Coordination between Project Participants through Constraint-Management

Hossam El-Bibany, Research Assistant
Boyd C. Paulson, Jr., Ohbayashi Professor of Engineering
Lai Heng Chua, Research Assistant

Stanford University
Department of Civil Engineering
Stanford, CA 94305-4020
U.S.A.

ABSTRACT

Project parties need to coordinate their efforts throughout the project life-cycle. This is especially the case in fast-tracked projects that have become commonplace in the construction industry. The crude methods available currently have become inadequate for these severe demands. In this paper, we describe the utility of constraint-management theory as a tool for building an integrated knowledge base to coordinate across disciplines involved in the overall project life-cycle. The paper starts with a brief description of the coordination problem in the AEC industry. A constraint-based view of a project's life-cycle is then described in section 2. Section 3 discusses the functionality and different issues related to implementation of a computer system to facilitate coordination. Section 4 describes the scope of and the environment that we have chosen for implementation. The paper ends with some conclusions based on the current state of this project.

1. INTRODUCTION AND BACKGROUND

The waning competitiveness of the U.S. construction industry in the past 20 years has been well-documented. Since the 1960's, the productivity of the nation's largest industry has been declining at a rate of 1% to 2% per year [Howard et al 89]. Practically all observers agree: one of the root problems is poor coordination in a highly fragmented and specialized industry. The planning, design and construction phases of a single project—whether a road, a bridge, a building, an industrial plant, or some other facility—are typically carried out by different planners, architects, and engineers, often working for different public or private organizations. Even at a particular phase of a project, such as design, the major structural, mechanical, and electrical engineering design tasks are likely to be performed by separate individuals and organizations. At the same time, fast-tracked projects, where design proceeds almost concurrently with construction and thus require tight coordination, have become the rule rather than the exception. While specialization has produced high quality results in specific cases, it has also created problems of coordination in an increasingly complex and time-pressured environment. An example of the results of errors in coordination is the 1981 collapse of the skywalk in the lobby of the Hyatt Regency Hotel in Kansas City [Marshall et al 82].

The methods currently used for coordination are quite primitive. Most of the information is passed by voice and paper up and down the organization as well as between organizations. Groups at all levels of the project organization meet often to sort out the issues and solve problems. Not only is this meeting process expensive, but the participants in the meetings often do not bring all their knowledge to bear on the problem. Rather, they work as best they can within their narrow areas of expertise and responsibility. Jurisdictional considerations result in having many "untouchables" — i.e., "it is not within

our jurisdiction." There are personality problems: people with brilliant and valuable economic solutions may not choose to communicate their ideas during the short time available in meetings, or they may not get the chance. Meetings are also expensive since management time is expensive. Ideally, project participants should all work as a team to solve the problem better, but we cannot achieve this with technological tools.

In this paper, we present a research topic that we started recently to investigate the utility of constraint-management theory as a tool for coordination across disciplines involved in the overall project life-cycle, and to identify the main issues in constraint representation and reasoning for achieving this coordination. It is not meant to be an encompassing integrative framework, but as a tool that participants can choose the extent to which to use. Earlier research on constraints focused on using them in the single area of design, planning or scheduling. However, the methodology of constraint-management can be extended in various directions to encompass a wider range of engineering problem-solving. Constraint-management provides a uniform framework for expressing the relationships that are necessary in engineering problems. It also provides transparent mechanisms for maintaining consistency among these relationships.

2. A CONSTRAINT-BASED VIEW OF THE WORLD

2.1 A Constraint View of Project Life-Cycle Integration

We can look at a constructed facility as a set of related objects. For example, in a multi-story steel residential building, beams and columns are objects. Each party involved in the creation of the building imposes different relationships among these objects. In the final facility, all of these relationships are satisfied. The architect forms the spatial relationships among the beams and columns and imposes more constraints on the final shape of these objects. The structural engineer forms a different set of constraints on the attributes of the same objects. Some of these constraints are derived from the structural analysis, and other constraints come from specifications imposed on the attributes of the objects. The construction planner might have another constraint on the weight of these objects due the capacity limitations of the cranes that can be available on site. In real life, these different sets of constraints imposed on the same object are known by different parties. They can only be communicated through organizational channels which are not very reliable and thus risk the delays that can occur either because one of these constraints is missed (e.g., crane-capacity limit which may require the redesign and manufacturing of another object), or because the set of constraints has no common solution (e.g., the depth of a beam cannot satisfy the architecturally required limit due to the high load intensity that it carries). In any case, the problem has to be communicated back to the parties involved and a redesign must then be initiated.

