DSM 기반의 프로세스 구조화 방법론

설현주*† · 김철현** · 이창용*** · 박용태***

* 공군사관학교 산업공학과
** 인덕대학 테크노경영학과
*** 서울대학교 산업공학과

A new approach to structuring the process based on design structure matrix (DSM)

Hyeonju Seol*† · Chulhyun Kim** · Changyong Lee*** · Yongtae Park***

* Department of Industrial Engineering, Korea Air Force Academy
** Department of Techno MBA,Induk Institute of Technology
*** Department of Industrial Engineering, Seoul National University

Key Words : Process Structuring, Module, Process Modularization, Design Structure Matrix, Algorithm, Supporting Tool

Abstract

This paper suggests a new process structuring method, which we call process modularization, for decomposing and grouping activities in a process. Above all, we propose the concept of a module that is a group of activities positioned on the same flow before and after control constructs. Since activities in a module are relatively strongly interrelated with one another, it is important to take into consideration of these together. A design structure matrix (DSM) is used to structure the process because it has a lot of advantages in process modeling and analysis. We developed two algorithms: the restricted topological sorting (RTS) algorithm for ordering activities and the module finding (MF) algorithm for detecting modules in a process, which utilize the DSM. The suggested approach enables a firm's manager to design and analyze the process effectively. We also developed a supporting tool to accelerate the progress of process modularization. The supporting tool aids the process manager in finding the module and understanding the process structure easily. An illustrative example is addressed to show operations of the suggested approach.

1. Introduction

Over the last several decades, firms have confronted with unprecedented changes such as globalization, political realignments and the rapid progress of information and communication technology. To successfully compete in a changing environment, it is important that firms have the ability to adapt to the environment. Business process reengineering (BPR) helps firms to be more responsive to a dynamic environment by implementing process orientated structures [18]. The BPR is the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in the measure of performance such as cost, time, quality, speed, and services [11]. It means that the BPR focuses on how work is executed rather than what kind of work is done. Therefore, a process plays a key role in the overall competitiveness of modern organizations and designing and implementing processes effectively is

† 교신저자 hjseol@afa.ac.kr

a crucial factor.

A process is a set of activities designed to produce a specified output for a particular customer or market [7]. In order to design or redesign a process, it is necessary to capture description of the relationships between the activities first. This is because the process must be fully evaluated and analyzed for the desired outcome before actually implementing the process. There are various process modeling methods such as flow charts and data-flow diagrams [26], Petri nets [22], IDEF techniques (IDEF 0, IDEF3) [19, 20], UML [2] and so on. These models allow organizations to be able to reinforce process understandability, increase communication capability, assess process feasibility, compare alternatives, and evaluate changes in the process, etc.

Once process modeling is done, it is required to structure the process, since it is very difficult to analyze all activities and their relationships simultaneously in one-dimension. The most general method for process structuring is decomposing the process into several sub-processes at a lower-level. Decomposition, which means dividing the problems into sub-problems, has long been recognized as a useful approach for simplifying large and complex systems [7]. There have been some studies related to the organization of the process by using decomposition approach. Johnson and Benson [16] assumed that all of sub-processes are separable. Kusiak *et al.* [20-22], Steward [29], Rogers and Bloebaum [28], and Eppinger *et al.* [10-12] applied the decomposition to group activities that have a high degree of cohesion within groups and low coupling among groups. Chen and Lin [5] used the decomposition concept to divide large interdependent task groups into smaller and manageable task groups. Yassine and Braha [31] proposed a unified solution approach to solve four critical problems involving decomposition problem.

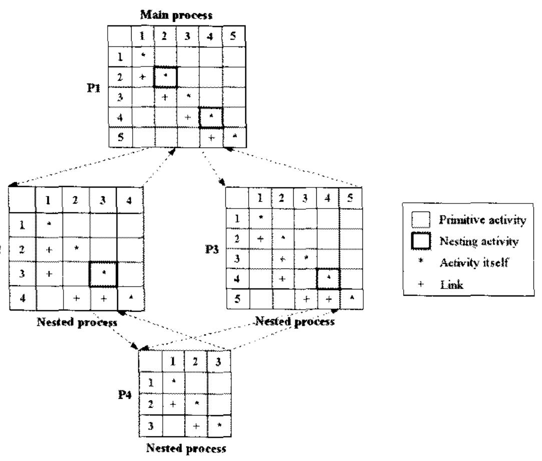
This paper presents a new approach to structuring the process on the basis of modular synthesis. Chen and Lin [5] developed three types of modular synthesis of mechanisms: structural fractionation based on product structure, functional fractionation based on same function, and kinematic fractionation based on kinematic influence. Contrary to Chen and Lin [5], the approach in this paper focuses on process flows. To do this, the concept of a module in a process is newly suggested. A module is defined as a group of activities which are divided by split or merge points in a process flow. In addition, to structure the process in a hierarchical architecture, the concept of a nested process suggested by Kim et al. [18] is employed. In a nested process, a complex process is broken into several sub-processes and structured in a hierarchical form. A nested process contains two types of activities: primitive and nesting. Primitive activities cannot be broken into smaller elements while nesting activities are all deployed into sub-processes. Furthermore, algorithms structuring the process in terms of a module are developed.

We propose a method using a design structure matrix (DSM) in structuring business processes. The DSM is a useful method for representing complex systems and their relationships in a simple and visual manner. And it gives helpful ideas for process improvement by operating rows and columns. Its effectiveness has been demonstrated in the analysis and management of the process [3, 4, 21, 23]. Along this line, the previous mentioned studies [5, 10-12, 16, 20-22, 28, 29] utilized the DSM to decomposing the process.

The remainder of this paper is organized as follows. Section 2 gives an overview of the DSM. Section 3 explains and illustrates the concept of a module suggested in this study. Section 4 addresses each steps of process modularization and explains algorithms related with modularizing process. Section 5 gives an illustrative example to demonstrate the operation involved in process modularization. Section 6 presents the supporting tool for structuring the process automatically. Finally, in section 7, some conclusions are drawn and the various implications and future research initiatives are discussed.

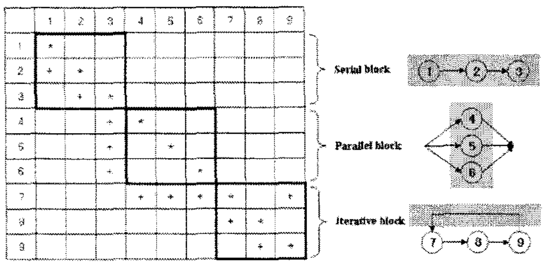
2. Design Structure Matrix

The DSM was originally developed by Steward [29] for information flow analysis and has been widely used to manage and analyze projects in the 1990s. The DSM have been applied to various areas such as building construction, semiconductor, automotive, photographic, aerospace, telecom, small-scale manufacturing, factory equipment and electronics industries [3]. Similarly to the incidence matrix in graph theory [9], a DSM is a square matrix with identical rows and columns. In the DSM, like a project task or system component, an activity in a given business process is positioned in a row and a corresponding column. The relationships between activities are represented by marking the cell that is formed by rows and columns related with the rows. <Figure 1 and 2> give illustrations of DSMs, each of which showing the nested process and the activity block. Further details are below.



