Design Process Modularization: Concept and Algorithm

Hyeonju Seol, Chulhyun Kim, Changyong Lee and Yongtae Park

Concurrent Engineering 2007; 15; 175
DOI: 10.1177/1063293X07079321

CONCURRENT ENGINEERING: Research and Applications

Design Process Modularization: Concept and Algorithm

Hyeonju Seol,[1] Chulhyun Kim,[2] Changyong Lee[2] and Yongtae Park[2,*]

[1]*Department of Industrial Engineering, Korea Air Force Academy, Namil, Cheongwon, Chungbuk 363-849, Korea*
[2]*Department of Industrial Engineering, School of Engineering, Seoul National University*
San 56-1, Shillim-Dong, Kwanak-Gu, Seoul 151-742, Korea

Abstract: The product design process is a complex set of integrated efforts, including generating ideas, developing concepts, modifying details, and evaluating proper solutions. The difficulties in designing complex products stem not only from their engineering complexity but also from the managerial complexity. First and foremost, what should be done to deal with the complexity problems is to capture all the constituent design activities and identify inter-dependency among respective activities. Further, once process modeling is done, it is required to structure the design process for better understanding of the process. This study presents a new approach to structuring the design process on the basis of modular synthesis. To do this, the concept of a module is newly proposed in the view point of a process. A module is defined as a group of activities which are divided by split or merge points in a process flow. The design structure matrix (DSM) is used to structure the process because it has a lot of advantages in process modeling and analysis. Two algorithms are developed: the restricted topological sorting (RTS) algorithm for ordering activities and the module finding (MF) algorithm for detecting modules in a process. Both of the algorithms are based on the DSM. Structuring the process in terms of a module, which is called as process modularization, allows a process manager to manage and analyze the process effectively. The overall process and detailed procedures of the suggested approach are presented and an illustrative example is addressed to show the practical operation of the approach.

Key Words: product design process, modular synthesis, module, process modularization, design structure matrix, module finding algorithm.

1. Introduction

The product design process is a complex set of integrated efforts, including generating ideas, developing concepts, modifying details, and evaluating proper solutions [12]. The difficulties in designing complex products do not stem simply from the engineering complexity. The managerial complexity, which occurs when managing the collaborations among different engineering disciplines, imposes additional challenge on the design process [30,32]. In order to deal with the complexity problems, it is necessary to capture all the constituent design activities and identify inter-dependency among respective activities [32]. Since the vast amount of knowledge used in the design process is certainly more than any individual can manage, there arises indispensable need for clear representation of process which will aid the understanding of design procedures [10]. In response, there are various process modeling methods, such as flow charts and data-flow diagrams [34], project evaluation and review technique/critical path method (PERT/CPM) [33], structured analysis and design technique (SADT) [26], integration definition (IDEF) techniques (IDEF 0, IDEF3) [21,22], Petri nets [24], design structure matrix (DSM) [28], and so on. These models facilitate and/or enhance the process understandability, communication capability, and process feasibility.

Once process modeling is done, it is required to structure the design process to achieve a better understanding of the process. In design research, the most widely used approach is decomposition which divides a larger design activity into smaller, more tractable component design problems [10,17]. Decomposition is useful for allocating the work among multiple designers or design teams and thereby accelerating the design process [27]. Since decomposition is valuable for moderating the complexity in the design process, many previous studies have adopted the approach. To illustrate, Johnson and Benson [13] proposed an assumption that all of sub-processes are separable. Kusiak et al. [16–18], Steward [28], Rogers and Bloebaum [25], and Eppinger et al. [6–8] applied the decomposition to group activities that have a high degree of cohesion within groups and low coupling among groups. Chen and Lin [3] used the decomposition concept to divide large interdependent task groups into smaller and manageable ones. Partitioning approach also can be used to structure the design process. In short, partitioning is to resequence the design tasks, and to identify cyclical and acyclical tasks. The goal of

*Author to whom correspondence should be addressed.
E-mail: parkyt@cybernet.snu.ac.kr

1063-293X/07/02 0175–12 $10.00/0 DOI: 10.1177/1063293X07079321

partitioning is to maximize the availability of information required at each stage of the design process [10]. Aforementioned studies, which are based on decomposition, also used partitioning or similar approaches, directly or indirectly. Although the above approaches have contributed considerably to organizing the complex design process for the purpose of increasing the efficiency of the process, little attention has been given to the modular synthesis of the process.

Along this line, the current study presents a new approach to structuring the design process on the basis of modular synthesis. Chen and Liu [4] 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 Liu [4], the approach in this study 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 design process in a hierarchical architecture, the concept of a nested process suggested by Lawrence [20] and Kim et al. [14] 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 product design process in terms of a module are developed.

Structuring the process based on a module, called as process modularization, allows a process manager to decompose the complex design process into smaller and managerial activity groups. By doing so, the process manager can manage and analyze the process effectively. Moreover, this will be conducive to both reducing the product development cycle time and accelerating the launch of products to the marketplace. Indeed, that is what concurrent engineering (CE) aims to do. Among various modeling methods, the DSM is used to describe a design process. 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 product development process [1,2,23,28].

The remainder of this study 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 step of process modularization and explains algorithms related with modularizing process. Section 5 gives an illustrative example to demonstrate the operation involved in process modularization. Finally, in Section 6, some conclusions are drawn and the various implications and future research initiatives are discussed.

2. Design Structure Matrix (DSM)

The DSM was originally developed by Steward [28] for analyzing information flow and has been widely used in managing complex projects. Typically, the DSM has been applied to such various areas as building construction, semiconductor, automotive, photographic, aerospace, telecom, small-scale manufacturing, factory equipment, and electronics industries [1]. Similar to the incidence matrix in graph theory, the DSM is a square matrix with identical rows and columns. In the DSM, like a project task or a system component, an activity in the design process is positioned in a row and the corresponding column. The relationships between activities are represented by marking the cell formed by rows and the corresponding columns.

