Integrated Data Exchange and Concurrent Design for Engineered Facilities[1]

Professor H. Craig Howard[2,3]
Professor Jeffrey Ullman[4]
Professor Kincho Law[2]
Professor Paul Teicholz[5]
Dr. Arthur Keller[3]
Sanjai Tiwari[2]
Ashish Gupta[3]
Karthik Krishnamurthy[2]

Abstract

The architecture-engineering-construction industry is highly fragmented, both vertically (between project phases, e.g., planning, design, and construction) and horizontally (between specialists at a given project phase, e.g., design). We need software that (1) translates and communicates project data dynamically, overcoming both logical barriers (differing views of overlapping data) and physical barriers (data distribution); and (2) detects, analyzes, and manages changes efficiently during concurrent design processes. We have a software prototype, KADBASE, that addresses the distributed query problem. In addition, we are adapting and applying a high-level constraint language to support distributed constraint processing for preliminary design.

1. Introduction

The U.S. architecture-engineering-construction (AEC) industry is highly fragmented compared with many of its Asian and European competitors [Howard 89a]. This fragmentation exists both within individual phases of the construction process (e.g., the design phase), as well as across project phases from planning through design and construction and into facility maintenance and operation. The problems arising from fragmentation affect productivity and competitiveness throughout the AEC industry. Since constructed facilities account for over half of the capital investment of manufacturing industries, AEC fragmentation increases costs across the entire economy.

Over the past thirty years, computers have become very widely employed by specialists in all phases of the AEC process, and yet the AEC industry is still exchanging data and design decisions as it did a century ago, with paper drawings and reports. Introducing computers to the process has changed the means of generating the paper, but it has not fundamentally changed the methods of sharing data across organizational boundaries. The result is that we have many little "islands of

1 This work is supported by NSF Grant IRI-9116646.
2 Department of Civil Engineering, Stanford University, Stanford, CA 94305-4020
3 phone: 415-723-5678; e-mail: hch@cive.stanford.edu
4 Department of Computer Science, Stanford University, Stanford, CA 94305-2140
5 Center for Integrated Facility Engineering, Stanford University, Stanford, CA 94305-4020

automation." It is not uncommon to find that in one office a designer uses a powerful computer-aided design and drafting (CADD) package to produce a project drawing and then, in another office, a construction estimator uses a digitizer to put the information from the same drawing back into a computer (in effect, *unCADD).* The result is a substantial net loss in efficiency and an increase in the ever-present potential for errors. Consequently, the use of computers in the AEC industry has tended to reinforce rather than weaken the organizational fragmentation that already exists.

As more and more of the facility engineering process is automated via computer-based integration, the data management and communication requirements will grow exponentially. The AEC databases of the 2000's will have to track multiple versions of designs, provide a spatial reference system for the automated control of construction machines, represent the as-built status of the completed facility, and accumulate changes that occur over the facility's operating life. The challenges are particularly great in "fast-track" construction, where the contractor begins constructing the foundation before the architects and engineers are finished designing the superstructure.

Our objective in the proposed research is to develop a virtual integrated computing environment that helps the project participants work concurrently to design and construct complex facilities. The computing environment will have to provide capabilities to represent and integrate the data views of each specialist, to communicate and translate data between the databases and applications of different organizations, and to intelligently manage changes across the physically and logically distributed databases. This proposal presents the requirements for a solution, discusses the engineering and computer science challenges in providing that solution, describes our own work so far in attacking parts of the larger problem, and outlines our proposed plan for integrating those approaches to solve the larger problem.

2. Different Views of Project Data: An Example

To provide the reader with a more tangible sense of the data communications issues, Figure 1 illustrates several of the key stages involved in a building project. The figure emphasizes the differing views that the various project participants have about the data describing the process. The following discussion elaborates on the management of data at each stage of the process:

- The *architect* may start with grand ideas for the structure, but must shape these into a more practical design based upon the constraints (financial, schedule, technical, operational) generated by the owner, engineers, users and others in the process. The architect is normally the overall coordinator and leader in this multidisciplinary process, and increasingly is likely to use advanced CADD software. The architect also generates many, if not most, of the constraints and criteria that govern others in the process, but communicates with them mainly through traditional paper documents and meetings.

