A Design Notification Scheme for Distributed AEC Framework

SANJAI TIWARI[1]
H C HOWARD[2]
R E LEVITT[3]

ABSTRACT

The Architecture, Engineering and Construction design process (AEC) involves participation of many inter-dependent specialists, who nevertheless make autonomous design decisions in their specialized domains. This fragmented and multidisciplinary design process often suffers from inadequate communication among specialists and a lack of coordination of design information. In this paper, we present a design-change notification scheme for a distributed engineering environment. The notification scheme facilitates conflict resolution by providing the relevant information about the design objects involved in a conflict. It operates on a global design repository, which uniformly captures the design data and cross-functional design dependencies. Upon detection of a design conflict, notification messages are automatically generated and delivered to concerned project participants in a particular sequence that is based on the nature of the conflict and the roles of the parties involved. We assert that this type of notification system in a design environment will facilitate identification of design conflicts early in the design process, and should decrease the number of change orders and rework during the construction phase. Fewer change orders and reduced project meetings can result in enhanced trust, improved cooperation, and lower project costs.

Key Words

design conflict; architecture, engineering and construction; engineering databases; constraint management; distributed systems

INTRODUCTION

The facility construction process involves participation of many specialists who need to exchange design data and information throughout the design process. A typical facility design project may involve one or more architects, structural engineers, mechanical engineers, electrical engineers, and contractors. These specialists often belong to separate design firms and have domain-specific views of the design data: an architect may be concerned with

1 *Graduate Assistant, Department of Civil Engineering, Stanford University, Stanford, California, USA 94305-4020.*
2 *Assistant Professor.*
3 *Professor.*

spatial characteristics of beams and columns while a structural engineer evaluates their load transfer capabilities.

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 amount of design data that can be accessed concurrently by the design group. A high-level query language or application programming interface (API) of a DBMS can support data input/output requirements of domain-specific applications, like construction scheduling and estimation. The results of such applications can be stored in the database itself and can be queried later for supervision and maintenance purposes (Tiwari and Gupta, 1993).

In addition to the independent views and concerns of the participants (Levitt, Yan and Dym, 1991), the geographical distribution of the participants leads to further fragmentation of the AEC design process (Howard, Levitt, Paulson, Pohl and Tatum, 1989). The autonomy of the participants and the distribution of design data motivated us to propose a distributed database model for the AEC design process. However, the design data is inter-dependent in that a set of inter-domain constraints exist between specialists' design. For example, if the size of the windows is changed by the architect, the HVAC (Heating, Ventillation and Air Conditioning) engineer needs to be notified about the change since it may effect the air-conditioning of the room.

The constraints are cross-functional because they deal with the spatial and functional characteristics of the design objects stored in different domain-specific databases. A facility design project can have large number of such constraints over a much larger number of the design objects, which necessitates efficient constraint-management techniques to provide real-time performance in a design environment (Tiwari and Howard, 1993). Identification of constraint violation can be thought of as identification of conflicts arising out of spatial or functional inconsistencies in the design.

Under existing AEC industry practices, design conflicts are often detected much later (at the execution stage) and are resolved through project meetings or correspondence, which can be time-consuming and expensive. Identification and detection of design inconsistencies early in the design process will lead to a decrease in design time and cost and a reduction in the number of change orders during the construction phase. Fewer change orders and reduced project meetings can result in enhanced trust, improved cooperation, and lower project costs. Designers will be able to explore a variety of alternatives since awareness of other participants' design criteria will give them more confidence before committing themselves to a particular design solution.

A powerful notification scheme should provide a minimum amount of relevant information to assist in conflict resolution. If redundant (or incomplete) information is presented to the designer, she/he may have to sort through the notification message (or request additional information). The system should be able to deliver intelligent notifications based on the context, and requirements of a particular designer. Our notification scheme is an extension of a constraint specification language being implemented for management of design constraints on a distributed AEC project data (Tiwari and Howard, 1993). In this paper, we present specific syntactic and semantic extensions for an automatic generation of a design notification. We will touch on a few organizational issues such as design information sharing, automated conflict identification and notification, and on-line versus periodic notifications, which must be addressed for effective implementation of this approach in a real collaborative AEC design environment.