In the DSM, an off-diagonal mark represents the dependency among activities and a diagonal cell indicates the activity itself. In Figure 1, activity 1 has a relationship with activity 2. This means that activity 1 is processed prior to activity 2 or the output of activity 1 is used by activity 2. A design process is composed of three types of basic behavioral patterns, namely, serial (dependent or decoupled), parallel (independent or uncoupled), and iterative (interdependent or coupled) ones. Since activities 1, 2, and 3 are connected consecutively, they are in a serial building block and executed sequentially. In a serial building block, each of the activities has only one activity before and after the activity. Activities 4, 5, and 6 can proceed in parallel, which means they compose a parallel building block, and thus they do not depend on one another. Activities 7, 8, and 9 form a cycle. This is called an iterative building block and it appears when some activities are carried out repeatedly.

As well as facilitating 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

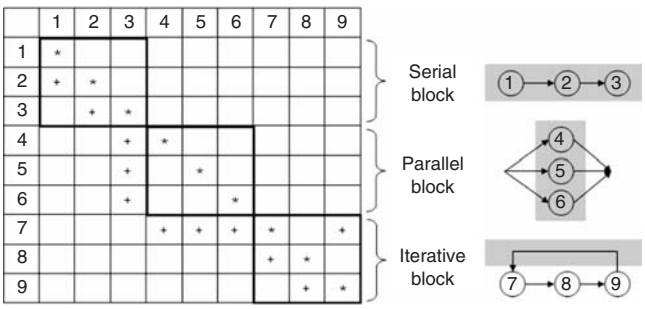

Figure 1. DSM representing three types of building blocks.

more compact representation [3]. Second, crucial information about a process can be obtained through the DSM operations. For example, mutually separable processes are found with cluster identification algorithm introduced by Kusiak and Wang [17], the order relationships among activities are discovered with topological sorting algorithm devised by Horowitz and Sahni [11] and triangularization algorithm developed by Kusiak and Wang [18], and cycles are detected with power of adjacency matrix method introduced by Deo [5] and path searching algorithm introduced by Sargent and Westerberg [29]. Third, the form of a matrix is amenable to program and calculate by using computers [22,30,31].

As mentioned thus far, the DSM has many advantages, vis-à-vis other methods, in process modeling and analysis and, therefore, the DSM was employed as a process representation method. In this study, all the algorithms in the proposed process structuring method, called as process modularization, are based on the DSM.

3. Concept of a Module

The concept of a module is the underlying basis of the authors' approach. A module, like a building block, is a group of activities which represents a logical unit of a process. However, they are different in that a module is further segmented than a building block, according to a process flow between activities. In addition, a module can be comprised of several activity blocks. The process in Figure 2 describes a simplified and summarized product design process which is designed for incorporating ergonomic characteristics into product design procedure. Therefore, the process takes only those activities that are related with the purpose into account, ignoring all other activities. Although the objective of the process is not to design specific products or parts, it is enough to demonstrate the concept of a module defined in this study. In Figure 2, activities in the process are grouped into four modules: selecting the target product, identifying ergonomic variables, identifying functional variable, and defining the direction of improvement. The first module, selecting the target product, is on the far left among modules. This module is a starting point of the whole process and is made of three activities: list products, prioritize products, and select the target product. Since these activities are interrelated with one another to select the target product which is supposed to be improved, they need to be considered together to progress or improve the process. After activities in this module are processed consecutively, the process meets new modules: identifying ergonomic variables and identifying functional variables. Although these two modules constitute a parallel block together in a view point of a building block, they have different process flows: one is to identify ergonomic variables and the other is to identify functional variables. Therefore, these activity groups are differentiated definitely and each of them is handled as an independent logical unit, which is referred to as a module. As such, a module is a group of activities divided by split or merge points in a process flow. In addition, it is a set of activities which share a common objective. Finally, the rest of activities in Figure 2 form another module and represent defining the direction of improvement. Figure 3 shows the DSM presenting the modules in Figure 2.

Like a nested process, a module can be further broken into sub-modules. To do this, two types of modules are defined: a primitive module and a nesting module. As shown in Figure 4(a), the whole process is decomposed into four main modules: A, B, C, and D at the first 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 second level, respectively. These sub-modules are modules at the relatively lower level and can also have sub-modules

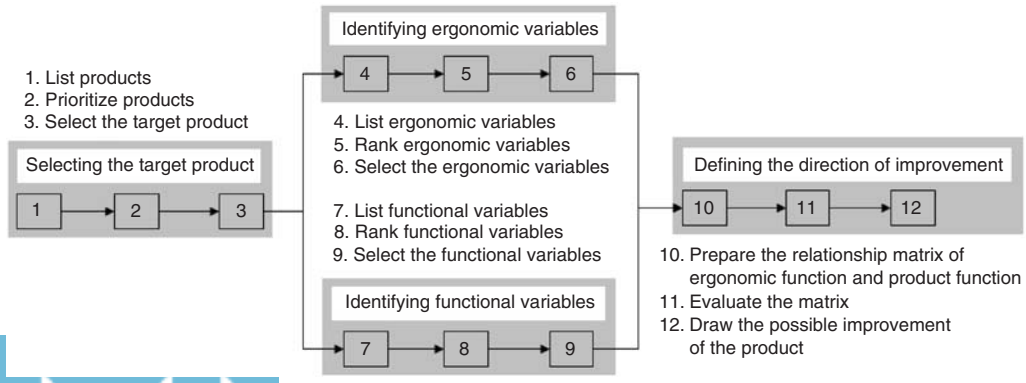

Figure 2. Example of modules in the design process.

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 4(b) shows the DSM representing the modules in Figure 5(a).

Structuring the product design process based on a module gives several advantages in managing and analyzing the process.

1. It is conducive to the management and analysis of a complex process in that it reveals a process structure according to a logical unit.
2. It allows a process manager to organize a 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.
3. It enables a process manager to think about process improvement with a broader perspective by offering both interrelated activities and a problematic activity.
4. 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.