Other kinds of constraints appear in a fast-track project where the requirements change as the facility is being erected. A decision to add one more floor would, of course, depend on the maximum capacity of the supporting foundation and columns that are already erected.

2.2 Restrictive vs. Suggestive Constraints

The previous constraints fall into two different types, *restrictive* (hard) and *suggestive* (soft). Restrictive constraints are imposed through specifications or analysis (failure or serviceability requirements). Failure to satisfy a restrictive constraint on an object should alert the parties involved in setting the constraints on this object to solve the problem. Suggestive constraints are mostly imposed by different parties for economical reasons (e.g., reducing cost). The crane-capacity constraint mentioned above can be

suggestive if higher capacity cranes are available. A non-satisfied suggestive constraint would not bring the system to a halt, but may notify the user of the implication of the failure to satisfy the constraint.

2.3 Conceptual View of the Project

The structure of a system that uses knowledge about objects in the form of constraints is envisioned as shown in Figure 1. Knowledge about project objects forms the basic core. The constraints layer describes the relationships that we wish to maintain concerning the properties of the various objects. The connectivity layer ties all parties working on the project together. Figure 2 illustrates some examples of project objects and project constraints.

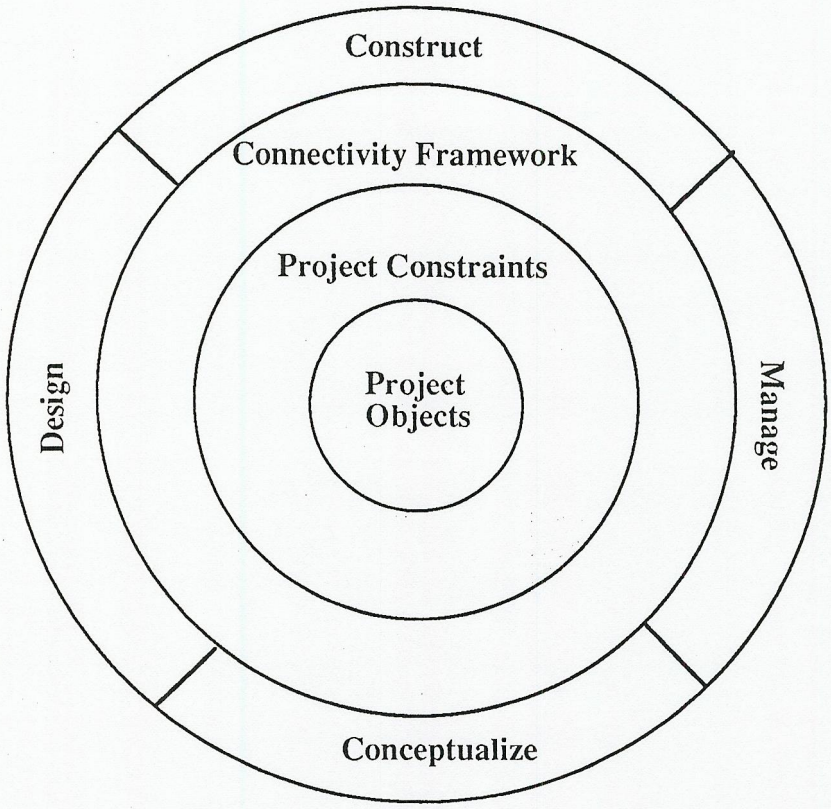

Figure 1- Framework for Coordination between Project Parties

3. COMPUTER SYSTEM REPRESENTATION AND REASONING

Our main objective is to unite the analytical and the heuristic aspects of design, construction and facility management under a common constraint-based framework to build and maintain an integrated object knowledge base.

