

DETC2002/DTM-34031

PARTITIONING TASKS TO PRODUCT DEVELOPMENT TEAMS

Dan Braha
Center for Innovation in Product Development
Massachusetts Institute of Technology
30 Memorial Drive
Cambridge, MA 02139
and
Ben-Gurion University, Israel

ABSTRACT

The principle of partitioning tasks among product development teams so as to minimize the cost of interactions across design teams is an important characteristic of complex engineered systems. Although there is growing literature that deals with the proper organization of product development tasks, little attention is given to rigorous modeling of the phenomena. To fill the void, we present a mathematical formulation for the problem. Two main issues are addressed by the model: 1) how to specify task dependencies, and 2) how to optimally partition the tasks among a number of teams. Characteristics of the problem are analyzed, and an efficient solution procedure is proposed. The developed model and solution technique can be applied to various scales of the product design and development process, and may open a variety of interesting questions.

Keywords: Partitioning, Clustering, Decomposition, Nearly Decomposable Systems, Product Development

1 INTRODUCTION

Product development processes are generally divided up into tasks and subtasks. These segments may then be assigned ("partitioned") to multifunctional design teams [4, 9, 10, 13, 16, 17]. A multifunctional team consists of a group of individuals from different disciplines working closely together toward the development of the product. Problem solving tends to be more rapid when individuals have face-to-face contact [3]. By working together, team members enhance their awareness and understanding of the design problems worked out by other team members. As a consequence, a sense of teamwork arises, which helps advance the team's objectives.

Design teams can achieve larger gains from specialization when they have fewer tasks to design. Thus, to realize

specialization economies, the firm divides the work of development by partitioning tasks into several teams. For instance, there are hundreds of teams in the firms that develop various parts of the Saturn car and the Toyota Crown [4]. Proper partitioning of design development tasks is concerned with assigning into the same team tasks that are anticipated to require high problem-solving interaction, while assigning to different ("independent") teams tasks that require low problem-solving interaction. Minimizing the need for problem-solving across teams can have important consequences on product development performance, particularly since much of the activity of innovative product development projects involves problem solving and the creation of new knowledge [17].

To illustrate the partitioning problem consider the development of a certain car model, which includes four tasks: developing the engine, instrumentation², dashboard, and power train system. There are several ways that these four tasks can be assigned to two design teams. Three partitions are considered (see Figure 1): 1) assigning the development tasks of the engine and instrumentation to one team, and assigning the development tasks of the dashboard and power train system to a second team; 2) assigning the development tasks of the engine and power train system to one team, and assigning the development tasks of the instrumentation and dashboard to a second team; or 3) assigning the development tasks of the engine and dashboard to one team, and assigning the development tasks of the instrumentation and power train system to a second team. The second partitioning appears more efficient than the other two in terms of the needed cross-boundary problem-solving that is required to carry out the development project. This is due to the strong problem-solving

² Automotive instrumentation performs the crucial role of monitoring vehicle operation and supplying information to both the driver and the vehicle subsystems.

interdependence between 'developing the engine' and 'developing the power train system' as well as the between 'developing the dashboard' and 'developing the instrumentation.' The first and third partition options will likely require more communication and coordination between the two teams, which may render the entire development process less effective and efficient. Thus, by properly managing partitioning an increase in the project's efficiency and effectiveness may be achieved.

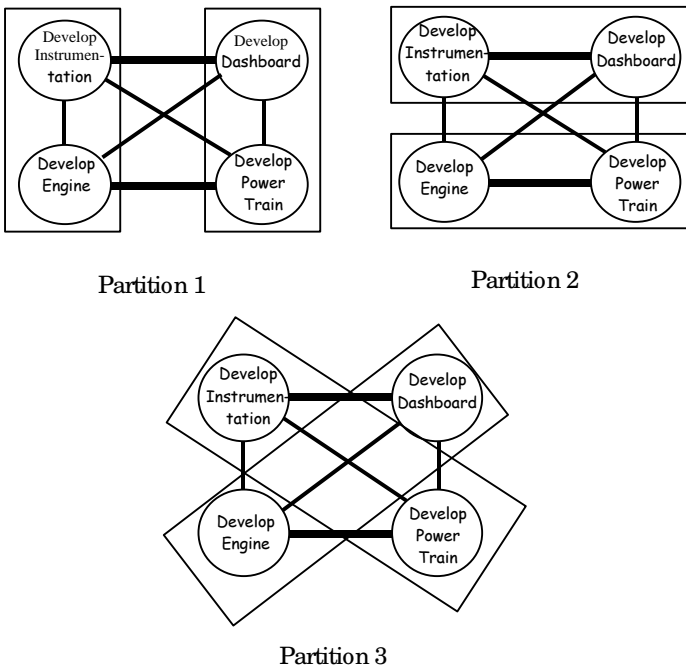


Figure 1: Partitioning the work of development of a car model

There are several advantages that are associated with proper partitioning. First, partitioning all the decisions that need to be made into nearly independent subsystems is an inevitable consequence of the limitations on the cognitive and information-processing capabilities of the designer's decision making (referred to as 'bounded rationality' in [13]). Alexander proposes that the design process can be greatly facilitated if the problem can be break apart to clusters in such a way that its clusters of requirements are as richly internally connected as possible [2]. Similarly, von Hippel [17] claims that the cost of changes in component tasks will be less, if tasks are arranged so as to reduce the problem-solving interdependence among them. Clark [8] and Henderson [12] claim that sub-optimal partition of the product development tasks (e.g., by depending on historically derived partitions) may result in ineffectively responding to emerging innovation problems. In the auto industry, Clark [8] observes that outsourcing certain detailed design tasks to external firms, who both design and manufacture the components, could save about 8 months of development time. Baldwin and Clark [4] show how modular

product development can speed the rate of technological change and increase product variety. von Hippel observes that problem-solving that extends beyond a single individual involves communication and coordination among problem-solvers [17]. He identifies a task boundary between problem-solvers to be often associated with physical and organizational barriers. Such barriers can add to the cost of problem-solvers' efforts to achieve cross-boundary communication and coordination, and thus reduce problem-solving efficiency [17]. He concludes that assigning tasks that have a lot of interconnected problem-solving to the same design team can help reduce the cost of communication and coordination across design teams.

Although there is a burgeoning literature that deals with the proper organization of product development tasks with respect to the requirements of problem solving [4, 8, 9, 10, 13, 16, 17], little attention is given to the rigorous formulation of the problem. In this paper, several techniques for specifying task dependencies are addressed, and a new integer linear programming formulation for task partitioning is suggested. Several properties based on the mathematical model are derived, and an effective procedure for solving the problem is presented. An example is provided to illustrate the problem formulation and solution procedure. We conclude by presenting several future implications.

