CONCOORD: A Framework for Design, Management and Coordination in a Collaborative AEC Environment

by

Hossam El-Bibany and Boyd C. Paulson, Jr.

TECHNICAL REPORT
Number 62

January, 1992

Stanford University

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

ConCoord: A Framework for Design, Management and Coordination in a Collaborative AEC Environment

Technical Report # 62
Center for Integrated Facility Engineering
Stanford University

Hossam El-Bibany
and
Boyd C. Paulson, Jr.
Construction Engineering and Management
Stanford University

TABLE OF CONTENTS

ABSTRACT

This report describes a CONstraint-based design, management and COORDination framework (**CONCOORD**) for collaborative AEC (Architecture Engineering Construction) software. The framework offers fundamental changes to the way that project design and management computer systems are built and used. The main goal is to define a new framework for integrated project management systems that is flexible enough to accommodate the various tasks required to create the product (facility) under a unified representation and reasoning methodology. The unified methodology could serve as the basis for robust knowledge integration, information processing, and coordination among the various participants over the life-cycle of the project.

The basic approach of this research is to integrate operations research models and artificial intelligence techniques to create an architecture for the proposed framework. Low-level mathematical models represent the structure and behavior of various project objects (spaces, beams, activities, resources, etc.) in a standard representational model (mathematical constraints). Mathematical equalities and inequalities also represent project constraints (requirements) among the attributes of the various objects. The architecture handles parametric as well as logic constraints in a standard representation and inference mechanism. A sophisticated constraint-management methodology coupled with interval-based mathematics is the basis for model integration as well as the quantitative and qualitative reasoning capabilities of the framework. Formalization and categorization of domain knowledge, as related to the framework, serve as the core of future metaknowledge modules to help in setting the system in various organizations.

Project participants interact with the system independently to create new project objects, browse through the system's knowledge, add or delete constraints on the project objects and perform efficient *what-if* queries.

The advantages of such an approach include the following:

- Coordinate human players over the life-cycle of the project (e.g., avoid decision conflicts);
- Serve as a computer-based framework for horizontal and vertical integration, process and product coordination, knowledge integration and exchange, optimization through user interaction, and multidisciplinary education;
- Create a change history for future use and liability tracking;
- Provide a basis for evaluating proposed changes;
- Provide a technical vehicle to achieve constructibility benefits.

PROJECT DESCRIPTION

A. Introduction and Project Summary

This research deals with the use of computers to support collaboration over the life-cycle of a project in the AEC (Architecture Engineering Construction) industry. It assumes that process coordination and human interaction are the basis for collaboration. The main objective is to provide collaborative information systems with five main capabilities:

- Coordinate human participants over the life-cycle of the project.
- Serve as a knowledge integration and exchange tool.
- Avoid decision conflicts that can lead to failures.
- Create a change history for future use and liability tracking.
- Provide a basis for evaluating proposed changes.

Computers have long supported various tasks over the life-cycle of the project (e.g., architectural modelling, structural design, project scheduling). For certain tasks they are generally indispensable, while in other areas their shortcomings often overshadow their advantages [Skibniewski 90]. A great deal of research is ongoing into integration between various computer tools for the purpose of collaboration among the various project participants. Most try to achieve this integration through knowledge-based support relying on various software architectures (e.g., blackboard architecture, cooperative distributed problem solving architectures, etc.) [Levitt et al 91]. However, the nature and project organization of the AEC industry makes it almost impossible to just rely on knowledge-based support without standard representation.

This research offers fundamental changes to the way that project design and management computer systems are built and used. The main goal is to define a new framework for integrated project management systems that is flexible enough to accommodate the various tasks required to create of the product (facility) (e.g., computer aided design (CAD), finite element analysis, project organization, planning and control) under a unified representation and reasoning methodology. The unified methodology serves as the basis for robust knowledge integration and information processing among the various project participants over the life-cycle of the project.

The basic approach of the reported research is to integrate operations research models and artificial intelligence techniques to create an architecture for the required integrated system. Low-level mathematical models represent the structure and behavior of various project objects (spaces, beams, activities, resources, etc.) in a standard representational model. Mathematical equalities and inequalities also represent project constraints (requirements) among the attributes of the various objects. A sophisticated constraint-management methodology coupled with interval-based mathematics is the basis for model integration as well as the quantitative and qualitative reasoning capabilities of the framework. Formalization and categorization of domain knowledge, as related to the framework, serve as the core of future metaknowledge modules to help in setting the system in various domains.

This research's main focus is to provide an overall conceptual design and overall architecture for such an approach and to test the approach by developing a prototype system to perform the integrated tasks over the life-cycle of the project.

Reader's Guide.

Section B outlines the research's specific objectives. The present state of knowledge in related fields is presented in Section C. Section D describes the basic approach for pursuing the research objectives and outlines the developed framework. Section E illustrates two AEC tasks formulated as constraint-management problems. Finally, section F describes the significance of the work.

B. Objectives of the Research

The broad objective of this research is to provide collaborative information systems with the main goals to coordinate human participants performing concurrent activities over the life-cycle of an AEC project.

The specific objective of this work is to develop a **CON**straint-based design, management and **COORD**ination framework (**CONCOORD**) that describes a new approach to integrated AEC software systems. This new approach is expected to offer improvements (see Section E) in accordance with the broad research objectives.

Another research objective is to design a software architecture with its corresponding representation standards to support the development of integrated AEC software with capabilities ranging from space (3-D) modelling in architecture, to construction management activities (e.g., planning, scheduling and operation simulation).

C. Present State of Knowledge in Related Fields

The topics of constraint management and cooperative behavior in computer systems are related to research occuring in several main fields: (1) computer science research on simulation, planning and design; (2) civil engineering applications on design, management and integration; (3) artificial intelligence research on reasoning for multi-agent domains; (4) manufacturing research on design and process planning. This section outlines some of these fields. For a complete review of related work on constraints, the reader is referred to [Serrano 87, Serrano and Gossard 88].

Computer Science Research on Simulation, Planning and Design.

[de Kleer and Sussman 78] and [Stallman and Sussman 77] used constraint-based representations to analyze and synthesize electrical circuits. Their work introduced the idea of dependency-directed backtracking to correct contradictory confluences, i.e., different values for the same parameter proposed by different constraints. The work implemented in Thinglab [Borning 79] introduced the idea that constraints could not only check descriptions but propose new ones that satisfy the constraints. It had no symbolic algebra capabilities, dealt only with equalities and used relaxation when simple techniques failed. [Stefik 80] developed a hierarchical planner, MOLGEN, to aid in the design for molecular genetic experiments. It propagates constraints arising at different levels of planning abstraction to generate plans for gene-splicing experiments. [Sussman and Steel 80, Zima 85] developed constraint languages. Most of the recent work on constraints has been influenced by the work of the pioneers mentioned above.

Civil Engineering Applications on Design, Management and Integration.

[Levitt et al. 91] review the research on and architectures for knowledge-based support for automating concurrent multidisciplinary design. [Holz 82] represents design constraints as algebraic inequalities and uses algebraic simplification to derive the values of certain designable items in terms of given parameters. [Garrett and Fenves 87] presented a design strategy in which structural components are designed automatically by applying different types of knowledge. Other closely related work is that by [Rasdorf and Fenves 86] and [Rasdorf et al 87] on structural design constraints maintained in a relational database. Related work on the use of "Constraint Logic Programming" (CLP) for representing design constraints and processing them within the single framework of logic can be found in [Lakmazaheri and Rasdorf, 89a, 89b].

[[Fenves et al. 89] report the IBDE environment for integrated design and construction planning of buildings. The environment combines a number of knowledge-based expert systems developed at CMU. A Blackboard is used to coordinate communication between the different knowledge systems by means of messages and data.

DICE is a distributed and integrated environment for computer-aided design and construction being developed at MIT [Sriram et al. 89]. DICE can be envisioned as a network of computers and users where several knowledge modules communicate and coordinate through a Blackboard by a control mechanism. Each knowledge module can be viewed as either a knowledge-based expert system, a CAD tool, a user of a computer, or a combination of the above.

A wide variety of object-oriented and knowledge-based project management software projects have much to offer for suggesting representation approaches for project information. These include [Darwiche et al 89], [Levitt & Kunz 87], [Ito et al 89], [Axworthy and Levitt 89], [Kartam 89], [Tommelein 89], [Waugh 90] at Stanford and [Logcher 87], [Cherneff et al 89], [Abu-Hijleh and Ibbs 90], [Best & Inkpen 90], and [Kellett et al 90] elsewhere.

Artificial Intelligence Research on Multi-agent Planning and Problem-solving.

Although most of the research points of the proposal are related to research topics in AI, the main AI topic that closely relates to the architecture proposed in this research is reasoning for multi-agent domains and distributed problem solving. As an example, [Lansky, 85, 88] proposed an architecture GEM (the Group Element Model) as a behavioral model for concurrent action. GEM was used to build GEMPLAN, an AI planning system which is quite different from traditional planners in at least two respects: (1) the variety of domain constraints, and (2) the way in which constraints are applied. The order in which GEMPLAN applies constraints is guided both by the domain structure and a set of tailorable regional search tables. Each region is associated with a constraint-satisfaction search tree that builds a local plan for that region. The resultant search structure is thus extremely flexible.