- The *structural engineer* takes the architect's ideas and designs the skeleton that allows the building to resist the loads that derive from its function (e.g., furniture and people) as well as the loads caused by the environment (e.g., wind and earthquakes). As with the architect, the operations of the structural engineer may be highly computerized (e.g., sophisticated CADD systems and analysis packages); yet, even in-house data communications are usually not automated (e.g., few CADD systems can effectively exchange data with analysis packages).

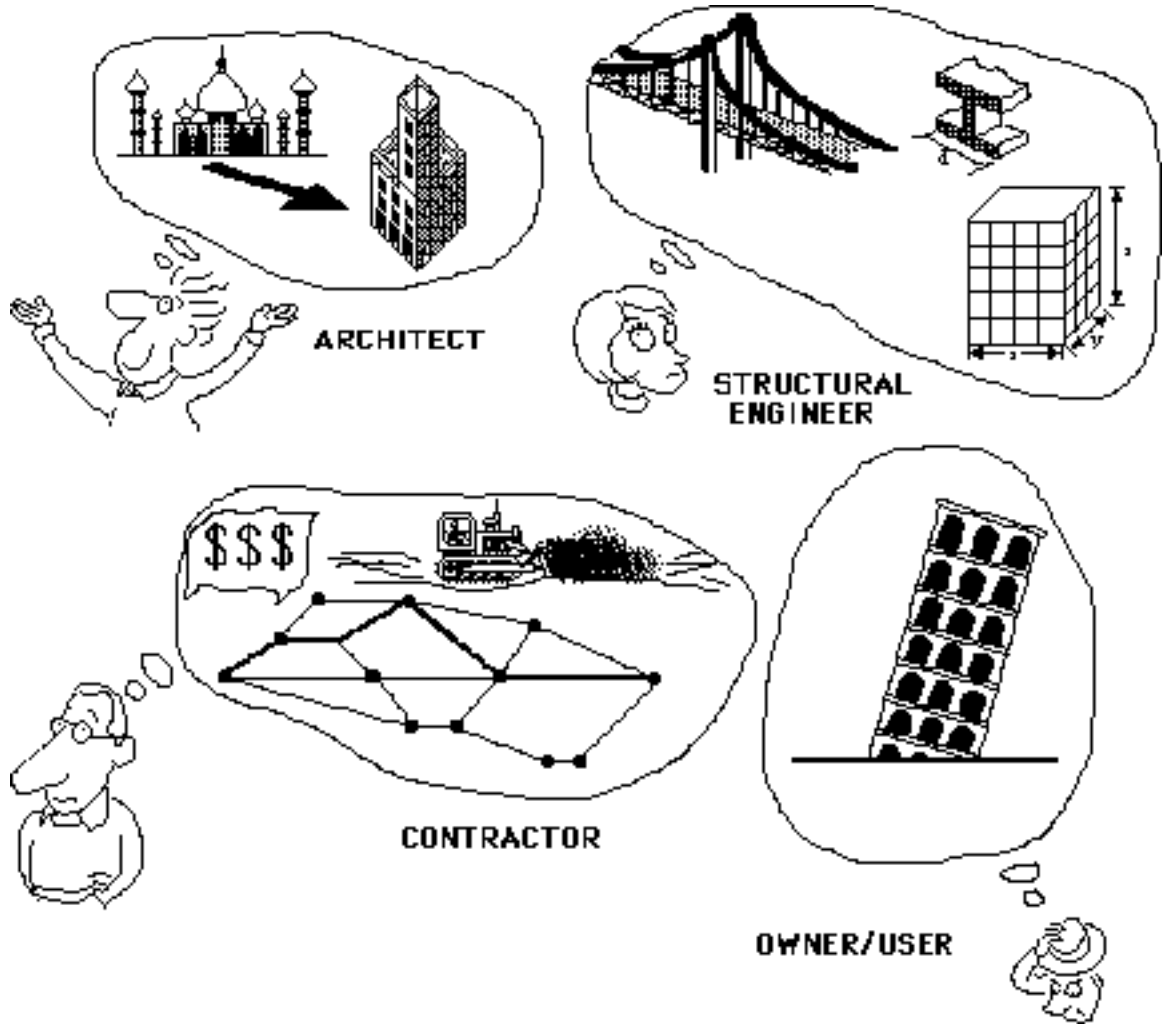

Figure 1: Differing Views of the Project Data

- The *contractor* takes the contract plans and specifications that result from all of these prior deliberations, and adds his knowledge of costs, schedules, methods and materials to produce an estimate and resource-based schedule for completing the project most efficiently. The general contractor normally retains specialty subcontractors who, in turn, may further subcontract specific tasks. There is little use electronic data communication across these boundaries, and coordination of changes is a time-consuming and error-prone process.

- The *owner* inherits the resulting performance of the structure, which hopefully stands up better that shown in the Figure 1. For maintenance, operations and future modification of facilities, owners are becoming increasingly conscious of the need for an accurate record of what was designed and built, and are seeking the record in electronic form. As an example, Stanford University is now trying to digitize key utility and building data from an existing file of some 45,000 paper drawings just to create a database to manage existing facilities. The university is also faced with the problems of having future architects, engineers and contractors to produce documents in a machine-readable, *machine-usable* form. Many public agencies and other private owners are facing this problem and are taking actions that may or may not be productive in the long run. Large sums are being expended to digitize existing drawings of dubious reliability into representations that may be difficult to integrate with other sources and types of data.

The intent of this discussion is to highlight the differing views of each of the project participants, to emphasize the current lack of electronic data communications among the participants, and to underscore the potential for efficiencies arising from effective methods for sharing data. The following sections further refine our view of the information to be communicated and describe methods for intelligently automating that communication.

3. Requirements for Data Exchange & Concurrent Design

This research proposal addresses the problem of concurrent design of architectural and engineered facilities (buildings, plants, roads, tunnels, etc.). Our goal is to *develop database technologies to automate data exchange and support concurrent design*. While there are many design systems currently available, they normally are dedicated to only one discipline (e.g., architectural or structural design) and do not allow multiple disciplines to work together in an efficient and effective manner. The current linear, paper-based design process is too slow and error prone, particularly when there are multiple changes to the design under tight time pressures with the various designers working for separate companies in different locations. Thus, we need data management technology that satisfies the following requirements:

1. **Access to distributed data.** The database will not be located in a single computer (or server). Rather, it will be located in different locations where the various designers are working. The project team will be linked via local-area or wide-area networks. Each discipline needs to be able to work independently for some period of time, and then link together with the other members to determine the validity and consistency of the design. At these times, validation programs are needed that will check for design consistency (among each discipline and with external codes). Thus, a distributed database capability is needed that will be secure (cannot lose or corrupt data) and permits both batch and on-line communication.

2. **Access to heterogeneous hardware and software.** The design and construction of a complex facility involves a large number of different computer programs, each attacking parts of the overall problem. We see that heterogeneity increasing instead of decreasing as programmers develop new and more powerful ways to solve both general and specialized problems—it is not feasible to consider building one integrated system on one platform to serve all the diverse and evolving of the project participants. In response to this software balkanization, the integrated environment must be able to support a variety of hardware and software platforms.

3. **Support for a variety of data.** The problems of exchanging data among diverse organizations are further compounded by the many and different kinds of data needed in varying degrees by each contributor to a project. Broad categories of these data include administrative (costs, schedule, procurement, etc.), technical (engineering analysis, quality specifications, etc.), graphical (drawings and sketches), real-time construction process automation (monitoring and control), and facility operations (environmental monitoring, structural modifications, etc.). The integrated data representation must accommodate spatial data (3D object models), graphical data (pictorial representations), textual data (object attributes, specifications), and temporal data (design versions and construction plans). The primary issue in providing an integrated approach to engineering data is to represent it in a form that is not only *machine-readable*, but *machine-usable* as well. For example, typical CADD systems create machine-readable vector and symbol representations, but additional information is required to quantify the design elements as a bill of materials, to provide a spatial reference system for the automated control of the construction machines, to represent the as-built status of the completed facility, and to accumulate changes that occur over the facility's operating life.