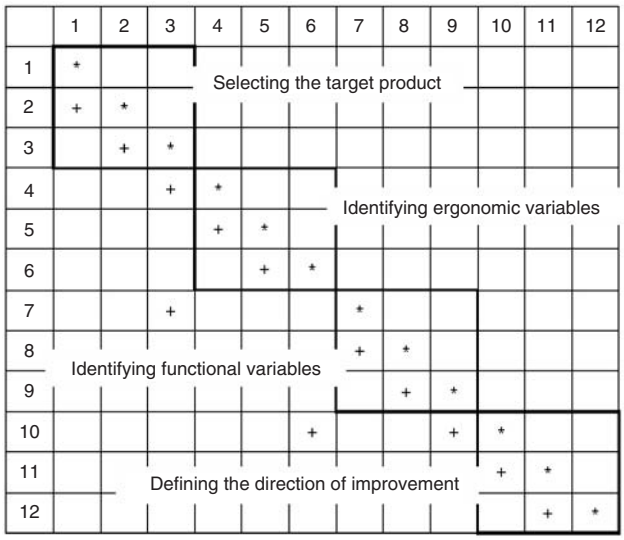

Figure 3. DSM representing modules in Figure 2.

4. Modularizing the Design Process

In order to modularize a design process, it is necessary to represent the activities required for designing a new product and their relationships with the DSM. Then, the modularizing process proceeds.

Figure 5 shows the overall process of modularizing the design process including major procedures represented by rounded rectangles colored gray. Before modularizing the design process, two preliminary steps are needed. One is to identify mutually separable processes in the whole design process. In a design process, some sub-processes are mutually exclusive. These sub-processes are groups of activities that have interactions within groups and have no interactions with other groups. In that case, identification of these sub-processes should be conducted prior to modularizing the process. In general, these sub-processes are called clusters in design research. Even though activity groups having minimal interactions are also clusters, this study limits the concept of clusters as sub-processes that have no interactions between them.

The other preliminary step is to detect iterative building blocks in the given design process. The purpose of this step is to check whether or not cycles exist in the design process. When an activity needs to go back to a certain previous activity, it forms cycles in the process. The information is registered and then, removed from

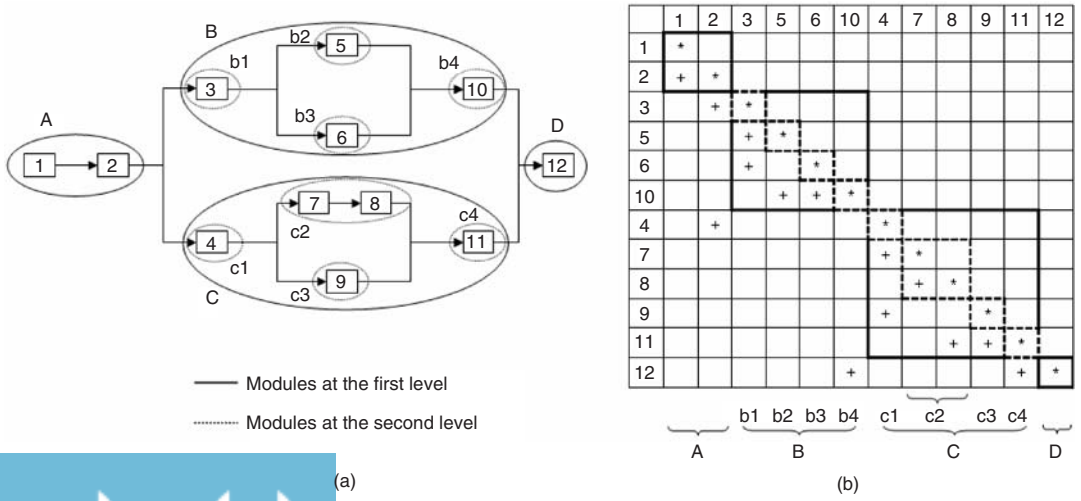

Figure 4. Example of modules for both at the first and second levels.

the process. This procedure is repeated until there is no more cycle. 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 to only keep the information. These two preliminary steps do not need to go into more detail since several algorithms for each of them are already developed. Decomposing mutually separable processes can be executed with cluster identification algorithm introduced by Kusiak and Wang [17]. Detecting iterative building blocks, or finding cycles, can be conducted with power of adjacency matrix method introduced by Deo [5] and path searching algorithm introduced by Sargent and Westerberg [29].

After two preliminary steps are performed, main steps for modularizing the design process proceeds. The first step is to order design activities based on the process flow. 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. However, it is required to draw activity order by a process viewpoint because a module is a group of activities decomposed according to process flows. Therefore, this study newly suggests the restricted topological sorting algorithm (RTS). Then, the detecting module step is carried out by the module finding (MF) algorithm developed in this study. More details on these steps 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 detecting module process is reapplied to these modules, called nesting modules, to find modules at a lower level. The other is representing cycle information to the DSM when the original process has a cycle relationship between activities.