There are ongoing projects that address the problems of management of design information in collaborative design environments. The DICE project (Sriram, Ahmed and Logher, 1991) addresses the issue of coordination and communication problems in engineering through a centralized blackboard and a global database. Our approach emphasizes data distribution, while DICE emphasizes on transaction management and concurrency control for a multiuser environment. The KADBASE project at Stanford (Howard, 1991) aims at providing a transparent interface to knowledge-based applications and databases in a heterogenous and distributed environment. Similarly, Morse *et al* (Morse and Henderickson, 1991) and Levitt *et al* (Levitt, Yan and Dym, 1991) have proposed models for communication and coordination in an engineering design environment.

DISTRIBUTED AEC FRAMEWORK

Figure 1 shows the distributed AEC framework on which the design notification can be effectively supported. The constraint management system assists in the identification of the design conflicts that may result due to the changes made to the design databases. In the following subsections, we highlight the important features of the proposed framework and present an overview of the constraint management system (Tiwari and Howard, 1993) that generates conflict notifications.

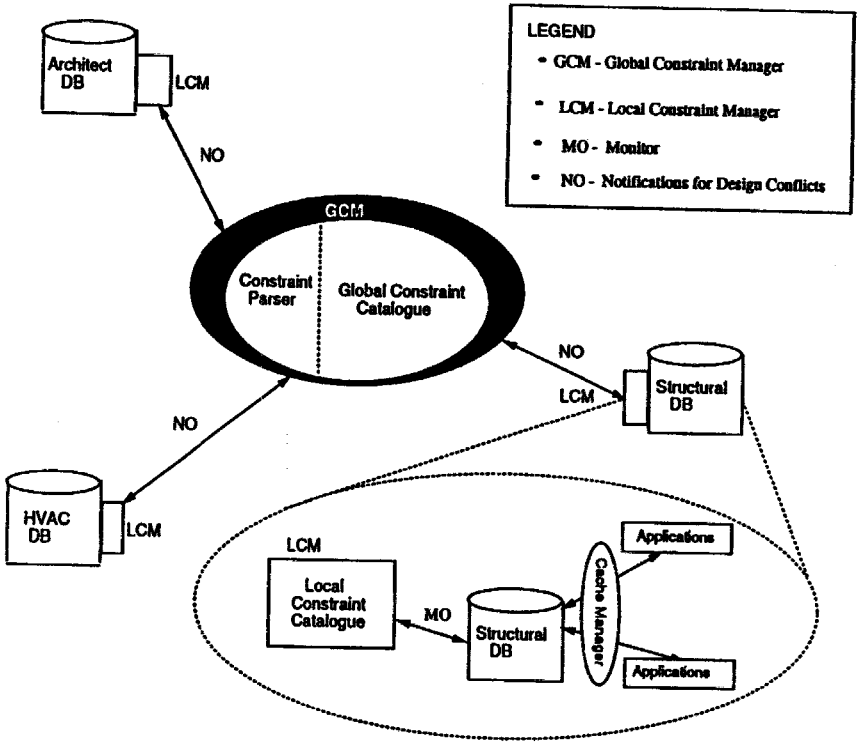

Figure 1.

Sharing of Computing Resources

Managing computing resources and applications, such as sophisticated CAD tools, database management systems and operating systems, tends to be expensive. These systems operate on domain-specific data and the distribution of this data will lead to the distribution of applications. Distributed databases will have higher throughput and performance since individual DBMSs can support local data requirements locally, thereby minimizing the retrieval of data from a remote centralized database. A centralized-database framework offers ease of maintaining consistency since all the data is stored at one site, while applications have to query this data remotely (Sriram, Ahmed and Logher, 1991). However, centralized databases may run into bottlenecks of excessive data communication and overheads of concurrency in engineering domains, for which greater amounts of data transfer are required to run the applications than are required for consistency checking. Since the domains have overlapping data requirements, the data and applications should be

distributed as much as possible for autonomy, performance and economic reasons (Ceri and Pelagatti, 1984).