We *formulate* constraints when we conceptualize the problem and decide what important objects and relationships occur in the domain of every party involved in the project. We *propagate* constraints when we derive new information based upon what is currently known by using the causal knowledge encoded in the constraints. We *integrate*

constraints when locally (to one party) consistent descriptions are checked for global consistency (with all project parties) [Stefik 80].

Examples of project objects:

beams, pipes, wiring, formwork, scaffolding, cranes, compressors, pumps, locations, activities

Examples of sources of project constraints:

Architectural Design	- distance between adjacent columns must be at least 8m
	- distance from floor to bottom of next floor beam must be at least door height
Structural Design	- maximum shear stress, maximum bending moment
Construct on Architect	- use other materials as concrete is very expensive
Design on Architect	- largest bridge span is 1500m
Design on Construct	- maximum concentrated load on floor
Construct on Design	- maximum reachable height for concrete pump
	- maximum weight liftable by crane

Examples of formulation of project constraints:

height of door + height of beam < distance between floors
weight of object to be moved < maximum weight liftable by crane
weight of equipment on floor + weight of load < maximum concentrated load on floor
no more than two large cranes can be on the site at the same time

Figure 2- Some Examples of Project Objects and Constraints

3.1 System Functionality

In the process of constraint propagation and integration, every step will be recorded in a multi-layer truth-maintenance (see section 3.2) to form an integrated knowledge source about the relationships between the attributes of the different project objects, and the way the constraints were solved (e.g., what are the heuristics used to solve each problem, who supplied these heuristics, etc.). The user can, then, query the integrated object knowledge base for the existence of certain relationship between the attributes of different objects. He can even plan his future decisions by querying for the effect of any change in an object's attribute, without really making the change, on any other object or total project cost. This allows the user to make efficient *what-if* queries without affecting the use of the system by other parties.

For the user to change an object's attribute under his control, the system would access the truth-maintenance module to decide on the constraints that would be affected, reschedule these constraints for propagation, and update the truth-maintenance module as these constraints are being successfully solved. The system would send messages to every participant identifying the party that required the change, the affected objects, and the way the attributes of these objects have been affected. The users can query the system for the details of any change, or run a what-if analysis for the effects of any changes which they might be proposing, on the different objects. In case of constraint violations, the system would prompt the user responsible for the entry to withdraw his change that resulted in the violation or relax one of his requirements. The user may withdraw the change that he

requested or, if he insists, the system would notify the different parties who are responsible for the violated constraints to tell them they need to solve the problem through meetings or any other type of communication. The solutions should then be entered into the system.

3.2 Supporting Theory

In developing the proposed theory, there are three major items that need to be investigated. The first concern is the development of the multi-layer truth-maintenance system as a project knowledge base, and its interaction with the constraint-management engine. Analogous to Stefik's meta-planning layers [Stefik 80], we are designing a multi-layer truth-maintenance system [de Kleer 86 a,b and c]. Each layer forms an abstraction and holds more information on the facts which led to the formation of the lower layer. For example, while the constraints are being solved, a lower layer in the truth-maintenance module would hold the dependencies between the attributes of the different objects. These dependencies have resulted from applying certain heuristics and meta-heuristics to solve the constraints. A higher layer in the truth-maintenance module would hold the information on the application and dependencies between these heuristics. It would probably be very helpful for the user if another higher layer would hold information about the value intervals of each attribute which would, or would not, affect the value of a certain dependent important attribute. The types and number of layers required are being studied.

The second concern is to determine the type of actions that the system should take when it detects a constraint violation or when it incorporates new knowledge. Updating the knowledge in the machine is not enough since the human agents may take actions based on outdated knowledge leading to problems. The system will have to update the knowledge of the different participants with any changes.

The third concern is to determine how to gather the information required to maintain the system and interact with participants. One of these facilities would be a browsing facility allowing the different users to browse through the different project objects and constraints. Every participant has the authority to change only the constraints for which he is responsible.