4.1 Ordering Activities based on Process Flows

One of the major procedures for process modularization is to order activities according to process flows. Since the ordering in the DSM is temporal, the activities need to be reordered appropriately to precede the module detection process. Many studies (e.g., Horowitz and Sahni [11], Kingston [15], Lawler [19], and Steward [28]) presented numerous procedures and algorithms to order a set of activities. Regardless of whether the procedures are based on a matrix or a graph, the general procedures of these algorithms are as follows: One of the activities that have no preceding activity is selected arbitrary. The selected activity is deleted from the process and ordered. This process is repeated for the remaining activities until all of the activities in the process are ordered. Figure 6 shows this process with graphs instead of matrices for better understanding. Activity 1 without the preceding activity

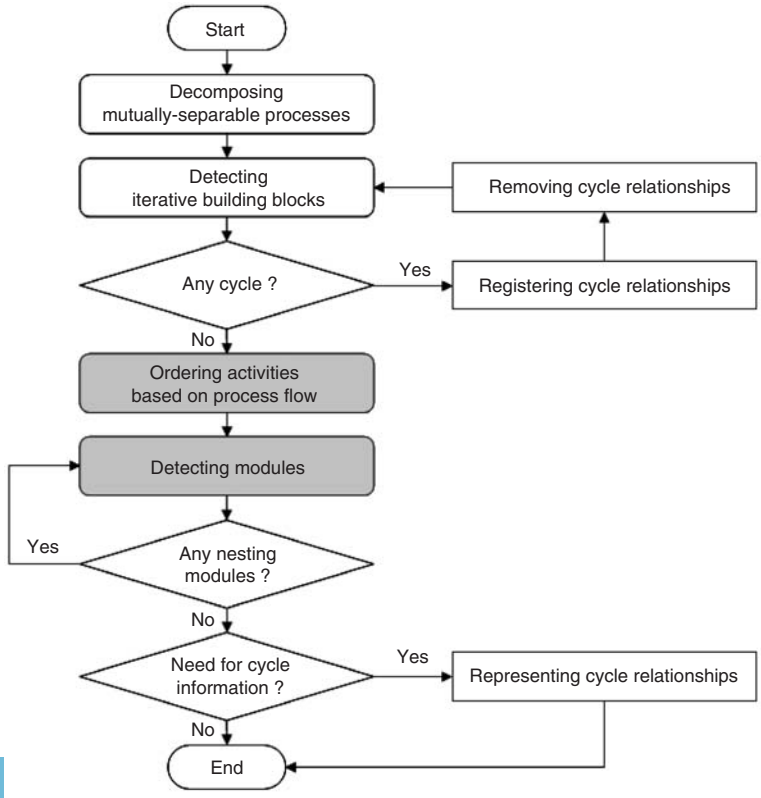

Figure 5. Overall process of modularizing the design process.

is selected and deleted from graph (a). In graph (b), activity 2 is chosen freely from two activities (2 and 5) and deleted. When the processes of (c), (d), and (e) are executed sequentially, the ordering of $\{1, 2, 5, 3, 4\}$ is derived. The above result is not unique because results can be different according to which activity is selected out of activities without a preceding activity.

The existing ordering procedures, however, is not appropriate for ordering and modularizing activities involved in the design process based on process flows because ordering should be achieved on the basis of the precedence relation between activities. Therefore, the new algorithm is developed to determine order based on predecessor requirements when activities without a preceding activity are selected. As a choice of an activity is restricted, it is named as the RTS algorithm. Thus, the existing topological sorting algorithm is revised into the RTS algorithm adding the restriction that the immediate successor of the preceding activity removed in the previous stage should be selected first. From the example in Figure 6, if activity 2 is selected as the subject of deletion and deleted from the graph, activity 3, which is the immediate successor of activity 2, must be selected as the subject of deletion. As a result, according to the RTS algorithm, possible orderings are

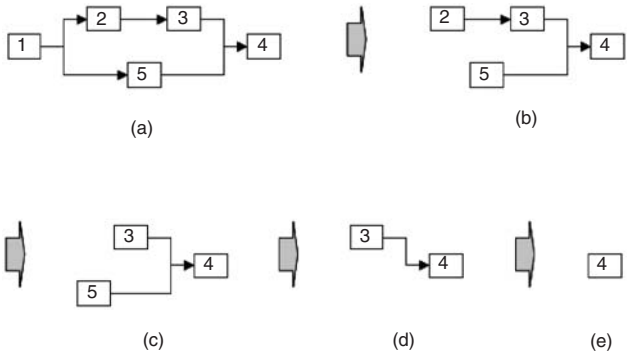

Figure 6. Example of general ordering procedures.

$\{1, 2, 3, 5, 4\}$ and $\{1, 5, 2, 3, 4\}$. In the same way, multiple activity orderings are possible when the RTS algorithm is applied to the design process. This depends on which activity is selected from activities without a preceding activity. However, this dose not become problematic since the drawn modules are the same whichever ordering outcome is used to detect modules. In the case of the process in Figure 6, two ordering results such as $\{1, 2, 3, 5, 4\}$ and $\{1, 5, 2, 3, 4\}$ can be derived from the RTS algorithm. Whichever ordering outcome may be used, if the orderings keep the precedence requirements between activities, the process will be grouped into four modules: $\{1\}$, $\{2, 3\}$, $\{5\}$, and $\{4\}$. Figure 7 shows the pseudo code for the RTS algorithm.

The algorithm first finds an activity without a preceding activity arbitrarily and puts this activity to the first column and row of the DSM. Among the remaining activities, the activity having no preceding activity is found and placed in the next position of the DSM. If the activity that is an immediate successor of the preceding activity is positioned on the DSM already, it should take priority over all other activities. This process continues repeatedly until there is no remaining activity. 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 Detecting Modules

The purpose of detecting modules is to structure the design process by the unit of a module. This step proceeds with the DSM which is reordered by the RTS algorithm. To detect modules, MF algorithm is developed and the pseudo code of it is presented in Figure 8.

First, the algorithm finds the first column which has two or more '+' elements. If no such column is detected, the design process is composed of only one serial block (i.e., single module). Otherwise, the elements, or

```
        RTS( )
1:      k ← Find_Only_One_Nonempty_Element( {K} )
2:      {K} = Get_Every_Element( {K} – SL ) // every element {K'} in {K} which are not in a search list.
3:      if ( k = 1 ) then goto 10
4:      else if ( k = 0 )
5:              do if ( Is_Empty(SL) ) then EXIT;
6:              k ← Pick_Last_Element_And_Delete_From( SL )
7:      else
8:              k ← Pick_Random_Element_From( {K} )
9:              Add_Elements( SL, {K} – k ) // add the other elements of {K} to the search list.
10:     Draw_Horizontal_Line_Through_Column( k )
11:     Draw_Vertical_Line_Through_Row( k )
12:     Cross_Out_Matrix( k )
13:     I ← I+1
14:     goto 1
```

Figure 7. Pseudo code for the RTS algorithm.

activities related to the found column, become the criteria to group the design activities into modules. Modules are identified by tracing activities that precede the criteria, respectively. All activities before the first criterion are grouped into one module. Activities positioned between the first criterion and the second criterion is grouped into another module with the first criterion. In this way, except the first criterion, activities positioned between two criteria comprise modules with the precedence criterion. The priority of the criteria is determined by the activity order in the DSM.