Increased Autonomy

As mentioned earlier, the participants in the AEC design process are independent and autonomous entities, each providing the specialized knowledge needed to construct the artifact. The design data specific to a particular domain is best understood by the design group which uses it to run the in-house and customized software. Any integrated system, no matter how smart, should not update this data arbitrarily, that is, access privilages need to be maintained. A design conflict is a constraint violation detected by the constraint management system. A conflict may result from errors of omission, or due to conflict between the design goals of the participants themselves. The conflict should not be resolved independently by the system since it may require updates to the distributed AEC databases. Appropriate notifications should be generated to allow the specialists to make decisions and to adjust the design parameters in their specialized domains.

Design Constraint Management

We use a high-level constraint language to express, optimize and validate design constraints at the runtime of the design environments. The constraints are expressed over the design states stored in distributed AEC databases. An example of a constraint specification appears in the next section and additional inter-domain constraints are listed in the Appendix. A global constraint manager (GCM) coordinates the DBMSs at the respective sites to query the changes that have occured since the last time the constraints were validated. Thus, constraint validation is periodic and, as envisaged in our project, the constraint manager will be invoked by the design cache-manager— responsible for management of the design data at the local database site. Distribution of constraints closer to the actual data also helps in local validation of design constraints. If an update occurs on a local database, the local constraint manager (LCM) first attempts to check the constraint with the locally available data. A global data collection is done and notifications are generated for a given constraint violation, only when the local validation fails.

An efficient and useful way to express constraints would be to propagate their specifications in order to limit the number of constraints that need to be specified for a large project. A constraint may be defined on higher level of abstraction, for example architectural spaces, which can then be propagated to different types of spaces in the building such as rooms, coridoors, and staircases. The view mechanism supported by the databases (Ullman, 1989) can be useful in keeping the number of constraints for a large project within a reasonable limit. Views represent derived or virtual data, and constraints

may be specified over views, which can then enforce these constraints at the desired level of abstraction.

Once the constraints are expressed and compiled at the global site, they are distributed for efficiency reasons. Also, a constraint repository is maintained by the GCM to facilitate the query and updates on the constraints themselves. Since the constraints are cross-functional and may be prioritized during different phases of the project, constraint specifications can be stored in a DBMS to provide a Constraint Repository. A constraint repository can be very useful for recording cross-functional design dependencies and the design rationale, and the information stored therein can be be used to generate meaningful notifications.

DESIGN NOTIFICATION

Notifications are generated when a given constraint is violated due to an update of a database by the designer or domain-specific applications. Before discussing the notification characteristics, we present an example of a design constraint specification over distributed design databases.

Example 1a:

```
Contractor::Cranes: Capacity  <= any  (  select Weight
                                         from Designer::Columns
                                         where Columns.Floor-Id =
                                         Contractor::Cranes.Floor-Id)
actions :
        Notify (Contractor, Designer, Project Manager);
```

For this simplified scenario, assume that a structural designer's database contains information about the columns' characteristics, and the contractor's database contains information about the cranes assigned to particular floors in a building construction project. The sample constraint says that capacity of the crane used at site should be greater than the weight of any building column it has to lift. If the constraint is violated, then the system should notify the concerned participants, *ie*, designer, contractor and project manager in this case. The details of constraint management issues are presented elsewhere (Tiwari, 1991); for the present discussion, the following characteristics of the design notifications are important.

Notification Content

The notification message should contain minimal relevant information about a design conflict. As a first cut, we consider only automatically generated notifications; one way to classify various notifications based on their content can be:

Design Conflict:

Conflict notifications are generated by the constraint management system and are the primary focus of our discussion. Designers can also issue such notifications, but this would require them to query others' databases and detect a conflict manually. This scenario also accomodates the constraints that are ad-hoc or could not be expressed in the system and, therefore, need to be checked manually.

Conflict Resolution:

Conflict resolution notifications are issued as a response to the conflict notification, *ie*, as a first step toward resolving a design conflict.

Follow up:

Follow up notifications include all other types of notifications that do not fit in the above categories. Such notifications are generated manually and may include the information that a participant considers relevant for the recipients. In order to support a collaborative design environment, it is important that all the above categories need to be supported. We present a method to deal with the first case, that is, automated generation of conflict notifications. In order to include the relevant information about the design attributes that can result in a constraint violation, we parse and compile a given constraint specification to derive the following low-level production, which are enforced at the individual databases level (Tiwari, 1991).

