool to aid the design manager in decomposing large, complex, multidisciplinary processes. A diagram of DeMAID is shown in figure 1. This program is based on a project

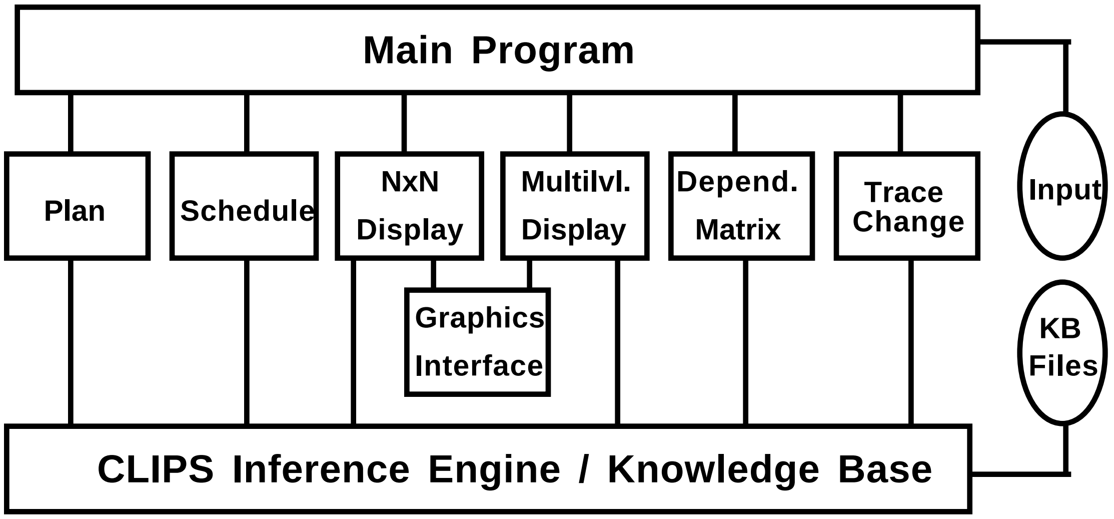

Fig. 1 Diagram of DeMAID

management tool developed by Steward (ref. 3) which organizes and displays the interactions among tasks in the format of a design structure matrix (NxN). This tool replaces Steward's matrix manipulations with a knowledge-base (ref. 4) to provide more flexibility in solving new problems and adding new enhancements.

In the design structure matrix (fig. 2), each box along the diagonal represents a task. A task requires some input and computes an output. Input for a task are shown as vertical lines entering the box from either above or below. Output from a task are shown as horizontal lines exiting the box from either side. A small circle indicates an interaction between tasks (the output from one task is required as input to another). Any circle in the upper portion of the matrix implies feedforward data, while circles in the lower portion of the matrix imply feedback data. Since feedback data indicates an iterative process which is typically costly and time consuming, DeMAID minimizes the amount of feedback by reordering the boxes along the diagonal. In most large and complex design problems, not all feedback data can be removed, therefore DeMAID groups this data into a single box called a circuit, the larger box surrounding tasks two and three in the figure.

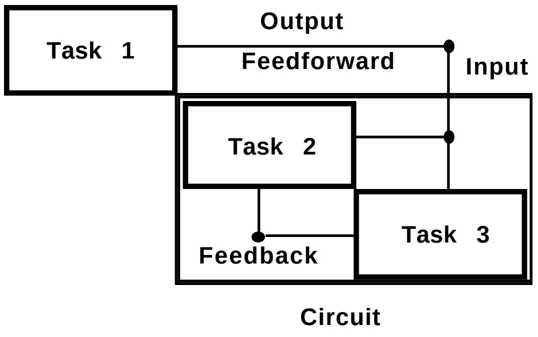

Fig. 2 Design Structure Matrix

Once the boxes have been reordered and grouped into circuits, the design process can be broken down into a hierarchical display of the circuits (fig. 3). From this display, the design manager can determine the order in which tasks are to be completed and which tasks (those on the same level) can be accomplished in parallel.