4. **Support for multiple views of data.** We have already described in the previous section how the various project participants have differing views of the facility data. The integrated environment must support the identification of common data and the translation of that data from one view to another.

5. **Management of changes.** A report or message is required when the changes of one designer have some impact on the efforts of another designer. For example, the location of columns must be consistent with the floor layout so that if the architect changes the layout, the structural designer must automatically be informed of this change. A more subtle example involves logical relationships among each element of the design. When the cooling loads for a building are increased by the architect increasing the size of the windows, the engineer designing the cooling system must be informed of this fact. The first step in this process is the detection of changes and monitoring data dependencies between design participants.

6. **Detection of conflicts.** In a complex multi-designer project, individuals could be blitzed by changes from every angle. Therefore, the environment should provide a mechanism for defining a designer's constraints and for filtering out changes that cause physical or logical conflicts. Ideally, the designer's constraints are specified declaratively, and methods for responding to or resolving constraint problems are also declaratively described.

7. **Adequate performance.** Finally, the performance of the databases must be adequate to support the design process for each of the project team. This requires that normal design functions within a given discipline must happen quite fast (within the threshold of user tolerance), while some functions that require checking against other disciplines can run as nightly processes. The environment must include features for both dynamic and batch transactions for data exchange, change notification, and conflict detection.

The above requirements are common to many complex design processes in discrete manufacturing and electronic chip design and fabrication. A significant amount of research has been done at Stanford and elsewhere to develop database and application systems that allow concurrent design with strong links to manufacturing. The proposed research will build on these efforts, but will reflect the unique needs of the AEC industry. These include: one of a kind facilities (with replications of common modules and components), fragmentation of designers (different companies in different locations), and a variety of data views used by each discipline.

4. Data Integration

Craig Howard's previous work on integrated engineering databases examined the basic issues involved in interfacing knowledge-based systems with database management systems. KADBASE (Knowledge Aided Database Management) is a flexible, knowledge-based interface in which multiple applications and multiple databases can communicate as independent, self-descriptive components within an integrated, distributed engineering computing system [Howard 86, 89]. In the KADBASE model, each application and database contains knowledge describing its data structures and data access methods. The data spaces of the components are integrated into a single global data space. The KADBASE architecture consists of *knowledge-based system interfaces* for each application, *knowledge-based database interfaces* for each database, and a *network data access manager* as shown in Figure 2. The network data access manager accepts requests from the applications through their KADBASE interfaces, locates the required data in the available databases, and sends requests to those databases through their KADBASE interfaces. Within the application and database interfaces, requests undergo a two-step translation: *syntactic* translation between the component request language and the interface request language, and *semantic* translation between the context of the component's data representation and the context of the global data representation. Each application or database is connected only to the central network data access manager. To add or update a component, one need only describe the syntactic and semantic mapping between that component and the network data access manager, thereby linking it with every other component without having to prescribe mappings to each of those systems.

The original design for KADBASE built on the ideas from early research into heterogeneous, distributed databases. There are a number of systems available that integrate relational databases, usually providing a relational global model, but sometimes providing an object-oriented or “semantic” model like KADBASE. Elmagarmid and Pu surveyed this class of systems in [Elmagarmid 90]. KADBASE has previously been implemented in demonstration systems at Carnegie-Mellon University and at Stanford. We are currently considering how to extend KADBASE within our current research.

5. Constraint Management

The design specialists in the AEC process are autonomous, geographically distributed and have domain-specific views of the design data: an architect may be concerned with spatial characteristics of beams and columns while a structural engineer evaluates their load transfer capabilities. We propose a distributed database model to take into account the autonomy and geographical distribution of the data. Autonomous domain-specific databases can cater to the management of design information in a given domain. Database management systems (DBMS) facilitate efficient retrieval and storage of massive amounts of design data that can be accessed concurrently by various domain-specific applications.

