The Journal of Korean
Institute of CALS/EC
Vol. 1, No. 2, Dec. 1996

동시공학/설계 환경에서 Conflict 중재

김명옥*

Mediating Conflicts in Concurrent Engineering / Design Environment

Myong -Ok Kim

Abstract

It is a typical scenario in concurrent engineering/design that different perspectives of each team members participating in the project exist, and those perspectives lead to conflicting decisions. The model 'resolution network' proposed in this work provides system-mediated resolution for all related team engineers to consider to optimize the manufacture in general. This paper focuses on development of the general architecture of the model and a search engine called Mediator to determine a resolution network from a given constraint network. The Mediator manages the constraint network, determines the most optimistic resolution called optimal point in terms of satisfying overall production goal, and use the optimal point to mediate controversial issues among teams. The biggest merit of our model is that it provides teams with resolution with logical and rational reasoning.

---

* 이화여자대학교 상경대학 상경학부

1. Introduction

Engineering projects generally involve a large number of components and the interaction of multiple technologies and participating engineers. For a single project, multiple team members from multiple firms geographically scattered over different locations often have to work together in cyber space through certain interfaces between agents and/or communication protocols. Maybe the most important and biggest task in such an environment is how to integrate multiple heterogeneous physical systems. Advances in database and networking technology, groupware, multimedia, and graphical user interfaces, and a precipitous drop in the cost of computing, all point the way to creating a truly collaborative environment to transcend the barriers of distance, time, and heterogeneity in computer equipment. Three well-known dimensions of the task are [Cutkosky et al., 1993]

1) cooperative development of interfaces, protocols, and architecture,
2) sharing of knowledge among systems that maintain their own specialized knowledge bases and reasoning mechanisms, and
3) computer-aided support for the negotiation and decision-making that characterize concurrent engineering.

The main focus of this work is on resolving conflicts among team engineers participating in concurrent design. There has been little coordination among various participants, and engineers typically find coordination among themselves very difficult. Coordination is critical for effective functioning of multidisciplinary product-development teams. These teams must influence each other so that a high-quality product is produced within a short turnaround. Potential conflicts among participants often are not recognized until manufacturing begins. This lack of coordination among team engineers can cause serious undesirable consequences.

Although concurrent engineering/design is almost universally advocated today, and many researches have been and are being done, it is still hard to execute when large multidisciplinary projects are involved. To illustrate some of the issues implied in design, consider engineering teams depicted in Figure 1. Here, teams are groups of engineers/designers from separate firms or functional groups within a firm. At any instant, the team members may be working on common product design, each employing its own expertise.

Suppose that the project involves product design on object x. Team1, Team2, and Team3 express their goals on attributes A and/or B of object x to be '$x[A] < 0$', '$x[A] + x[B] > 0$', and '$2 < x[A] < 10$', respectively where $x[A]$ and $x[B]$ denote attributes A and B of object x (Figure 2). What should be the goal on $x[A]$ in

terms of the product design? What type of logical reasoning should be applied to come up with a consensus? Or is it even possible?