Manufacturing Research on Design and Process Planning.

[Krishnan et al. 90] use a constraint-management methodology similar to the one proposed here to manage concurrent mechanical design under one organization. The Edinburgh Designer System (EDS) [Popplestone 84,87] provides a high level of integration of knowledge about function and form for mechanical engineering design. PRIDE, an expert system for the design of paper handling subsystems in copiers was developed by [Mittal et al 86]. PRIDE uses data-directed dependency backtracking to do hierarchical refinement. [Araya and Mittal 87] show how design plans can be generated using an artifact's description and problem solving heuristics. [Frayman and Mittal 87] describe a constraint-based expert system, COSSACK, for configuration tasks to be used as a tool to understand the configuration task as a generic problem solving activity. [Ramchandran & Shah, 88] proposed a model for mechanical design based on refinement, constraint propagation and parameter selection. The main type of constraints they have dealt with is the constraint knowledge that decides what are the legal moves in a search space depending on the state of the problem. This knowledge is stored in the form of dependency graphs. [Smith 88] described a framework for reactive factory management based on scheduling constraints. [Meunier & Dixon, 88] presented a hierarchical computational model for parametric mechanical system design. In this model, system managers solve their assigned problems by specifying subproblems to, and by coordinating communication between, subordinates. When the subsystem solutions are returned, the manager integrates them into a whole, evaluates the resulting system, and prepares new specifications for subordinates iteratively as needed until an acceptable result is obtained. This model of cooperation assumes that the design problem has been decomposed into systems and subsystems a priori. A model for cooperation between different project parties, as proposed here, cannot practically be built on this assumption. [Dixon & Simmons, 85, Dixon, 86] have also proposed a model of the design process based on iterative respecification. The respecification model does not allow for information to be passed between subproblems. Constraints are only propagated up and down the hierarchy. [Stefik et al 82] have developed a knowledge-based system for circuit design that takes a user through several distinct levels of abstraction. Each level of abstraction considers only certain details.

D. Overview of the Research Approach and the ConCoord Framework

As mentioned above, our overall objectives were to define a framework and design the corresponding standard architecture for integrated AEC software. The framework considers the nature of the AEC industry, its market and organizational requirements. Section E shows that this will achieve coordination through knowledge integration as well as define methodologies to increase the software's intelligence potential, and greatly improve the level of integration among applications.

This section will describe the proposed approach in greater detail, discuss the role of constraint-management and outline the framework for the integrated software.

Factors behind the Choice of Constraint-Management as Basis for Coordination

The design of information systems in general should reflect the needs of the organization operating in its specific industry. Specific AEC industry organizational factors call for special needs in the design and management of collaborative information systems. In the first phase of the research [El-Bibany et al, 91], we have identified these needs, studied various collaborative information systems models developed for other industries, as well as designed a more generic model (Figure 1) that fits these needs.

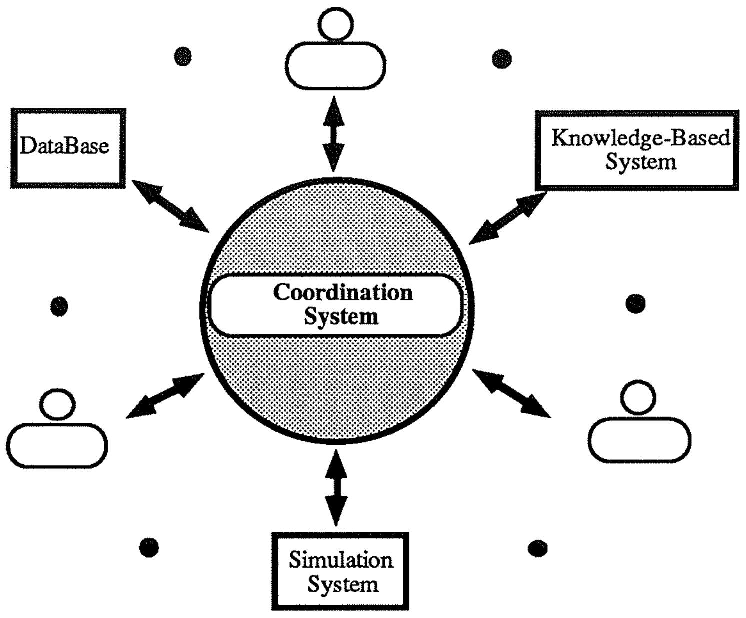


Figure 1 - Coordination System Concept

In this model a generic and flexible coordination mechanism is the core of the system, facilitating interaction between all types of agents — human, machine, databases, knowledge-based systems, simulation systems, etc. A central project database is no longer considered the focus of the system. Databases are merely agents that contribute to the constraints imposed on the overall problem. The coordination system will ultimately be much more flexible, but will require a very low level representation of knowledge. To achieve this flexibility in a standard representation, the main reasoning technique should handle various types of AEC problems in the same manner. Constraint-based reasoning will achieve this goal [Chan and Paulson 87, 88, Serrano 88, Lansky 88].

Constraint-management was also chosen based on the following:

- the flexibility to accommodate the various design and management tasks
- the existence of mathematical models for various tasks
- domain-independent flexible methodology
- The ability to model the overall problem in the form of real life objects and constraints representing their behavior and relationships

A Constraint-based 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, which may be viewed as constraints, 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 (analysis and specification constraints). 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. Figure 2 illustrates other examples of project objects and project constraints.

Examples of project objects:

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

Examples of sources of project constraints:

Architectural Design	- 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 Construct	- maximum concentrated construction 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

Figure 2- Examples of Project Objects and Constraints

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.

Constraint-Management Methodology

Once the problem is formulated in terms of mathematical constraints, a constraint analysis and solution 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 (any predefined model would constrain the capabilities of the system to the extent that general coordination would not be feasible). 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 the proposed research, we use standard graph algorithms for solving this problem. The major advantages of such representation are: 1) it is a very general task and domain independent representation, 2) it can analyze arbitrary sets of constraints, and 3) it allows both qualitative and quantitative reasoning.

The graph-based representation for constraints and the problem-solving strategy using constraint networks that we are using in this research is based on the work described in [Serrano 88]. We enhance Serrano's work by integrating interval-based mathematical capabilities. In Section E, we illustrate the methodology as applied to design and construction planning and scheduling.

The power of the proposed constraint-based integration engine is that the machinery for updating and adapting to the changes is built-in. An important change will propagate through the system and different parties would then be notified of any problem and be queried for solutions (e.g., relaxing one of their constraints, if possible).

In our constraint-based framework, we make the distinction between the constraints (which are the desired relationships) and the means of enforcing these constraints. The mechanism of enforcing constraints is invariant across different problems; only the constraints differ from one problem to the next. This dichotomy proves useful because, in theory and as proved by the current status of this research, we can concentrate on the correct formulation of a problem in terms of constraints, assuming that the computational mechanism has been correctly implemented [Chan 86]. We use this property to create standards for constraint-based representation.

Core of the Framework: Objects and Constraints in a Standard Representation

The framework is based on the fact that coordination can only be achieved through integration of knowledge about, and various requirements imposed on, project *objects*. Project participants look at the project in terms of individual objects (spaces, doors, beams, pipes, formwork, cranes, compressors, pumps, locations, activities, etc.). There are two things that we want to capture: the knowledge that is encapsulated within the objects, and the relationships among the objects. Both may be expressed in the form of object internal constraints (representing the internal structure and behavior of an object) and object-to-object constraints (representing the relationships among objects).

Based on this view of the world, the structure of a system that uses knowledge about objects in the form of constraints is envisioned as shown in Figure 3. 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.

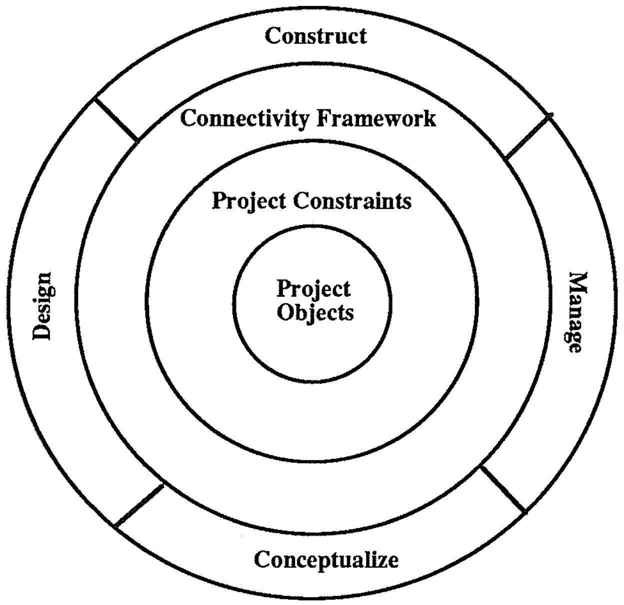


Figure 3 - Framework for Coordination

For the framework to reflect the AEC organizational needs, it should allow the creation of new types of objects in a standard format. The **representational standards** and the **CONCOORD architecture** are documented in [El-Bibany 92].