Next, the algorithm finds the first row which has two or more '+' elements from the last row in the DSM. This step is performed only when any column has two or more '+' elements in the previous stage because this means not only that the process has multiple modules but also that the DSM has the row having two or more '+' elements. Likewise the previous stage, drawn elements become the criteria to group the activities into modules. In this process, modules are discovered by tracing activities that follow the criteria. All activities after the first criterion are grouped into one module. The other modules are derived in the same way as the previous stage. The priority of the criteria is determined by the reverse activity order in the DSM.

Through the above two stages, the algorithm detects modules at the uppermost level. Among the found modules, if some modules have the column involving two or more '+' elements, nesting modules must be included in the modules. Therefore, the process of finding modules should be reapplied to each of the nesting modules to look for modules at the next lower level. This process ends when no more nesting modules in the DSM exist. The time complexity of the algorithm is $O(n^2)$, 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. In this table, the branch represents the possible forks at a given activity, and the level means the depth of a module. It appears from the results that the computation time is affected by not the number of branches but both the number of

```
        MD( DSMAfterRTS, NumOfEl )
1:      column ← getStartingCol( DSMAfterRTS )
        // Finding the first column which has two or more '+' elements.
2:      if ( column exists ) then
3:              row ← getStartingRow( DSMAfterRTS )
                // Finding the last row which has two or more '+' elements.
4:              while ( retrieving column in DSMAfterRTS )
5:                      if ( element = '+') then
6:                              {D} ← element
7:              while ( retrieving row from right to left in DSMAfterRTS )
8:                      if ( element = '+' & element is not in {D} )
9:                              {D} ← element
10:     Visualize_Boundary_of_Module( every element described in D )
12:             if ( DSMAfterRTS has n nesting modules ) then
13:                     MD ( eachNestingModule, revised NumOfEl )
```

Figure 8. Pseudo code for the MF algorithm.

Table 1. Performance results of both the RTS and the MF algorithms.

Number of activities	Maximum number of branches	Maximum number of levels	Computation time (ms)[a]	
			RTS algorithm	MF algorithm
25	2	2	1312	234
		4	1344	618
	4	2	1328	297
		4	1375	641
50	2	2	7672	828
		4	7766	2468
	4	2	7681	897
		4	7802	2653

[a]Computation time is the sum of 1000 iteration times.

activities and the number of levels in a process. Increasing the number of activities and levels causes an increase of computation time not only in the RTS algorithm but also in the MF algorithm.

5. An Illustrative Example

To illustrate the newly suggested approach in this study, a hypothetical design scenario composed of 14 activities has been made. The activities and their relationships are described through the DSM representation, shown in Figure 9(a). Two preliminary steps are omitted in this illustrative example because of the many procedures or algorithms for them.

5.1 Ordering Activities based on Process Flows

The first major step of modularizing the design process is to derive the ordering of activities grounded on process flows by using the RTS algorithm. Table 2 shows the procedures of the RTS algorithm applied to the hypothetical process and Figure 9(b) presents the final result. In Table 2, k is the selected activity to draw 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 9(a) is the original DSM and the DSM in Figure (b) is the outcome of the rearrangement of activities with the RTS algorithm. The DSM (a) has a cycle relationship (colored gray) between activities

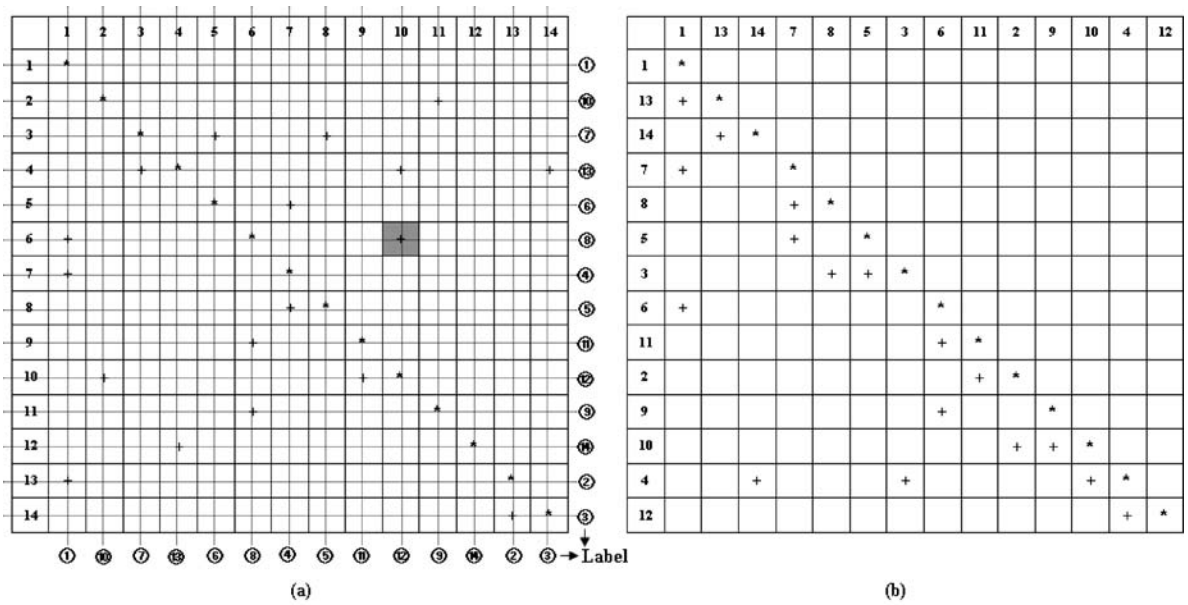

Figure 9. Result of the RTS algorithm applied to the virtual process.