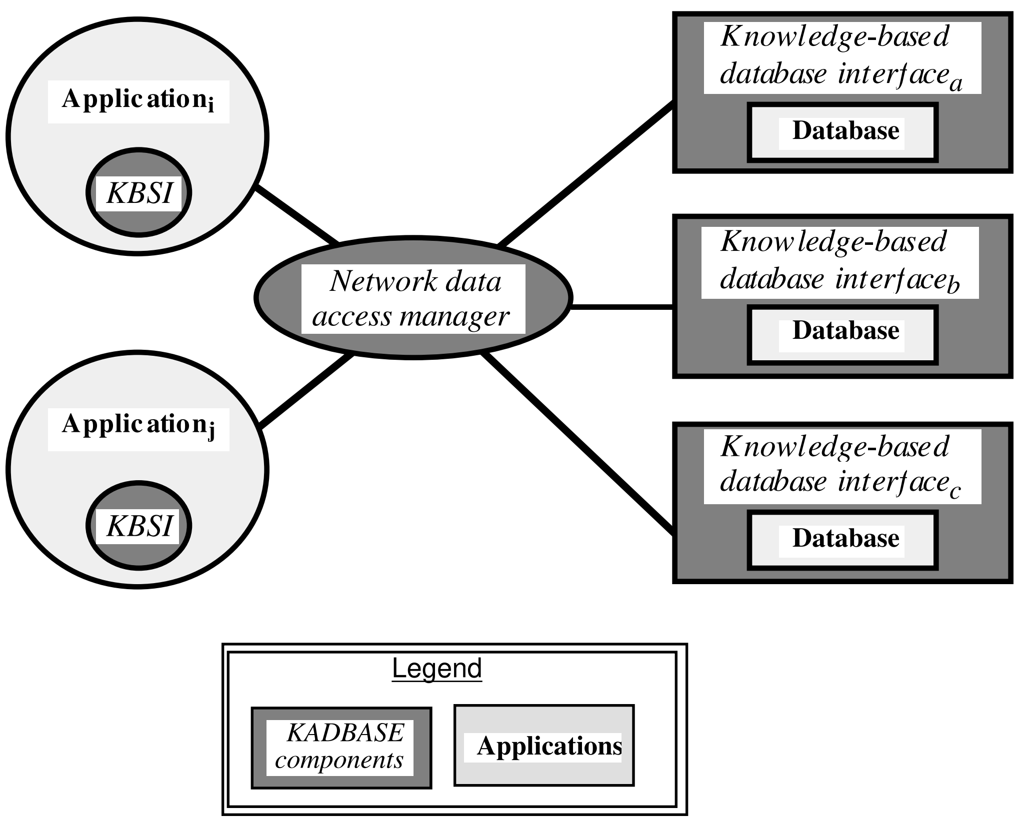

Figure 2: Overview of KADBASE architecture

However, the autonomous design databases need to be maintained consistent to insure the spatial and functional consistency of the design. A set of inter-domain constraints exist between the objects stored in different databases. For example, if window sizes are changed by the architect, the HVAC engineer (Heating, Ventilation, and Air Conditioning) needs to be notified about the change for its effect on the air-conditioning of the room. We are adapting a high-level constraint language to express, optimize and efficiently manage such cross-functional constraints.

Both the domain-specific data and application programs need to be distributed for autonomy of the individual disciplines and efficient management of computing resources. Distributed design databases can support data requirements of the domain-specific applications. A high-level query language or application programming interface (API) of a DBMS can support data input/output requirement of various applications, like construction scheduling and estimation software. The output of such applications can also be stored in the database itself and can be queried later for the supervision and maintenance purposes.