2 THE TASK PARTITIONING PROBLEM

In this section, we embark on a new formulation for the task partitioning problem. The new formulation of task partitioning is divided into three steps: defining task dependencies, determining partitioning costs, and devising formulation for optimal partitioning of tasks among teams. These steps are described in more detail below. Step One of the new formulation is related to setting up a framework for specifying tasks and their dependencies.

2.1 A Framework for Specifying Task Dependencies

Before presenting the proposed framework for specifying task dependencies, we briefly describe Steward's Design Structure Matrix (DSM), which is a useful tool for representing and analyzing task dependencies of a design project. The DSM is a binary square matrix, where a project task is assigned to a row and a corresponding column. A row, associated with a given task, is marked with 1s if the corresponding tasks (in columns) are dependent on it (e.g., by a precedence relationship [10]). Otherwise, the value of the matrix elements is 0 -- excluding the elements on the diagonal. Off-diagonal marks in a single row of the DSM represent all of the tasks whose output is required to perform the task corresponding to the row; while reading down a column shows all of the tasks that receive information from the task corresponding to the column. The DSM representation has been successfully used in concurrent engineering management and implementation by several researchers (e.g., [10, 18, 19]). However, this technique has

several major drawbacks: 1) tasks are considered monolithic elements with no explicit specification or characterization. Moreover, the DSM describes the project in the physical domain, and thus results in a one dimensional representation of task dependencies (expressed by a binary task/task matrix); 2) the extraction of task dependencies is based on interviews with team's members at the detailed design stage. As a result, partitioning the DSM merely reflects the way the tasks are actually distributed among the multifunctional design teams [15]; 3) although the classical DSM representation has been extended by including measures of the degree of dependence between tasks [10], these numerical coupling values have not been explicitly incorporated in an optimization framework that aims at minimizing the between-task communication and coordination costs.

To address the above concerns, it is suggested to incorporate a *multidimensional* approach where 'design tasks' are defined in terms of a set of 'attributes' that need to be 'processed' or 'attained' during the development of the related task. Koopman [23] has suggested to characterize a design as having attributes that fall into the three attribute categories called structures, behaviors (or functions), and goals. This classification is supported by several structured design methods; e.g., conceptual design with a 'function-structure' in Pahl and Beitz [24], or systematic concept generation for technical products in Hubka and Eder [25]. Structural attributes are physical components, geometric information, logical objects, etc. that are generally related to the various aspects of the design implementation. Examples include springs, cylinders, materials, geometric shapes, layout, process and physical parameters, databases, or electric fields. Behavioral attributes are control processes; actions; forces; storing, delivering or converting energy; flows of material, energy or forces; and signals or control between subsystems, etc. They are usually concerned with the design's functions and processes, behavior over time, state and modes, as well as the conditions and events that cause modes to change [7]. The behavioral attributes also deal with concurrency, synchronization, and causality [7]. Examples include the ability to resist gravity loads, convert electrical energy into translational energy, deliver powder, control temperature, run software, etc. Goal attributes are aggregate design properties [23] or 'holistic' requirements [20] that emerge in a complex way based on the components of a product, and satisfy the intended needs of the design. Examples include system level performance objectives, costs, aesthetics and ergonomics, size, weight, mass, or external constraints. We assume that 'attributes' are characterized according to a finite number of types. Given a particular task, its set of underlying attributes can be classified as those attributes that are only associated with the task, and those attributes that are shared with other tasks in the product development process.

Tasks and their attributes can be derived using several techniques. For example, by utilizing the DSM methodology, one can create a 'task network' related to the DSM. The task

network is a directed graph that represents the input/output relationships among the various tasks in the system as shown in Figure 2. Each node in the task network represents a task; each arc entering a node represents an input parameter needed for carrying out the task; and each outgoing arc represents a parameter that is generated by the task.

Design Structure Matrix:

		Tasks				
		1	2	3	4	5
Tasks	1	.	0	0	0	0
	2	X	.	0	0	0
	3	X	0	.	0	X
	4	0	0	X	.	0
	5	0	X	0	0	.

Task Network:

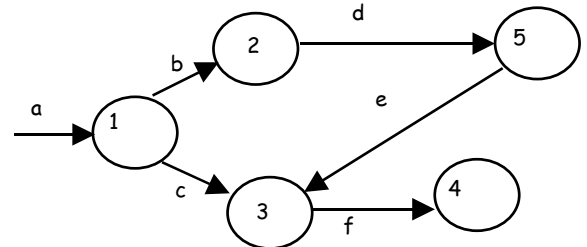


Figure 2: A Design Structure Matrix (DSM) and the related task network

Tasks and attributes can also be derived by analyzing the underlying *schematic* of the product. A schematic is a diagram, obtained at the end of the concept development phase, representing the team's understanding of the elements of the product [16]. Some of the elements in the schematic are described as physical concepts (e.g., a motor) while others are described by their function (e.g., absorb shocks). Elements in a schematic are connected by input flow lines and output flow lines, which represent the desired interactions that are fundamental to the product's operation. These lines indicate the flow of material, energy or forces, and signals or control [16]. Each element in a schematic is associated with flow lines entering the element and flow lines outgoing the element. Thus, the representation of a schematic is similar to the task network description (related to the DSM) discussed above. Specifically,

'elements' and 'flow lines' correspond to 'tasks' and 'attributes', respectively. In addition to 'flow lines', other attributes can be identified after establishing the product architecture³. For instance, undesired interface attributes that arise because of the particular physical implementation of functional elements or because of the geometric arrangement of the elements [16]. Examples of undesired interface attributes include relative motion between two physical elements, vibration, drifted radiation, poor geometric alignment between elements, undesired heat transfer, losses in signal, material or force transfer, and thermal expansion. Such undesired attributes can have a harmful effect and should be reduced. Other attributes that may be identified are attributes that are associated with 'global' or 'holistic' requirements that a task may fulfill; e.g., total product weight, size, efficiency, or reliability [20].

Specifying task dependencies based on the underlying schematic of the product may lead to ineffective partitioning of design development tasks. This is related to the fact that, in general, the specification of tasks is not necessarily related to the underlying product that is being designed [17]. For example, a product may be composed of several major physical components, but the project tasks leading to the development of this product may be partitioned according to non-physical attributes. For instance, when physical components have multiple functional attributes, partitioning may be performed according to the functional attributes.