Once the system is set with specific object classes, project participants interact through the instantiation of project specific objects and imposing the object-to-object constraints (through domain and project specific logic) to represent the required relationships. The framework translates their actions into the underlying constraint-based representation for analysis and solution.

As participants propose changes, new information requirements, constraints are automatically propagated to other participants who need to know about them, while maintaining the constraints that were previously specified and remain unchanged. Design parameters are automatically adjusted to satisfy the new requirements if possible; if not, all parties concerned are notified immediately so that negotiations can proceed to find a mutually agreeable solution. The system flags conflicts and assists in determining the cause of problems, allowing participants to focus their attention on solving the right problem in a better way. Users can query the integrated object knowledge base for the existence of certain relationships between the attributes of different objects and evaluate the global impact of proposed design changes. Advanced qualitative reasoning capabilities may also be supported by the constraint-management system.

The Framework

The main contributions of the framework to be presented in Section F are based on the following capabilities:

Unified Representation and Reasoning. The framework is based on one uniform constraint-management methodology for various tasks based on the argument that computers can look at various tasks in the same way using mathematics and logic.

External Knowledge Access Capabilities. Every object in the framework has the ability to access external information (external databases, knowledge-based systems, etc.) under a flexible graded-reachability architecture.

User-Controlled, Flexible Representation. To fit the AEC industry needs for dynamic project organization and flexible, free-to-define task and task integration methodologies, the framework provides system managers (first level of users) with the ability to define prototypes of their own objects with their behavior. Day-to-day users (second level of users, e.g., designers, planners, etc.) would interact with the system instantiating project-specific objects and their logical constraints. The constraint-management system would handle interaction and information management.

Context-Based Reasoning. Context-based reasoning and change-management to support life-cycle integration will be based on extensive truth-maintenance capabilities [de Kleer 86 a, b, c, Shoham 87]. Internal truth maintenance will control dependency-directed propagation. An external truth maintenance system will control context-based reasoning.

Customized User Interfaces. The framework is a decision support tool based on the assumption that unconstrained user interaction is the basis for coordination. Customized user interfaces are therefore supported.

Formalization of Representation. For a general representation, we differentiate between domain and non-domain knowledge. Furthermore, to apply the framework in different organizations with different requirements, specification, categorization and formalization of domain control knowledge representation become the basis for initialization of the system. In the proposed architecture, we represent domain objects and their behavior in a non-domain specific mathematical and parametric representation. Object relationships and their implications are categorized and formalized with respect to this representation.

Metaknowledge Control. The formalization of representation is envisioned as the basis for future metaknowledge modules (in the form of a knowledge-based system) with the main task of helping system managers define the objects and their required interaction in various organizations.

Prototype's Implementation Environment

Object-Oriented Logic Programming is the primary tool for prototype implementation. Prolog^{++} on a MacII provides us with the object-oriented programming environment based on advanced logic programming techniques. It proved its power in most of the utilities that we need for this research [Lansky 88].

Prolog as a logic programming language has a sound theoretical basis, being modelled on first-order predicate calculus. As a declarative type-free language with built-in unification and inference engine, Prolog results in an increase in both productivity and creativity of the programmer. Its type-free capabilities increase its capability for meta-programming with ease of updating and editing the programs. Ongoing theoretical research [Cohen 90, Colmerauer 90] will render Prolog the most capable language for handling constraint-management.

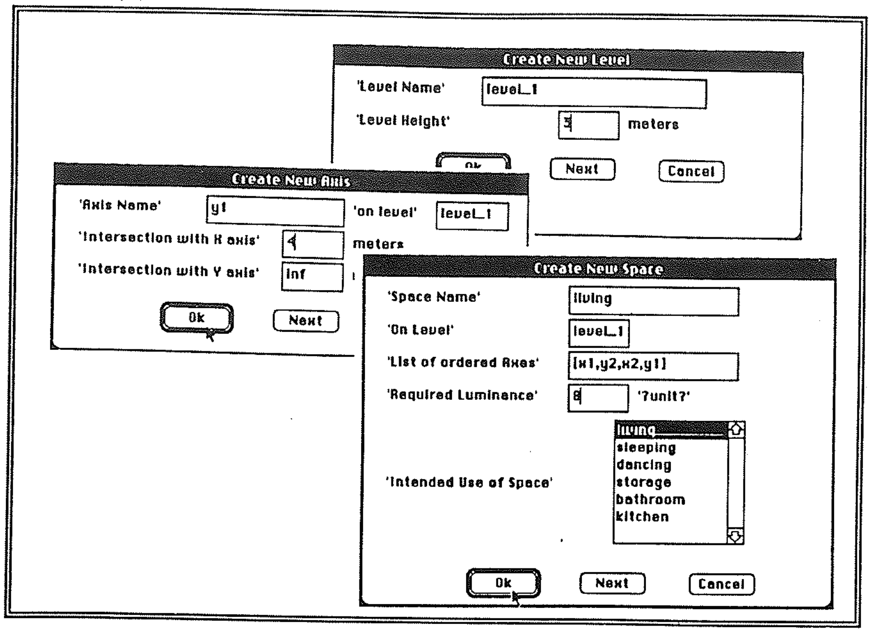
Prolog imposes few restrictions on the structure and organization of a program. This can lead to problems when constructing large applications. The use of Prolog^{++} as an object-oriented programming environment (class hierarchies, abstraction, encapsulation, message passing, polymorphism, and inheritance) helps programmers impose discipline for organizing and managing their code as well as increase their productivity. Analysis of advantages of object-oriented programming may be found in [Cox 86], [Booch 91].

Figure 4 shows sample screens from the current tests of the prototype in various fields. The upper block illustrates architectural design dialogues creating levels, axes and spaces. The created space in this case is a living room with a specific usage and luminance requirements (a parametric or database model relating luminance to volume of the space and sources and sinks of light is needed in this case). The middle block illustrates sample interaction by the structural engineer creating a beam in a certain location. Other structural elements may be created in the same manner. Notice that the user has the option to set the direction of information flow at any time based on the constraint-management methodology previously described. In this case, the user chooses to perform normal design by entering values for the concrete flexural strength and the thickness of the beam (actually in this case the system searches default values, e.g., the default value for beam thickness is the wall thickness under the beam defined by the architect). The lower block illustrates sample construction planning and scheduling interaction with the system. The user is still in control of information flow by defining the known parameters of the activity *construct_beam112*.

E. Examples

The constraint-management methodology used as the basis for the CONCOORD framework is based on the graph theory [Serrano 88]. Constraint networks are presented as directed graphs, where the nodes represent parameters and arcs represent constraint relationships. Parameter dependencies are generated automatically. Constraints can be continually added, deleted and modified throughout the evolving design of a new artifact.

This section illustrates the constraint-management methodology as applied to two types of examples suitable for CONCOORD. The first concentrates on decisions made mainly at the design stage, with some input from construction. The second focuses on construction planning and scheduling.



Create New Level

'Level Name' level_1

'Level Height' 3 meters

Next Cancel

Create New Axis

'Axis Name' y1 'on level' level_1

'Intersection with X axis' 4 meters

'Intersection with Y axis' inf

Ok Next

Create New Space

'Space Name' living

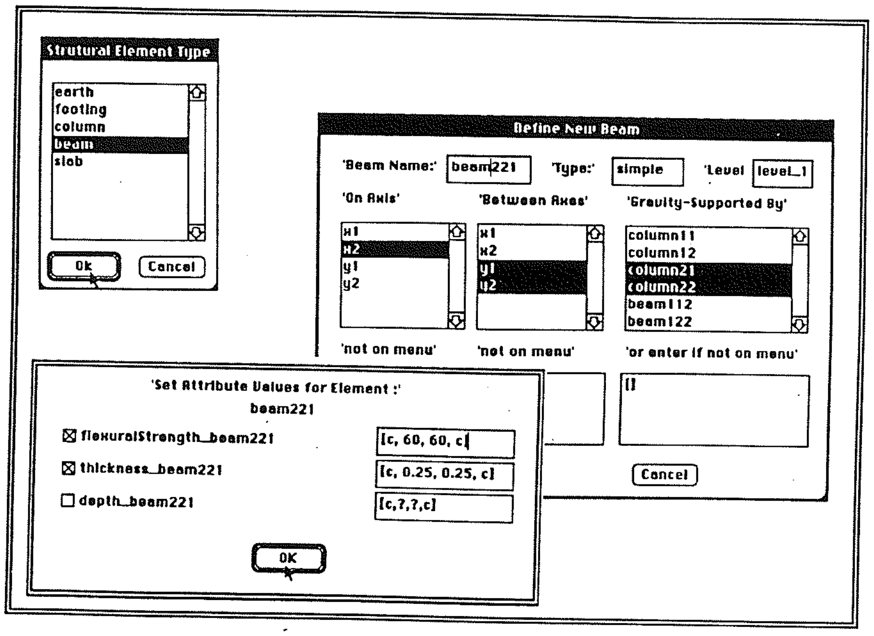
'On Level' level_1

'List of ordered Axes' [x1,y2,x2,y1]

'Required Luminance' 8 '?unit?'

