APPLYING THE DSM METHOD IN PRODUCT DESIGN

Daniel-Constantin ANGHEL[1]
[1]University of Pitesti, Romania

Article history:
Received: 10.06.2013; Accepted: 20.09.2013.

Abstract: *The design of product is an activity that involves a lot of tasks. These tasks can be achieved in various ways: sequentially, in parallel or overlapping. Proper piloting of these tasks can increase the performance of design process and consequently the success of the company. In this paper we present the method DSM. The aim of this method is to reduce the design iterations by reorganizing the design tasks from a project, based on the informational dependencies between them. A design experiment was conducted in order to apply this method on a real case.*

Keywords: DSM, design of product

INTRODUCTION

Today, in order to be competitive on the market, the companies are trying to improve the performance of their products and of their processes. The design stage is a very important phase in the product development cycle. In addition, the design process is subjected iterations so that a proper organization of the design tasks can have significant results on the performance of the design process and product.
The experiment was realized at the Laboratory Design and Product Development at the University of Pitesti, and his objective was the optimization of mode of completion the tasks necessary for design of a gearbox, using DSM method, in order to reduce the number of iterations.

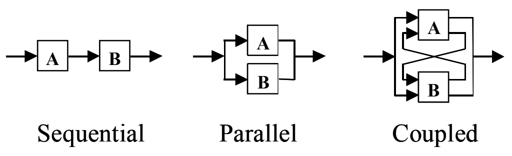
THE DESIGN STRUCTURE MATRIX (DSM)

In the DSM method, each design task can be modelled as an information processing task, using information supplied by other tasks, and creating information for other ones. In DSM, design tasks and their information dependencies are encapsulated within a compact matrix representation. The DSM matrix is square with one row and one column per task. The tasks are listed in a roughly chronological sequence of execution. In a simple DSM, diagonal elements of the matrix are not used. Off-diagonal elements indicate information flows between design tasks.

	A	B	C	D	E	F	G	H	I	J	K
A	■			X							X
B		■		X							
C			■	X			X		X		
D				■							
E					■		X				
F						■		X		X	
G			X		X		■				
H								■			
I			X	X			X	X	■		
J										■	
K				X		X					■

Fig. 1. Example of a DSM

Reading across a row indicates all of the tasks whose output is required to process the task corresponding to the row, whilst reading down a column indicates all of the tasks which receive output from the task corresponding to the column. There are three types of sequence relationship that links design tasks: serial (or independent), parallel (or dependent) and coupled (or interdependent). Figure 2 shows the three relationship types and their DSM representations.



	A	B
A	■	
B	X	■

	A	B
A	■	
B		■

	A	B
A	■	X
B	X	■

Fig. 2. Task relationships in a DSM

Upper diagonal elements of a DSM depict the existence of cyclic information flows. Matrix elements are manipulated in an attempt to eliminate or minimize the number of upper diagonal elements, a process known as partitioning. The remaining information cycles are resolved through iteration within the design process. The reordered elements of the DSM matrix shown in figure 1 are presented in figure 3 below.

	H	J	F	D	E	G	C	I	B	K	A
H	■										
J		■									
F	X	X	■								
D				■							
E					■	X	▨	▨			
G					X	■	▨	▨			
C				X	▨	X	■	X			
I	X			X	▨	X	▨	■			
B				X					■		
K			X	X						■	
A				X						X	■

Fig. 3. Reordered matrix with a group of coupled tasks

The original DSM method [1], [2] does not contain quantitative information about the strength of interaction between design tasks. Several extensions have been therefore developed to allow quantitative analysis.

THE DSM TOOL

For this work we have used a computer program DSM@MIT. This program is an application tool of the integrated project management framework proposed in the M.S. thesis, “An Integrated Method for Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation” by Soo-Haeng Cho at the Massachusetts Institute of Technology [3].
In the presented study, the program will be used for structuring the design tasks in order to reduce iterations, to reduce the developing time, to reduce the cost and for reducing the design errors.

Table. 1. Inputs and results in a DSM@MIT matrix

Inputs	Analyses Results
• A list of tasks • Information flows among tasks • Information flow patterns	• Design Structure Matrix (Identification of loops and process hierarchies among tasks) • Differentiation of planned & unplanned iterations • Identification of non-binding dependencies • A critical dependency sequence

The tool receives two types of information flows between tasks as shown in figure 4. The first type represents the case that a downstream task requires final output information from an upstream task to begin its work. The second type represents the case that a downstream task uses final output information in the middle of its process and/or begins with preliminary information but also receives a final update from an upstream task.