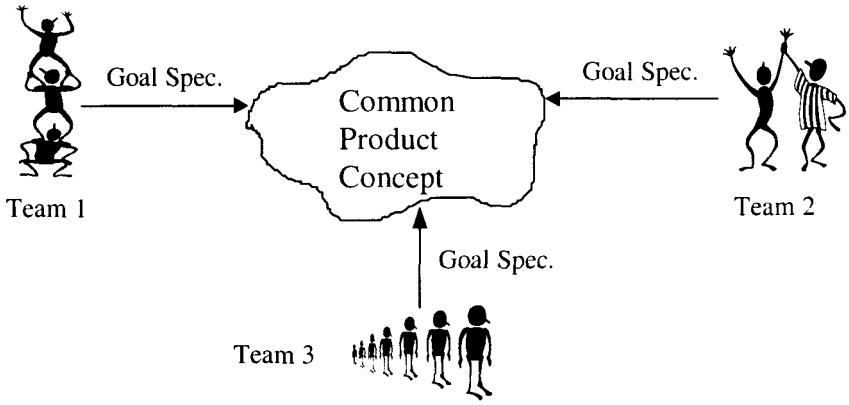

Figure 1. A Multidisciplinary Design Team

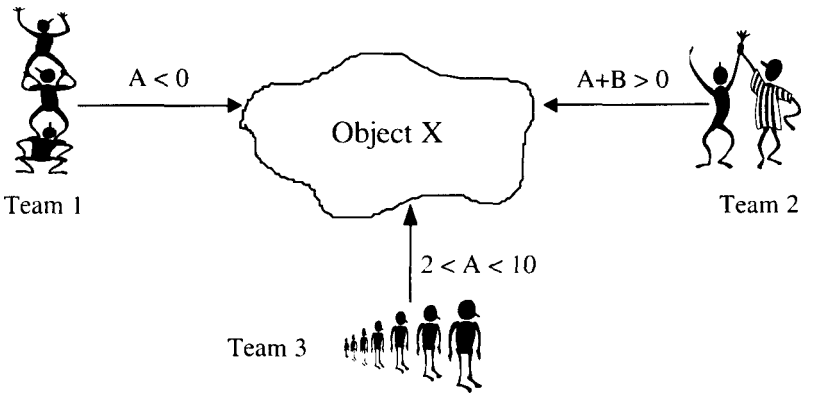

Figure 2. Controversial Goals of Teams

To deal with detection and resolution of complicated conflicts between various teams, constraint network, originally proposed by Bowen and Bahler [Bowen et al., 1991], has been greatly extended in this work. The whole research project is to model the Mediator that (1) forms the constraint network graph (CNG) from various goals of all participating teams and (2) finds the most optimistic resolution called *optimal point* in terms of satisfying overall constraints and uses the optimal point to transform the CNG to a resolution network graph (RNG). Here, we only concentrate on the second part. The major purpose of this study is to explore how to generate a RNG by using a given CNG which is assumed to be generated by the user interface module. User interface module is currently being developed.

2. Related Work

Concurrent engineering is predicated on the ability of each virtual team member to negotiate with the group and reach consensus. This means that the evolving design must be visible globally and its ramifications to any member's interest must be highlighted [Klein, 1991]. Blackboard technologies and research on constraint management are potential enablers in solving this problem. Also, to improve communication between teams, some work have been done on common visibility of activities and data, planning and scheduling of activities, notifying team members of changes, and managing constraints across multiple perspectives are some of the needs to be addressed by these services [Sriam et al., 1992].

During the course of product development, various decision-making tools are required by different members of the team. However, most of the present conflict-handling methods are centered on a single perspective. Team members need advances in the area of group-decision support, design assessment, and quality function-deployment tools to assist in group decision-making. It might be worthwhile to pay attention to some of the leading research projects in the field: the MIT Dice, the Stanford Designworld, and the constraint-based software [Sriam et al., 1993].

The MIT Dice program is aimed addressing coordination and communication problems in engineering. Dice can be envisioned as a network of agents or knowledge modules that communicate through a shared workspace called a blackboard. The present Dice framework is a collaborative agent-based architecture, and an agent is viewed as a combination of a user and a computer. The Dice has implemented a system called Coplan (constraint planner), which uses planning techniques to solve constraint-satisfaction problems. A planner is used as a top-level control process, guiding the search for a solution

and producing an appropriate solution plan when the problem is solvable. The Dice system provides cooperation and coordination among multiple designers working in separate engineering disciplines, using knowledge to estimate interface conditions between disciplines, recording who used any piece of design data created by others and how such data was used, and checking for conflicts among disciplines, manufacturability, and manufacturing cost and schedule impacts of design decisions.