'Intended Use of Space' living, sleeping, dancing, storage, bathroom, kitchen

Ok Next Cancel



Strutural Element Type

earth, footing, column, beam, slab

Ok Cancel

Define New Beam

'Beam Name:' beam221 'Type:' simple 'Level' level_1

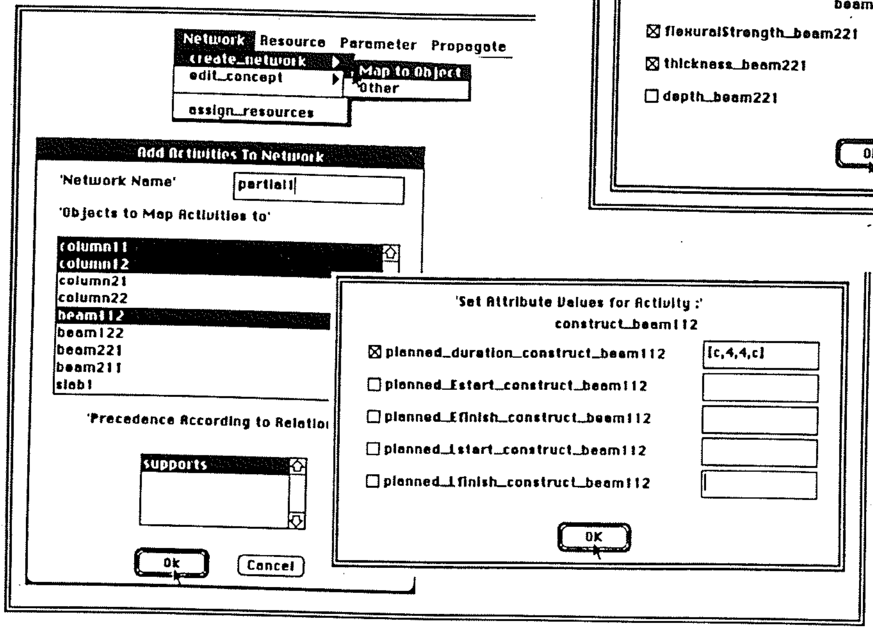
'On Axis'	'Between Axes'	'Gravity-Supported By'
x1	x1	column11
x2	x2	column12
y1	y1	column21
y2	y2	column22
		beam112
		beam122
'not on menu'	'not on menu'	'or enter if not on menu'
		[]

Cancel

'Set Attribute Values for Element :'
beam221

☒ flexuralStrength_beam221	[c, 60, 60, c]
☒ thickness_beam221	[c, 0.25, 0.25, c]
☐ depth_beam221	[c,?,?,c]

OK



Network Resource Parameter Propogate

create_network ▸ Map to Object / Other

edit_concept ▸

assign_resources

Add Activities To Network

'Network Name' partial1

'Objects to Map Activities to'

column11, column12, column21, column22, beam112, beam122, beam221, beam211, slab1

'Precedence According to Relatio

supports

Ok Cancel

'Set Attribute Values for Activity :'
construct_beam112

☒ planned_duration_construct_beam112	[c,4,4,c]
☐ planned_Estart_construct_beam112	
☐ planned_Efinish_construct_beam112	
☐ planned_Lstart_construct_beam112	
☐ planned_Lfinish_construct_beam112	

OK

Figure 4: Sample Screens from Prototype Testing

Example 1

In a two-story commercial building, the structural elements (e.g., footings, columns, beams and slab panels) are the main objects to be designed and constructed. We assume that, in the computer system, an object-based representation will be used to represent the attributes and relationships between these objects (e.g., column C21 supports slab panel S21, column C11 supports column C21 and slab panel S11, and footing F1 supports column C11). From this representation, we can form the equations (constraints) that constrain the values of column C21's attributes. The different parties in the project would form different constraints on C21. Figure 5 illustrates column C21 and its attributes as a part of the overall structure.

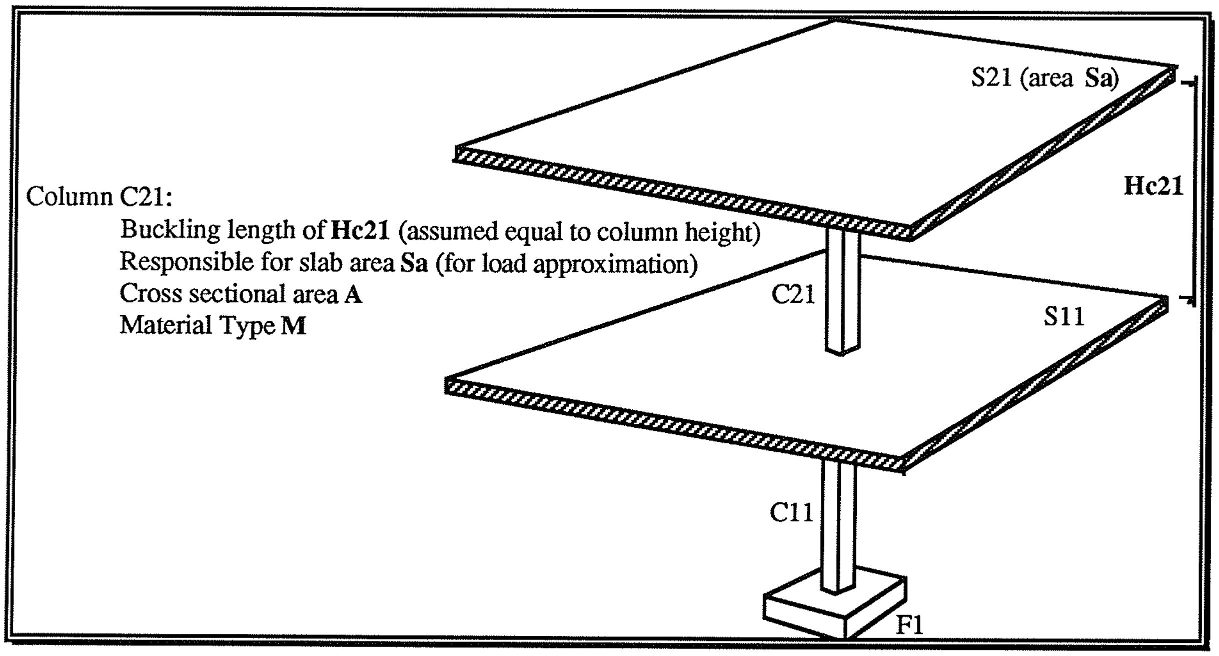

S21 (area Sa)
Hc21
Column C21:
Buckling length of **Hc21** (assumed equal to column height)
Responsible for slab area **Sa** (for load approximation)
Cross sectional area **A**
Material Type **M**
C21
S11
C11
F1

Figure 5 - Column Design in an Example Structure

We look at three different parties involved in the design and construction of this project: the architect, the structural designer and the construction manager. The scenario is as follows:

In current practice, as design starts, the architect forms some constraints on column C21, such as the type and the surface area of the slab it supports, the height of C21 and some restrictions on its cross sectional area. Taking the architectural constraints into account, the structural designer tries to define the appropriate cross section for C21. The height of C21, the type and surface area of the slab it supports (as defined by the architect) would respectively affect the buckling length, the load intensity and the total load on C21. In the construction phase, the construction manager would have constraints on the capacity of the cranes required to put C21 in place. The decisions are made sequentially, which most frequently leads to the violation of some of the constraints pertaining to one or more of the involved parties. Figure 6 illustrates sample constraints imposed by the project participants.

The erected C21 must satisfy all of these constraints. CONCOORD considers all the constraints in parallel. In this way, we can look at any party as taking the other parties' requirements into consideration in making its decision. For example, construction constraints (constructibility) are taken into account simultaneously with structural constraints as the architect is defining his requirements.

Architectural Constraints:

H = Floor Height (e.g., 3 meters) (A1)

A $\leq$ 0.02 square meters (architectural requirement) (A2)

Sa is fixed = slab area supported by C21 (depends on chosen spacing of columns) (A3)

Slab-Type = Type of Area Supported (e.g.,internal shop area for special use of owners) (A4)

Structural Constraints:

F $\leq$ w A S (S1)

F = Sa x (LI + DI) (S2)

DI = St xρ (S3)

LI = f(Slab-Type) (S4)

w = f(Hc21, A) (S5)

Hc21 = f(H) (S6)

where **F** = total force on column
A = cross sectional area of column C21
S = allowable compression stress of the material of column C21
w = factor to reduce the allowable compressive stress according to buckling length and cross sectional dimensions
LI = Live load intensity on the supported area
DI = the dead load intensity
St = the thickness of the slab supported by column C21
ρ = the material density of the slab

For our purpose, we will not go into complicated details of structural design; instead we assume that a direct factor **w** takes buckling into account. (**w** is actually a function of the height and radius of gyration of the column)

Construction Constraints:

Through the construction planning for the erection operations, the available crane stationed at the planned location would have a maximum allowable load capacity of "**CraneCap**" for putting column C21 in place. This constraint can be formulated as follows:

A H Ω $\leq$ CraneCap (C1)
where Ω = unit weight of column material

Figure 6 - Sample Constraints Imposed on a Column (C21)

If, after erection of C21, the architect changed the type of the area supported by C21 to a storage area, a higher load intensity would be generated for the structural designer. A recheck (by the structural designer) would find out if the capacity of the actual erected C21 is adequate for the change in the architectural requirements. In case of inadequate capacity, a decision must be made either by the architect to withdraw his new requirement or by the structural designer to reinforce C21.