3.3 Framework for Integration

The framework for integration that we plan to use can be described by the following points:

i) *Object-oriented representation.* Our representation of objects attributes and their relationships is based on the concept of a *module* [Chan 86, Lansky 88]. The module concept is associated with encapsulation, information hiding and controlled interface between the module and its environment. The module is a named collection of methods and constraints on the objects' attributes which are encapsulated so that access to the objects' attributes or the methods can be controlled via the interface protocol. This partitioning is very useful in controlling the human-machine interface. An attribute is simply a named entity that represents a parameter and can accept design descriptions, which are the values of the design parameters (examples are depth of beams, columns dimensions, crane capacity, etc.). In each module, frame-based structures will be used to represent object attributes and their relationships. Objects can be physical (beams, columns, etc.), or conceptual (constraints, construction activities, etc.). Slots will represent structure of objects (e.g., physical attributes of a beam), behavior of objects in the form of procedural attachments and active values (e.g., compression stress check for a column), relationships (e.g., supportedBy relationship), or conceptual information (e.g., constraintProvidedBy).

ii) *Constraints representation imposed by different project parties.* Constraints can be generic, represented as behavioral attachments to the objects, or can be imposed by different project parties. In the latter case, information about the provider of the constraint, the date and time on which it was entered, and the attributes of the constraints will be represented.

iii) *Planning.* Solving any problem using constraints needs a plan of the sequence of the constraints to be solved. In a cooperative system, where different users interact and enter constraints, we cannot assume that there is a certain sequence of constraint utilization beforehand. The constraint-management system will, hence, have to find this sequence as well as identify conflicting and redundant constraints, which we call planning the use of constraints. In this system we will use standard graph algorithms for solving this problem [Serrano 87]. Domain-specific Knowledge (but not project-specific) can also be used for simplifying this problem.

iv) *Causal knowledge.* Causal knowledge tries to explain behavior in terms of cause-and-effect relationships. Mathematical constraints implicitly represent these kinds of relationships. The design interpreter uses constraint propagation to transmit information from one constraint to another in the module. Causal reasoning leads to more robust problem-solving since appropriate actions for all possible situations (including unanticipated situations) are implicitly coded in the causal representation [de Kleer and Brown 86, Iwasaki and Simon 86, Shoham 87].

v) *Heuristic knowledge.* Heuristics are ad hoc means of finding values for certain objects' attributes. They are used when the processes that determine the description for an object's attribute cannot be adequately or efficiently described by the cause-and-effect mechanism (failure of propagation). The party who is most knowledgeable about the problem provides this heuristic knowledge.

vi) *Truth-maintenance.* Truth-maintenance is a method for keeping track of changes and undoing them if necessary. A multi-layer truth-maintenance system will be constructed. The first layer keeps track of the changes in the values of attributes of project objects and constraints. The second layer keeps track of the heuristics used, etc.

vii) *What-if query capability.* The framework provides users with the capability to perform what-if queries. What-if queries are tested against the normalized knowledge base. A powerful query system would be built that helps users formulate their questions.

viii) *Locking and unlocking for multiple user access.* The system allows several parties to input constraints, make what-if queries and create new project objects. The designer does not have to be finished before the construction manager inputs his constraints, nor does he have to wait until the architect has input his constraints or defined his project elements. However, to make sure that a value or constraint cannot be changed under the user's nose, some database locking, such as read and write locks, will be designed and provided.

ix) *Tracking knowledge.* Since multiple parties are involved, it is necessary to keep track of who created the project objects and define their attributes, defined constraints, etc. This knowledge is called *tracking knowledge* and will be incorporated into the framework for use in controlling the interaction with the users.

x) *Interface Elements.* All interactions are processed through a rich human-machine interface. Each party has an interface designed to suit his or her viewpoint of the project. The interface includes the ability to browse the defined project objects and constraints.

4. IMPLEMENTATION AND SCOPE

The research is based on a constraint-management system which we are designing and implementing. We see the constraint-management system as a tool that will *assist* personnel at all levels by *flagging conflicts* and enabling them to determine the *cause* of problems and thereby solve the right problem in a better way.