Table 2. Procedures of the RTS algorithm applied to the virtual process.

I	Initial search list	Initial K	New $K=K'$	k	New search list	Label
1	{ }	(1)	(1)	1	{ }	1
2	{ }	(6, 7, 13)	(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*!!		

6 and 10. However, this information does not appear in the DSM (b), because the cycle relationship is registered and removed from the DSM before the RTS algorithm is applied.

The order of activities in the DSM of Figure 9(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. As previously mentioned, however, an arbitrary selection from these activities does not become problematic because the drawn modules are the same. In the DSM of Figure 9(a), 20 activity orderings are possible. Of these, one is produced in this case, which is represented in the DSM of Figure 9(b).

5.2 Detecting Modules

Based on the ordering resulted from the RTS algorithm, the module detecting process is executed. Figure 10 shows the process according to the stages when the MF algorithm is applied to the DSM in Figure 9(b).

Figure 10(a) presents the first stage of detecting module. The process in Figure 10(a) has multiple modules since activity 1 in the first column has three '+' elements corresponding to the rows of activities 13, 7, and 6. Therefore, these three activities ($C_{c1} = 13$, $C_{c2} = 7$, and $C_{c3} = 6$) are used as the criteria to group the process into modules. By tracing all activities before the activity 13 and grouping them, one of the modules in the process, {1}, is elicited. Activities between activities 13 and 7 form another module with activity 13. In addition, activities 7, 8, 5, and 3 are grouped as a module. Through this stage, three modules are derived: {1}, {13, 14}, and {7, 8, 5, 3}. The second stage for detecting module starts with finding the row which has two or more '+' elements. According to Figure 10(b), activity 4 in the second row in the view point of the last row has three '+' elements. These three elements ($C_{r1} = 10$, $C_{r2} = 3$, and $C_{r3} = 14$) are also used as criteria of detecting modules. All activities after activity 10 in the DSM are grouped into a module, or {4, 12}. The second module is {6, 11, 2, 9, 10} which is composed of activities between criteria $C_{r1} = 10$ and $C_{r2} = 3$ with activity 10. Through the second stage, the algorithm draws three modules: {4, 12}, {6, 11, 2, 9, 10}, and {7, 8, 5, 3}. As shown in Figure 10(a) and (b), the module {7, 8, 5, 3} is duplicated because it is derived from both stages. Eventually, five modules in the process are determined at the uppermost (first) level.

Figure 11 shows modules at the first level. Of these, some modules have the column including two or more '+' elements. Since these modules are nesting modules, the whole module detecting process should be reapplied to each of them to find modules at the next lower (second) level. The module finding process ends when no more nesting modules exist in the design process. Figure 12 shows the final results after the MF algorithm was applied to the hypothetical design process. The bold lines represent the modules at the first level and the dotted lines represent the modules at the second level. Consequently, the design process is composed of five modules at the first level: three primitive modules and

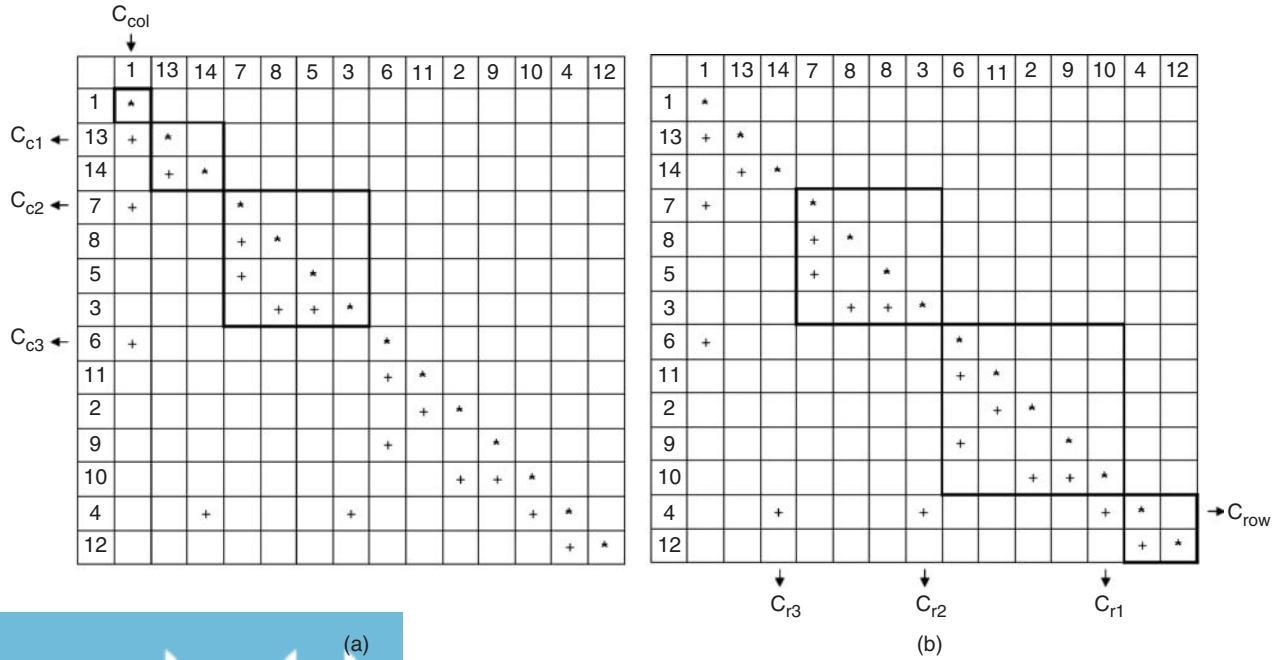

Figure 10. Module detection process applied to the DSM in Figure 9(b).

two nesting modules. Two nesting modules have four sub-modules which are {7}, {8}, {5} and, {3} in the nesting module N1, and {6}, {11, 2}, {9}, and {10} in the nesting module N2, respectively. The gray colored cell indicates the cycle relationship which was in the original process. The DSM in Figure 12(a) represents the modules in one dimension while the DSMs in Figure 12(b) show the modules in a hierarchical form.

