Integration through Standard Project Models

Martin Fischer[1], Thomas Froese[2]

Abstract

Integration of data and knowledge among project participants and their computer applications holds the promise to improve the quality and efficiency of the project delivery process in the AEC industry. We propose that integration can be achieved by sharing a standard project model. In this paper, we describe such models. We summarize two research projects that developed and used shared project models and we outline essential characteristics of these models.

Introduction

Integration in the Construction Industry

The technical, social, and economic complexity of construction projects has increased significantly over the past decades. This increase is likely to continue into the next century [Froese 91]. The increased complexity of projects has also led to an increase in specialization and thus fragmentation in the architecture, engineering, and construction (AEC) industry. This fragmentation and the resulting inefficiencies in the AEC industry have been well discussed and documented over the past years [Howard 89, Tatum 87a, Tatum 89]. Strategies to overcome these inefficiencies have mainly focused on organizational and contractual measures, such as early assignments of project managers [Tatum 87b], partnering [Tarricone 92], design-build, construction management, etc.

Increased project complexity and participant specialization have also greatly increased the need for information transfer. Traditionally, this has been achieved through the exchange of drawings and reports. Today, much project information is stored and processed on computers. Thus, integrating project participants' computer systems would greatly enhance the effectiveness of a project team by increasing data sharing, reducing time requirements and errors for data input and output, accelerating communication among participants, and improving the completeness of information received by each team member. Several communication protocols have been proposed and are already in use (e.g., DXF, IGES). Unfortunately, most of these communication standards are largely based on computer paradigms of the past and focus on data communication only. "Integration" through data communication—for example, exchange of drawings—has been the modus operandi of the construction industry for hundreds of years and has often proven insufficient to address the challenges of current projects mentioned above.

Today, effective integration requires the continuous and interdisciplinary sharing of project goals, data, and knowledge among all project participants. Therefore, we advocate that the AEC industry pursue integration through shared standard project models. These models should be based on object-oriented paradigms to allow the capture and exchange of data <u>and</u> knowledge.

---

[1] Act. Asst. Prof., Dept. of Civ. Engrg., Stanford, CA 94305-4020

[2] Grad. Res. Asst. and Ph.D. Cand., Dept. of Civ. Engrg., Stanford, CA 94305-4020

Computer-Supported Integration

This paper focuses on computer integration, which encompasses electronic information sharing and inter-process communication among project participants' computer systems. Computer integration is necessary at many levels: to link the various applications used on a single platform by one user, to unite all the project participants' systems together, and to provide industry-wide capabilities for the electronic communication of information. Three basic mechanisms support computer integration (Figure 1):

- *Direct translators:* each program is treated as an independent agent, or self-directed entity, and establishes direct links and translation with other programs.
- *Global control and translation:* a global controller and translator can be created and linked to each application (Howard and Rehak 1989). This controller manages all integration and performs all translation among different applications.
- *Standard models:* standard project models can be defined and adopted by all applications. Project models are "conceptual structures specifying what kind of information is used to describe buildings and how such information is structured (Björk and Penttilä 1989)." Project models can be thought of as schemas, ontologies, or vocabularies for representing AEC project information in computers. Applications that adopt a standard project model need no translation or central control to interact with each another.