In solving the constraints of Figure 6, a graph network is constructed (Figure 7). Nodes represent constraint objects (i.e., variables or exogenous variables appearing in the constraint set). Each arc represent a relationship between two objects through a certain constraint. Exogenous variables (enclosed by rectangles) can be defined either using domain knowledge or by the user.

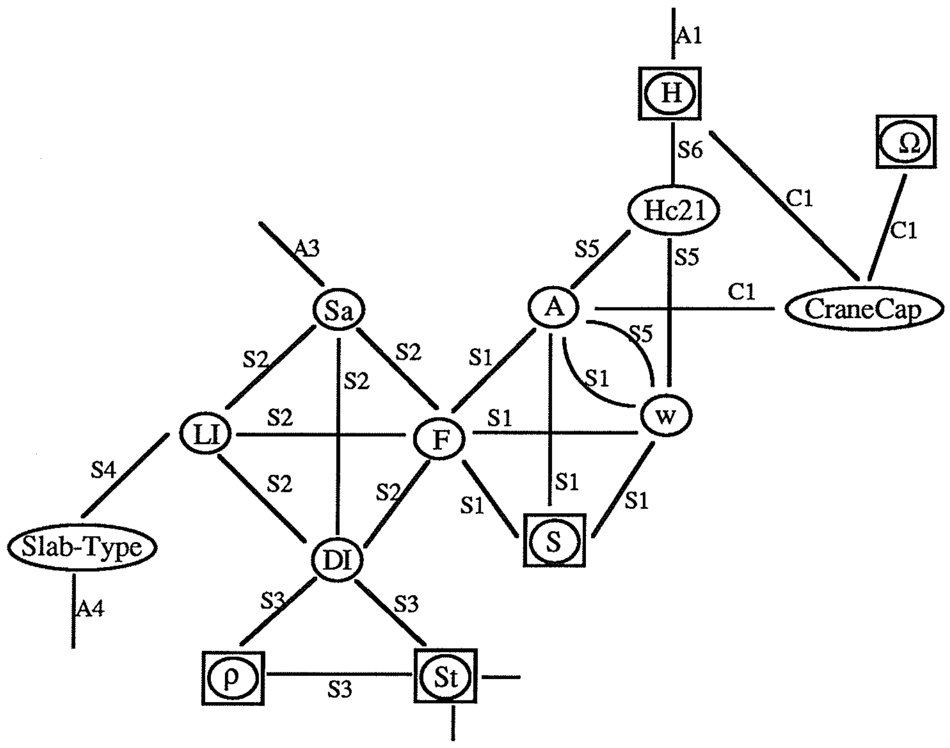


Figure 7 - Constraint Network as Non-directed Graph

A bipartite graph (Figure 8) is then constructed, the two sets of nodes being the variables and the constraints in which they appear. A maximum matching algorithm will be applied to the bipartite graph. The basic bipartite matching graph will be a good tool to decide if the constraint system is under or over constraint. The under-constraint variables and the redundant constraints can also be discovered (the figure does not show these cases).

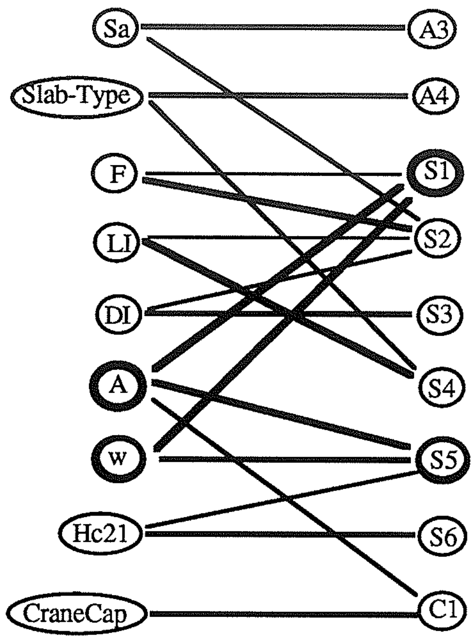


Figure 8 - A Bipartite Graph

The outcome of the bipartite matching is used to transform the original undirected graph network to a directed graph (Figure 9). This directed graph can be cyclic due to mutual dependence of some variables.

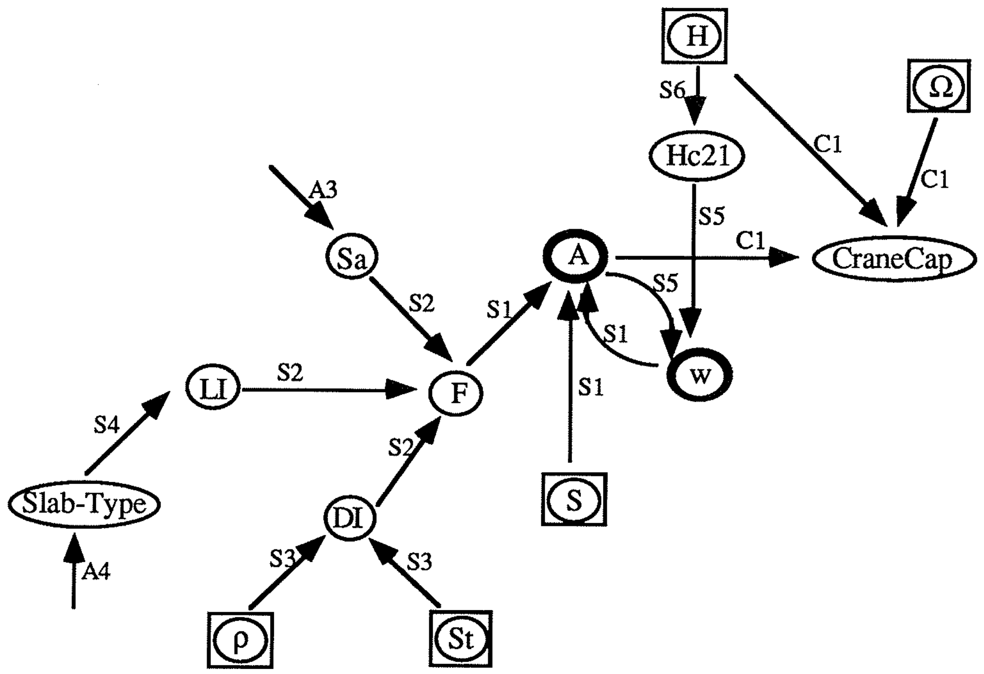


Figure 9 - Directed Cyclic Constraint Network

The bipartite matching also discovers the mutually dependent variables and their constraints. The set of mutually dependent variables will be reduced to one node (supernode) in the directed graph transforming it to a directed acyclic graph (DAG) (Figure 10). This graph is the basis for value propagation. The encountered supernodes can be solved independently either numerically or using more powerful symbolic manipulation techniques. All the graphs will represent nodes on one level of the overall truth maintenance hierarchy.

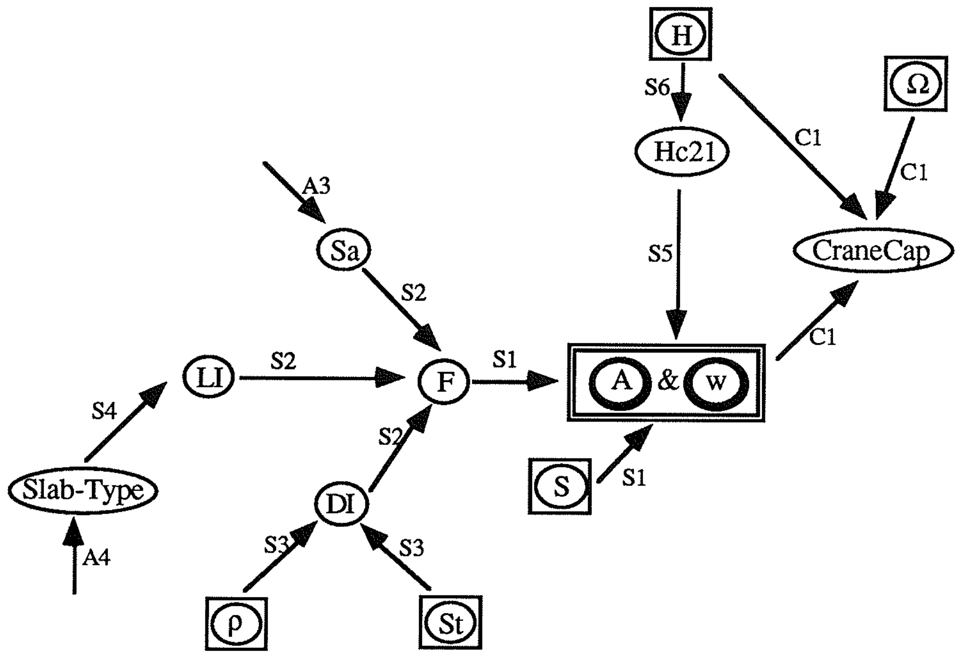


Figure 10 - Directed Acyclic Constraint Network

After collapsing the strong components, the solution plan is determined by a topological sort on the resulting directed acyclic network (Figure 10). The solution plan is used to solve the constraints for their matched variables, to obtain the causal relationships among the parameters, to explain the solution procedure to the users, to identify conflicting constraints, and as a basis for advanced sensitivity analysis and qualitative reasoning. Value propagation is handled through interval-based mathematics.