<Figure 1> DSM representing nested process

In the DSM, an off-diagonal mark represents the dependency among activities and a diagonal cell indicates the activity itself. Here, an activity in a column is the preceding activity. For example, in the main process in <Figure 1>, activity 1 has some relationship with activity 2. This means that activity 1 is processed prior to activity 2 or that the output of activity 1 is used by activity 2. By structuring a large DSM into a hierarchy of smaller DSMs, the DSM can express the nested process. As shown in <Figure 1>, the main process is composed of five activities and has two nesting activities. These nesting activities are further deployed into processes P2 and P3. P4 is another nested process which is positioned at a lower level and shows how a nested process can be shared or can be used several times.



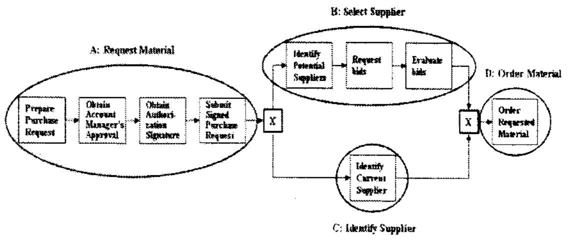
<Figure 2> Three types of an activity block

<Figure 2> shows how the DSM can represent activity blocks. In a process, there are three types of behavioral patterns, namely serial, parallel and iterative ones [1]. Since activities 1, 2 and 3 are connected consecutively, called a serial block, these activities can be executed sequentially. In a serial block, each of activities has only one activity before and after the activity. Activities 4, 5 and 6 can proceed in parallel, called a parallel block, and thus they do not depend on one another. A parallel block can be classified further according to the split and merge form: AND, OR, XOR-parallel block and so on. Activities 7, 8 and 9 form a cycle. This is called an iterative block and it is appears when some activities are carried out repeatedly.

As well as supporting the representation of a process, the DSM is useful for analyzing a process. First, it overcomes the size limitation of other graph-based process representation methods and offers a much more compact representation [5]. Second, important information about a process can be obtained through the DSM operations such as finding a mutually-separable process, discovering the order relationship among activities, and detecting cycles in a given process. They can be conducted with cluster identification algorithm [21, 22], topological sorting algorithm [15, 18], and Power of Adjacency matrix method [13] or triangularization algorithm [22], respectively. Third, the form of a matrix is amenable to program and calculate using computers [6, 30, 31].

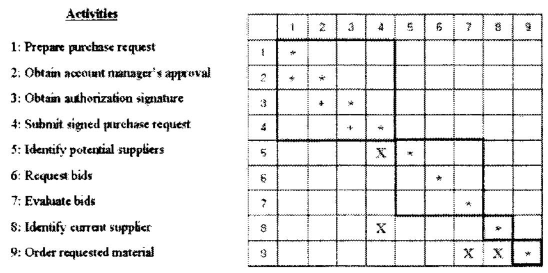
3. Concept of a module

The concept of a module is the underlying basis of our approach. A module, like an activity block, is a group of activities which are positioned between control constructs. However, they are different in that a module focuses not behavioral patterns but process flows. A module is the further segmented than an activity block, according to the flow existing between activities. Also, a module can be comprised of several activity blocks.



<Figure 3> An example of modules in a process

In <Figure 3>, a process with IDEF3 [25], the process is partitioned into four modules: A, B, C, and D. Modules A and B are composed of several activities but modules C and D are composed of only one activity. The first module in a process is a group A which is on the far left among the modules. The module is a starting point of a whole business process and is made of four activities: prepare purchase request, obtain account manager's approval, obtain authorization signature and submit signed purchase request. Since these activities are interrelated with one another for requesting material, it is necessary to consider together for progressing or improving a process. After activities in a module A are processed consecutively, the process meets new modules: B and C. Although these two modules constitute a parallel block together in a view point of an activity block, they have different flows: one is for selecting a new supplier and the other is for identifying a current supplier. Therefore, we differentiate these activity groups definitely and handle each of them as an independent logical unit, which is referred to as a module. As such, a module is defined as a group of activities divided by split or merge points in a process flow. Also, it is a set of activities which share a common objective. D is the last module for ordering material. <Figure 4> shows the DSM presenting the modules in <Figure 3>.



Activities

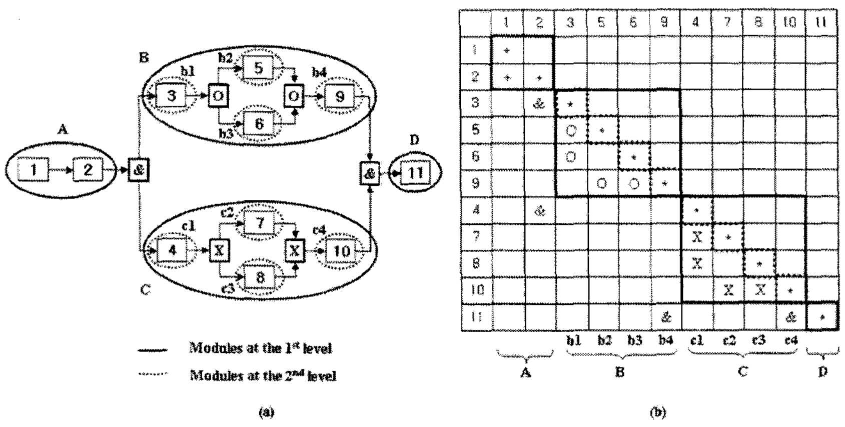
1: Prepare purchase request
2: Obtain account manager's approval
3: Obtain authorization signature
4: Submit signed purchase request
5: Identify potential suppliers
6: Request bids
7: Evaluate bids
8: Identify current supplier
9: Order requested material

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

<Figure 4> DSM representing modules in <Figure 3>

Like a nested process, a module can be further broken into sub-modules. To do this, we defined two types of modules: a primitive module and a nesting module. As shown in <Figure 5(a)>, a whole process is partitioned into four main modules: A, B, C, and D at the 1st level. These modules are at the highest level. An investigation of these modules reveals that not modules A and D but modules B and C can be further decomposed into sub-modules. Modules A and D are primitive modules while modules B and C are nesting modules. Nesting modules B and C are further decomposed into sub-modules b1, b2, b3, and b4, and c1, c2, c3, and c4 at the 2nd level respectively. These sub-modules are modules at relatively lower level and can also have sub-modules at further lower levels. That is, sub-modules comprise the module at a higher level, but at the same time, each sub-module can possess sub-modules at a lower level. <Figure 5(b)> shows the DSM representing the modules in (a) with IDEF3.