In this paper, we suggest to utilize the 'design matrix' representation of Axiomatic Design [21] or Quality Function Deployment (QFD, [11]) as a means for specifying tasks and their associated attributes (we use the term 'design matrix' specification technique). The design matrix represents the mapping between functional requirements in the functional domain and design parameters in the physical domain. Using our previous terms, the design parameters correspond to tasks while the functional requirements to attributes. The structure of the design matrix can be determined early in the process, during the conceptual design stage. The detailed design of the various tasks (design parameters) is established at the detailed design stage by the product development teams. Each task (design parameter) in the design matrix affects a set of attributes (functional requirements). The interdependence between two tasks is captured by the *shared* attributes (functional requirements) that are affected by *both* tasks.

The 'design matrix' specification technique has several merits. First, it provides the means for identifying the interdependence among tasks based on the underlying physical principles governing the functioning of the product. By deducing task dependencies based on physical principles, a more objective (and unbiased) specification technique is obtained. This also avoids the need to conduct long interviews with product development members in order to elicit the task

dependencies, which carries the risk of obtaining historically derived partitions that may not address innovation or novel design problems. Second, the 'design matrix' specification technique can be utilized to specify 'tasks' and 'attributes' at the early stages of the product development process; thus, enabling more effective partition decisions at the detailed design phase.

The definition of a task in terms of its associated attributes enables the dependency between a pair of tasks by the attributes that are shared by both tasks. Consider the two tasks sharing a common attribute as illustrated in Figure 3. If the two tasks are assigned to two different teams, and they are to effectively carry out the tasks, then the two teams will be involved in problem-solving communication and coordination with respect to the shared attribute(s). Coordinating attributes across teams generally demands interaction time and may involve the utilization of communication technology (e.g., distributed computer assisted design). Thus, the coordination of attributes across teams carries a certain cost, and may have a negative effect on the product development time. Intuitively, if tasks that exhibit 'strong' dependency are assigned to the same team, then the partitioning may be more effective. However, the proper decision about how tasks are assigned to design teams is affected by several technical and non-technical factors.

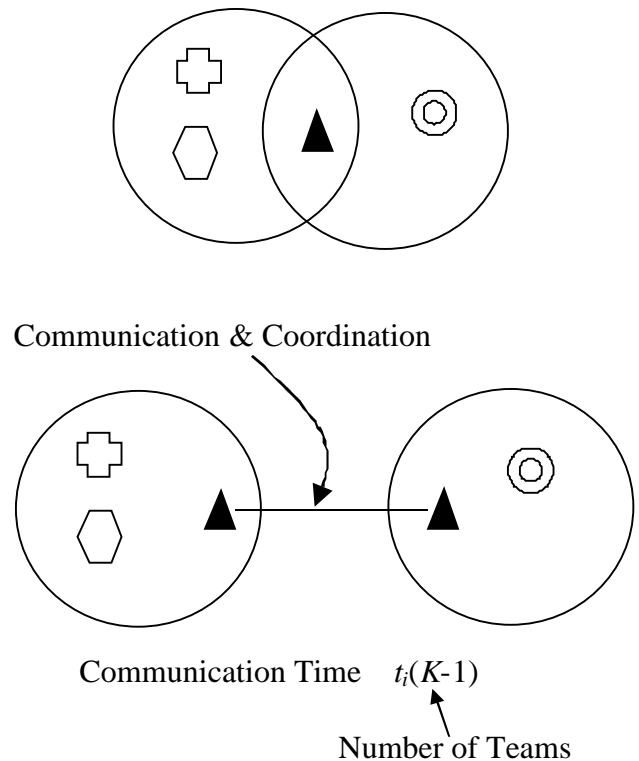


Figure 3: Assigning two tasks that share a similar attribute to different teams requires additional communication and coordination activities

³ The product architecture is the scheme by which the elements of the product are arranged into physical chunks, and by which the chunks interact and their approximate geometric layout is described [16].

2.2 Formulation of Task Partitioning

In this following, we discuss some of the factors affecting task partitioning with the aim of addressing the question of how design development tasks can be distributed among a number of design teams so as to minimize the need for problem-solving across teams. The task partitioning problem can be described as follows. We are given a set of tasks, each of which is associated with a set of attributes. Since the team's information processing capacity is limited in terms of the total number of attributes that it can process (often referred to as 'bounded rationality', [14]), the tasks are partitioned into several design teams as shown in Figure 4. This assumption is driven by the hypothesis that design teams can achieve larger gains from specialization when they have fewer attributes to attain. By introducing a limit on the total number of attributes that can be assigned to a team, we capture the idea that assigning a number of attributes above the threshold *considerably* reduces the problem solving performance expected from the design team. After assigning the tasks to several teams it is likely that an attribute (or a task) will be assigned to more than a single team as shown in Figure 4.