Figure 3 shows such a framework with autonomous architect, structural and HVAC databases [Tiwari 93, 93a]. A global constraint manager (GCM) is responsible for maintaining a set of inter-domain constraints, and it uses the facilities provided by the local constraint managers (LCMs) and monitors. A description of the components of the constraint management system follows:

- **Constraint:** A constraint is expressed as invalid design state over a set of autonomous design databases. The inter-domain constraints can involve multiple design attributes of objects residing in different databases. In our approach, a high-level constraint language is used to specify constraints to the global constraint manager. The constraints are declarative in that a user specifies "what" is required to be enforced rather than how to enforce a constraint. Frequently, the constraints are not known a priori, i.e., at the time of database design—the constraint language provides {\it data independence} in this sense, constraints need not be put into the system with the data. Design data and constraints are important, but orthogonal issues in the facility design process.

  The constraint language is SQL-like as shown in the example below, with some extensions relevant to our framework.[6] In our implementation, the major extension has been the transition from a centralized case to a distributed environment [Ceri 90].

```
Architect::Floors:Floor_Ht < any
     ( select  sum(Rooms.Ceiling_ht, Beams.Depth, HVAC_Ducts.Depth)
       from    Architect::Rooms, Designer::Beams,  MEP::HVAC_Ducts
       where   Floors.floor_Id = Room.floor_Id and
               Rooms.Room_Id = HVAC_Ducts.Room_Id
               HVAC_Ducts.Beam_Id = Beams.Beam_Id)
actions:
       Notify(Architect, Designer, MEPengineer);
```

- **Application:** A program that needs input data for domain specific reasoning and analysis. The output of an application, e.g., structural analysis and material cost estimation, usually involves refinement or addition of design attributes to the database objects. Applications interact with the DBMS through transactions, which cause atomic changes to the database states.

---

6 The original language has been implemented for the Starburst system, which is a prototype centralized database management system developed at IBM Almaden Research Center.