Example 2

The purpose of this example is to illustrate how the constraint-management methodology, as currently implemented, can handle a non-design problem in a similar way as a design problem once the knowledge is represented in the form of constraints.

The treatment of construction planning/scheduling as a constraint-management problem allowed us to create a strong methodology for planning/scheduling under resource constraints. We are currently extending the methodology to incorporate more construction management tasks.

Suppose we represent a construction activity in an object-oriented hierarchy with the attribute, parameters and behavioral constraints as shown in Figure 11.

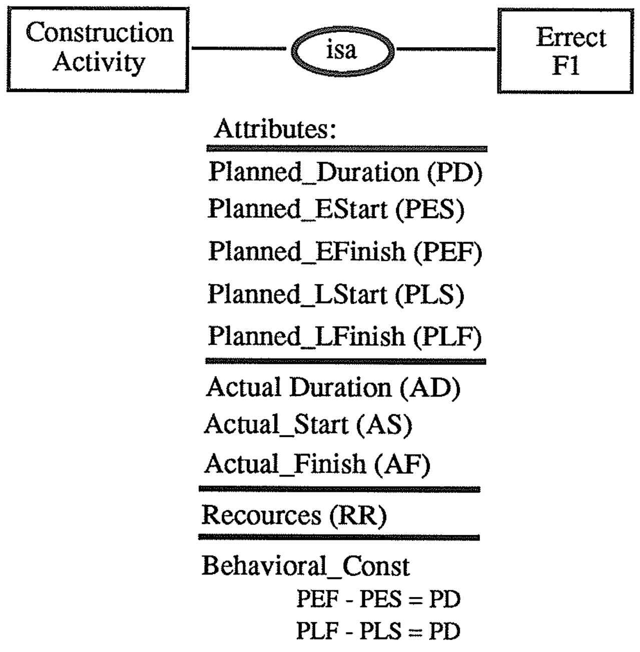


Figure 11 - Representation of a Construction Activity

Once the user creates project objects and their relationships, as the ones shown in Figure 12, The system will create a constraint set consisting of three types of constraints (Figure 13).

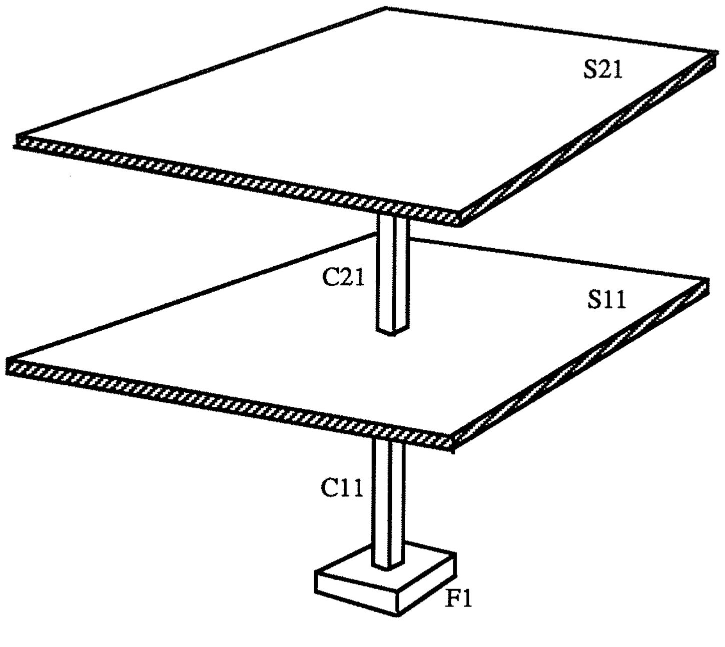
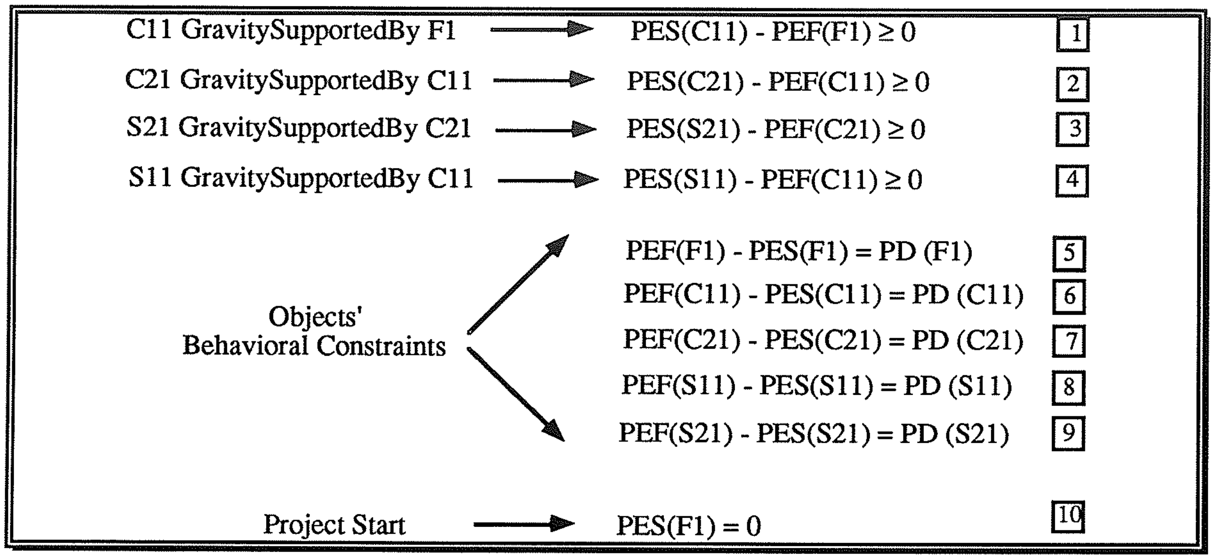


Figure 12 - Example Structure for Planning and Scheduling



C11 GravitySupportedBy F1 $\longrightarrow$ $PES(C11) - PEF(F1) \geq 0$ $\boxed{1}$

C21 GravitySupportedBy C11 $\longrightarrow$ $PES(C21) - PEF(C11) \geq 0$ $\boxed{2}$

S21 GravitySupportedBy C21 $\longrightarrow$ $PES(S21) - PEF(C21) \geq 0$ $\boxed{3}$

S11 GravitySupportedBy C11 $\longrightarrow$ $PES(S11) - PEF(C11) \geq 0$ $\boxed{4}$