Structuring the process based on the module gives



<Figure 5> An example of modules for both 1st and 2nd level

several advantages in modeling and analyzing the process.

- It is conducive to the management and analysis of a complex process in that it reveals the process structure according to the logical unit.
- It allows a process manager to organize the cross-functional team in an appropriate manner because a cross-functional team may be constructed by the unit of a module instead of the whole process.
- It enables a process manager to think about process improvement with a broader perspective by offering both interrelated activities and a problematic activity.
- It increases the reusability of a process because a module can be regarded as a manufacturing part or a software component, and hence reduces unnecessary efforts for process design.

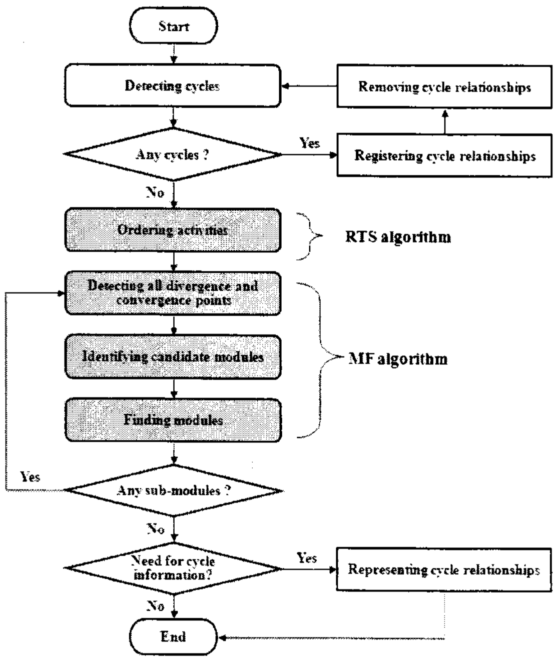
4. Process modularization

In order to modularize the process, it is necessary to represent the activities in the process and their relationships with the DSM. An activity is placed in a row and a corresponding column on the DSM. The relationships among activities are expressed by links and junction marks used in IDEF3. There are various approaches to represent the relationships between activities in the process. The DSM notation can be expressed with numerical values to describe the degree of relationships. On the other hand, the relationship is described with Boolean operators to only show the existence or nonexistence of the relationships. In this study, we augment DSM notation by links and junction marks used in IDEF3 since these well show the relationships between activities in the process. In representing a process with the DSM, we assume that there are no successive junction marks between activities. This is because the DSM cannot indicate successive junction marks in one cell formed by a row and a column. We can solve this problem by introducing 'dummy' activities.

<Figure 6> shows the overall flow of process modularization including major procedures represented in rounded rectangles colored gray. Before modularizing the process, a preliminary step is performed. This step, expressed in a rounded rectangle, is necessary to detect iterations in the process. When an activity needs to go back to a certain previous activity, cycles are formed in the process.

With the preliminary step, the information is registered and then, removed from the process. This procedure is repeated until there are no more cycles. A cycle is not a basic element of the process in a view point of process modularization since it appears when some activities are performed repeatedly.

Therefore, cycles are regarded as additional information and this step is used only to find and keep the information. This step does not need to go into more detail since there are several algorithms for finding cycles on the DSM such as path searching and power of adjacency matrix methods [13].


Start
Detecting cycles
Removing cycle relationships
Any cycles ?
Yes
Registering cycle relationships
No
Ordering activities
RTS algorithm
Detecting all divergence and convergence points
Identifying candidate modules
MF algorithm
Finding modules
Yes
Any sub-modules ?
No
Need for cycle information?
Yes
Representing cycle relationships
No
End

<Figure 6> Overall flow of process modularization

After the preliminary step is performed, main steps for modularizing the process proceed. The first major step is to order activities in the process. In general, the ordering in the DSM is a temporal sequence. That is, the DSM focuses on only two activities and their relationship in representing activities. So, it is required to induce activity order from a process view point. This is because a module is a group of activities divided by split or merge points in a process flow and process modularization is a process structuring method based on a module. To do this, we developed the restricted topological sorting (RTS) algorithm, which is further described later in this section. Then, three consecutive procedures, which are detecting all divergence and convergence activities, identifying candidate modules, and finding modules, play a crucial role in process modularization. These procedures are performed by the module finding (MF) algorithm that we developed to identify modules in a process. Each of them is corresponding to the phase Ⅰ Ⅱ and Ⅲ in the MF algorithm, respectively. More details on these procedures will be given in this section. Finally, two checking steps are undertaken. One is examining whether or not sub-modules exist at a lower level. If they do exist in detected modules, the module finding process, from detecting all divergence and convergence activities to finding modules, is reapplied to these modules to find modules at a lower level. The other is representing a cycle relationship to the DSM when the original process has a cycle relationship between activities.

4.1 Ordering activities

One of major procedures for process modularization is ordering the activities according to the process flows. Since the ordering in a DSM is temporal, the activities need to be reordered appropriately to precede the module finding process. For this purpose, we developed the RTS algorithm. The RTS algorithm originates in the topological sorting (TS) algorithm [15, 19] used to induce the order relations among activities. Two algorithms are different in selecting which activity is arranged first among the activities having no predecessor. The TS algorithm gives a priority to any activity without preceding activity in ordering while the RTS algorithm orders the immediate successor of the preceding activity which is already ordered in the previous stage first. Therefore, we give the name RTS due to its characteristics that a choice of an activity is restricted. The procedure of the RTS algorithm is as follows;

Step 1: Set i = 1 and make a search list having no elements.

Step 2: Find the every row {K} of the matrix with only one non-empty element and go to Step 9.

Step 3: If the number of {K} is 1, go to Step 6. But, if the number of {K} is 0, go to Step 5.

Step 4: Select one element k from {K} randomly, add the other elements of {K} to the search list and go to Step 6.
Step 5: If there is no element in search list, stop; Otherwise select the last element k of search list and delete that from the search list.
Step 6: Draw horizontal line through column k.
Step 7: Draw vertical line through row k.
Step 8: Label i the cross-out row k and column k of the matrix, set I = I + 1 and go to Step 2.
Step 9: Find every element {K`} in {K}which are not in a search list.
Step 10: Set {K} = {K`} and go to Step 3.

The algorithm first finds an arbitrary activity without a preceding activity and puts this activity to the first column and row of the DSM. Among the remaining activities, an activity having no preceding activity is found and placed in the next position of the DSM again. In this time, if there are activities that are immediate successors of the preceding activity positioned on the DSM already, these activities should have priority over all other activities. This process continues repeatedly until there is no remaining activity left. The computational complexity of the algorithm is O (n^2), where n is the number of activities in the DSM. The performance of the algorithm will be given in detail later with the numerical examples.