6. Conclusions

Process modeling, representing activities and their relationships, is useful for understanding and analyzing a design process. However, it is very difficult to fully understand the design process as the number of activities in the process increases. That is why structuring the process has become more essential. Although existing decomposition and partitioning approaches have been contributed to organizing the complex design process to increase the efficiency of the process, little attention has been given to the modular synthesis of the process.

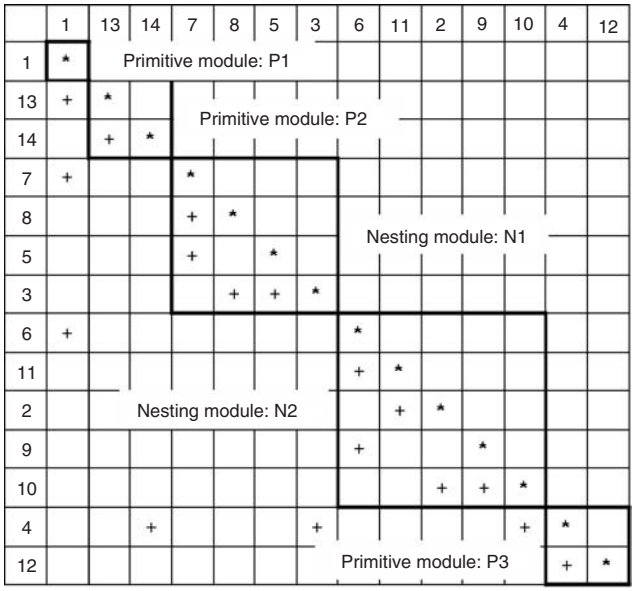

Figure 11. Results of modularizing the design process at the first level.

This study presents a new approach to structuring the design 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 module is defined as a group of activities which are divided by split or merge points in a process flow. In order to structure the design 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 design 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 design process in an efficient manner. However, it must be noticed that there is some room to be considered when modularizing the

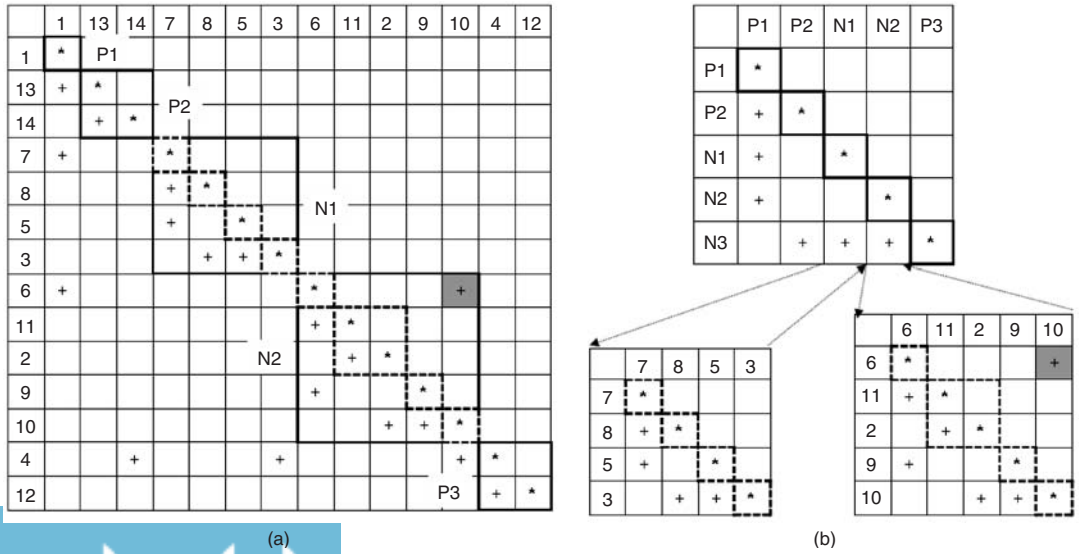

Figure 12. Final results of the modularized design process.

design 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 also should be considered as a whole. Second, related to the first consideration, 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 design process depend on people who have specific knowledge, experience, or skills. Nevertheless, considering that a process view is an ever-increasing importance in product development with the advent of CE, the suggested approach is implicative in that the emphasis is on process flows rather than on functions.

To illustrate the proposed approach, the hypothetical product 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 product design 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 different degrees of complexity in activities and their relationships. This is a further research issue to be considered.