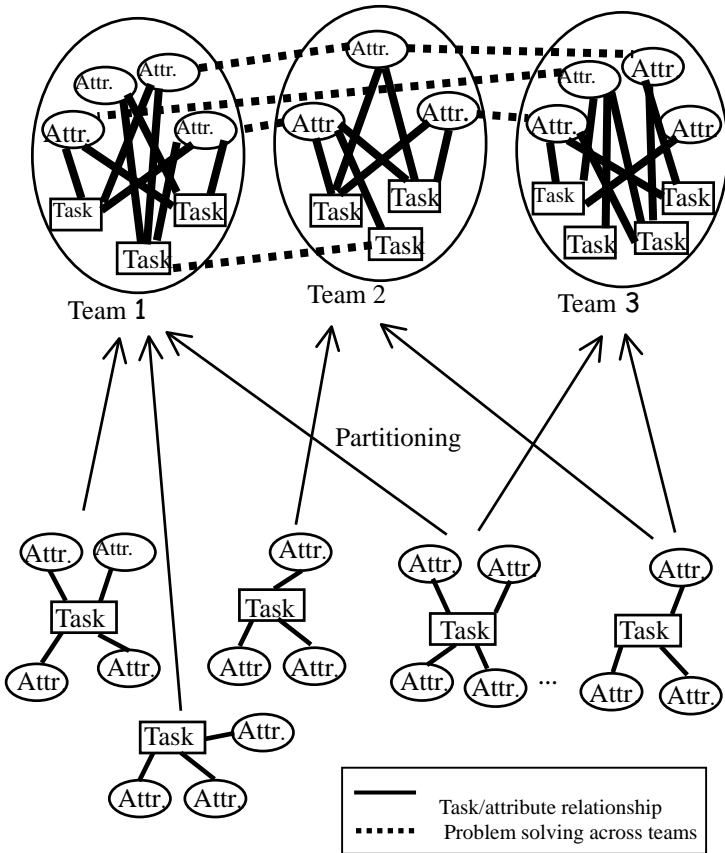


Figure 4: Partitioning tasks to product development teams

Step Two of the new task partitioning problem formulation

determines the various costs related to the problem. We consider two costs associated with a particular partition. The first cost is incurred when the same attribute is included in more than a single team. As explained above, the coordination of an attribute across teams incurs a cost. For illustration, we use a linear communication cost per attribute. That is, if an attribute i is included in K teams, then the communication cost that is incurred is $t_i \times (K - 1)$, where the communication cost coefficient t_i is a constant associated with coordinating attribute i . By associating different communication cost coefficients with attributes, we capture the idea that coordinating some attributes is more difficult than others. For example, in designing a certain building, the attribute "resist lateral wind loads" may be more critical than the attribute "resist gravity loads." In general, critical attributes require more careful coordination than less critical attributes. Thus, coordinating (across teams) the more critical attribute tends to have a higher cost than coordinating the less critical attribute. In another example, the attribute "personal safety" may be more critical than "screen parked cars." As a result, it is desirable to minimize the assignment of an attribute with high communication costs to too many design teams.

The second cost is related to the assignment of a task to several teams. In this case, each team that includes the task will have to expend an initial preparation cost. This cost may be associated, for example, with learning the schematic and engineering drawings, scheduling the development activities, or preparing the resources needed to carry out the task (e.g., setting up a 3-D computing platform). For the sake of exposition, we use a linear preparation cost per task. That is, if a task j is included in K teams, then the communication cost that is incurred is $T_j \times K$, where the preparation cost coefficient T_j is a constant associated with preparing for task j . Similar to attributes, different tasks may have different preparation cost coefficients depending on the complexity of the underlying task.

The goal of task partitioning is to assign tasks and attributes to teams so as to minimize the total task and attribute costs, while not exceeding the team's information processing capacity expressed in terms of the maximum number of attributes assigned to a team.

Finally, Step Three of the task partitioning problem is related to encompassing the above considerations using mathematical formulation, namely integer linear programming. The integer linear programming formulation may be found in the Appendix.

2.3 Illustrative Example

In this section, the task partitioning problem is illustrated by presenting an example of designing a parking garage [1]. In order to identify the tasks and their associated attributes, we apply the 'design matrix' specification technique that was

described earlier. At the end of the conceptual design process, 12 tasks (design parameters) are identified. Each task affects a set of attributes (the functional requirements associated with the task). At the detailed design stage of the project, the identified tasks are distributed among a number of design teams for detailed design. The various tasks and attributes are given as follows:

Tasks	Attributes
• Task 1: Red Granite Facade	• Attribute A: Screen Parked Cars
• Task 2: Fire Stairs	• Attribute B: Confine Fire
• Task 3: Sprinkler System	• Attribute C: Suppress Fire
• Task 4: Artificial Lighting System	• Attribute D: Provide Adequate Visibility
• Task 5: Reinforced Concrete Parapet Wall	• Attribute E: Provide Perimeter Barrier
• Task 6: Security Guard	• Attribute F: Personal Safety
• Task 7: Gate Control System	• Attribute G: Restrict Vehicle Access
• Task 8: High-Strength Concrete	• Attribute H: Design for Long-Term Durability
• Task 9: Drainage System	• Attribute I: Design for Ease of Maintenance
• Task 10: Post-Tensioned Flat Plate	• Attribute J: Resist Gravity Loads
• Task 11: Rigid Frame System	• Attribute K: Resist Lateral Wind Loads
• Task 12: Column Spacing	

The set of attributes associated with each task is described in the attribute/task binary matrix presented in Table 1.

Table 1: Attribute/Task Design Matrix for a parking garage design

Tasks	1	2	3	4	5	6	7	8	9	10	11	12
Attributes												
A	1	0	0	0	0	0	0	0	0	0	0	0
B	0	1	0	0	0	0	0	1	0	1	0	0
C	0	0	1	0	0	0	0	1	0	1	0	0
D	0	0	0	1	0	0	0	0	0	0	0	0
E	0	0	0	0	1	0	0	0	0	0	0	0
F	0	0	0	1	1	1	0	0	0	0	0	0
G	0	0	0	0	1	1	1	0	0	0	0	0
H	0	0	0	0	0	0	0	1	0	0	0	0
I	0	0	0	0	0	0	0	1	1	0	0	0
J	0	0	0	0	0	0	0	1	0	1	0	1
K	0	0	0	0	0	0	0	1	0	0	1	1

For example, the task 'develop high-strength concrete' is associated with the following attributes: 'design for long-term durability,' 'design for ease of maintenance,' 'resist gravity loads,' and 'resist lateral wind loads.' We assume that the maximum number of attributes that can be assigned to a team is 6. This number may be determined based on the previous performance of teams. For simplicity, all communication cost coefficients of attributes are identical and equal to 1 unit of time (i.e., $t_i=1$), and all preparation cost coefficients of tasks

are identical and equal to 1.5 units of time (i.e., $T_j=1.5$). Since the total number of attributes associated with the various tasks is greater than the information processing capacity of a team, we need to partition the set of tasks and attributes into several teams.

Many partitions are possible. Two feasible solutions are presented in Figure 5. Based on the formulation presented in the Appendix, the total cost of each solution is obtained as follows. In Solution One, tasks are assigned 12 times, thus incurring a total preparation cost of 18 units of time (1.5×12). We also observe that each of the attributes B, C, G, I, and J is included in two teams, while attribute K is included in three teams. Thus, the total communication cost that is incurred is 7 units of time ($5 \times 1 + 1 \times 2$). Therefore, the total cost of Solution One is 25 units of time. Similar calculations show that the total cost of Solution Two is 22 units of time.