4.2 Module finding

The module finding process is composed of three procedures: detecting all divergence and convertgence activities, identifying candidate modules, and finding modules. These procedures all utilize the DSM which is reordered by the RTS algorithm. To perform these procedures, we developed the MF algorithm comprising three phases, each of which supports these three procedures individually. To enhance comprehension, we described the algorithm by the phase. Listed below is the notation used in the algorithm.

- s: the number of activities in DSM
- p: the level of modules
- $\{M\}_{p,T}$: a set of modules of level p with the size of T
- $\{L\}_{p,W}$: a set of modules of level p with the size of W which may have lower level modules
- $\{C\}_{N,}$: a set of activities having three elements in a column with the size of N
- $\{R\}_N$: a set of activities having three elements in a row with the size of N
- $\{G\}_{2N}$: a set of activity groups with the size of 2N
- c_i: the i_{th} element of $\{C\}_N$
- r_i: the i_{th} element of $\{R\}_N$
- g_i: the i_{th} element of $\{G\}_{2N}$
- U_i: the number of elements in g_i
- Y_p: the number of unmarked modules in level p

In phase I, the algorithm finds all divergence and convergence points in the design process, i.e. the activities from which following activities split or on which preceding activities merge are found. This is performed by finding all activities having three elements in a row or a column respectively. The step 0 in phase Ⅰis used for initialization of the whole MF algorithm. The phase I of the MF algorithm as follows;

- Phase I / *Detecting divergence and convergence activities.*

Step 0: Set p = 1 and DSM=the target matrix for module find- ing.
Step 1: Set loop = 1.
Step 2: Set i = 1.
Step 3: If i_{th} column of the DSM has three elements, add i_{th} activity to $\{C\}_N$.
If i_{th} row of the DSM has three elements, add i_{th} activity to $\{R\}_N$.
Step 4: If i < s, set i = i + 1 and go to Step 3.
Step 5: If p = 1 and N = 0, stop.
Step 6: If p > 1 and N = 0, go to Step 14 of Phase Ⅲ otherwise go to Step 1 of Phase Ⅱ

In phase II the MF algorithm finds all activities existing between divergence activities and convergence activities by tracing activities that follow the divergence activity and that precede the convergence activity. More details about this process are appeared in step 3 and step 8 shown below. Through these procedures, the algorithm finds candidate modules based on divergence and convergence activities.

– **Phase II/ *Identifying candidate modules.***

Step 1: Set $i = 1$.

Step 2: Select i_{th} activity c_i in $\{C\}_N$.

Step 3: Among the activities following the selected activity in a row, find all activities from the activity which has the first junction mark to the activity immediately prior to the next junction mark

Step 4: Make these activities a set g_i and add g_i to $\{G\}_{2N}$.

Step 5: If $i < N$, set $i = i + 1$ and go to Step 2.

Step 6: Set $i = 1$.

Step 7: Select i_{th} activity r_i in $\{R\}_N$.

Step 8: Among the activities preceding the selected activity in a column, find all activities from the activity immediately after the activity having the first junction mark to the activity which has the other junction mark immediately prior to the selected activity.

Step 9: Make the these activities found in Step 8 a set g_{i+N} and add g_{i+N} to $\{G\}_{2N}$.

Step 10: If $i < N$, set $i = i + 1$ and go to Step 7 otherwise, go to Step 1 of Phase III

In phase III the algorithm begins by identifying the modules among the candidate modules resulted from phase II These modules are discovered by eliminating the candidate modules generated from divergence or convergence activities which are also member of the activities in other candidate modules. Next, modules which are not based on the divergence or convergence activity are obtained. This process is executed by drawing vertical lines and horizontal lines on the DSM across all activities that exist in the modules created from divergence and convergence activities. When vertical and horizontal lines are drawn, activities in the DSM that are separated by these vertical and horizontal lines form groups. These groups and modules produced from divergence and convergence activities become modules that exist in the DSM, the target matrix for module finding.

– **Phase III// *Finding modules.***

Step 1: Set $i =1$, $j = 1$, $k = 1$.

Step 2: If $i > N$, go to Step 4 otherwise go to Step 3.

Step 3: If k_{th} element of g_j is c_i, mark g_i and go to Step 5 otherwise go to Step 5 directly.

Step 4: If k_{th} element of g_j is r_{i-N}, mark g_i and go to Step 5 otherwise go to Step 5 directly.

Step 5: If $k < U_j$, set $k = k + 1$ and go to Step 2 otherwise set $k = 1$.

Step 6: If $j < 2N$, set $j = j + 1$ and go to Step 2 otherwise set $j = 1$.

Step 7: If $i < 2N$, set $i = i + 1$ and go to Step 2.

Step 8: If the number of unmarked g_i is 0, go to Step 11; otherwise draw vertical lines and horizontal lines in DSM across all activities in all unmarked g_i. Find all activity groups separated by a border that is formed by crossing out the rows and columns in DSM.

Step 9: Add all unmarked activity groups and activity groups found in Step 8 to $\{M\}_{p,T}$.

Step 10: Among the activity groups in $\{M\}_{p,T}$, add activity groups generated from divergence and convergence activities and having activities that generate marked g_i to $\{L\}_{p,W}$.

Step 11: If $p = 1$ and $Y_1 = 0$, stop.

Step 12: If $p = 1$ and Y_1 is not 0, set DSM = l_1 of $\{L\}_{1,W}$, $p = p+1$ and go to Step 1 of Phase I.

Step 13: If loop < W of $\{L\}_{p-1,W}$, set loop=loop+1, DSM = l_{loop} of $\{L\}_{p-1,W}$ and go to Step 2 of Phase I.

Step 14: If $Y_p = 0$ stop otherwise, set $p = p + 1$ and go to Step 1 of Phase I.

The steps from 11 to 14 in phase III are used for checking whether or not sub-modules exist. If sub-modules do not exist, all the procedures for finding the module suspended otherwise, the MF algorithm defines the new DSMs and repeats to all the new DSM. Among the modules identified from divergence or convergence, modules having divergence or convergence activities of the eliminated candidate modules are new DSM for finding modules at a lower level. Then, the module finding process is reapplied to each of the new DSM to look for modules at the next lower level. This process ends when there are no more sub-modules in the DSM. The time complexity of the algorithm is $O(n^3)$, where n is the number of activities in the DSM. Table 1 shows the computation times on a PC with a Pentium 4 processor (2.53GH) and 1 GB RAM when both the RTS and the MF algorithms were applied to the numerical examples. It appears from the results that the computation time is affected by not the number of branches but both the number of activities and the number of levels in a process. Increasing the number of activities and levels caused an increase of computation time not only in the RTS algorithm but also in the MF algorithm.