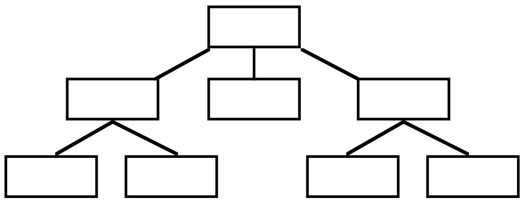

Fig. 3 Hierarchical Display oF Circuits

DeMAID performs several functions which are displayed in the boxes below the "Main Program" box in figure 1. In the planning function, tasks with output not used as input to another task are removed. In addition, the user receives a warning message when an input to a task is not satisfied by the output from another task. The scheduling function orders the tasks by minimizing the feedbacks and groups the iterative processes into circuits. Once the tasks have been ordered and grouped they can be displayed in a design structure matrix. While in the display function, the user can refine the sequence by manually reordering the tasks. Since there is no feedback among circuits, they can be displayed in a hierarchical multilevel format. In this format, tasks or groups of tasks (circuits) which appear at the same level can be executed in parallel. The tasks can also be displayed in a matrix format which indicates the dependency of one task on other tasks. The "Trace Change" function is new and is described in detail below.

After DeMAID was distributed to the public and applied to some in-house projects, it was determined that some new enhancements would be useful. Since DeMAID is concerned with tasks and their interactions, it was felt that it would be appropriate to enhance the tool by enabling it to order the tasks of an assembly line problem. For this type of problem, DeMAID could determine the order in which pieces are to be added and connected in an assembly process, and if any of the tasks could be accomplished in parallel. There should be no feedbacks in the assembly process. The second enhancement applies to complex analysis tasks. It allows the design manager to see what tasks must be redone if a change is made in some input data. This saves time and money since not all tasks need to be redone for every change.

Because DeMAID is composed of a knowledge-based system, the addition of new enhancements is quite simple. A new enhancement is added by generating a new file of rules, adding a new line to a menu or a sub-menu, and writing a small driver routine to load in the rules and call the inference engine. Since all the rule files are independent of one another, adding a set of rules for a new enhancement does not affect any rules already in the system.

Enhancement for Assembly Line Problems

The rules for the ordering of a complex assembly process are based on research at the Charles Stark Draper Laboratory (ref. 5). An example assembly is shown in figure 4. It is broken down into 11 parts labeled A-L (no I). Table 1 lists 18 liaisons connecting the different parts. The list in table 1 has five fields for each liaison. Field 1 indicates that the element is a liaison. Field 2 is the liaison number. Field 3 is a status, initially uk (for unknown). Fields 4 and 5 are the parts to be connected by this liaison. Table 2 lists 37 precedence constraints. Each precedent constraint in the list contains a variable number of fields to indicate the liaisons which must be done before or after other liaisons. Field 1 indicates that the element is a constraint. There is a variable number of fields between the delimiters "constraint" and "before" to indicate which liaisons must be done before the liaison in the last field of each line.

Fig. 4 Assembly Problem

The list of precedence constraints is developed by answering the two questions:

What liaisons must be done prior to doing liaison i?
What liaison must be left to be done after doing liaison i?

The development of this list is described in detail in reference 5.

A new file of rules was created for the knowledge base for application to an assembly problem. These rules are executed from the "plan" submenu in DeMAID and act as a preprocessor to the original DeMAID by ordering the liaisons for assembling the model and creating a standard input file for input to the planning function. There are many possible combinations of liaison sequences to accomplish the assembly. DeMAID creates a subset of these possible sequences where each sequence in this subset begins with a different unconstrained liaison. An unconstrained liaison has no other liaison which must be done before it. In this problem, the unconstrained liaisons are 3, 4, 6, 14, 15, and 18. New liaisons are added to each sequence as their constraints are satisfied, or if it is an unconstrained liaison and one of its parts has already been added by a previous liaison.

Once the potential sequences are complete, the engineer must select the preferred sequence. At this time, this is done subjectively based on the engineer's knowledge of the problem. For this example, the potential sequences of liaisons are listed in table 3. The sequence beginning with liaison 6 was chosen arbitrarily. DeMAID then lists the liaisons showing the connection of parts (table 4).

DeMAID then creates a list of modules (Note: The words modules and tasks are used interchangeably in this paper) in the standard format for displaying the design structure matrix in DeMAID. The ordered sequence is shown in table 5. There are three type of modules. (1) There are part modules named P# where the # is associated with a particular part. This type of module adds a part to the assembly line. (2) There are liaison modules named L# where the # is associated with a particular liaison. (3) Finally, there is a goal module named fin . This module takes as input all the modules not used as input to another module.

In the third field of the table, part modules are designated as type 1 (formally called weight in reference 1) to differentiate them from liaison modules (designated type 2) and the goal module (designated type 3). The output of the module is in the fourth field. A

P# module has the part name as its output, while an L# module has C# (for connection) as its output. The input requirements follow the uk status parameter in the fifth field.