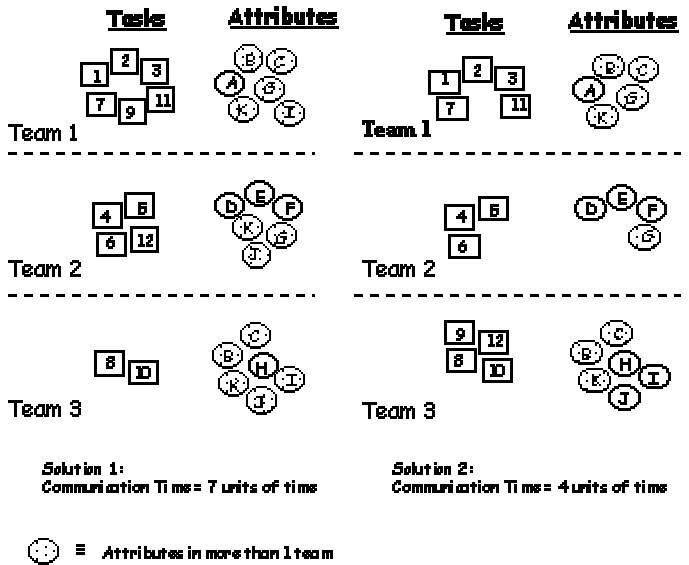
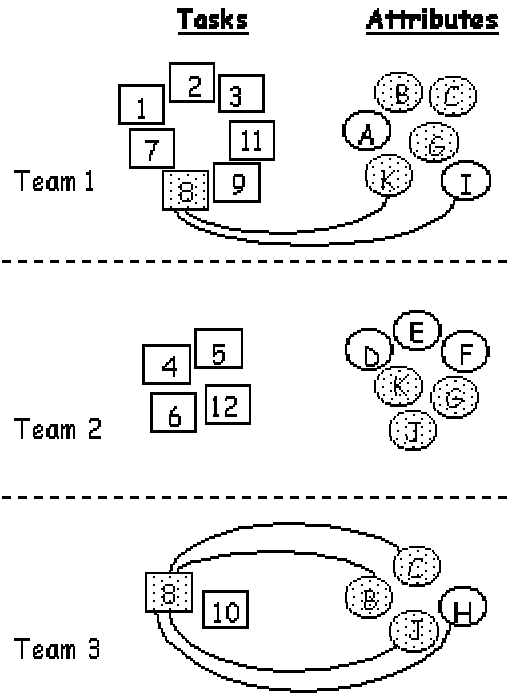


Figure 5: Two partitions for the parking garage problem

In the above two solutions, a task is assigned to only a single team. However, other partitions may be considered where a task is assigned to several teams. For example, in the third partition presented in Figure 6, task 8 is assigned to two teams. We observe that Solution Three is more effective than Solution One since the total cost that is incurred is 24.5 units of time (compared to 25 units of time). Indeed, by assigning task 8 to teams 1 and 3, the communication costs associated with attributes I and K are reduced (saving 2 units of time); however, additional preparation costs associated with task 8 are added. Since the reduction in communication costs (2 units of time) is greater than the added preparation cost of task 8 (1.5 units of time), assigning task 8 twice is justified. This

demonstrates the important tradeoff of reducing communication costs versus adding preparation costs.



Solution 3:
Communication Time = 5 units of time

- ≡ Attributes in more than 1 team
- ≡ Tasks in more than 1 team

Figure 6: A Partition that assigns task 8 to more than 1 team

2.4 Problem Characteristics

In this section, we draw several characteristics of the task partitioning problem. These characteristics provide insight regarding task partitioning, and are instrumental in devising solution techniques for the problem. The illustrative example provided in Section 2.3 shows that an effective solution may be obtained by assigning a task more than once. However, assigning a task to several teams may have a negative effect on the overall product quality. For instance, two teams that are assigned the same task may have conflicting interpretations of the information generated at early stages of the product development process. Thus, it may be useful to consider partitions where each task is assigned to only a single team. The following theorem shows that (1) if the team's information

processing capacity is greater than or equal to the number of attributes associated with each task, and (2) if all of the task preparation cost coefficients, T_j , are very large relative to the attribute communication cost coefficients, t_i , then it will be optimal to assign each task only once. This is formalized as follows.

Theorem 1. Let the total number of attributes that are associated with task j be less than or equal to the team's information processing capacity (i.e., $|N_j| \leq B$, see Appendix).

Let $Shared(j)$ be the set of indices of attribute types i associated with task j and at least one other task. In an optimal solution, task j should be assigned only to a single team if the following condition holds:

$$T_j > \sum_{i \in Shared(j)} t_i$$

where T_j is the preparation cost coefficient of task j and t_i is the communication cost coefficient of attribute i .

In the remainder of this paper, we assume that the condition stated in Theorem 1 holds for every task so that each task will be assigned exactly once in an optimal partition. Since each task will be assigned only once in an optimal partition, all attributes associated with a task will be assigned together. Thus, we will be interested only in partitions with respect to tasks (and not with respect to attributes). This also simplifies the procedure required to solve the task partitioning problem as shown in Section 3.

It is possible to manage problem-solving interdependence across team boundaries by, at a given level, reducing the cost of problem-solving coordination of attributes. This can be achieved by purchasing better communication technology that allows teams to reduce attribute coordination loss. Examples of information technology tools that enhance the coordination of project teams include meetings, electronic and voice mail, computer-assisted software, networks, co-location of teams, computer and other structured information technology. By incorporating better communication technology, it is possible to render the task preparation cost coefficients, T_j , very large relative to the attribute communication cost coefficients, t_i . In which case, the condition stated in Theorem 1 may become valid, and we will be primarily interested in the solution to the problem when each task is assigned to exactly a single team.

The next theorem addresses the computational complexity of solving the task partitioning problem. It is shown that the task partitioning problem is inherently intractable in several cases. The intractability of the task partitioning problem means that it is likely that no polynomial time (i.e., quick) algorithm exists for optimally solving the problem. In other words, the CPU time required to solve the task partitioning problem, based on known algorithms, grows exponentially with the 'size' of the problem. This property is formally stated as follows:

Theorem 2. The general task partitioning problem is NP-hard. The problem remains NP-hard even when considering the following restricted variants of the problem: 1) each task will be assigned to only one team; 2) each team is constrained to including 3 or fewer tasks; and/or 3) all attribute communication cost coefficients are identical and all task preparation cost coefficients are identical.

Theorem 2 implies that the potential of solving the task partitioning problem depends on the availability of certain heuristics. A heuristic algorithm finds a good solution fast (though sub-optimal). In Section 3, we propose an effective heuristic for solving the task partitioning problem.

The next theorem shows that a simple variant of the task partitioning problem can be solved in polynomial time (i.e., an optimal solution is found fast).

Theorem 3. If each team is constrained to have 2 or fewer tasks and each task is assigned to exactly a single team, the problem can be solved in polynomial time by performing $O(J^3)$ operations, where J is the total number of tasks.