<Table 1> Performance results of algorithms

Number of activities	Maximum number of branches	Maximum number of levels	Computation time(msec)*	
			RTS algorithm	MF algorithm
25	2	2	1187	610
		4	1203	1391
	4	2	1203	797
		4	1235	1432
50	2	2	7265	2110
		4	7331	4486
	4	2	7398	2311
		4	7604	4969

* Computation time is the sum of 1000 iteration times

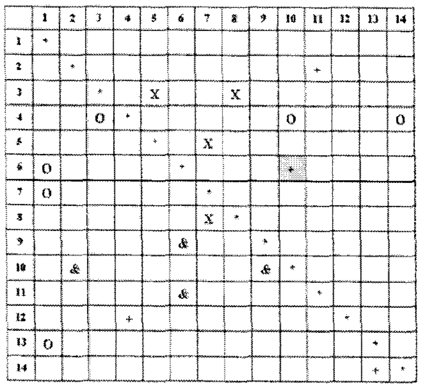
5. An illustrative example

To explain the newly suggested approach in this study, a hypothetical process composed of 14 activities has been made. The activities and their relationships are described through the DSM representation, which is appeared in <Figure 7(a)>.

5.1 Ordering activities

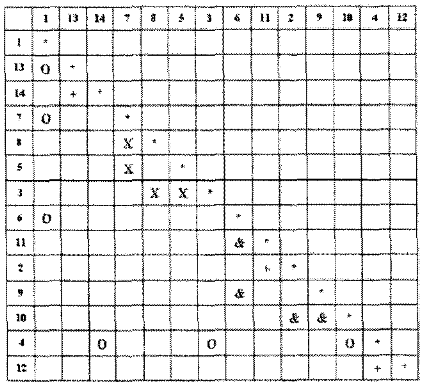
The first major step of process modularization is inducing the ordering of activities grounded on the flow among activities by use of the RTS algorithm. <Table 2> shows the procedures of the RTS algorithm applied to the hypothetical process and <Figure 7(b)> presents the final result. In <Table 2>, k is the selected activity for inducing order and the number of label indicates the sequence. The order of activities is 1-13-14-7-8-5-3-6-11-2-9-10-4-5.

The DSM in <Figure 7(a)> is the original DSM and the DSM in (b) is the outcome of the rearrangement



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	*													
2		*									+			
3			*		X			X						
4			O	*						O				O
5					*		X							
6	O					*				+				
7	O						*							
8							X	*						
9						&			*					
10		&							&	*				
11						&					*			
12				+								*		
13	O												*	
14													+	*

(a)



	1	13	14	7	8	5	3	6	11	2	9	10	4	12
1	*													
13	O	*												
14		+	*											
7	O			*										
8				X	*									
5				X		*								
3					X	X	*							
6	O							*						
11								&	*					
2									+	*				
9								&			*			
10										&	&	*		
4			O				O					O	*	
12													+	*

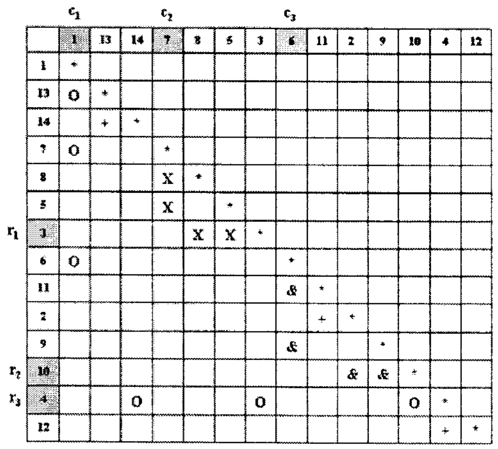
(b)

<Figure 7> Result of the RTS algorithm applied to the virtual process

<Table 2> Procedures of the RTS algorithm

I	Initial search list	Initial K	New K=K'	k	New search list	Label
1	{ }	(1)	(1)	1	{}	1
2	{ }	(6,7,3)	(6,7,13)	13	{6,7}	2
3	{6,7}	(6,7,14)	(14)	14	{6,7}	3
4	{6,7}	(6,7)	()	7	{6}	4
5	{6}	(5,6,8)	(5,8)	8	{6,5}	5
6	{6,5}	(5,6)	()	5	{6}	6
7	{6}	(3,6)	(3)	3	{6}	7
8	{6}	(6)	()	6	{}	8
9	{}	(9,11)	(9.11)	11	{9}	9
10	{9}	(2,9)	(2)	2	{9}	10
11	{9}	(9)	()	9	{}	11
12	{}	(10)	(10)	10	{}	12
13	{}	(4)	(4)	4	{}	13
14	{}	(5)	(5)	5	{}	14
15	{}	()	()	Stopped		

of activities with the RTS algorithm. The DSM (a) has a cycle relationship (colored gray) between activity 6 and activity 10, but this information does not appear in the DSM (b). This is because cycle relationship is registered and removed from the DSM before the RTS algorithm is applied. The order of activities in the DSM of <Figure 7(b)> is not a unique result of the RTS algorithm but merely one of the possible orders. The ordering results are dependent on which activity is chosen among the activities (K in <Table 2>) having the same priority in ordering. However, an arbitrary selection from these activities does not become problematic because the induced modules are the same irrespective of whichever or-

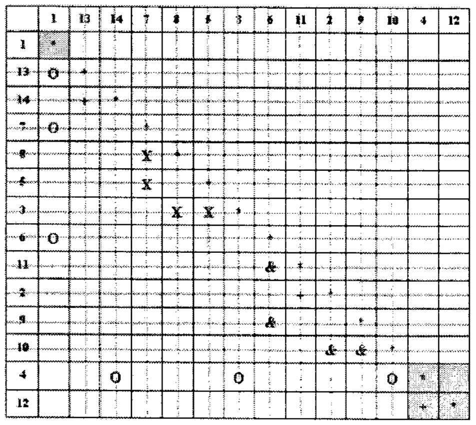


Phase	Divergence & convergence points

I	C = {1, 7, 6}
	R = {3, 10, 4}

Phase	Divergence & convergence activities	Candidate modules
II	c_1 = 1	g_1 = {13, 14}
		g_2 = {7, 8, 5, 3}
	c_2 = 7	g_3 = {8}
	c_3 = 6	g_4 = {11, 2}
	r_1 = 3	g_5 = {5}
	r_2 = 10	g_6 = {9}
	r_3 = 4	g_7 = {7, 8, 5, 3}
		g_8 = {6, 11, 2, 9, 10}

(a)



Phase	Candidate modules	Remarks
III (based on divergence & convergence activities)	g_1 = {13, 14}	Module
	g_2 = {7, 8, 5, 3}	Module
	g_3 = {8}	Eliminated
	g_4 = {11, 2}	Eliminated
	g_5 = {5}	Eliminated
	g_6 = {9}	Eliminated
	g_7 = {7, 8, 5, 3}	Eliminated
	g_8 = {6, 11, 2, 9, 10}	Module