Example 1b:

```
when
/* when any of these database operations are executed at the Designer or
Contractor site */
    /* triggers for Designer Site */
    updated Designer::Columns.Weight
    updated Designer::Columns
    inserted into Designer::Columns
    /* triggers for Contractor Site */
    updated Contractor::Cranes.Capacity
    updated Contractor::Cranes.Floor-Id
    inserted into Contractor::Cranes
/* then check whether there exist any tuples in Conflict-Set */
if exists Conflict-Set : ( select * from Contractor::Cranes
    where Capacity <= any (select Weight
    from Designer::Columns
    where Contractor::Cranes.Floor-Id =
            Columns.Floor -Id))
```

```
/* User Specified Actions */
then select Crane-Id from Conflict-Set
    Notify (Field Engineer, Contractor, Project Manager);
```

The above production deals with the low-level database objects and is more efficiently enforced than the high-level constraint specification shown in Example 1a (Ceri and Widom, 1990). This production evaluates the constraint when any of the operations in the conditional clause occur at various local sites. The validation process involves a search through elements in the Conflict Set , which is exactly those cranes that fail to satisfy the constraint. Thus, only information about the members of the conflict set need to be notified. At present, the constraint management system collects all the attributes (select * statement of SQL query in Example 1b); we are working on further optimizations to notify only the relevant design attributes required to resolve a given design conflict.

Notification Frequency

The frequency of notifications is a very practical consideration since the in-coming notifications should not hinder the design process itself. The system should provide means to control the constraint validation period, or the delivery of notifications— notifications may be buffered and delivered on request. A better approach would be to pass this responsibilty to a local design-cache manager, which supports the checkin-checkout process on a design database. The cache manager can invoke the LCM when requested by the designer. This approach can also accommodate the what-if changes since the cache-manager need not commit the changes to the database. However, to support checkin-checkout mechanism over many database objects, the constraint management system needs to deal with sets of design changes. For the very same reasons, our constraint language is set-oriented (Ullman, 1989), that is, it supports sets of changes to the database.

Notification Database

After a conflict notification is issued, the system needs to keep track of which conflicts have been resolved or are in the process of being resoved. In our example, the contractor may be responsible for allocating appropriate crane capacity to the site; alternatively, in certain circumstances, the designer may be required to stay within crane capacity limits in selecting the column sizes. To model this non-directionality of constraints, there can be a particular order in which notifications are issued to the participants who are responsible for the design or planning decisions that have generated the conflict. If the initial conflict notification to the field engineer fails to receive any attention, or if an authorized participant so requests, the notification can

be forwarded to other interested parties in the design conflict, *eg*, the structural steel designer and the contractor's project manager in our crane-column example. If all issued notifications are kept in a database, they can be tracked to find out which notifications were issued when, and how many of these design conflicts have been resolved. The system can be enhanced to monitor the conflicts that have not received any attention for a given time period and can send a reminder of them or forward them to the participant's supervisors. The notification database can also assist in keeping the log or history of all the conflicts, which can be queried later to get useful information about the status of the design. This information will be required not only for the duration of the project but also during the service, maintenance and retrofit of the facility.

The architecture we have described allows the system to model the organizational structure of the AEC design team—*ie*, who can be permitted to arbitrarily require a change in another's design decision—as well as the generic hierarchy of design decisions. For instance, in the type of crane-column conflict described in our example, the project design manager may decide that crane capacity is generally more difficult to change than maximum column weights so that the designer should always be asked to try to resolve a constraint by reducing the maximum weight of columns, before asking the field engineer to obtain a larger crane. Formalizing AEC knowledge about the hierarchy of design decisions and the organization structure of the design team is likely to have many benefits beyond facilitating more rational conflict notification. This issue is highlighted as needing more research; suffice it to say that the architecture of the constraint management system is capable of supporting this kind of prioritizing of constraints and the ranking of participants authority to dictate conformance.

Notification Format

The format of the notification messages can vary from textual to audio or video messages. Multimedia technologies will prove helpful, especially if the database allows drawings and images to be stored. The advanced features such as query by image content on the databases - QBIC project (Cody, 1991) - allows retrieval of images by their content. The constraint management system should certainly be able to include multimedia attributes in the notification message if efficient means to query and transport multimedia data across the network exist. Thus, an architect may query the layout of a given room and electronically mail the drawing to a contractor at the project site as a follow-up notification.