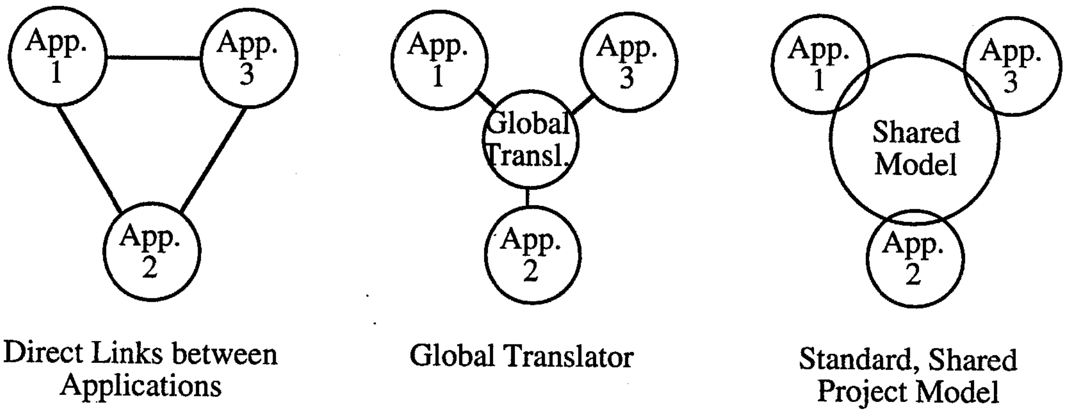

Figure 1 Integration Mechanisms

Each of these approaches has advantages and disadvantages. Direct translators are problematic for systems with many applications since each program must contain a translator for every other program that it communicates with. While a global controller and translator solves this problem, it still requires translators from each program to some universal neutral format that encompasses all the programs' data representations. A standard model approach also requires a universal project model that supports all AEC software; however, no translation is required since each application uses the same model for external, and possibly internal, information representation and communication.

Furthermore, programs that adopt a standard project model can store their information in shared databases. Integration then comes for free since each application reads and writes a single version of the data in a single location. Data consistency problems are solved since each application no longer maintains redundant copies of the data, and control of the integration requires only traditional database concurrence principles.

Therefore, we propose that AEC computer applications adopt standard project models. These applications can then communicate and share information through these models. If this is to support industry-wide integration, it requires the creation of industry-wide project model standards. These standard models must be comprehensive, rich, and flexible representations of AEC projects. The specification of these standards must include the data model (a formalism for representing information in general) as well as the project model (a schema or vocabulary for expressing AEC concepts in terms of the data model). If shared databases are used their characteristics and functionality must also be specified.

The next section discusses two examples of AEC computer integration using standard project models. In the section on the Characteristics of a Project Model, we then describe the characteristics required of standard project models to support integration. We cannot, at this point, provide the answers to all the questions related to the development and use of shared product models. Instead, we would like to raise issues that became apparent in our research and illustrate possible solutions to foster discussion of the content, use, and benefits of shared, standardized project models.

Examples

Introduction to Examples

This section outlines two examples of integration of AEC computer applications using a shared project model. It highlights particular integration issues and briefly describes the specific solutions chosen to address these issues.

The purpose of the first system (COKE) is to automate constructibility feedback for the preliminary design of a reinforced concrete structure. It links a constructibility expert system with a project model and contributes to the sharing of knowledge among project participants. OPIS, the second system, combines several traditional and knowledge-based project planning applications around a shared object-oriented project database.

COKE

In a research project sponsored by Stanford's Center for Integrated Facility Engineering (CIFE), one of the authors developed COKE (Construction Knowledge Expert) [Fischer 91a]. COKE provides a designer of reinforced concrete building structures with constructibility feedback related to the layout and dimensioning of structural elements. A major goal for the development of COKE was to automate this feedback as much as possible.. This was achieved by creating a link between a CAD system (representing the project) and an expert system (containing the constructibility knowledge base). The following paragraphs describe the architecture, implementation, and use of COKE.

System Architecture (Figure 2)

COKE was developed using CIFECAD as the CAD package [Ito 89] and KAPPA-PC as the expert system shell. CIFECAD is a modified version of AutoCAD (Release 10). Menus and functions programmed in AutoLISP make AutoCAD an object-based CAD system that can capture the project data (outlined below) necessary for constructibility reasoning. These menu-functions allow a designer to draw structural elements without drawing the individual lines of each object. The user of CIFECAD also indicates connection information with the mouse (e.g., by clicking on the columns that support a slab). CAD data are then written to an ASCII file that can be interpreted by KAPPA-PC to build a symbolic (geometrical and topological) model of the project's structure. KAPPA-PC also contains the constructibility knowledge base and functions that compare the constructibility knowledge to the project data to give a designer feedback about the constructibility of his/her design.