Phase	Activity group	Remarks
III (based on the border by Modules)	{1}	Module
	{4, 5}	Module

(b)

<Figure 8> Module finding process applied to the DSM (b) in <Figure 7>

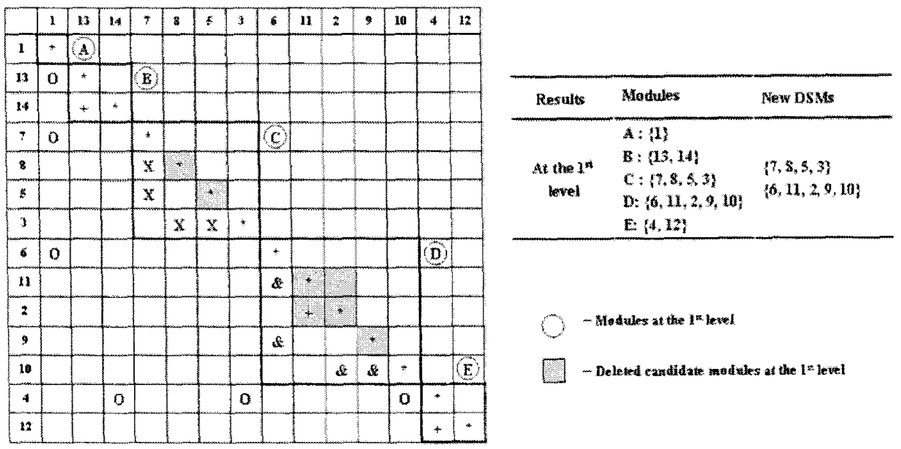
dering outcome is used to find modules in a process. In the DSM of <Figure 7(a)>, there are 20 possible activity orderings. Among them one is produced in this case which is represented in the DSM of <Figure 7(b)>.

5.2 Module finding

Based on the ordered result of activities with the RTS algorithm, the module finding process is executed. <Figure 8> shows this process according to the phases when the MF algorithm is applied once to the DSM in <Figure 7(b)>. <Figure 8(a)> presents the process for detecting the divergence and convergence activities and candidate modules. Since activity 1 in the 1st column has four elements (more than two) corresponding to the rows of activities 1, 13, 7 and 6 in the DSM, activity 1 becomes one of the divergence activities ($c_1=1$). Activities 7 and 6 in columns are also divergence activities because each of them has three elements in the corresponding row. Unlike the divergence activities, activities in rows having more than two elements in the corresponding columns are the convergence activities ($r_1=3$, $r_2=10$, $r_3=4$).

In phase II the algorithm finds all candidate modules composed of activities that follow the divergence activities and that precede the convergence activities. By tracing from activity 13 which has the first junction mark (O) to activity 14 immediately prior to the next junction mark (O), one of candidate modules ($g_1=\{13, 14\}$) is elicited. This is a candidate module produced from a divergence activity $c_1=1$. The other candidate module from a divergence activity $c_1=1$ is $g_2=\{7, 8, 5, 3\}$ which is composed of activities positioned between activity 7 and activity 3. In this case, activity 7 is the activity having the second junction mark whereas activity 3 is the activity just before the next junction mark. As such candidate modules based on divergence activities are obtained by tracking from an activity having a junction mark to an activity just prior to an activity having the next junction mark. There are four candidate modules in a given process which are based on the divergence activities: $g_1=\{13, 14\}$, $g_2=\{7, 8, 5, 3\}$, $g_3=\{8\}$ and $g_4=\{11, 2\}$. In case of a convergence activity, among all preceding activities of the convergence activity, from activities positioned immediately after junction mark to the activities having the next junction mark become the components of the candidate module. Candidate module $g_5=\{5\}$, $g_6=\{9\}$, $g_7=\{7, 8, 5, 3\}$ and $g_8=\{6, 11, 2, 9, 10\}$ are generated from the convergence activities $R_3=\{3, 10, 4\}$.

After finding all candidate modules based on the divergence and convergence activities, the algorithm looks for modules. <Figure 8(b)> shows the results of this process. Phase III begins by identify-



	1	13	14	7	8	5	3	6	11	2	9	10	4	12
1	*	A												
13	O	*		B										
14		+	*											
7	O			*				C						
8				X	*									
5				X		*								
3					X	X	*							
6	O							*					D	
11								&	*					
2									+	*				
9								&			*			
10										&	&	*		E
4			O				O					O	*	
12													+	*

Results	Modules	New DSMs

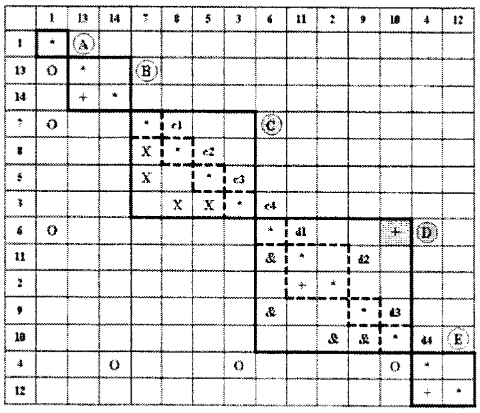
At the 1st level	A : {1}
	B : {13, 14}
	C : {7, 8, 5, 3}
	D: {6, 11, 2, 9, 10}
	E: {4, 12}
		{7, 8, 5, 3}
		{6, 11, 2, 9, 10}

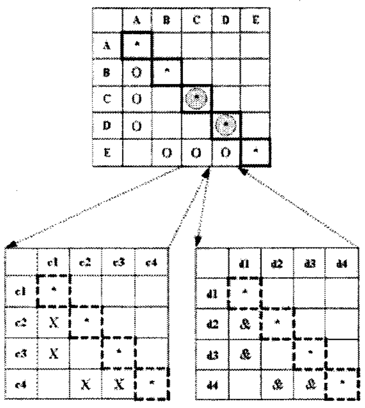
– Modules at the 1st level
– Deleted candidate modules at the 1st level

<Figure 9> Results of the modularized at the 1st level

ing modules among the candidate modules discovered in phase II These modules are found by removing the candidate modules. There are two cases when candidate modules are eliminated: one is when candidate modules are also members of other candidate modules and the other is when there are several candidate modules comprised of identical activities. For example, in (b), candidate module g_3={8}, g_4={11, 2}, g_5={5}, and g_6={9} are eliminated, since these are also subsets of candidate module g_2={7, 8, 5, 3}, g_7={7, 8, 5, 3}, and g_8={6, 11, 2, 9, 10}. A candidate module {7, 8, 5, 3} is also deleted because of being duplicated. The candidate module {7, 8, 5, 3} is duplicated because it is generated from both a divergence activity (c_1=1) and a convergence activity (r_3=4). In this study, we removed the candidate module g_7={7, 8, 5, 3} produced from a convergence activity (r_3=4). As a result, the remaining candidate modules g_1={13, 14}, g_2={7, 8, 5, 3}, and g_8={6, 11, 2, 9, 10} became modules based on the divergence and convergence activities of a virtual process in <Figure 6>. Next, the horizontal and vertical lines are drawn on the DSM through activities (13, 14, 7, 8, 5, 3, 6, 11, 2, 9, 10) that are members of module g_1, g_2, and g_8. These lines separate and group the remained activities in the DSM and each group formed by these lines is also a module in a given process. In this case, activity group {1} and {4, 12} are the modules that are generated from the horizontal and vertical lines. This procedure is represented in the DSM in (b).