In [5], a heuristic procedure is suggested where the task partitioning problem is solved by repeatedly incorporating constrained partitioning subproblems as described in Theorem 3. Furthermore, each such constrained subproblem can be solved fast as indicated by the Theorem. Finally, the following useful observation is made:

Observation 1: If the set of attributes associated with task j_1 is included in the set of attributes associated with task j_2 , then at least one optimal partition will have task j_1 and task j_2 in the same team.

3 AN EFFECTIVE SOLUTION PROCEDURE

In this section, we present a fast heuristic procedure⁴ for solving the task partitioning problem, and evaluate its performance on various data sets. The solution procedure is divided into two phases. Phase One constructs a feasible partitioning of the tasks into several teams, while Phase Two uses the feasible solution created by Phase One as a starting point, and iteratively improves the partition by moving to a better "neighbor" solution. These two phases of the task partitioning procedure are described in more detail below:

Task Partitioning Procedure

Phase One (constructing a feasible partition):

1. The construction of a new team is initiated by identifying the unassigned task that is associated with the smallest

number of attributes.

2. Additional tasks are sequentially added to the same team. A task can be added to the current team if the total number of attributes included in the team after adding the task is less than the team's capacity. If several unassigned tasks can be added to the team, then the task that adds the smallest number of *additional* attributes is added first.
3. If no task can be added to the same team, and there are tasks that have not yet been assigned to a team, then a new team is initiated and Step 2 is repeated. Otherwise, Phase One stops and Phase Two begins.

Phase Two (local improvements):

To improve the feasible solution that has been obtained in Phase One, two possible operations are sequentially performed. The first operation is related to transferring a single task (along with its attributes) from one team to another, where the second operation is related to exchanging two tasks (along with their attributes) that are included in two different teams. The executed operation at each iteration is the one that results in the largest cost saving. The local improvement operations are continued until no feasible cost saving operation can be identified, in which case Phase Two stops and returns the current partition.

To illustrate the task partitioning procedure, we consider the illustrative example presented in Section 2.3. Solution One in Figure 5 presents the three teams that have been constructed by Phase One of the task partitioning procedure. Phase Two uses the feasible partition returned by Phase One as a starting point, and iteratively improves Solution One (shown in Figure 5) by either removing a single task from its current team and including the task in another team, or removing two tasks from different teams and including each task in the team in which the other was included. Figure 7 shows the consecutive steps performed by Phase Two: removing task 12 from the second team and assigning it to the third team, removing task 9 from the first team and assigning it to the third team, removing task 11 from the first team and assigning it to the third team, removing task 3 from the first team and assigning it to the third team, and removing task 7 from the first team and assigning it to the second team. At this point, since neither a feasible cost saving move nor a feasible cost saving swap can be identified, the task partitioning procedure stops and the final partition solution is obtained as presented in Figure 8. We observe that at the first step of Phase Two, a reduction in total cost (of one unit of time) can also be obtained by swapping tasks 7 and 12. However, moving task 12 from team 2 to team 3 is executed since a larger reduction in total cost can be identified (2 units of time). Interestingly, the solution presented in Figure 8 reflects the structure of the parking garage system. Specifically, tasks that are assigned to team 1 correspond to the 'architectural design', tasks that are assigned to team 2 correspond to the

⁴ This is based on the generic partitioning algorithms in [7, 28].

'security system', and tasks that are assigned to team 3 correspond to the 'structural system' and 'fire safety system' of the parking garage. This also suggests the possibility of utilizing the proposed model for defining the product architecture.

To evaluate the performance of the task partitioning procedure, two data sets are examined. The characteristics of the underlying design matrix (e.g., percentage of 1s in the matrix) of each data set have been randomly generated based on data in [28] and is presented in Table 2. Many different task partitioning problems have been generated for each data set by varying the maximum number of attributes assigned to a team. To evaluate the quality of the solutions found by the task partitioning procedure, the relative error between the heuristic solutions and the optimal solutions⁵ has been determined. The results show that the average percentage error in the heuristic solutions is 6%, 7.2%, and 8.5% for data sets 1, 2, and 3, respectively. These results demonstrate the effectiveness of the task partitioning procedure, both in terms of CPU time (a few seconds) and solution accuracy. The results also suggest that the proposed heuristic could be easily applicable to large-scale product development processes.

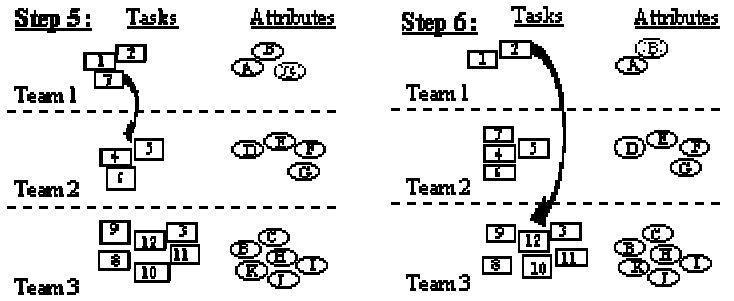
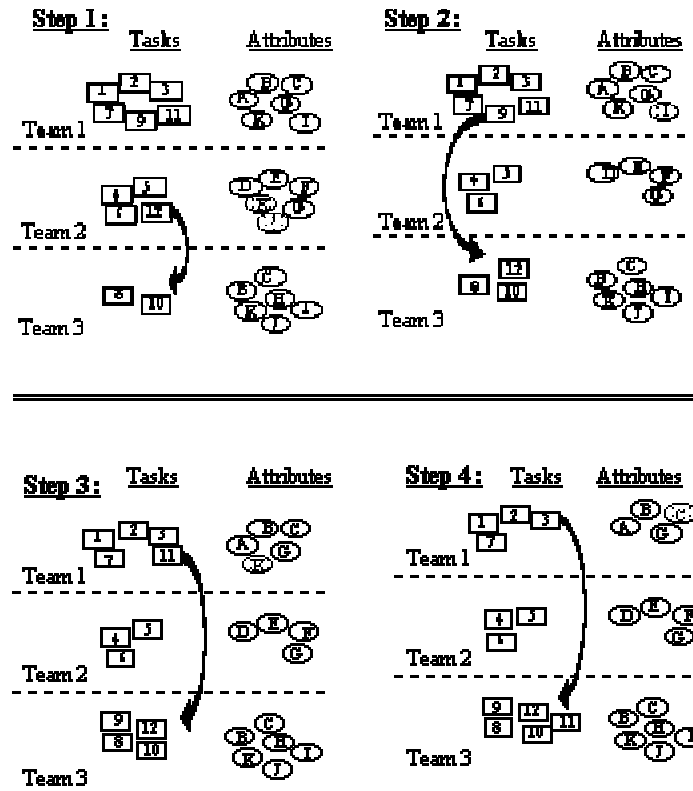