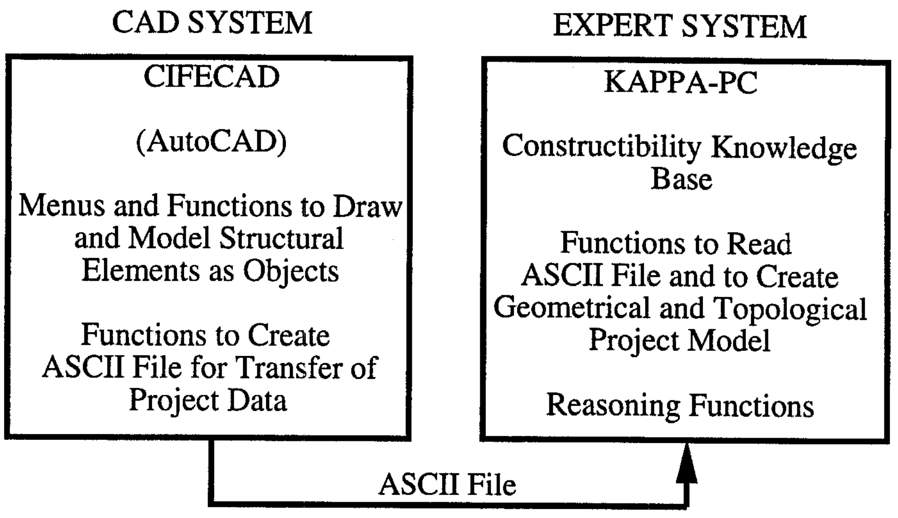
CAD SYSTEM
EXPERT SYSTEM
CIFECAD
(AutoCAD)
Menus and Functions to Draw and Model Structural Elements as Objects
Functions to Create ASCII File for Transfer of Project Data
KAPPA-PC
Constructibility Knowledge Base
Functions to Read ASCII File and to Create Geometrical and Topological Project Model
Reasoning Functions
ASCII File

Figure 2. COKE's System Architecture

The reasons for this architecture are threefold:

(1) *It is impractical to use an expert system shell to visualize a structure, or a CAD system to reason about constructibility.* This requires two separate systems, one to represent project data, and one to store constructibility knowledge and to perform the reasoning function.

(2) *It is tedious to input project data manually into COKE.* Therefore, data must be transferable from the CAD system to the expert system. Currently, these data need to be transferred in form of a batch file because many CAD systems do not run in a multi-tasking environment. Soon, dynamic data exchange will become a standard feature of most CAD systems.

(3) *Usually, CAD systems do not know what the vectors (lines, surfaces, solids) stored in the CAD file represent.* This makes the extraction and transfer of CAD data extremely difficult. AutoCAD provides a 'block' function by which the user can group lines to represent objects and attach additional attributes to the blocks. This simplifies the extraction of CAD data since block data are more readily available than vector data in the original CAD database. Therefore, menu-functions were programmed to insert structural elements into a CAD model.

CAD System and Data Transfer

An analysis of design-relevant constructibility knowledge showed that a geometrical and topological model of the project is necessary to support constructibility reasoning. This means that for each component the CAD system needs to store data about its type, location, dimensions, connections to other elements, and additional attributes. Table 1 summarizes the structural elements that can be modeled with CIFECAD, and the data stored with each object. These data items are written to an ASCII file for transfer to the constructibility expert system.

The data necessary to support constructibility reasoning show that a significant effort is required to enable a CAD system to represent these data. When specifying the data needed to model a structure topologically and geometrically, we tried to keep the user input to a minimum. We feel that the work load for the user to input this information during the design and drawing process is acceptable. Furthermore, within the limitations of CIFECAD, the designer can place and size structural elements any way he/she wants, thus giving him/her maximum design flexibility.

Data Type	Columns	Beams	Walls	Slabs	Drop Caps
Location	x, y, z coord.	x, y, z coord.	x, y, z coord.	x, y, z coord.	x, y, z coord.
Dimen-sions	side x side y (diameter) height	top width bottom width depth length clear span	thickness height length	thickness area	side x side y thickness
Connec-tions	column below column above drop cap walls beams slabs	column left column right slab left slab right	column left column right slab left slab right wall above wall below windows doors	boundary and supporting columns supporting beams and walls	column
Addtl. Attr.	name (id) concr. strength rebar ratio level orientation round	name (id) concr. strength rebar ratio level	name (id) concr. strength rebar ratio level orientation	name (id) concr. strength rebar ratio level	name (id) concr. strength level orientation type

Table 1. Data Stored with Structural Elements in CIFECAD

Representation of Project Data in KAPPA-PC

In the expert system shell KAPPA-PC, functions were implemented that read the ASCII file described in the previous section. These functions create an object (instance) for each structural element under the appropriate object class. Each class contains a slot for every data item in the ASCII transfer file. After creating a new instance for a new structural element, KAPPA-PC fills all the slots of this new instance with the corresponding values. This process works well, but is highly application-dependent since the functions that read the ASCII file need to be structured exactly according to the sequence of the data items in the ASCII file. See Figure 3 for the structure of the project model and examples of slots.