DeMAID orders the list of modules by taking the modules containing the parts of the first liaison and adding them to the list. The first liaison module is then added to the list. DeMAID then examines each liaison module in the preferred sequence. If both of its parts are available then the liaison is added to the list. If a part is not available (at least one part is always available), the module containing the part is added to the list followed by the module containing the liaison. This list is then input to DeMAID to display the assembly process in a design structure matrix (part of which is shown in figure 5). In this figure, a box with a single letter indicates when to a part to the process and a box with two letters indicates a liaison where two parts are joined together. The program can also create a hierarchical display showing which tasks of the assembly process can be executed in parallel.

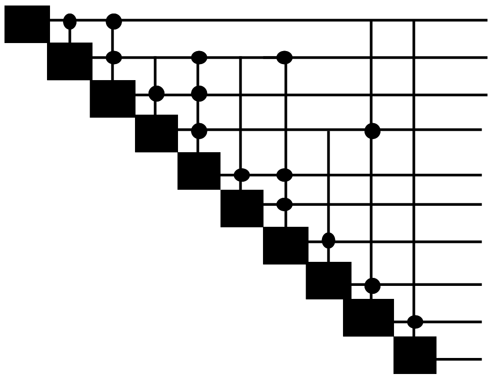

Fig. 5 Design Structure Matrix for Assembly Problem

Enhancement for Selecting Tasks for Re-Execution

The second enhancement to DeMAID addresses the situation where a change is made during an engineering analysis. Just because a change is made to some data in the design process, this does not mean that all the tasks in the analysis system must be re-executed. To incorporate this new enhancement into DeMAID required writing a new file of rules and adding a new option called "Trace Change" to the main menu in DeMAID to execute these rules. The planning and scheduling functions must be run before this set of rules can be applied.

The ordered list of tasks used to illustrate this enhancement is shown in table 6. Each line is divided into fields delimited by a space. Fields 1 and 2 contain the number and name of the task. The third field is a types field to differentiate among disciplines. For this sample problem, a type of 0 defines a task not in a discipline (such as input data or the final goal module), a 1 is for aerodynamics, a 2 is for structures, a 4 is for performance, and a 7 for a task common to all three disciplines. The uk status parameter in the fifth field is preceded by the output generated by the task and followed by the input requirements of the task.

This system carries out the analysis of an aircraft. It allows for the coupling existing among aerodynamic, performance, and structures disciplines. It calculates trim parameters, flexible static loads, flexible polar curves, elastic stresses and displacements, and aircraft performance (ref. 6).

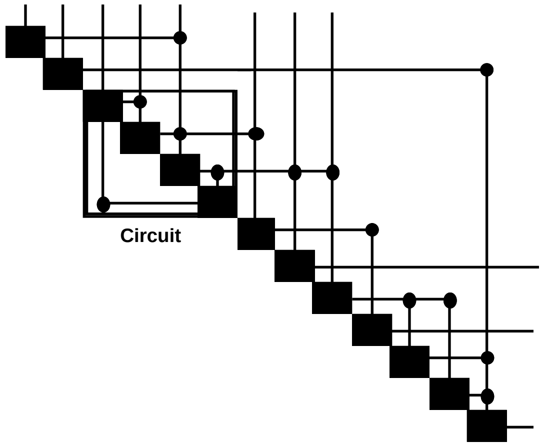

Fig. 6 Design Structure Matrix for Redesign Problem

Figure 6 displays a portion of the design structure matrix for this example. There is only one circuit in this problem and it contains modules 19 through 22, which correspond to the iterative calculations for trim and flexible static loads.

The engineer is asked whether or not to differentiate among disciplines. If there is no differentiation among disciplines, then the entire system is to be reanalyzed, assuming that all disciplines are changed. If there is to be differentiation among the disciplines then the system is to be reanalyzed assuming that only the discipline(s) to which the changed data belongs has changed, including tasks which might be common to two or more disciplines. The engineer is then asked what output data is to be traced back through the design system and what input has changed. DeMAID lists the tasks that must be re-executed to compute the new output given a change in the input. In addition, the listing indicates where there are iterations among tasks.