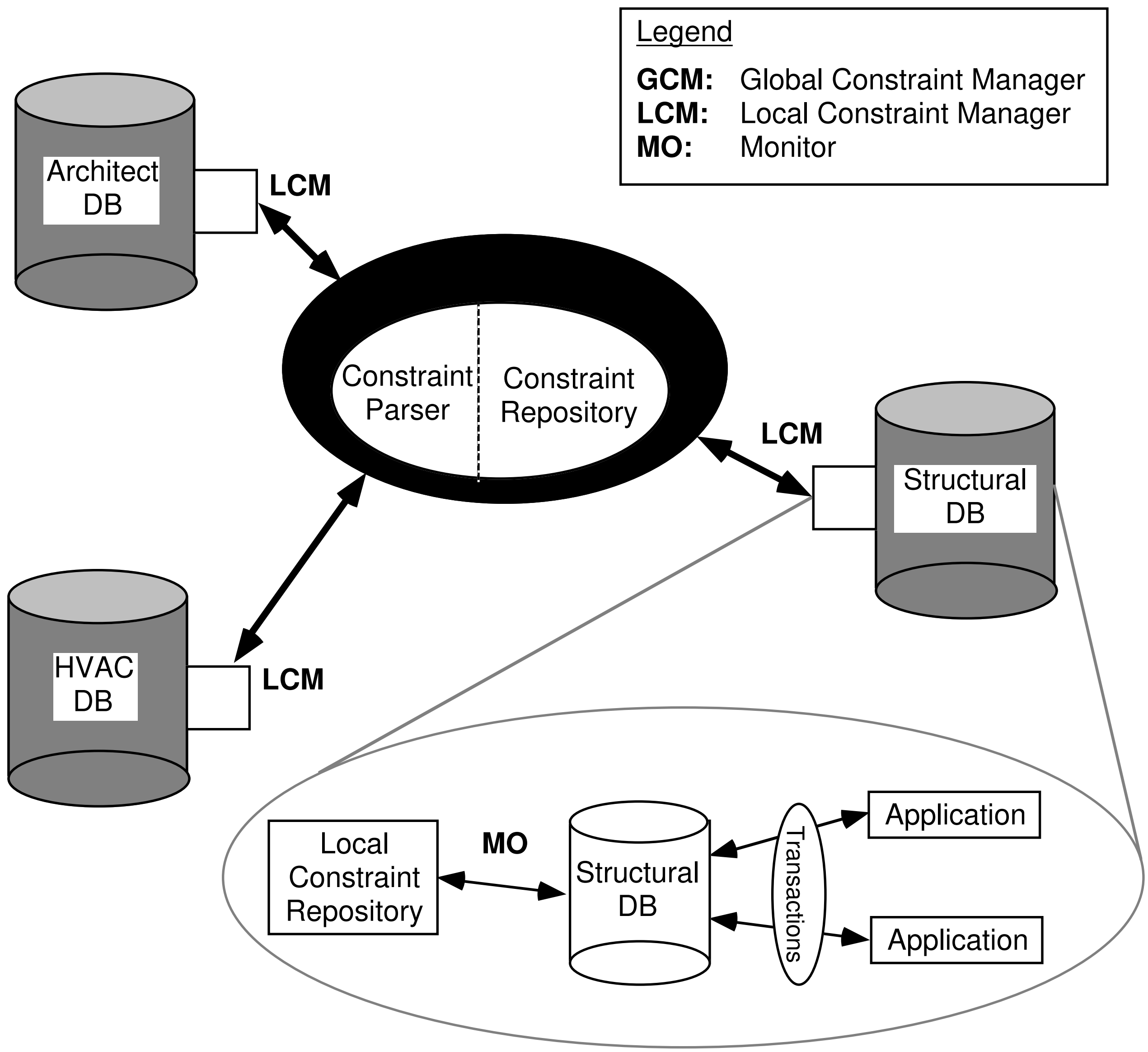

Figure 3: Architecture of A Distributed Constraint Management System}

- **Global Constraint Manager:** A global constraint manager is entrusted with the task of decomposing and distributing a constraint. The global constraint manager (GCM) is distinguished from local constraint managers in having a catalogue of all the design information at various sites (a global data dictionary). The GCM also has a repository of all the design constraints, and like other database objects, it provides facilities for the updates or queries on constraints themselves. The constraint processor resides within the GCM, which enables it to extract the relationships between constraints and database objects. The partitioning of a global constraint into a set of local constraints is based on the respective sites of the design objects referred to in that constraint.

- **Local Constraint Manager:** Local constraint managers (LCM) are stripped-down versions of their global counterpart and manage portions of constraint specifications relevant to their site. LCMs can communicate directly with the local database and query the data if a constraint violation is suspected. They can also send the relevant data to the GCM if the need arises. The primary function of an LCM is to maximize local constraint validation or else to communicate the minimal required data to the GCM, if local validation is not sufficient. LCMs make use of local DBMS integrity checking facilities to achieve this goal.

- **Monitors:** As the name suggests, monitors provide a monitoring facility for individual design objects in a database. There is one monitor per design object. Monitors interact closely with the local DBMS and are constantly on the lookout for updates to the design objects within a transaction that may lead to a constraint violation. The relevant update information is communicated to the LCM if it is a potentially invalid update.

A constraint processor has been implemented in C++ that works on top of Oracle and Starburst database systems. The processor uses *lexx* and *yacc* unix tools to lexically analyze and create as bottom-up parse of a constraint specification. A set of rules operate on this parse tree of a constraint to derive the potentially invalid operations (monitors in our terminology) on local design databases. Constraint catalogs are set up as persistent database tables, used by the global and local constraint managers and also as part of the constraint repository that can be used by the designers to query the constraints.

We are using Starburst database system at IBM-Almaden Research center because it provides better monitoring facilities [Widom 89]. A set of inter-disciplinary constraints have been tested on prototype architect's, structural and hvac databases for a commercial office building. The framework is general enough and can be ported on top of any commercial database systems like oracle and sybase. C++ is used to implement the constraint managers code and calls to Starburst API running on RS6000.