The Designworld concept is a similar to that of desktop publishing [Genesereth, 1991]. A desktop publishing system user typically interacts with a what-you-see-is-what-you-get editor to create the desired document on the computer screen and then produces a hard copy with a laser printer. In Designworld also, agent-based framework is employed to addresses many issues critical to effective concurrent engineering. In this approach, the idea is for programmers to write their programs as individual software agents. Communication is assured by using a standard agent communication language. Agent interaction is assisted through the services of a coordinated set of operating-system programs called facilitator agents or facilitators. In Designworld, an agent is a software that is capable of communicating with other programs in the agent communication language. The key feature of an agent communication language is its expressiveness. It communicates constraints, negations, disjunctions, rules, and quantified expressions in a variety of modes, such as questions, assertions, and commands. Agents do not communicate directly. Instead, they communicate with their local facilitators, and the facilitators communicate with each other. In effect, the agents form a "federation" in which they surrender their communication autonomy to the facilitators; hence, the name of the architecture.

The agent-based approach to software interoperation rests on the key ideas of communication standards and federation architecture. Neither idea is new. What is new here is the application of agent technology to help push these ideas to their limits. The novelty stems from (1) the use of a standard agent communication language to encode general knowledge about application areas and about programs, and (2) the application of automated reasoning and automatic programming technology to implement facilitators that can use this knowledge to match clients with servers in much more interesting ways than in the past.

The constraint-based software approach [Bowen et al., 1993] aims at developing a language and methodology to obtain comments or advice on the evolving design from other professional experts throughout the design process. Bowen and Bahler have investigated the possibility of developing a concurrent

engineering-oriented language based on the notion of constraint networks. A constraint restricts the values that can be assumed by a group of one or more parameters. A constraint network is a collection of such constraints. Figure 3 depicts an example network, with parameters in round nodes and constraints in rectangular nodes.

A constraint network constitutes the specification of a relation where each tuple represents a group of consistent value assignments to the network parameters. The set of admissible values for an individual parameter is the projection of this relation onto the parameter. The attraction of constraint network is that they can support multidirectional inference. This means that a constraint network can capture the impact of a decision – made concerning one phase of a product's life cycle by an expert in that phase – on the other phases of the life cycle.

Negotiation/conflict management techniques presented in this section and many other existing simple communication methods, in general, do not support very complicated and serious controversies. Also, all existing methods illustrate how the choice is made without telling much or none about why the choice made is rational.

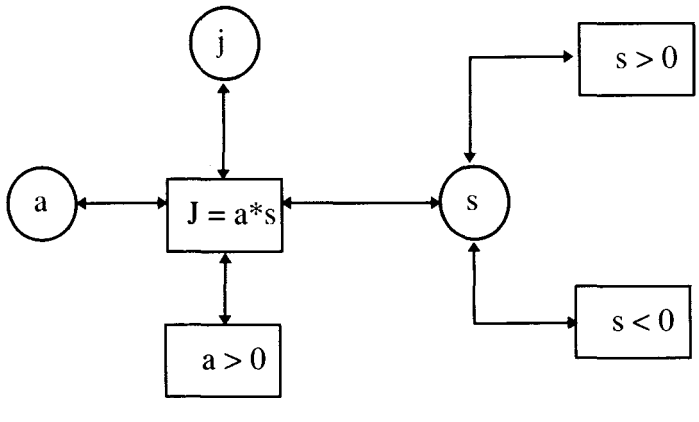

Figure 3. An Example of Constraint Network

3. Resolution Network Model

It is a typical scenario in concurrent engineering/design that different perspectives of each team members participating in the project exist, and those perspectives lead to conflicting decisions. The model 'resolution network' proposed in this work provides system-mediated resolution for all related team engineers to consider to optimize the manufacture in general. The long-term research covers several major phases: user interface and language development, constraint network management, and generation of resolution network. This paper focuses on the general architecture of the system and development of a search engine called Mediator to determine a resolution network from a given constraint network.

3.1 Environment

Product-development teams scattered over different locations persent their manufacturing goals and views to other teams through user interface and concurrent engineering-oriented language. Bowen and Bahler have worked on developing such a language, and discussion of the interface and language is excluded from this paper. Goals and views on a product are called constraints in this work. As in Figure 4 constraints from various users enter the network and stored as a labeled graph called constraint network in the central database.