For example, suppose there is to be differentiation among disciplines and a change in variable XA (planform variable), and the engineer needs to know what effect this will have on computing the variable YPP (measures of performance). DeMAID will produce the following listing:

You must rerun the following programs:

Program WDMOD1a to compute EGEi
(aircraft geometry)
Program WINGDES1 to compute COPT
(optimum camber)
Program WDMOD2a to compute EGOi
(aircraft geometry with
optimum camber)
Program TPWEIGHTa to compute WEIC (concentrated weights)
Program ASP/FLOPS to compute YPP
(measures of performance)

As another example, suppose there is to be no differentiation among disciplines and a change in variable XP (gross weight), and the engineer needs to know what effect this will have on computing the variable YAP (elastic polar curves). DeMAID will produce the following listing:

You must rerun the following programs:

Program TPWEIGHTb to compute YPS (fuel weight and concentrated weight)
---- loop ---- Program SUPERPOS to
compute PRSC
(pressure distribution)
---- loop ---- Program TRIM to
compute LOAD
(load distribution)
---- loop ---- Program ELAPSb to
compute YSA
(wing deformations)
---- loop ---- Program CAMBER to
compute YAS
(pressure due to
deformations)

Program WDMOD3a to compute EGLi (aircraft geometry with elastic deformations)
Program WINGDES3 to compute CDIN (induced drag)
Program AWAVE to compute CDWA (wave drag)
Program POLAR to compute YAP (elastic polar curves)

In this example, the "loop" containing programs SUPERPOS through CAMBER implies an iterative process. This can be seen in the large box around the four smaller boxes in figure 6 and the programs are listed in bold type in table 6. Also notice that modules 18, 23, 24, and 26 are not re-executed and are therefore "blacked out" in the figure.

Summary

Two new enhancements have been added to DeMAID. The first applies to sequencing a complex assembly process. It allows the user to order and display the successive tasks in the process. The second applies to sequencing the analysis tasks in a complex analysis process. It allows the user to determine what subset of tasks need to be re-executed to compute a particular output when a change has been made to an input. In keeping with the general philosophy of DeMAID, these enhancements were added to save the design manager time and expense in the design process. In addition, these enhancements increase the flexibility of DeMAID as well as offer proof to the ease of adding new capabilities to a knowledge-based system.

References

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

2. Rogers, James L. and Padula Sharon L.: An Intelligent Advisor for the Design Manager. Proceedings of the First International Conference on Computer Aided Optimum Design of Structures, Southampton, UK, June 1989, pp. 169-177.

3. Steward, Donald V.: Systems Analysis and Management: Structure, Strategy and Design. Petrocelli Books Inc. c. 1981.

4. Riley, Gary; Culbert, Chris; Savely, Robert T.; and Lopez, Frank: CLIPS: An Expert System Tool for Delivery and Training. Third Conference on Artificial Intelligence for Space Applications - Part I, J. S. Denton, M. S. Freeman, and M. Vereen, compilers, NASA CP-2492, PP. 53-57.

5. DeFazio, Thomas L. and Whitney, Daniel E.: Simplified Generation of All Mechanical Assembly Sequences. IEEE Journal of Robotics and Automation. Vol. RA-3, No. 6, December 1987, pp. 640-658.

6. Barthelemy, Jean-Francois M. et al: Integrated Design Analysis and Optimization of Aircraft Structures. AGARD Report 784, 1992, pp. 4.1-4.5.

(liaison	1	uk	A	C)
(liaison	2	uk	A	D)
(liaison	3	uk	A	G)
(liaison	4	uk	A	K)
(liaison	5	uk	A	L)
(liaison	6	uk	B	C)
(liaison	7	uk	B	E)
(liaison	8	uk	B	G)
(liaison	9	uk	B	H)
(liaison	10	uk	B	J)
(liaison	11	uk	C	D)
(liaison	12	uk	C	E)
(liaison	13	uk	E	F)
(liaison	14	uk	G	H)
(liaison	15	uk	H	J)
(liaison	16	uk	H	K)
(liaison	17	uk	H	L)
(liaison	18	uk	J	L)

Table 1
List of liaisons for assembly problem

(constraint	6	before	1)
(constraint	3 8	before	1)
(constraint	1	before	2)
(constraint		before	3)
(constraint		before	4)
(constraint	3 14 15 4	before	5)
(constraint	3 14 15 16 17	before	5)
(constraint	3 14 18 4	before	5)
(constraint	3 14 18 16 17	before	5)
(constraint	3 17 15 4	before	5)
(constraint	3 17 15 16	before	5)
(constraint	3 17 18 4	before	5)
(constraint	3 17 18 16	before	5)
(constraint	14 17 15 4	before	5)
(constraint	14 17 15 16	before	5)
(constraint	14 17 18 4	before	5)
(constraint	14 17 18 16	before	5)
(constraint		before	6)
(constraint	2	before	7)
(constraint	1	before	8)
(constraint	3	before	8)
(constraint	3 14	before	9)
(constraint	8	before	9)
(constraint	8 15	before	10)
(constraint	3 14 15	before	10)
(constraint	9	before	10)
(constraint	1	before	11)
(constraint	2	before	12)
(constraint	7	before	13)
(constraint		before	14)
(constraint		before	15)
(constraint	17	before	16)
(constraint	3 4	before	16)
(constraint	4 14	before	16)
(constraint	15	before	17)
(constraint	18	before	17)
(constraint		before	18)