Objects' Behavioral Constraints

$PEF(F1) - PES(F1) = PD\ (F1)$ $\boxed{5}$

$PEF(C11) - PES(C11) = PD\ (C11)$ $\boxed{6}$

$PEF(C21) - PES(C21) = PD\ (C21)$ $\boxed{7}$

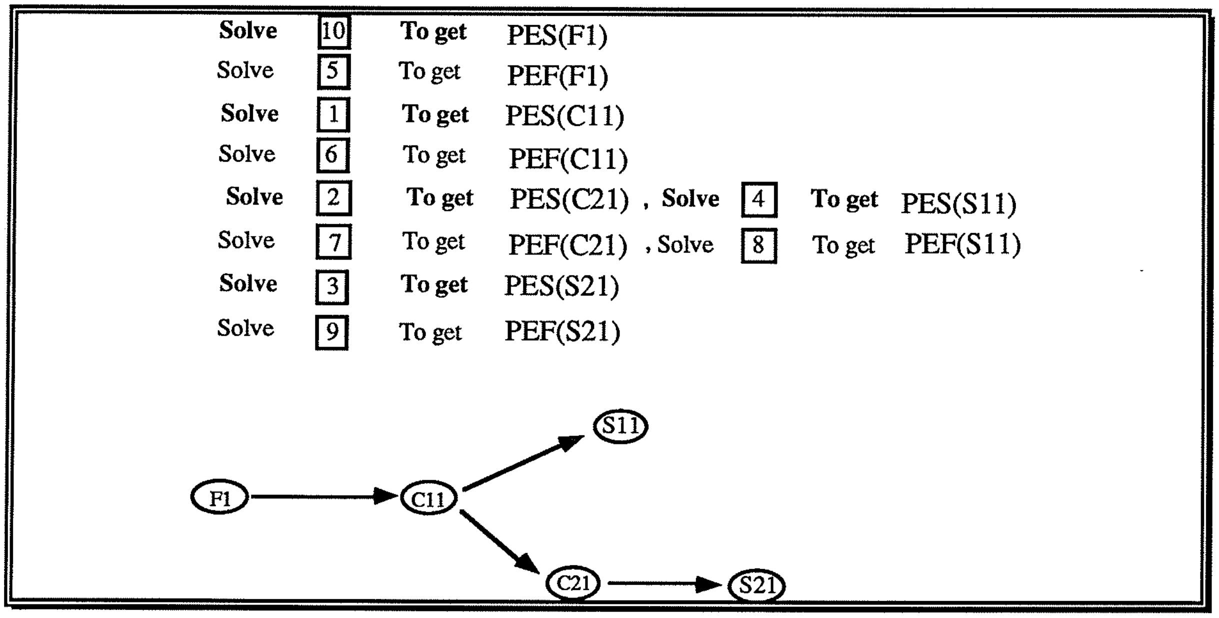
$PEF(S11) - PES(S11) = PD\ (S11)$ $\boxed{8}$

$PEF(S21) - PES(S21) = PD\ (S21)$ $\boxed{9}$

Project Start $\longrightarrow$ $PES(F1) = 0$ $\boxed{10}$

Figure 13 - Planning/Scheduling Constraint Set

Notice that constraints 1 to 4 represent internal behavior of the created activities and will be created whenever a construction activity is created. Constraints 5 to 9 represent precedence according to certain relationships between the corresponding objects. For example the user may choose to create these constraints based on the relationship "supported_by" between these structural objects. The system creates these constraints automatically. Notice also that constraint number 10 in the set represent a boundary constraint (in which PES(F1) is considered an exogenous variable for the constraint set) for the whole set and scheduling may not proceeded unless constraint 10 is present.



Solve $\boxed{10}$ **To get** PES(F1)

Solve $\boxed{5}$ To get PEF(F1)

Solve $\boxed{1}$ **To get** PES(C11)

Solve $\boxed{6}$ To get PEF(C11)

Solve $\boxed{2}$ **To get** PES(C21) , **Solve** $\boxed{4}$ **To get** PES(S11)

Solve $\boxed{7}$ To get PEF(C21) , Solve $\boxed{8}$ To get PEF(S11)

Solve $\boxed{3}$ **To get** PES(S21)

Solve $\boxed{9}$ To get PEF(S21)

$$\begin{tikzcd}
 & & S11 & \\
F1 \arrow[r] & C11 \arrow[ur] \arrow[dr] & & \\
 & & C21 \arrow[r] & S21
\end{tikzcd}$$

Figure 14 - Plan of Solution as Construction Plan

Application of the constraint-management methodology as described in Example 1 will result in a plan for solving the constraints as shown in the upper part of Figure 14. The corresponding constraint network represents the construction plan as shown in the lower part of the figure.

The construction schedule is then computed by delivering the plan in Figure 14 to the interval-based manager to propagate interval values.

F. Significance of the Research

Advantages of the CONCOORD Framework

The CONCOORD framework for integrated AEC software offers the following advantages:

1. **Horizontal Integration.** The framework is flexible enough to accommodate various tasks during the creation and maintenance of the facility in a unified methodology for facilitating their coordination (flagging source of conflicts, suggesting and tracking changes, what-if analyses) across various disciplines.
2. **Vertical Integration.** The framework incorporates mechanisms for change-management and context reasoning for the purpose of integration and coordination over the life-cycle of the project..
3. **Process and Product Coordination.** Process coordination is the management of the concurrent activities during the creation of the facility. Sound process coordination (e.g., identifying decision conflicts) requires product coordination to ensure that the final product satisfies the requirements of various project participants.
4. **Knowledge Integration and Information Processing.** The framework creates a robust model for knowledge integration (rather than data integration) and information processing based on an architecture that reduces knowledge items to their fundamental representation (mathematical and parametric models).
5. **Optimization through User Interaction.** The framework facilitates global optimization by enabling individual participants to evaluate the impact of their decisions on overall project goals. In this framework, the constraint-management methodology provides powerful and transparent *what-if* capabilities for both quantitative as well as qualitative reasoning.
6. **Multi-disciplinary Education.** With the previous capabilities, the framework may be used in education and learning (e.g., by project team members or in educational institutions).

The results of this research also provide computer systems with methodologies to handle the automatic update of changes and hence affect the evolution of design as well as project management systems. Some of the practical applications of the methodologies defined under this research would be:

1. **New generation of design and project management systems.** In this category we expect to see a first generation of integrated design and project management systems with the main themes of coordination and flexible standards' processing. In the AEC industry, for example, an international company would manage their projects much more effectively; design can be done locally and automatic accounting for different international standards as well as local requirements would be handled by the system. This will decrease the cost of managing these projects and hence increase the competitive advantage of the user companies.

2. **Intelligent CAD systems.** Applying the results of this research to the current CAD/CAM systems, through the use of the domain knowledge, would increase their capabilities and flexibility to handle intelligent design (functional requirements) and update. For example, as the user defines new objects for the system, the system will be able to design the beam automatically and ask the user to approve the design. Connections between these objects (e.g., beam column connection) would be automatically designed if the requirements for these connection were defined as objects.

3. **Intelligent Database systems.** Database systems may use the underlying methodology to automatically propagate any change to update all related information efficiently and answer more intelligent queries with a powerful explanation facility.

G. Bibliography

Abu-Hijleh, Samer, F. and Ibbs, C.W., 1990, "An Intelligent Exception Reporting System," presented at the *11th International Cost Engineering Congress and 6th AFITEP Annual Meeting,* Paris, France, April.

Araya, A., and Mittal, S., 1987, "Compiling Design Plans from Descriptions of Artifacts and Problem Solving Heuristics," *Proc. IJCAI-87*, Milan, Italy, July.

Axworthy, A. and Levitt, R., 1989, "An Object-Oriented Approach for Scheduling Construction Projects," *Sixth International Symposium on Automation and Robotics in Construction (6th ISARC),* San Francisco, USA, June, pp. 325-331.

Best, J.T. and Inkpen, C.J., 1990, "Resource and Activity Simulation in a Knowledge-Based Planning System," *First International Conference on Expert Planning Systems*, Institution of Electrical Engineers Conference Publication No. 322, Brighton, UK, June, pp. 119-123.

Booch, Grady, 1991, *Object Oriented Design with Applications*, Redwood City CA: Benjamin Cummings.

Borning, A., 1979, *ThingLab- A Constraint-oriented Simulation Laboratory,* Ph.D. thesis, Department of Computer Science, Stanford University, California.

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., 1988 Spring Convention, ASCE, May 9-11, pp. 188-202.

Cherneff, J., Logcher, R., and Sriram, D., 1989, "Automating Schedule Generation From Architectural Drawings," Proc. of ASCE Construction Congress, San Francisco.

Cohen, J., 1990, " Constraint Logic Programming languages", *Communications of the ACM*, vol. 33, No. 7, July, pp. 52-68.

Colmerauer, A., 1990, " An Introduction to Prolog III", *Communications of the ACM*, vol. 33, No. 7, July, pp. 69-90.

Cox, Brad J., 1986, *Object-Oriented Programming, An Evolutionary Approach,* Reading Massachusetts: Addison-Wesley.

Darwiche, A., Levitt, R.E., and Hayes-Roth, B., 1989, "OARPLAN: Generating Project Plans in a Blackboard System by Reasoning about Objects, Actions and Resources", *CIFE Technical Report No. 2*, Center for Integrated Facilities Engineering, Stanford University, Stanford, Ca 94305.

de Kleer, J. and Sussman, G. J., 1978, "Propagation of Constraints Applied to Circuit Synthesis," *M.I.T. Artificial Intelligence Laboratory Memo 485,* Cambridge, MA.

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

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

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

Dixon, J. R., 1986, "Artificial Intelligence and Design: A Mechanical Engineering View," *Proc. AAAI-86*, Vol. 2, pp. 872-877.

Dixon, J. R., and Simmons, M. K., 1985, " Expert Systems for Mechanical Design: A program of Research," *ASME Paper No. 85-DET-78,* ASME Design Engineering Conference, Cincinnati, OH.

El-Bibany, H., 1992, *A Man-Computer Framework and Architecture for Design, Management and Coordination in a Collaborative AEC Environment*, Ph.D. Thesis, Construction Engineering and Management, Stanford University, to be submitted.

El-Bibany, H., Katz, G., Vij, S., 1991, "Collaborative Information Systems: A Comparison of the Electronics and Facility Design Industries", *Technical Report # 48,* Center for Integrated Facility Engineering, April.