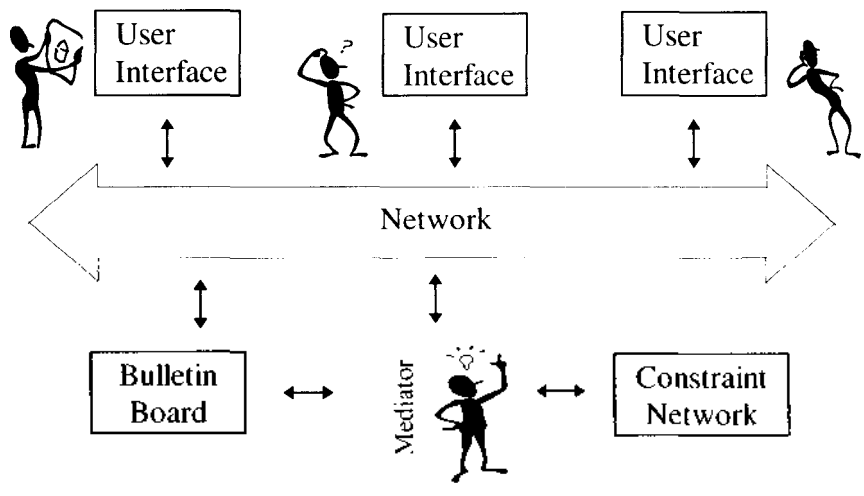

Figure 4. Architecture of the Mediator

Suppose that several team members entered four constraints on two attributes A and B of object *x*, '0 < x[A] < 10', '0 < x[A] < 5', 'x[A]+x[B] > 5', and 'x[B] = 4' as in Figure 5. A constraint network is a labeled graph which is defined as follows: each node is labeled by an attribute of an object to be manufactured; each arc is labeled by a constraint that user wants to impose on the attribute; an arc can be cyclic; a constraint which labels an arc is an arithmetic expression of intervals (Figure 6). After users' constraints are recognized as a graph with expressions shown in Figure 5, all constraints in the graph are transformed into corresponding intervals as shown in Figure 6. In this study all values are viewed as intervals. It is beyond the scope of this paper to discuss such transformation processes. It is assumed that the constraint network with intervals as in Figure 6 is provided to generate a resolution network graph.

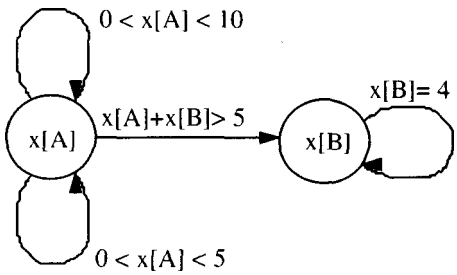

Figure 5. A Constraint Network Graph

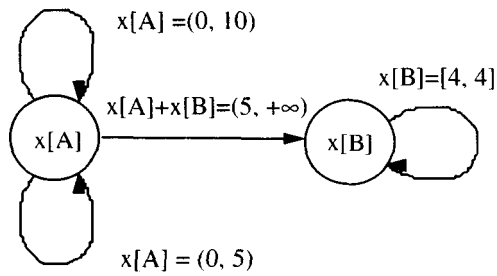

Figure 6. A Constraint Network Graph with Intervals

Using the transformed constraint network, the search engine, Mediator, tries to find *rational reason* to resolve conflicts among various teams. The main philosophy of the Mediator is to make an appeal to team engineers to consider *overall* satisfaction of all constraints (goals) of all teams, not an individual team's view. For example, in Figure 6, for the attribute A there exists a conflict between two teams: One team says that the value of A should be in (0, 10), and another team insists that it should be in (0, 5).

For a node with such a conflict, the Mediator checks all adjacent nodes' constraints and determines optimistic values within the search space for all attributes (nodes) in the constraint network graph. The value which

satisfies the most number of adjacent constraints is called the most *optimistic* value. If the optimistic value for attribute A is determined as interval (4, 8), then the Mediator informs all corresponding team members of the interval and persuade them to adjust their limits and boundaries according to that.