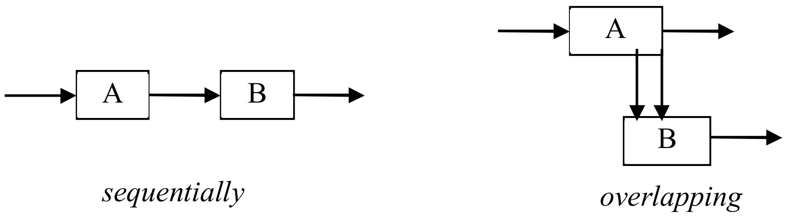
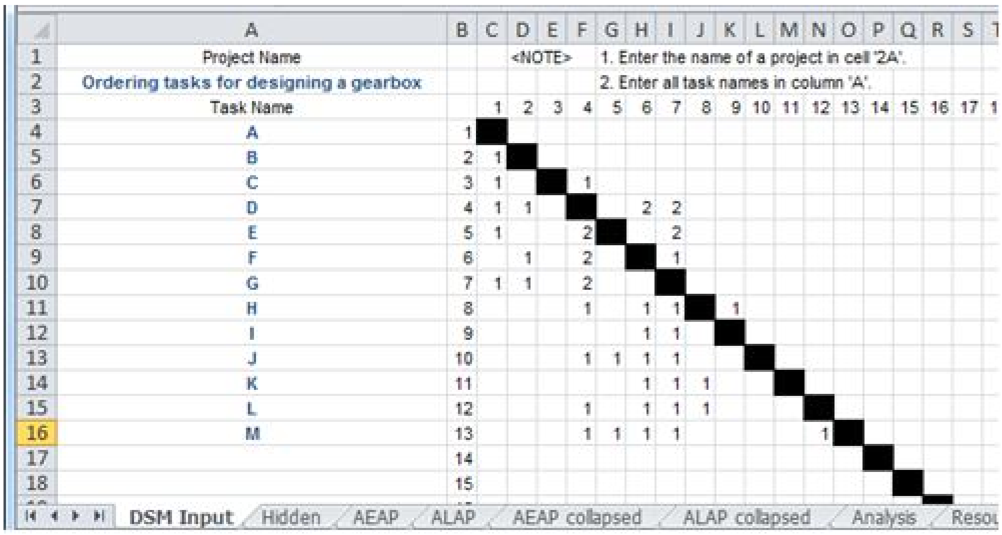


Fig. 4. The types of information flows between tasks in a DSM@MIT matrix

When there is the first type of information flow from task *A* to task B, enter ‘1’ into (*i*, *j*) of the square matrix where *i* and *j* are the unique indices representing tasks B and A, respectively. For the second type of information flow, enter ‘2’. Reading across a row reveals the tasks where the inputs of the task corresponding to the row come from. Reading down a specific column reveals the tasks receiving outputs from the task corresponding to the column. If the sequence of tasks in the matrix is the sequence of execution, a nonzero element in the upper diagonal represents a feedback. Figure 5 shows a sample of the inputs for the structuring module.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Project Name			<NOTE>			1. Enter the name of a project in cell '2A'.												
2	Ordering tasks for designing a gearbox						2. Enter all task names in column 'A'.												
3	Task Name		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	A	1	■																
5	B	2	1	■															
6	C	3	1		■	1													
7	D	4	1	1		■		2	2										
8	E	5	1			2	■		2										
9	F	6		1		2		■	1										
10	G	7	1	1		2			■										
11	H	8				1		1	1	■	1								
12	I	9						1	1		■								
13	J	10				1	1	1	1			■							
14	K	11						1	1	1			■						
15	L	12				1		1	1	1				■					
16	M	13				1	1	1	1					1	■				
17		14														■			
18		15															■		

DSM Input / Hidden / AEAP / ALAP / AEAP collapsed / ALAP collapsed / Analysis / Resou

Fig. 5. Sample of inputs for a DSM@MIT matrix

By executing *‘Analyze DSM’* in the menu, the tool performs analyses in the structuring module. The results are shown in the five worksheets: ‘AEAP’, ‘ALAP’, ‘AEAP collapsed’, ‘ALAP collapsed’, and ‘Analysis’.

By partitioning a project, iteration loops (cycles) are identified and tasks are rearranged. A *coupled block* represents a set of tasks constituting a loop, within which there is at least one dependency mark in the upper diagonal of the matrix. Due to its cyclic relationships among constituent tasks, iterations may take place potentially among the tasks within a coupled block. A *level* is determined such that tasks in the same level constitute a coupled block or they are independent of other tasks not belonging to the same coupled block. This multi-level structure presents a simple hierarchy of a project as well as concurrency of tasks – independent tasks and/or blocks in the same level can work in parallel.
In 'AEAP', tasks are sequenced based on the *as-early-as-possible rule* which assumes that a task starts immediately after all the inputs necessary to begin are available. In 'ALAP', tasks are sequenced based on the *as-late-as-possible rule* in which the starting time of a task is delayed to the extent that it does not delay entire project. 'AEAP collapsed' and 'ALAP collapsed' worksheets show the *collapsed views* of the DSMs in 'AEAP' and 'ALAP', respectively, where coupled blocks are collapsed to *block tasks*. In the collapsed views, information flow from a single task to a block task is marked as '2' when there exists at least one second-type flow between the single task and any constituent task within the block in the DSM. In contrast, any information flow from a block task to any single or block task is regarded as the second-type flow as it is very likely that preliminary outputs of tasks within the block are transferred to downstream tasks before converging to their final forms at the end of iterations.