Basically, we are looking at three principal agents in a hypothetical fast-track construction of a multi-story residential or commercial building. These agents, which the system will be assisting, are the architect, the structural designer and the construction manager. We are building a constraint-management system in which the new information requirements, constraints and field conditions are *automatically* propagated to the agents who need to know about them. At the same time, the system *automatically* continues to maintain the constraints that were previously specified and still remain unchanged, such as structural constraints. Designated variable parameters in the system are automatically readjusted to satisfy the new requirements if possible; if it is not possible to satisfy all of the new constraints, then the parties concerned can be *notified* immediately so that negotiations can proceed among them or new resources can be brought to bear on the project.

4.1 Implementation Environment

In our view of knowledge integration, there are two things that we want to capture: the knowledge that is encapsulated within the objects, and the relationships among the objects. A methodology that combines object-based representation (static) and a constraint-based engine (dynamic) would provide us with most of the necessary power. Neither methodology alone would be adequate. We have chosen an object-oriented logic programming system as an implementation environment. Logic programming will provide us with the strong capabilities of dynamic data manipulation [Chan 86, Chan and Paulson, 87,88].

5. CONCLUSIONS

In this paper, we presented a methodology for defining and communicating knowledge about project objects, as a part of the overall knowledge about design and field operations, in a way that can effectively be utilized by the project participants to coordinate their decision making. The methodology will improve productivity in the construction industry by facilitating effective coordination and communication across various disciplines involved in the project, improving constructibility by applying construction decisions early in the design phases, reducing errors in design, developing a model of intelligent interfaces, and advancing automation.

6. REFERENCES

Chan, W.T., 1986. *Logic Programming for Managing Constraint-Based Engineering Design,* Thesis submitted to Stanford University in partial fulfillment of the requirements for the degree of Ph.D., March.

Chan, Weng-Tat, and Boyd C. Paulson, Jr., 1987. "Exploratory Design Using Constraints," *Journal of Artificial Intelligence in Engineering Design and Management,* Vol. 1, No. 1, December, pp. 59-71.

Chan, Weng-Tat, and Boyd C. Paulson, Jr., 1988. "An Integrated Software Environment for Building Design and Construction," *Proceedings of the Symposium on Microcomputer Knowledge-Based Expert Systems in Civil Engineering,* Hojjat Adeli, Ed., Spring Convention, ASCE, May 9-11, pp. 188-202.

de Kleer, J., 1986 a. "An Assumption-based TMS," *Artificial Intelligence*, vol. 28, pp. 127-162.

de Kleer, J., 1986 b. "Extending the TMS," *Artificial Intelligence*, vol. 28, pp. 163-196.

de Kleer, J., 1986 c. "Problem Solving with the TMS," *Artificial Intelligence*, vol. 28, pp. 197-224.

de Kleer, J., and Brown, J. S., 1986. "Theories of Causal Ordering," *Artificial Intelligence*, vol. 29, pp. 33-61.

Howard, H. C., R. E. Levitt, B. C. Paulson, Jr., J. G. Pohl, and C. B. Tatum, 1989. "Computer-Integration: Reducing Fragmentation in AEC Industry," *Journal of Computing in Civil Engineering,* Vol. 3, No. 1, January, pp. 18-32.

Iwasaki, Y., Simon, H. A., 1986. "Causality in Device Behavior," *Artificial Intelligence*, vol. 29, pp. 3-32.

Lansky, A. L., 1988. "Localized Event-Based reasoning for MultiAgent Domains," *Technical Note 423*, Artificial Intelligence Center, SRI International, January.

Marshall, R.D., et al, 1982, *Investigation of the Kansas City Hyatt Regency Walkways Collapse*, Technical report Science Series 143, National Bureau of Standards, Washington, D.C., May.

Serrano, D., 1987, *Constraint-management in Conceptual Design*, Sc.D. Thesis, Massachusetts Institute of technology, December.

Shoham, Y., 1987. *Reasoning about Change: Time and Causation from the Standpoint of Artificial Intelligence,* Ph.D. Thesis, Dept. of Computer Science, Yale University, New Haven, Connecticut.

Stefik, M., 1980. *Planning with Constraints,* Ph.D. dissertation, Dept. of Computer Science, Stanford University, Stanford, California.