6. Summary

The degree of vertical fragmentation (between project phases, e.g., planning, design, and construction) and horizontal fragmentation (between specialists at a given project phase, e.g., design) in the U.S. AEC industry is unparalleled in any other manufacturing sector. The latest census of the U.S. construction industry reveals that it consists of over 1.4 million establishments, of which over 930,000 have no employees and the remainder have an average of 10 employees. The designers of constructed facilities are similarly fragmented by specialty area and by specialty within specialty. It is not uncommon to find 20 separate design firms involved in various aspects of a major high-rise building's definition. Moreover, the buyers of construction are very fragmented. Individual real estate developers, home buyers, entrepreneurs, and a myriad of state and local agencies constitute a very fragmented customer base. There are a few major consumers of construction; the U.S. Army Corps of Engineers, the General Services Administration, and the nation's largest private manufacturing organizations are large buyers of construction. However, their cumulative purchases are probably less than 25% of the industry's output.

The combination of vertical and horizontal fragmentation of the marketplace results in small specialized firms that frequently serve local geographical markets. Moreover, the U.S. tradition of open bidding for construction results in intense competition, unpredictable workloads, and low overheads, including essentially zero investment in research and development for the industry. The specialization may offer some flexibility and benefits to the industry, but produces tremendous costs in the form of fragmented decision-making. Lack of integration among the planners, designers, lenders, builders, and operators of constructed facilities misses important opportunities for improved project performance. Contingencies for risks arising from a lack of information at various levels throughout the development and life of a facility add further costs. To date, computer utilization in facility development has tended to reinforce rather than mitigate this existing organizational fragmentation—fragmentation which fosters inefficiencies in planning, financing, designing, constructing and managing capital projects.

While pushing the frontiers of automation in the AEC industry, the project will also advance the state of the art in the database area. We will develop a facility for integrating databases that is beyond the state of the present art in the degree of integration it supports and in the variety of languages and systems to which it will speak. We will also provide a significant advance in the power of deductive databases, introducing change-management facilities, optimization for remote database access, and integration—features not found in current prototypes.

Our objective in this project is to create the software tools to support a virtual integrated computing environment to manage data intelligently for the architecture-engineering-construction industry. That means communicating data, translating data, checking data for consistency across interdependent parts of the project design, and resolving the inevitable inconsistencies. The potential benefits are improved designs, reduced construction time, fewer design errors and omissions, minimization of costly rework, and better lifecycle management of the constructed facility.

References

[Ceri 90] Ceri, S., and Widom, J., "Deriving Production Rules for Constraint Maintenance," *Proceedings of the Sixteenth International Conference on Very Large Data Bases*, pages 566-577, 1990.

[Elmagarmid 90] Elmagarmid, A. and C. Pu (eds.) [1990]. *ACM Computer Surveys* 22:3.

[Howard 86] Howard, H. C., and Rehak, D. R., *Interfacing Databases and Knowledge Based Systems for Structural Engineering Applications,* Technical Report EDRC-12-06-86, Engineering Design Research Center, Carnegie-Mellon University, Pittsburgh, PA, November 1986 (PhD dissertation).

[Howard 89a] Howard, H. C., Levitt, R. E., Paulson, B. C., Pohl, J. G., and Tatum, C. B., "Computer-Integration: Reducing Fragmentation in the AEC Industry,"*Journal of Computing in Civil Engineering,* Vol. 3, No. 1, pp. 18-32, 1989.

[Howard 89] Howard, H. C., and Rehak, H. C., "KADBASE: A Prototype Expert System-Database Interface for Engineering Systems," *IEEE Expert,* Vol. 4, No. 3, pp. 65-76, Fall 1989.

[Tiwari 93a] Tiwari, S., and Howard, H. C., "The Management of Design: A Design Notification Scheme for Distributed AEC Framework," to appear in *First International Conference on the Management of Information Technology for Construction*, Singapore, August 1993.

[Tiwari 93] Tiwari, S., and Howard, H. C., "Constraint Management in Distributed AEC Databases," to appear in *Fifth International Conference on Computing in Civil and Building Engineering*, June 1993.

[Widom 89] Widom, J. and Finkelstein, S. J., "A Syntax and Semantics for Set-Oriented Production Rules in Relational Database Systems," IBM Research Report RJ 6880, IBM Almaden Research Center, Almaden, CA, 1989.