RESULTS

The design experiment has been monitored during the entire duration of its.
CATIA software was used to designing all the parts, the assembly and to realize the mechanical analysis.

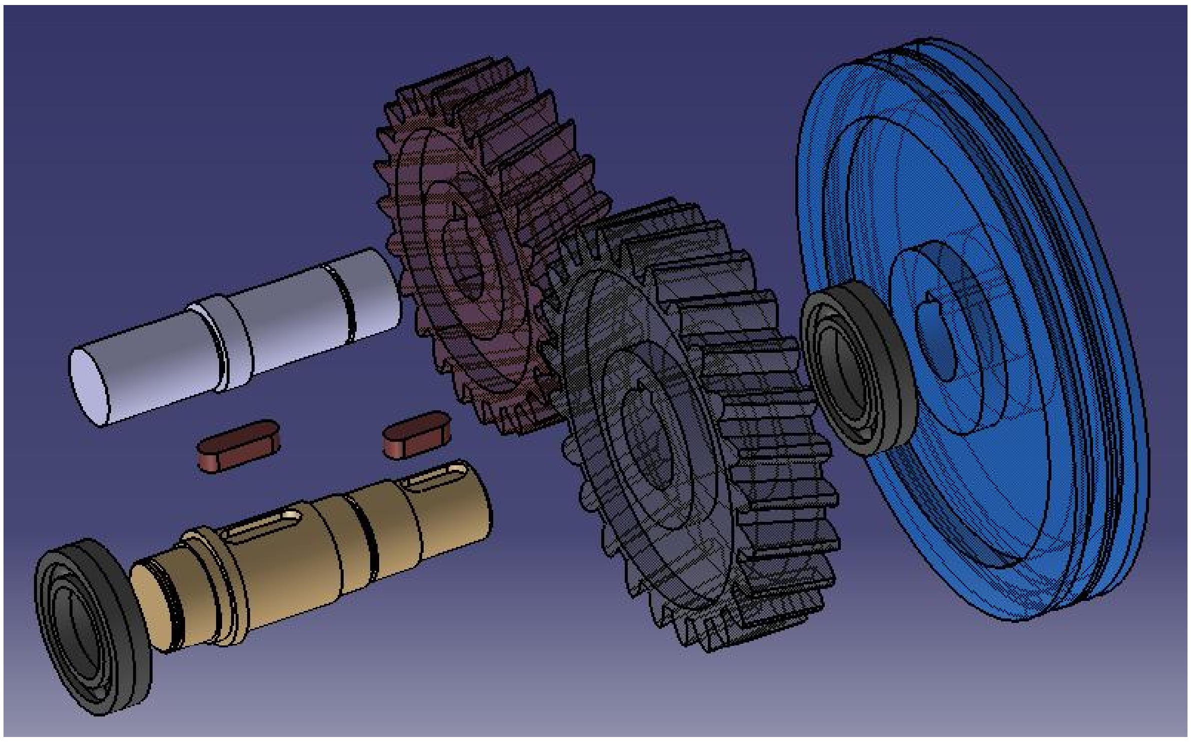
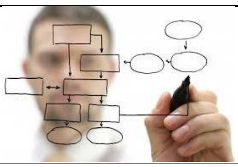
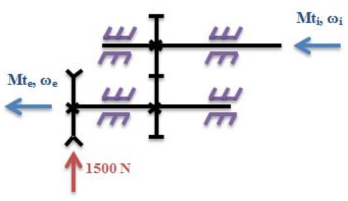
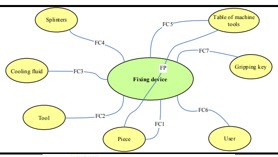
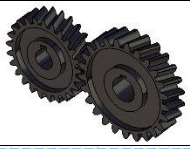
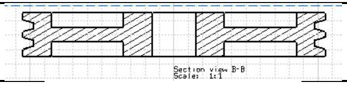
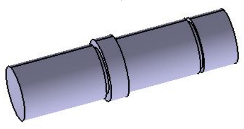
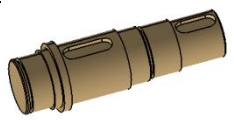
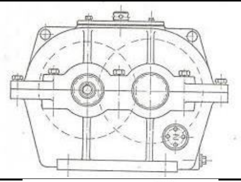
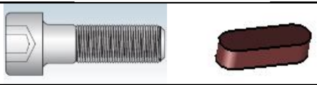


Fig. 6. Screen capture of CATIA software, during design

For our design experiment, we have identified 13 tasks, shown in table 2.