3.2 Resolution Algorithm

For a given conflict network graph G, the Mediator determines a search space S using intervals of all cyclic arcs of all nodes in graph G. After that the Mediator begins its navigation starting from the lowest vertex in S searching for the peak vertex which is known to satisfy the most number of adjacent constraints presented as arcs between nodes in G. The peak vertex is called *optimal point*. To find the optimal point, the generic *iterated next ascent hillclimbing* [Ackley, 1987] was modified to add some rules as follows:

1) Build the search tree from the attribute which is bounded most tightly in S.
2) Use branch-and-bound search with bounding heuristics to determine a move. For each branch, it is tested whether it is worthwhile to visit the branch or not. If it is not worthwhile, then that branch is pruned from the search tree.

function Determine-Optimal-Point:

{ input: constraint network graph (CNG)
output: optimal-point }

For each node i in the CNG
    U_i = union of all intervals of cyclic arcs of node i;
Determine n-dimensional search space S using U_i, for $1 \le i \le n$;
Let x be a vertex $(v_1,...,v_n)$ in S, where v_i is an integer for the attribute of node i;
Set an initial vertex x_i to the lowest point in S;
$dist_i \leftarrow$ Eval (x_i, CNG);
optimal-point $\leftarrow dist_i$; $x_c \leftarrow x_i$; $x_a \leftarrow x_i$; $j \leftarrow n$; $old_j \leftarrow j$;
$dist_c \leftarrow dist_i$; $dist_a \leftarrow dist_i$; flag $\leftarrow$ true;
go to step 2;

1. if jth bit in x_a has reached the upper boundary, then
   if ($j = old_j$) then
     flag $\leftarrow$ false; $old_j \leftarrow old_j - 1$; $j \leftarrow old_j$;
     if ($j = 0$) then return optimal-point;
     else $j \leftarrow j - 1$; go to step 1;
   if flag then
     select x_a from x_c by (1) changing jth bit in x_c to the next adjacent value and (2)reinitializing bits from j+1 to n with the least values in S;
     $dist_a \leftarrow$ Eval (x_a);
   else flag $\leftarrow$ true;
     select x_a from x_i by changing jth bit in

x_i to the next adjacent value;
$dist_a \leftarrow$ Eval (x_a);
if ($dist_a < dist_i$) then $dist_i \leftarrow dist_a$; $x_i \leftarrow x_a$; $x_c \leftarrow x_a$;
else go to step 1;

2. if ($dist_a \leq dist_i$) then
$x_c \leftarrow x_a$; $j \leftarrow n$; flag $\leftarrow$ true;
if ($dist_a \leftarrow dist_c$) then optimal-point $\leftarrow x_a$; $dist_c \leftarrow dist_a$;
else optimal-point $\leftarrow$ optimal-point $+x_a$;
go to step 1;
else $j \leftarrow j-1$; go to step 1;
endfunction;

function Eval:

{ input: vertex x, CNG
output: an integer, 'dist' }
for all arcs between nodes in the CNG
evaluate vertex x, producing dist;
if vertex x satisfies all constraints between nodes, then dist = 0;
otherwise, dist = the number of arcs between nodes which are not satisfied by vertex x;
return dist;
endfunction;

Suppose the given constraint network contains two nodes for x[A] and x[B] and two arcs between them, 'x[A]+x[B] > 5' and 'x[B] < 3'. Then, vertex x(3, 4) for x[A] and x[B] produces distance 1, since it satisfies the constraint 'x[A]+x[B] > 5' and violates the constraint 'x[B] < 3'. The final output of the above algorithm is an optimal point which is used to produce a resolution network.

function Resolution-Network:

{ input: optimal-point, CNG
output: modified CNG (resolution network: RN) }
for each node i in CNG,
add a cyclic arc labeled with v_i of optimal-point;
remove all other cyclic arcs from node i;
return RN;
endfunction;

Figure 7 shows the transformed network from the given constraint network in Figure 6. The Mediator puts the resolution network on the bulletin board for participating team engineers, and engineers are advised to reflect the recommendation. A resolution network is in effect until any change on the CNG is reported. When any of the constraints on the CNG is updated by a team in response to the recommendation, the Mediator is immediately notified and starts processing the CNG to generate a new resolution network graph.