References

1. Browning, T. (2001). Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions, *IEEE Transactions on Engineering Management*, **48**(3): 292–306.
2. Browning, T. (2002). Process Integration using the Design Structure Matrix, *System Engineering*, **5**(3): 180–193.
3. Chen, S. and Lin, L. (2002). A Project Task Coordination Model for Team Organization in Concurrent Engineering, *Concurrent Engineering: Research and Applications*, **10**(3): 187–202.
4. Chen, D. and Liu, C. (2001). Frameworks for the Modular Synthesis of Mechanisms, *International Conference on Engineering Design*, ICED 01, Glasgow, August 21–23.
5. Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*, NJ: Prentice-Hall.
6. Eppinger, S., Whitney, D. and Gebala, D. (1992). Organizing the Tasks in Complex Design Projects: Development of Tools to Represent Design Procedures, *Proceeding NSF Design and Manufacturing System Conference*, Atlanta, GA.
7. Eppinger, S., Whitney, D., Smith, R. and Gebala, D. (1990). Organizing the Tasks in Complex Design projects, In: *Proceedings 2nd ASME International Conference on Design Theory and Methodology*, Chicago, IL, pp. 39–46.
8. Eppinger, S., Whitney, D., Smith, R. and Gebala, D. (1994). A Model-based Method for Organizing Tasks in Product Development, *Research in Engineering Design*, **6**(1): 1–13.
9. Fernandez, C. (1998). Integration Analysis of Product Architecture to Support Effective Team Co-location, SM Thesis, Massachusetts Institute of Technology, Cambridge, MA.
10. Gebala, D. and Eppinger, S. (1991). Methods for Analyzing Design Procedures, In: *Proceeding of the ASME 3rd International Conference on Design Theory and Methodology*, **31**: 227–233.
11. Horowitz, E. and Sahni, S. (1982). *Fundamentals of Data Structures*, Rockville: Computer Science Press.
12. Hsiao, S. and Chou, J. (2004). A Creativity-based Design Process for Innovative Product Design, *International Journal of Industrial Ergonomics*, **34**(5): 421–443.
13. Johnson, R. and Benson, R. (1987). A Basic Two-level Monotonicity-Based Decomposition Method, In: *Proceedings of the ASME Design Automation Conference*, Boston, MA, pp. 41–48.
14. Kim, Y., Kang, S., Kim, D., Bae, J. and Ju, K. (2000). WW-FLOW: Web-Based Workflow Management with Runtime Encapsulation, *IEEE Internet Computing*, **4**(3): 55–64.
15. Kingston, J. (1990). *Algorithms and Data Structure*, New York: Addison-Wesley.
16. Kusiak, A., Larson, T. and Wang, J. (1994). Reengineering of Design and Manufacturing Processes, *Computers and Industrial Engineering*, **26**(3): 521–536.
17. Kusiak, A. and Wang, J. (1993). Decomposition of the Design Process, *Journal of Mechanical Design*, **115**(4): 687–695.
18. Kusiak, A. and Wang, J. (1993). Efficient Organizing of Design Activities, *International Journal of Production Research*, **31**(31): 753–769.
19. Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids*, New York: Holt, Rinehart & Winston.
20. Lawrence, P. (1997). *Workflow Handbook 1997*, Workflow Management Coalition, Chichester: Wiley & Sons Ltd.
21. Mayer, R., Benjamin, P., Caraway, B. and Painter, M. (1995). *A Framework and a Suite of Method for Business Process Reengineering in Business Process Change: Concepts, Methods and Technologies*, PA: Idea Group Publishing.
22. Mayer, R., Menzel, C., Painter, M., deWitte, P., Blinn, T. and Perakath, B. (1995). *Information Integration for Concurrent Engineering (IICE): IDEF3 Process Description Capture Method Report*, Texas: Knowledge Based Systems.
23. Park, H. and Cutkosky, M. (1999). Framework for Modeling Dependencies in Collaborative Engineering Process, *Research in Engineering Design*, **11**(2): 84–102.
24. Reisig, W. (1985). *Petri Nets: An Introduction*, Berlin Heidelberg: Springer-Verlag.
25. Rogers, J. and Bloebaum, C. (1994). Ordering Design Tasks Based on Coupling Strengths, In: *Proceedings of 5th AIAA/NASA/USAF/ISSMO Symposium on*

Multidisciplinary Analysis and Optimization, Panama City, FL, pp. 708–717.
26. Ross, D. (1977). Structured Analysis (SA): A Language for Communicating Ideas, *IEEE Transactions on Software Engineering*, **3**(1): 16–34.
27. Smith, R. and Morrow, J. (1999). Product Development Process Modeling, *Design Studies*, **20**(3): 237–261.
28. Steward, D. (1981). The Design Structure System: A Method for Managing the Design of Complex System, *IEEE Transactions on Engineering Management*, **28**(3): 71–74.
29. Sergeant, R. and Westerberg, A. (1964). Speed-up in Chemical Engineering Design, *Transactions of the Institute of Chemical Engineers*, **42**(a): 190–197.
30. Tang, D., Zheng, L., Li, Z., Li, D. and Zhang, S. (2000). Re-engineering of the Design Process for Concurrent Engineering, *Computers and Industrial Engineering*, **38**(4): 479–491.
31. Yassine, A. and Braha, D. (2002). Complex Concurrent Engineering and the Design Structure Matrix Method, *Concurrent Engineering: Research and Applications*, **11**(3): 165–176.
32. Yassine, A., Falkenbug, D. and Chelst, K. (1999). Engineering Design Management: An Information Structure Approach, *International Journal of Production Research*, **37**(10): 2957–2975.
33. Wiest, J. and Levy, F. (1977). *A Management Guide to PERT/CPM*, Prentice-Hall, Englewood Cliffs: NJ.
34. Yourdon, E. (1989). *Modern Structured Analysis*, NJ: Prentice-Hall.

Hyeonju Seol

Hyeonju Seol is an assistant professor in the Department of Industrial Engineering at Korean Air Force Academy (KAFA). He holds a BS, MS, and PhD in industrial engineering from Seoul National University (SNU). In addition, he received a BS from KAFA. He is interested in the areas of engineering design, new product development, process management, and technology management.

Chulhyun Kim

Chulhyun Kim is a PhD candidate in the Department of Industrial Engineering at SNU. He holds a BS and MS in industrial engineering, both from SNU. His research interests are in new product development, new service creation, and technology management.

Changyong Lee

Changyong Lee is a PhD candidate in the Department of Industrial Engineering at SNU. He holds a BS in computer science from Korea Advanced Institute of Science and Technology (KAIST). His research interests are in the areas of service process management, process improvement of software engineering, and network analysis.

Yongtae Park

Yongtae Park is a professor in the Department of Industrial Engineering at SNU. He holds a BS in industrial engineering from SNU, and an MS and PhD in operations management, both from the University of Wisconsin-Madison. His research topics cover a wide variety of areas including process management, technological innovation management, knowledge management, and information management.