Table. 2. The tasks of the design experiment

Tasks	Name of the tasks	
A.	Analysis of the design requirements	
B.	Functional scheme proposal	
C.	Functional analysis of the product	 Splinters Table of machine tools Cooling fluid Gripping key FP Fixing device Tool Piece User FC1 FC2 FC3 FC4 FC5 FC6 FC7
D.	Gear design	
E.	Pulley design	
F.	Design of the input shaft	
G.	Design of the output shaft	
H.	Casing design	
I.	Choice of bearings	
J.	Choosing of fixing elements	
K.	Design of sealing elements	
L.	The choice of lubricant	

M.	Preparation of technical documentation	

Depending on the information dependencies between the tasks has been identified a total of six iterations: a long iteration (■), two medium iterations (■) and three short iterations (■), figure 7.

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	■												
B	1	■											
C	1		■	1									
D	1	1		■		2	2						
E	1			2	■		2						
F		1		2		■	1						
G	1	1		2			■						
H				1		1	1	■	1				
I						1	1		■				
J				1	1	1	1			■			
K						1	1	1			■		
L				1		1	1	1				■	
M				1	1	1	1					1	■

Fig. 7. The initial DSM matrix for the experiment

By reorganizing the matrix of connections between tasks, we are trying to reduce the number of iterations and transforming the long iterations into short iterations.
DSM @ MIT Tool allows reorganization of tasks in order to reduce the number of iterations based on two strategies: AEAP and ALAP, offering to designers the possibility to choice the appropriate strategy for each context.
Firstly, the AEAP (as-early-as-possible) rule was applied, figure 8. The number of iterations was reduced from 6 to 3.
Note that we could to transform the long iteration between the tasks G and D in a medium iteration, which reduces the time spent for the iterative process.
Thus, of the six iterations was remaining three: one medium and two short.

Number of tasks in the DSM : 13 AEAP

Task Name	Level		1	2	3	4	5	6	7	8	9	10	11	12	13	
A	1	1	■													1
B	2	2	1	■												2
D	3	3	1	1	■	2	2			Block1						3
F	3	4		1	2	■	1									4
G	3	5	1	1	2		■									5
C	4	6	1		1			■								6
E	4	7	1		2		2		■							7
I	4	8				1	1			■						8
H	5	9			1	1	1			1	■					9
J	5	10			1	1	1		1			■				10
K	6	11				1	1				1		■			11
L	6	12			1	1	1				1			■		12
M	7	13			1	1	1		1					1	■	13
			1	2	3	4	5	6	7	8	9	10	11	12	13	

Fig. 8. The reorganized DSM matrix by AEAP rule

Then, the ALAP (as-late-as-possible) rule was applied, figure 9. The DSM result is the same as above for this case.

Number of tasks in the DSM : 13 ALAP

Task Name	Level		1	2	3	4	5	8	9	7	12	6	10	11	13	
A	1	1	■													1
B	2	2	1	■												2
D	3	3	1	1	■	2	2			Block1						3
F	3	4		1	2	■	1									4
G	3	5	1	1	2		■									5
I	4	8				1	1	■								8
H	5	9			1	1	1	1	■							9
E	6	7	1		2		2			■						7
L	6	12			1	1	1		1		■					12
C	7	6	1		1							■				6
J	7	10			1	1	1			1			■			10
K	7	11				1	1		1					■		11
M	7	13			1	1	1			1	1				■	13
			1	2	3	4	5	8	9	7	12	6	10	11	13	

Fig. 9. The reorganized DSM matrix by ALAP rule

CONCLUSIONS

Currently, the reorganization of tasks within a project is a must. By applying the DSM method, we can control the design iterations, which can improve the organization of information necessary in product design.
DAM@MIT tool is a friendly and strong tool.
By applying the DSM@MIT tool, the designers can reduce the iterations, can reduce the developing time and can reduce the cost and the design errors.

REFERENCES

[1] STEWARD, D.V., *The design structure system: A method for managing the design of complex systems*, IEEE Transactions on Engineering Management, Vol. EM-28, N°3, 1981, pp. 71-74;
[2] CARRASCOSA, M., EPPINGER, S.D., WHITNEY, D.E, *Using the Design Structure Matrix to Estimate Time to Market in a Product Development Process*, ASME Design Automation Conference, Atlanta 98-6013, 1998;
[3] CHO, Soo-Haeng. *An integrated method for managing complex engineering projects using the design structure matrix and advanced simulation*. 2001. PhD Thesis. Massachusetts Institute of Technology.