Figure 7: Consecutive steps performed by Phase Two of the task partitioning heuristic

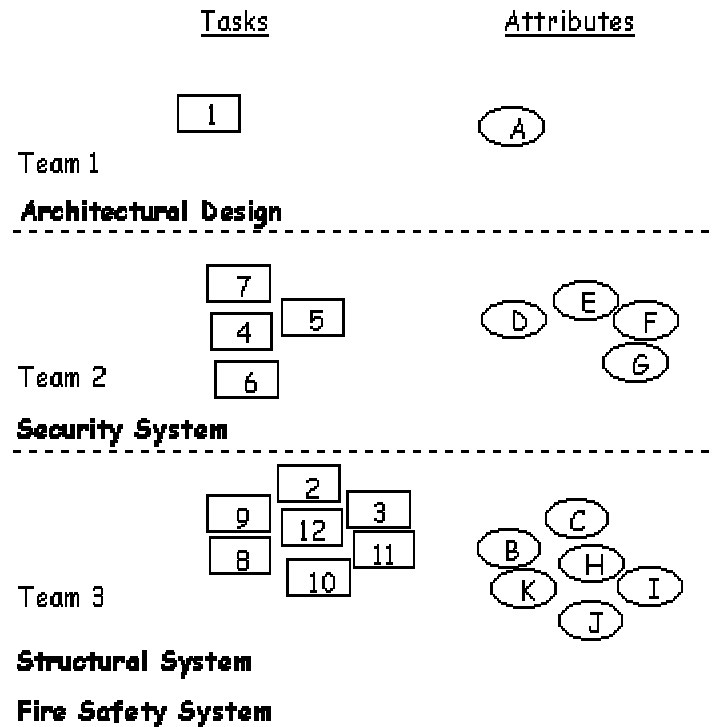


Figure 8: The optimal solution of the Parking Garage partitioning problem

⁵ The optimal solution has been found by employing a branch-and-bound enumeration algorithm.

Table 2: Characteristics of tested data

Data set	Number of tasks	Number of attributes	Average number of tasks sharing the same attribute	Number of problems tested	% of 1s in matrix
1	16	53	7	25	13.3
2	26	52	4.73	25	14.1
3	34	162	5.92	25	12.1

4 CONCLUSION

4.1 Summary

Empirical research in various industries has shown that task partitioning affects problem-solving interdependence, and that by properly managing task partitioning the efficiency and effectiveness of the product development process may be improved [13, 17]. Ineffective product development processes may be attributed to the failure of some companies to consider task problem-solving interdependence as an important aspect of partitioning decisions [17]. In such ineffective projects, partitioning is performed according to economies of specialization (e.g., all electrical design work is done by the same team), or according to traditional and rigid problem-solving patterns.

In this paper, we embark on a new mathematical formulation for task partitioning. The formulation is based on several key elements. First, it has been suggested to specify a task in terms of its associated attributes. Using this approach, task dependency is interpreted as the attribute types shared by (affected by) the tasks. As a useful specification technique, we have proposed to identify attributes as the set of functional requirements that are affected by the various tasks. Second, partitioning costs related to the coordination of an attribute across teams, and the preparation for a task by a team have been incorporated into the model. Third, the task partitioning problem has been formulated through integer linear programming, which is stated as the distribution of tasks and attributes among a number of teams so as to minimize the total attribute communication and task preparation costs. Several characteristics have been derived based on the mathematical formulation. Since the task partitioning problem has been shown to be inherently intractable in the general case (see Theorem 2 in [5]), a heuristic algorithm that finds "good" solutions fast has been developed. The heuristic algorithm is particularly useful for partitioning large-scale product development projects.

4.2 Future Implications and extensions

In the following, we delineate several ways by which the proposed formulation can be extended in order to address additional characteristics of task partitioning. In this paper, we assume that the tasks and their related attributes are completely specified prior to distributing the tasks among a number of design teams (e.g., by specifying the tasks and attributes at the end of the conceptual design stage or after establishing the product architecture). However, task specifications and task dependencies may change during project development due to new information that is generated as product development unfolds. For example, assume that a new requirement for an 'economic engine' arises during the project development process. This new requirement may lead to the generation of new tasks such as the development of 'electronic ignition' and 'fuel injection' systems. These two tasks, in turn, may create new dependencies with existing tasks of the project. Thus, new partitioning problems will be introduced with the generation of new tasks and attributes. If the specification of tasks and attributes is not relatively stable over time, then the various tasks and attributes may be re-partitioned as the project unfolds. To carry out the re-partitioning activity, we need to develop techniques for identifying changes in task specification.

Tasks and attributes can be specified at a number of levels of abstraction depending on the company's objectives. For example, consider the task of designing a model of a family car, which is associated with a set of aggregate attributes. This task can be further broken down into an entire task network; e.g., to a level including 'develop the chassis,' 'develop the power plant,' 'develop the power train,' and 'develop the body.' This level may be further broken down; for instance, the 'develop the power plant' task, in turn, may be broken down to a level of detail such as 'develop the engine,' 'develop the fuel system,' and 'develop the cooling system.' The problem of task partitioning, thus, becomes hierarchical and recursive. That is, a partition on one level of aggregation may affect the partition on the next level. Consequently, an effective partitioning on one level of aggregation ('local' partition) may not necessarily render the entire partitioning ('global' partition) of the system effective. We also observe that if tasks and attributes are specified at a level of aggregation related to the *schematic* of the underlying product ([15, 16]), then the formulation developed in this paper can be utilized to form and define the *product architecture*.

The structure of the attribute/task design matrix affects the efficiency and effectiveness of the underlying task partitioning problem. Given two design matrices defined by the *same* set of task and attribute types and same cost coefficients for the two problems, we can determine their comparative merit by examining their optimal task partitioning solutions. This approach of comparing design matrices is different from the dichotomy considered by Axiomatic Design, where design matrices are classified as uncoupled, decoupled (or

quasicoupled), and coupled [21] in descending order of preference. Indeed, this difference is exemplified in [5] where coupled design matrices may be more efficient than decoupled design matrices when comparing their corresponding optimal task partitioning solutions. The mathematical criteria for comparing design matrices have yet to be determined.