El-Bibany, Hossam, Paulson, B.C. and Chua, Lai Heng, 1990, "Coordination between Project Participants through Constraint Management," *Proceedings of the 7th International Symposium on Automation and Robotics in Construction,* Bristol, England, June 5-7, pp. 505-513.

Fenves, S.J., Flemming, U., Hendrickson, C., Maher, M.L., Schmitt, G., 1989, "An Integrated Software Environment for Building Design and Construction," *1st Symposium of the Center of Integrated Facility Engineering*, Stanford University, March 26-29.

Fischer, M. and Tatum, C.B., 1989, "Partially Automating the Design-Construction Interface: Constructibility Design Rules for Reinforced Concrete Structures" *Sixth International Symposium on Automation and Robotics in Construction (6th ISARC),* San Francisco, USA, June, pp. 127-134.

Fox, M. S., 1983, "Constraint-Directed Search: A Case Study of Job Shop Scheduling," *Technical Report CMU-RI-TR-83-22,* The Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Froese, Thomas M. and Paulson, B.C., 1990, "Object-Oriented Programming for Project Management Software," *Proceedings of the 7th International Symposium on Automation and Robotics in Construction,* Bristol, England, June 5-7, pp. 513-521.

Fryman, F., and Mittal, S., 1987, "Making Partial Choices in Constraint Reasoning Problems," *Proc. AAAI-87*, WA., July.

Garrett, J.H., and fenves , S.J., 1987, "A Knowledge-based Standards Processor for Structural Component Design," *Engineering with Computers 2*, pp. 219-238.

Gosling, G., 1983, *Algebraic Constraints,* Ph.D. Dissertation, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Holtz, N.M., 1982, *Symbolic manipulation of Design Constraints - an Aid to Consistency Management*, Ph.D. dissertation, Dept. of Civil Engineering, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

Howard, H. C., and Rehak, D. R., 1989, "KADBASE -- A Prototype Expert System-Database Interface for Engineering," *IEEE Expert*, 1989, in press.

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.

Ito, K., Ueno, Y., Levitt, R.E., and Darwiche, A., 1989, *Linking Knowledge-Based Systems to CAD Design Data with Object-Oriented Building Product Model*, CIFE technical report No. 17, Center for Integrated Facility Engineering, Stanford University, August.

Kartam, N. and Levitt, R., 1989, "A Least-commitment Approach to Planning Construction Projects with Repeated Cycles of Operation," *Sixth International Symposium on Automation and Robotics in Construction (6th ISARC),* San Francisco, USA, June, pp. 189-196.

Kellett, J.M., Brown, N. and Boardman, J.T., 1990, “A Structure for an Interactive Project Management System,” *First International Conference on Expert Planning Systems*, Institution of Electrical Engineers Conference Publication No. 322, Brighton, UK, June, pp. 163-168.

Koo, Charles C., 1987. *Synchronizing Plans among Intelligent Agents via Communication,* thesis submitted to Stanford University in partial fulfillment of the requirements for the degree of Ph.D., August.

Krishnan, V., Navinchandra, D., Rane P., Rinderle, J. R., 1990, "Constraint reasoning and Planning in Concurrent Design," *CMU-RI-TR-90-03*, The Robotics Institute, Carnegie Mellon University, February.

Lakmazaheri, S., Rasdorf, W.J., 1989a, "The Analysis and Partial Synthesis of Truss Structures via Theorem Proving," *The International Journal of Engineering with Computers.*

Lakmazaheri, S., Rasdorf, W.J., 1989b, "Constraint Logic Programming for the Analysis and Partial Synthesis of Truss Structures," *The International Journal of Engineering, Design, Analysis, and Manufacturing.*

Lansky, A. L., 1985, "Behavioral Specification and Planning for MultiAgent Domains," *Technical Note 360*, Artificial Intelligence Center, SRI International, November.

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

Levitt, R. E., Dym, C. L., Jin, Y., 1991, "Knowledge-Based Support for Concurrent Multidisciplinary Design," *CIFE Working Paper # 10*, Center for Integrated Facility Engineering, Stanford University, January.

Levitt, Raymond E. and Kunz, John C., 1987 “Using Artificial Intelligence Techniques to Support Project Management,” *AI EDAM*, Vol.1, No. 1, pp. 3-24.

Logcher, Robert D., 1987, “Adding Knowledge Based Systems Technology to Project Control Systems,” *Project Controls: Needs and Solutions,* Ibbs, C. William, Jr. and Ashley, David B. Eds. New York: American Society for Civil Engineers, pp. 76-87.

Meunier, K. L., and Dixon, J. R., 1988," Iterative Respecification: A Computational Model For Hierarchical Mechanical System Design," *Proc. ASME International Computers in Engineering*, Vol. 1, pp. 25- 32.

Mittal, S., Dym, C.L., and Morjaria, M., 1986, "PRIDE: An Expert System for the Design of Paper Handling Systems," *Computer*, vol. 19, no. 7, July.

Popplestone, R.J., 1984, "The Application of Artificial Intelligence Techniques to Design Systems," *International Symposium on Design and Synthesis*, Japan Society of Precision Engineering, Tokyo.

Popplestone, R.J., 1987, "The Edinburgh Designer System as a framework for Robotics," *Proc. 1987 IEEE International Conference on Robotics and Automation*, vol. 3, pp 1972-7.

Ramchandran, N., and Shah, A., 1988," Expert System Approach in Design of Mechanical Components," *Proc. 1988 ASME International Computers in Engineering*, Vol. 1, pp. 1- 10.

Rasdorf, W.J., and Fenves, S.J., 1986, "Constraint Enforcement in Structural Design databases," *Journal of the Structural Division*, American Society of Civil Engineers, vol. 112, no. 12, December, pp. 2565-2577.

Rasdorf, W.J., Ulberg, K.J. and Baugh, J.W., 1987, "A Structure-Based Model of Semantic Integrity Constraints for Relational Databases," *Engineering with Computers*, Springer-Verlag, vol. 2, no. 1, spring, pp. 31-39.

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

Serrano, D., Gossard, D., 1988, "Constraint management in MCAE", *Proc. International Conference on Applications of Artificial Intelligence in Engineering:Design"*, Palo Alto, California, Editor: J.S.Gero, Elsevier-Computational Mechanics Publications.

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, 1987.

Skibniewski, Miroslaw J., 1990, "On the Use of Microcomputers by Small Contractors: Implications of a survey and Recommendations for the Future," *Project Management Journal*, Vol.21, No. 1, March, pp. 25-31.

Smith, S. F., 1988, "A Constraint-Based Framework for Reactive Management of Factory Schedules," *Intelligent Manufacturing, Proceedings of the First International Conference on Expert Systems and the Leading Edge in Production Planning and Control,* Oliff, M. D. ed., 113-130.

Sotoodeh-Khoo, H., and B. C. Paulson, Jr., 1989, "Sensors and Expert Systems in Production Optimization of Earthmoving Scrapers," *Proceedings of the ASCE Conference on Computing in Civil Engineering*, Orlando, Florida, September.

Sriram, D., Logcher, R.D., Groleau, N., and Cherneff, J., 1989, "DICE: An Object Oriented Programming Environment for Cooperative Engineering Design," *Technical Report No: IESL-89-03*, Intelligent Engineering Systems Laboratory, Dept. of Civil Engineering, MIT, Cambridge, MA 02139, April.

Stallman, R.M., and Sussman, G.J., 1977, "Forward Reasoning and Dependency-directed Backtracking in a System for Computer-aided Circuit Analysis," *Artificial Intelligence*, vol.9, pp. 135-196.

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

Stefik, M., Bobrow, D., Brown, H., Conway, L., and Tong, C.,1982, "The partitioning of Concerns in Digital System Design," *Proc. of the Conference on Advanced Research in VLSI*, MIT, Cambridge, MA.

Sussman, J. G. and Steele, G.L., Jr., 1980, "CONSTRAINTS-A Language for Expressing Almost-hierarchical Descriptions," *Artificial Intelligence,* vol. 14, no. 1, pp. 1-39.

Tatum, C.B., 1989, "Design and Construction Automation: Competitive Advantages and Management Challenges," *Sixth International Symposium on Automation and Robotics in Construction (6th ISARC),* San Francisco, USA, June, pp. 332-339.

Tommelein, I., Levitt, R. and Hayes-Roth, B., 1989, "Sightplan: An Artificial Intelligence Tool to Assist Construction Managers with Site Layout," *Sixth International Symposium on Automation and Robotics in Construction (6th ISARC),* San Francisco, USA, June, pp. 340-347.

Warthen, B., 1988, "PDES—A CAD Standard for Data Exchange," *Unixworld*, December, pp. 103-104.

Waugh, Lloyd M., 1990, *A Construction Planner*, Ph.D. Thesis Department of Civil Engineering, Stanford University, June.

Zima, Hans P., 1985, "A Constraint Language and its Interpreter," *Research Report RJ 4714 (50192),* IBM Research Laboratory, San Jose, California 95193.