As far, the algorithm found all modules existing in the 1st level of a given process and the DSM as shown in <Figure 9>. This Figure shows modules A={1}, B={13, 14}, C={7, 8, 5, 3}, D={6, 11, 2, 9, 10}, and E={4, 12}. Among the modules generated by the divergence and convergence activities in the 1st level, modules involving the deleted candidate modules (C and D) are defined as the new DSMs. Then, the whole module finding process is reapplied to each new DSM to find modules at the 2nd level. <Figure 10> shows the final results after the MF algorithm was applied to a hypothetical process. In this <Figure 10>, the bold lines represent the modules at the 1st level and the dotted lines represent the modules at the 2ndlevel. Consequently, a hypothetical process is composed of five modules at the 1st level and two modules (C and D) have sub-modules at the 2nd level, each of which has four modules. They are c1={7}, c2={8}, c3={5}, and c4={3} in a module C, and d1={6}, d2={11, 2}, d3={9}, and d4={10}in a module D. The gray colored cell indicates the cycle relationship which was in the original process. The DSM in <Figure 10(a)> represents the modules one-dimensionally while the DSMs in (b) show the modules in a hierarchical form.


(a)


(b)

<Figure 10> Final results of the modularized process

6. Supporting tool

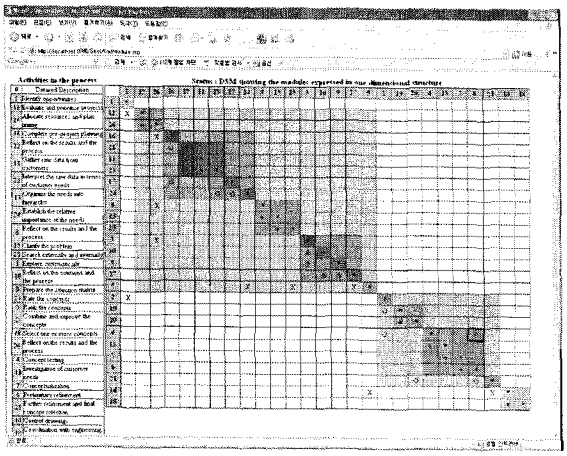
In order to support the modularization process, we developed a web-based system with JSP, J2SDK1.4.2, and Tomcat 4.1. The system can create DSM comprising any number of activities a user wants. The relationships between activities are expressed by selecting links and junction marks provided by a selection button. <Figure 11 and 12>, a process with 27 activities, display the final results in a different point of view when the whole modularization process is performed. The left sides of these figures show a detailed description of the activities and the right sides show the DSMs. The DSM in <Figure 11> exhibits modules in a one-dimensional way while the DSM in <Figure 12> exhibits modules in a hierarchical structure. These two interfaces are supplementary to each other. The interface in <Figure 11> provides detailed information about modularized process with activities involved in modules, whereas the interface in <Figure 12> gives more abstracted information about the modularized process. In <Figure 11>, the level of modules is differentiated according to the brightness of gray color. The modules at the same level are represented by the same brightness. And the lower the level of a module is, the darker the color representing the module is. For example, modules {16}, {22, 11, 23}, {17}, and {24} are modules at the same level and at the most lower level. The cell which is drawn bold line indicates the cycle relationship between activity 6 and activity 4.

In <Figure 12>, we can see that the entire process is composed of four modules at the 1st level and two modules (B and C) have sub-modules at the 2nd level respectively. In addition, sub-modules B2 and B4 have sub-modules at a lower level. However, it is unknown what kinds of activities each of the modules comprises of. Although, the interface in <Figure 12> gives abstract information, we can see detailed activities information tentatively by placing a mouse on the modules. For example, among the modules at the first level, we can see the module B composed of activities 12, 26, 16, 22, 11, 23, 17, 24, 8, 15, 25, 3, 10, 9, 27, 5 by positioning a mouse to the module.

7. Conclusion

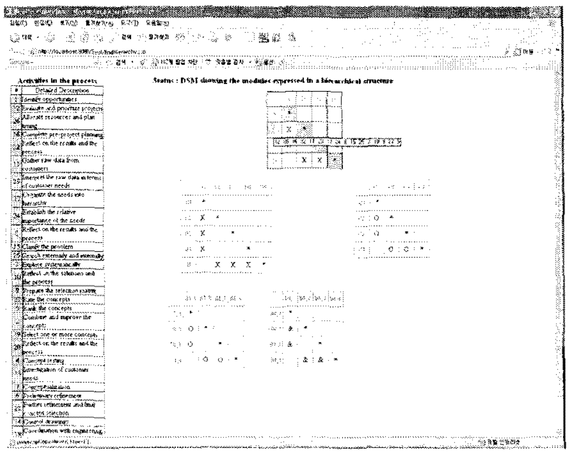
Process modeling, that represents activities and their relationships, is helpful in understanding and analyzing the process. However, since it is very difficult to fully understand the process with the increasing number of activities in a process, structuring the process has become essential.

This paper presents a new approach to structuring the process on the basis of modular synthesis. In connection therewith, the concept of a module is newly suggested in a view point of a process. A


Activities in the process

Status : DSM showing the modules expressed in one dimensional structure

<Figure 11> Modules expressed one-dimensional way


Activities in the process

Status : DSM showing the modules expressed in a hierarchical structure

<Figure 12> Modules expressed in a hierarchical form

module is defined as a group of activities which are divided by split or merge points in a process flows. In order to structure the process by the unit of a module, which is called process modularization, activities and their relationships need to be described first. The DSM is employed as a process representation method because it has a lot of advantage in expressing the process and gives useful information in analyzing the process. Process modularization is composed of two main steps: ordering activities based on the process flow and detecting modules. For the purpose of modularizing the process, two algorithms are developed. One is the RTS algorithm for ordering activities and the other is the MF algorithm for finding modules.