CONCLUSIONS

The future information systems will emphasize increased distribution of

data and applications for autonomy, performance and economic reasons. We have presented a distributed AEC framework to model a facility design process where each participant runs customized applications on a domain specific database. The important problem of identification and notification of inconsistencies resulting from updates made to the database was discussed. In particular, we studied the characteristics of the design notification scheme to support collaborative AEC framework.

A powerful notification scheme will enable designers to work simultaneously on multiple projects since the burden of keeping track of various changes that may effect other participants' design is passed to the constraint management system. Notifications will enable designers to identify design conflicts and take measures to resolve them early in the design process. Early detection of design conflicts will lead to reduction in expensive change orders and project meetings. Designers will have greater confidence before committing themselves to a particular design solution since they will become more aware of other participants' design goals and criterias. Ultimately, a global constraint management system coupled with intelligent notifications will prove its worth by bringing a reduction in project duration and costs.

References

Cody, William (1991), *The QBIC Project.* Technical Report, IBM Almaden Research Center, San-Jose, California.

Ceri, S and Pelagatti, G (1984), *Distributed Databases: Principles and Systems*, McGraw-Hill.

Ceri, S and Widom, W (1990), Deriving Production Rules for Constraint Maintenance. *In Proceedings of Sixteenth International Conference on Very Large Databases*, pp 566-577.

Howard, H C, Levitt, R E, Paulson, B C, Pohl, J G and Tatum, C B (1989), Computer Integration: Reducing Fragmentation in the AEC Industry. *Journal of Computing in Civil Engineering*, volume 3(1), pp 18-32.

Howard, H C (1991), *Linking Design Data with Knowledge Based Construction Systems.* Technical Report CIFE Spring Symposium Proceedings. Center for Integrated Facility Engineering, Stanford University, Stanford, California.

Levitt, R E, Yan Jin, Dym, C L (1991), Knowledge Based Support for Management of Concurrent, Multidisciplinary Design. *AI in Engineering, Design and Manufacturing*, volume 2(5), pp 77-95.

Morse, D V and Hendrickson, C (1991), Model for Communication in Automated Interactive Engineering Design. *Journal of Computing in Civil Engineering*, volume 5(1), pp 4-24.

Sriram, D, Ahmed, S and Loghcher, R (1992), A Transaction Management Framework for Collaborative Engineering. Engineering with

Computers, volume 8(4).

Tiwari, S and Gupta, A (1993), DScheduler: A Deductive Database for Scheduling Building Construction Tasks. To appear in Proceedings of Fifth International Conference on Computing in Building and Civil Engineering, Anaheim, California.

Tiwari, S and Howard, H C (1993), Constraint Management on Distributed AEC Databases. To appear in Proceedings of Fifth International Conference on Computing in Building and Civil Engineering, Anaheim, California.

Tiwari, S (1991), *Constraint Management on Distributed AEC Databases*, Unpublished Thesis Proposal Submitted to the Department of Civil Engineering, Stanford University, Stanford, California.

Ullman, J D (1989), *Principles of Database and Knowledge Base Systems*, volume 1 and 2, Computer Science Press.

APPENDIX

Sample Project Constraints

- The capacity of the crane used at site should be greater than the maximum weight of the building column it has to lift..
- The constraint: Floor Elevation 1.10 * (Ceiling Ht. + Depth of the Beam + Depth of HVAC Duct) should be satisfied under all circumstances. Notify all professionals in case of conflict.
- The size of the reinforcement bar (diameter, bar) used by the contractor should be within 10 range as specified in the structural engineering database.
- If Wall thickness changed by more than 20, notify the structural engineer for possible change of depth of the supporting beam section.
- Exterior Walls could not be deleted by Architect until no reference exist in other databases, *ie*, they have been removed from structural designer's and contractor's database.
- If any wall partitions are removed by the Architect, delete the corresponding windows and doors from the designer's and contractor's database.
- If Slab thickness changed by more than 20, notify the structural engineer for possible change of depth of the surrounding beam section.
- The availability and cost of the formwork for the Round columns dictates that their diameter be even number in the range of 12"-32" and that there should be at least twelve such columns in the building.