In this paper, task partitioning has been treated as a manageable variable. Problem-solving interdependence can also be managed by assuming a given partitioning, and by trying to reduce the cost of communication and coordination across task boundaries with the proper information technology tools. For instance, since the procurement of information technology tools carries a certain cost, it would be desirable to extend the model proposed in this paper so as to incorporate this cost.

To solve the task partitioning problem, tasks and attributes need to be completely specified. In this paper, it is suggested to apply the mapping between functional requirements in the functional domain and design parameters in the physical domain as a means for specifying tasks and their associated attributes. The use of data mining and knowledge discovery in databases may also be used for identifying dependencies between design tasks [6]. Having specified tasks and attributes, the next step is to derive the task preparation and attribute communication costs. In this paper, we utilize 'units of time' as an objective measure of project performance. In addition, for exposition purposes, a linear communication cost modeling has been introduced. Other aspects may be considered instead. First, a subjective project performance measure can be developed; e.g., (1) by ranking the attributes according to their extent of criticality, and assigning values to the various attribute communication costs by utilizing the Analytic Hierarchy Process [22], (2) by introducing a team performance measure that would be dependent on the number of tasks and attributes such that larger assigned tasks and attributes degrade the team performance. Second, a non-linear attribute communication cost modeling may be incorporated; e.g., by employing a *concave* rather than a linear function.

In this paper, we assume that a team can only manage a certain number of attributes. This assumption can be relaxed by allowing the possibility of adding more resources (e.g., people) to the team; thus, being able to manage more tasks and attributes.

Another consideration is related to pre-imposed constraints. Such constraints may require that specific tasks (or attributes) be included in the same team, or that specific tasks (or attributes) be included in particular teams.

The model developed in this paper can be incorporated within a decision support system. In order to find out whether a potential improvement in product development process efficiency and effectiveness is expected, one may compare the optimal partitioning as obtained by the model with respect to the partitioning that is adopted in practice. A major discrepancy in the way tasks are assigned to various teams may be an indicator for potential improvement or the need to incorporate

additional considerations.

Finally, we hope that the model developed in this paper will lead to a better understanding of other partitioning related problems, including decomposition, specialization, multifunctional integration, team formation, the role of suppliers in the product development process, and the effect of partitioning decisions on the overall product design.

REFERENCES

- [1] Albano, L. D., Connor, J. J., and Suh, N. P., 1993, "A Framework for Performance-Based Design," *Research in Engineering Design*, Vol. 5, Number 2, pp. 105-118.
- [2] Alexander, C., 1964, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, MA.
- [3] Allen, T. J., 1977, *Managing the Flow of Technology*, MIT Press, Cambridge, MA.
- [4] Baldwin, C. Y. and Clark, K. B., 1994, "Modularity-in-Design: An Analysis based on the Theory of Real Options," *Technical Report, Harvard University*.
- [5] Braha, D., 2001, "Notes on Partitioning in Product Design and Development," *Working Paper, Center for Innovation in Product Development*, MIT, Cambridge, MA.
- [6] Braha, D. (Ed.), 2001, *Data Mining in Design and Manufacturing*, Kluwer Academic Publishers, Boston.
- [7] Braha, D. and Maimon, O., 1998, *A Mathematical Theory of Design: Foundations, Algorithms, and Applications*, Kluwer Academic Press, Boston.
- [8] Clark, K. B., 1985, "The Interaction of Design Hierarchies and Market Concepts in Technological Evolution," *Research Policy*, Vol. 14, pp. 235-251.
- [9] Clark, K. B. and Fujimoto, T., 1991, *Product Development Performance*, Harvard Business School Press, Boston.
- [10] 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*, Vol. 6, pp. 1-13.
- [11] Hauser, J. R. and Clausing, D., 1988, "The House of Quality," *Harvard Business Review*, Vol. 88, pp. 68-72.
- [12] Henderson, R. M., 1988, *The Failure of Established Firms in the Face of Technical Change: A Study of Photolithographic Alignment Equipment*, *Ph.D. Thesis, Harvard University*.
- [13] Shirley, G. V., 1990, "Models for Managing the Redesign and Manufacture of Product Sets," *Journal of Manufacturing and Operations Management*, Vol. 3, pp. 85-104.
- [14] Simon, H. A., 1976, *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*. Free Press, New York.
- [15] Sosa, M. E., Eppinger, S. D. and Rowles, C. M., 2000, "Understanding the Effects of Product Architecture on Technical Communication in Product Development Organizations," *Working Paper Number 4130, MIT Sloan School of Management*.
- [16] Ulrich, K. T. and Eppinger, S. D., 1995, *Product Design*

and Development. McGraw-Hill, New York.

- [17] von Hippel, E., 1990, "Task Partitioning: An Innovation Process Variable," *Research Policy*, Vol. 19, pp. 407-418.
- [18] Steward, D., 1991, "Planning and Managing the Design of Systems," in *Proceedings of Portland International Conference on Management of Engineering and Technology*, Portland, Oregon.
- [19] Proceedings of the DSM Workshop at the Sloan School of Management, Massachusetts Institute of Technology, September, 2000.
- [20] Ulrich, K. T. and D. J. Ellison, 1999, "Holistic Customer Requirements and the Design-Select Decision," *Management Science*, Vol. 45, pp. 641-658.
- [21] Suh, N. P., 1990, *The Principles of Design*. Oxford University Press, New York.
- [22] Saaty, T., 1980, *The Analytic Hierarchy Process*. McGraw-Hill, New York.
- [23] Koopman, P. J., 1995, "A Taxonomy of Decomposition Strategies based on Structures, Behaviors, and Goals," *Design Theory & Methodology Conference*, September.
- [24] Pahl, G., and Beitz, W., 1984, *Engineering Design*, The Design Council, London.
- [25] Hubka, V., and Eder, W. E., 1988, *Theory of Technical Systems: A Total Concept Theory for Engineering Design*, Springer-Verlag, New York.
- [26] Perry T. S., 1992, "E-mail at Work," *IEEE Spectrum*, October, pp. 24-28.
- [27] Sproull, L., and Kiesler, S., 1991, *Connections: New Ways of Working in the Networked Organization*, MIT Press, Cambridge, MA.
- [28] Daskin, M. S., Maimon, O., Shtub, A., and Braha, D., 1997, "Grouping Components in Printed Circuit Board Assembly with Limited Component Staging Capacity and Single Card Setup: Problem Characteristics and Solution Procedures," *International Journal of Production Research*, Vol. 35, pp. 1617-1638.