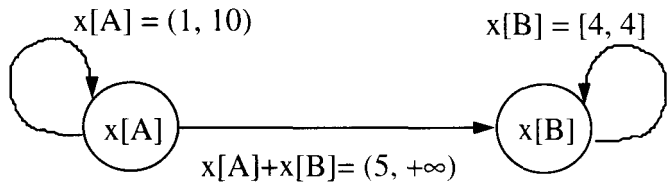

Figure 7. A Resolution Network Graph

The above resolution algorithm is justified by intuition. The underlying philosophy of the model, mediating conflicts by introducing the optimal point for all participating teams' constraints (goals), is a new rational approach, and for the first time the resolution algorithm is introduced in this work. However, the model has to be expanded mainly in following areas in the future: user interface module / language design to form CNG and manage updates on it, fuzzy data value handling, etc.

4. Conclusion

One way to carry out concurrent engineering is to use product-development teams comprising professional experts in each phase of the product life cycle. Throughout the design process, the designer obtains comments on the evolving design from other team members. However, product-development teams present many logistic, scheduling, and other management difficulties. One way to address some of these difficulties is to use software that provides life-cycle design advice.

The main purpose of this work was to develop the core method, the resolution network transformation algorithm to mediate conflicts between participating engineering teams. The resolution proposed in this study is the first kind to reflect overall goal satisfaction in managing and judging conflicts. The Mediator, a search engine, was introduced to find the optimal point, and the search algorithm was based on the modified iterated next ascent hillclimbing. We recognize that this model must be improved in various fields. Designing user interface module of the Mediator and more comprehensive support for data values such as uncertain, imprecise, or fuzzy values constitute a major issue in our ongoing research.

References

[Ackley, 1987] D. Ackley, *A Connectionist Machine for Genetic Hillclimbing*, Kluwer Academic Publisher, Boston, 1987.

[Bowen et al., 1991] J. Bowen and D. Bahler, "Supporting Cooperation Between Multiple Perspectives in a Constraint-Based Approach to Concurrent Engineering," *J. Design and Manufacturing*, (1) 1991.

[Bowen et al., 1993] J. Bowen and D. Bahler, "Constraint-Based Software for Concurrent Engineering," *IEEE Computer*, January 1993.

[Cutkosky et al., 1993] M. Cutkodky and W. Mark, "PACT: An Experiment in Integrating Concurrent Engineering System," *IEEE Computer*, January 1993.

[Genesereth, 1991] M. Genesereth, "Designworld," *Proc. IEEE Conf. Robitcs and Automation*, L.A. Calif. U.S.A., 1991.

[Klein, 1991] M. Klein, "Support Conflict Resolution in Cooperative Design Systems," *IEEE Trans. Systems, Man, and Cybernetics*, 21(6), December 1991.

[Sriram et al., 1992] D. Sriram et al., "Dice: An Object-Oriented Programming Environment for Cooperative Engineering Design," *Artificial Intelligence in Engineering Design*, Academic Press, 1992.

[Sriram et al., 1993] D. Sriram and R. Logcher, "The MIT Dice Project," *IEEE Computer*, January 1993.

저자소개

김명옥

1980 이화여대 법정대학, 문학사
1981 Ohio State University, 경영학 석사
1989 Wayne State University, Michigan, 전산학 석사
1993 Wayne State University, Michigan 전산학 박사
1992 - 1995 Ford Motor Company 연구원
1993 - 1995 University of Michigan - Dearborn, 전산정보학과 객원조교수
1995 - 현재 이화여대 상경대학 부교수
주요 논문 : 퍼지데이타 환경에서 생산과정 최적화
관심 분야 : 전자계산(Database, Expert System, Fuzzy System, Office Automation, Information Retrieval, etc.)

연락처: (연구실 전화) 360 - 3465
(Fax) 393 - 3734
(E-mail) myongkim@mm.ewha.ac.kr