Table 2
List of precedence constraints for assembly problem

Potential sequences

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

Table 3
List of potential sequences of liaisons

Connection of parts:

Connect B to C
Connect A to C
Connect A to D
Connect A to G
Connect A to K
Connect C to D
Connect C to E
Connect B to G
Connect G to H
Connect H to K
Connect B to E
Connect B to H
Connect B to J
Connect E to F
Connect H to J
Connect A to L
Connect H to L
Connect J to L

Table 4
Preferred order for connecting parts (begin with liaison 6 from preferred list)

1	P4	1	B	uk	no-input
2	P5	1	C	uk	no-input
3	L6	2	C6	uk	B C
4	P1	1	A	uk	C6
5	L1	2	C1	uk	A C C6
6	P9	1	D	uk	C1
7	L2	2	C2	uk	A D C1
8	P2	1	G	uk	C2
9	L3	2	C3	uk	A G
10	P3	1	K	uk	C3
11	L4	2	C4	uk	A K
12	L11	2	C11	uk	C D C1
13	P10	1	E	uk	C11
14	L12	2	C12	uk	C E C2
15	L8	2	C8	uk	B G C3 C1
16	P6	1	H	uk	C8
17	L14	2	C14	uk	G H
18	L16	2	C16	uk	H K C14 C4 C3
19	L7	2	C7	uk	B E C2
20	L9	2	C9	uk	B H C14 C3 C8
21	P7	1	J	uk	C9
22	L10	2	C10	uk	B J C9
23	P11	1	F	uk	C7)
24	L13	2	C13	uk	E F C7
25	L15	2	C15	uk	H J
26	P8	1	L	uk	C4)
27	L5	2	C5	uk	A L C15 C14 C4 C3
28	L17	2	C17	uk	H L C15
29	L18	2	C18	uk	J L
30	fin	3	goal	uk	C18 C17 C12 C13 C10 C16 C5

Table 5
List of ordered assembly modules

1	INPUT1	0	EGE0	uk	no-input
2	INPUT2	0	XA	uk	no-input
3	INPUT3	0	IGE0	uk	no-input
4	INPUT4	0	XS	uk	no-input
5	INPUT5	0	XP	uk	no-input
6	WDMOD1a	7	EGEi	uk	EGE0 XA
7	WDMOD1b	7	IGEi	uk	IGE0 XA
8	WINGDES2a	2	PRSA	uk	EGEi
9	WINGDES1	7	COPT	uk	EGEi
10	WINGDES2b	2	PRSM	uk	EGEi
11	WINGDES2c	2	PRSR	uk	EGEi
12	WDMOD2b	7	IGOi	uk	IGEi COPT
13	WDMOD2a	7	EGOi	uk	EGEi COPT
14	WAREA	1	WARL	uk	EGOi
15	ELAPSa	2	YSP	uk	EGOi XS
16	TPWEIGHTa	4	WEIC	uk	XP EGOi
17	TPWEIGHTb	4	YPS	uk	XP EGOi
18	CD1	1	CDFR	uk	1 WARL
19	SUPERPOS	2	PRSC	uk	PRSM PRSR YAS
20	TRIM	2	LOAD	uk	PRSC PRSA IGEi YPS EGOi XS
21	ELAPSb	2	YSA	uk	LOAD EGOi XS
22	CAMBER	1	YAS	uk	YSA
23	ELAPSd	2	STRE	uk	LOAD EGOi XS
24	WDMOD3b	1	IGLi	uk	IGOi YSA
25	WDMOD3a	1	EGLi	uk	EGOi YSA
26	COLAPS	2	YSS	uk	STRE
27	WINGDES3	1	CDIN	uk	EGLi
28	AWAVE	1	CDWA	uk	EGLi
29	POLAR	1	YAP	uk	CDIN CDFR CDWA
30	ASP/FLOPS	4	YPP	uk	WEIC YAP YSP XP
31	FINI	0	goal	uk	YSS YPP IGLi

Table 6
Modules for redesign problem after ordering

[**] Senior Computer Scientist

[+] Senior Research Engineer, Senior Member AIAA