Modularizing a process based on the concept of a module has several advantages: it conduces to managing and analyzing a complex process, organizing the cross-functional team efficiently, giving the broad view points in process improvement, and increasing the reusability of a process. Process modularization enables a process manager to manage and analyze the process in an efficient manner. However, it must be noticed that there is some room to be considered when modularizing the process. First, since a module is not an independent unit entirely separated from the whole process but a member of the process, relationships between modules should also be considered as a whole. Second, related to the first remark, decision making that focuses on a specific module would be in conflict with the purpose of the whole process. Third, there are more meaningful cases when a module is defined by not process flows but resources such as people and equipments. For example, if subsequent activities are performed by different people in different organizational units, it would make sense to allocate them to different modules. Especially, this view point is very important when activities comprising the process depend on people who have specific knowledge, experience, or skills. Nevertheless, considering that a process view is an ever increasing importance in the overall competitiveness of modern organizations, the suggested approach is implicative in that the emphasis is on process flows rather than on functions.

To illustrate the proposed approach, the hypothetical design process was used. Although the process is not a real process, it is enough to understand the operation involved in process modularization and the promising value of the suggested approach. However, in the real world, the process is more complicated than the process used in the illustrative example in terms of the number of activities and the relationships between activities. Therefore, it is necessary to apply this approach to various processes which have the different degree of complexity in activities and their relationships. This is a further research issue to be considered.

References

[1] Bae, J., Bae, H., Kang, S., Kim, Y., "Automatic Control of Workflow Processes Using ECA Rules", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 8, pp. 1010-1023, 2004.

[2] Booch, G., Rumbaugh, J., Jacobson, I., The Unified Modeling Language: User Guide, Addison-Wesley, 1999.

[3] Browning, T. R., "Applying the design structure matrix to system decomposition and integration problems: a review and new directions", IEEE Transactions on Engineering Management, Vol. 48, No. 3, pp. 292-306, 2001.

[4] Browning, T. R., "Process integration using the design structure matrix", System Engineering, Vol. 5, No. 3, pp. 180-193, 2002.

[5] Chen, S., Lin, L., "A Project Task Coordination Model for Team Organization in Concurrent Engineering", CONCURRENT ENGINEERING: Research and Applications, Vol. 10, No. 3, pp. 187-202, 2002.

[6] Chen, S., Lin, L., "Decomposition of interdependent task group for concurrent engineering", Computers and Industrial Engineering, Vol. 44, pp. 435-459, 2003.

[7] 6. Courtois, P. J. (1985).On Time and Space Decomposition of Complex Structures, *Communication of ACM*, **2**: 590-603.

[8] Davenport, T., Process Innovation, Harvard Business School Press, Cambridge, 1993.

[9] Deo, N., Graph Theory with Applications to

Engineering and Computer Science, Prentice-Hall, Englewood Cliffs, 1974.

[10] Eppinger, S. D., Whitney, D. E. and Gebala, D. A. (1992). Organizing the tasks in complex design projects: Development of tools to represent design procedures, *Proceeding NSF Design and Manufacturing System Conference*, Atlanta, GA.

[11] Eppinger, S. D., Whitney, D. E., Smith, R. P. and Gebala, D. A. (1990). Organizing the tasks in complex design projects, *Proceedings 2nd ASME International Conference on Design Theory and Methodology*, Chicago, IL, 39-46.

[12] Eppinger, S. D., Whitney, D. E., Smith, R. P. and Gebala, D. A. (1994). "A model-based method for organizing tasks in product development", *Research in Engineering Design*, **6**: 1-13.

[13] Gebala, D. A., Eppinger, S. D., Methods for analyzing design procedures, in: Proceedings of the 3rd international conferences on design theory and methodology, Miami FL, pp. 227-233, 1991.

[14] Hammer, M., Champy, J., Reengineering the Corporation: A Manifesto for Business Revolution, Harper Business, New York, 1993.

[15] Horowitz, E., Sahni, S., Fundamentals of Data Structures. Rockville: Computer Science Press, 1982.

[16] Johnson, R. C. and Benson, R. C. (1987). "A Basic Two-level Monotonicity-Based Decomposition Method", *Proceedings of the ASME Design Automation Conference*, Boston, MA, 41-48.

[17] Johnson, R. C. and Benson, R. C. (1987). "A Basic Two-level Monotonicity-Based Decomposition Method", *Proceedings of the ASME Design Automation Conference*, Boston, MA, 41-48.

[18] Kim, Y., Kang, S., Kim, D., Bae, J., Ju, K., "WW-FLOW: Web-Based Workflow Management with Runtime Encapsulation", IEEE Internet Computing Vol. 4, No. 3, pp. 55-64, 2000.

[19] Kusiak, A., Engineering Design: Product, Processes, and Systems, Academic Press, San Diego, 1999.

[20] Kusiak, A., Larson, T. N. and Wang, J. (1994). Reengineering of design and manufacturing processes, *Computers and Industrial Engineering*, **26**(3): 521-536.

[21] Kusiak, A., Wang, J., "Decomposition of the design process", Journal of Mechanical Design, Vol. 115, pp. 687-695, 1993.

[22] Kusiak, A., Wang, J., "Efficient organizing of design activities", International Journal of Production Research, Vol. 31, No. 4, pp. 753-769, 1993.

[23] Lindsay, A., Downs, D., Lunn, K., "Business processes-attempts to find a definition", Information and Software Technology, Vol. 45, pp. 1015-1019, 2003.

[24] Mayer, R. J., Benjamin, P. C., Caraway, B. E., Painter, M. K.,A Framework and a Suite of Method for Business Process Reengineering in Business Process Change: Concepts, Methods and Technologies, Idea Group Publishing, Harrisburg, 1995.

[25] Mayer, R. J., Menzel, C. P., Painter, M. K., deWitte, P. S., Blinn, T., Perakath, B., Information integration for concurrent engineering (IICE): IDEF3 process description capture method report, Knowledge Based Systems, College Station, Texas, 1995.

[26] Park, H., Cutkosky, M. R., "Framework for Modeling Dependencies in Collaborative Engineering Process", Research in Engineering Design, Vol. 11, pp. 84-102, 1999.

[27] Reisig, W., Petri Nets: An Introduction, Springer-Verlag, Berlin Heidelberg, 1985.

[28] Rogers, J. L. and Bloebaum, C. L. (1994).Ordering design tasks based on coupling strengths, *Proceedings 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama city, FL, 708-717.

[29] Steward, D. V., "The design structure system: A method for managing the design of complex system", IEEE Transactions on Engineering Management, Vol. 28, pp. 71-74, 1981.

[30] Tang, D., Zheng, L., Li Z., Li, D., & Zhang, S., "Re-engineering of the design process for concurrent engineering", Computers and Industrial Engineering, Vol. 44, pp. 435-459, 2000.

[31] Yassine, A., Braha, D., "Complex Concurrent Engineering and the Design Structure Matrix Method", CONCURRENT ENGINEERING: Research and Applications, Vol. 11, No. 3, pp. 165-176, 2002.

[32] Yourdon, E., Modern Structured Analysis, Yourdon, Englewood Cliffs, NJ, 1989.