Automation and Robotics in Construction XI
D.A. Chamberlain (Editor)

AEC Collaborative Information Systems: From Requirements to Architecture

Hossam El-Bibany

Assistant Professor, The Pennsylvania State University, Department of Architectural Engineering, University Park, PA 16802-1416, U.S.A.

Abstract

A collaborative information system must fit the market, organization, culture, and norms of the group it support. In the Architecture/Engineering/Construction (AEC) industry, a computer-based framework for collaboration needs to consider issues such as its application suitability in various project organizations, and the ability to operate under partial knowledge. This paper briefly presents the philosophy behind and the basis for a conceptual design of a human-computer architecture to facilitate design, management and coordination in AEC projects. Section 2 presents the framework and the need for parametric representation and constraint-based reasoning as a unifying methodology. The principles behind a system architecture are then described in Section 3 with a brief description of components and user interaction. The implementation environment and testing of a prototype system are finally presented in Section 4.

1. INTRODUCTION

The creation of a facility is a complex, highly-interdependent process. Typically, no individual possesses all of the required knowledge and skills, so design, construction and management teams that include participants from each required domain are formed to collaborate and jointly create a single product. As projects become increasing complex, additional participants are added to the team. While this approach supplies the needed expertise, it also introduces a significant coordination burden and can decrease the overall efficiency of the process.

Computers have long supported various tasks over the life-cycle of the project (e.g., architectural modeling, structural design, project scheduling). For certain tasks they are generally indispensable, while in other areas their shortcomings often overshadow their advantages [Skibniewski 90]. Computer technology may provide tools for design, management and coordination in human-computer collaborative systems. A great deal of research is ongoing into integration between various computer tools to support 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, as collaborative information systems must reflect the market,

organization, culture, and norms of the group it supports, the nature and project organization of the Architecture/Engineering/Construction (AEC) industry makes it almost impossible to just rely on knowledge-based support without standard representation [El-Bibany 92].

This paper briefly presents the philosophy behind and the basis for a conceptual design of a human-computer architecture to facilitate design, management and coordination in AEC projects. The framework that builds on specific AEC industry factors is first presented. The need for parametric representation and constraint-based reasoning are described within the framework requirements. The principles behind a system architecture are then described with a brief description of components and user interaction. Implementation environment and testing of a prototype system are finally presented.

2. THE FRAMEWORK

The framework that supports collaboration over the life-cycle of a project in the AEC industry assumes that process coordination and human interaction are the basis for collaboration. Specific organizational factors related to the team, product, process and managed information over the life cycle of an AEC project special design of collaborative information systems. The main objective of this framework is to provide collaborative information systems with six main capabilities:

- Provide design and management tools.
- 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.

In providing these capabilities, these systems should be flexible enough to consider the change of project organization over time, to allow work to proceed with partial knowledge, to create and maintain dependency information among data items as needed by various users, and to have powerful explanation facilities clarifying their state of knowledge.

2.1. Constraint-based Representation and Reasoning

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. [El-Bibany et al 91] identified these needs, studied various collaborative information systems models developed for other industries, as well as designed a generic model (Figure 1) that fits these needs.

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.

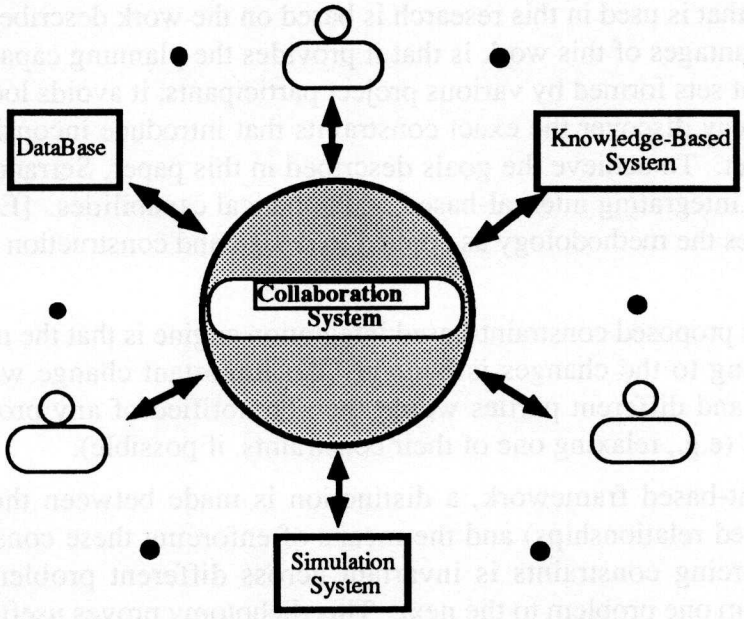

Figure 1. Collaboration System Concept

To achieve this flexibility in a standard representation, the main reasoning technique should handle various types of AEC problems in the same manner. Parametric constraint-based representation and reasoning will achieve this goal [Chan and Paulson 87, 88, Serrano 87, Lansky 88]. In addition, constraint-based reasoning was chosen based on the following:

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

2.2. 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, one 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 is a planning process for the use of constraints. The presented methodology uses 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 is used in this research is based on the work described in [Serrano 87]. The major advantages of this work is that it provides the planning capability needed for solving constraint sets formed by various project participants, it avoids local constraint propagation and it may discover the exact constraints that introduce inconsistencies in a specific constraint set. To achieve the goals described in this paper, Serrano's work was further enhanced by integrating interval-based mathematical capabilities. [El-Bibany and Paulson 94] illustrates 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 this constraint-based framework, a distinction is made 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, one can concentrate on the correct formulation of a problem in terms of constraints, assuming that the computational mechanism has been correctly implemented [Chan 86]. This property is used to create standards for constraint-based representation in the system architecture.

3. THE ARCHITECTURE

The architecture offers fundamental changes to the way that project design and management computer systems are built and used. The architecture 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 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 entities (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 entities. 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.

3.1. Core of the Architecture: Entities 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 entities. Project participants look at the project in terms of individual entities (spaces, doors, beams, pipes, formwork, cranes, compressors, pumps, locations, activities, etc.). There are two things that can be captured: the knowledge that is encapsulated within the entities, and the relationships among the entities. Both may be expressed in the form of entity internal constraints (representing the internal structure and behavior of an entity), entity-to-entity constraints (representing the relationships among entities) and boundary constraints (representing external constraints imposed on an entity set).

For the architecture to reflect the AEC organizational needs, it should allow the creation of new types of entities in a standard format. The representational elements of the architecture with their standards and reasoning mechanisms are documented in [El-Bibany 92].

3.2. User Interaction

Once the system is set with specific entity classes, project participants interact through the instantiation of project specific entities and imposing the entity-to-entity and boundary constraints (through domain and project specific logic) to represent the required relationships. The system 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 entities and evaluate the global impact of proposed design changes. Advanced qualitative reasoning capabilities may also be supported by the constraint-management system.

3.3. Principles

The architecture is built on the following principles:

Unified Representation and Reasoning. Representation and reasoning are 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 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 architecture provides system managers (first level of users) with the ability to define prototypes of their own entities with their behavior. Day-to-day users (second level of users, e.g., designers, planners, etc.) would interact with the system instantiating project-specific entities 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 are based on extensive truth-maintenance capabilities [de Kleer 86 a, b, c, Shoham 87]. Internal truth maintenance controls dependency-directed propagation. An external truth maintenance system controls context-based reasoning.

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

Formalization of Representation. For a general representation, differentiation between domain and non-domain knowledge is important. Furthermore, to utilize such an architecture 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, domain entities and their behavior are represented in a non-domain specific mathematical and parametric representation. Entity 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 entities and their required interaction in various organizations.

4. IMPLEMENTATION

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 needed for this research [Lansky 88].

Prolog as a logic programming language has a sound theoretical basis, being modeled 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] may 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].

The prototype has been tested on a simple problem in the AEC industry. The problem consisted of the collaboration of an architect, structural engineer and construction manager to create a simple building. The main purpose was to test the ideas presented in this paper and refine the architecture. This stage has been performed successfully as documented in [El-Bibany 92]. The next step is to model a more complicated problem and report on the outcome.

5. CONCLUSIONS

This paper briefly presented the philosophy and ideas of a new direction of research on collaborative information systems for the AEC industry. Specific industry factors related to the dynamic project organization, the nature of the product, the design construction process and the managed information influences the design of the presented framework. The resulting system architecture needed to be based on a unifying representation and reasoning methodology. Parametric representation and constraint-based reasoning played the role of this unifying methodology. They also provided the basis for knowledge integration, building and managing data dependency structures and identifying sources of conflicts among the requirements imposed by the various project participants.

The work presented in this paper has been implemented in a prototype using object-oriented logic programming techniques. Preliminary tests have served to build and refine the architecture. The tests were further convincing of the power of this approach in design, management and coordination tasks.

REFERENCES

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

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.

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

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.

El-Bibany, H., 1992, *Architecture for Human-Computer Design, Management and Coordination in a Collaborative AEC Environment*, Ph.D. Thesis, Civil Engineering Department, Stanford University, June.

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, H., Paulson, B.C. and Chua, L. H., 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.

El-Bibany, H., Paulson, B.C., 1994, "Collaborative Knowledge-integration Systems: A Tool for Design, Management and Coordination," *MicroComputers in Civil Engineering,* Volume 9, No. 1, February.

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.

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, 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.