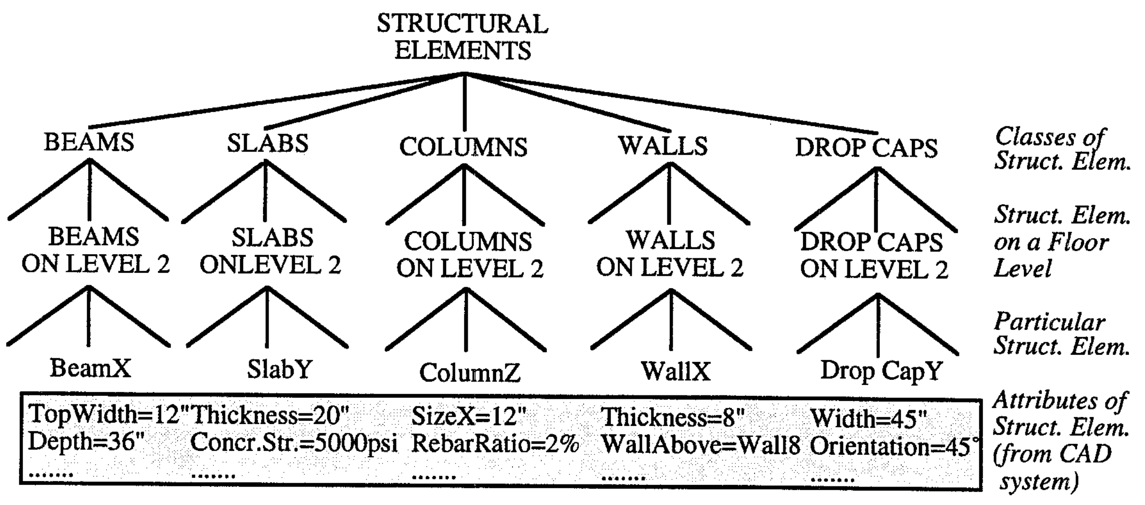
STRUCTURAL ELEMENTS
BEAMS
SLABS
COLUMNS
WALLS
DROP CAPS
Classes of Struct. Elem.
BEAMS ON LEVEL 2
SLABS ONLEVEL 2
COLUMNS ON LEVEL 2
WALLS ON LEVEL 2
DROP CAPS ON LEVEL 2
Struct. Elem. on a Floor Level
BeamX
SlabY
ColumnZ
WallX
Drop CapY
Particular Struct. Elem.
TopWidth=12" Depth=36" ....
Thickness=20" Concr.Str.=5000psi ....
SizeX=12" RebarRatio=2% ....
Thickness=8" WallAbove=Wall8 ....
Width=45" Orientation=45° ....
Attributes of Struct. Elem. (from CAD system)

Figure 3. Symbolic Representation of Project Data

Representation of Constructibility Knowledge in KAPPA-PC

I investigated the use of rules and frames for knowledge representation, and rules, message-passing, and functions for reasoning. I found that frames work well when combined with functions. In the expert system, each construction method is a frame with design-relevant constructibility items as slots (attributes) including the appropriate values (Figure 4). For example, a construction method might have limitations regarding the maximum slab thickness it can accommodate. Thus the knowledge base contains a frame "Flying Forms" with a slot "Maximum Slab Thickness" that is filled with the value "12 inches."

Reasoning Mechanisms in COKE

In COKE, functions perform the reasoning on the data contained in the project model. For each knowledge item, a function tests whether the structure fulfills the necessary requirements. To continue the previous example: if constructibility feedback for flying forms is desired, a function compares all slab thicknesses with the value stored in the "Maximum Slab Thickness" slot. These functions are grouped by construction methods to check all the requirements for one method with one function call. We found that three types of reasoning about CAD data are needed for constructibility feedback. The reasoning complexity and the data requirements to support reasoning increase significantly from type (1) to type (3). Figure 5 shows an example for each type.

a) *Reasoning about attributes of objects:* This is the simplest type of reasoning. It involves comparing an attribute-value of a structural element with the appropriate value from the knowledge base. For example, the size of a column (an item of project data) is compared with available formwork sizes (a knowledge item) to decide whether the column can be built using available formwork (Figure 5a).

b) *Reasoning about relationships between attributes of objects:* This is a slightly more complicated form of reasoning. It involves taking the attribute of an object and propagating its influence on attributes of a different object. For example, when Cunningham steel forms are used to form beams, the columns should be four inches wider than the beams (Figure 5b). This type of reasoning can become difficult—i.e., it might require some spatial reasoning—if the relationships among different attributes are complex.

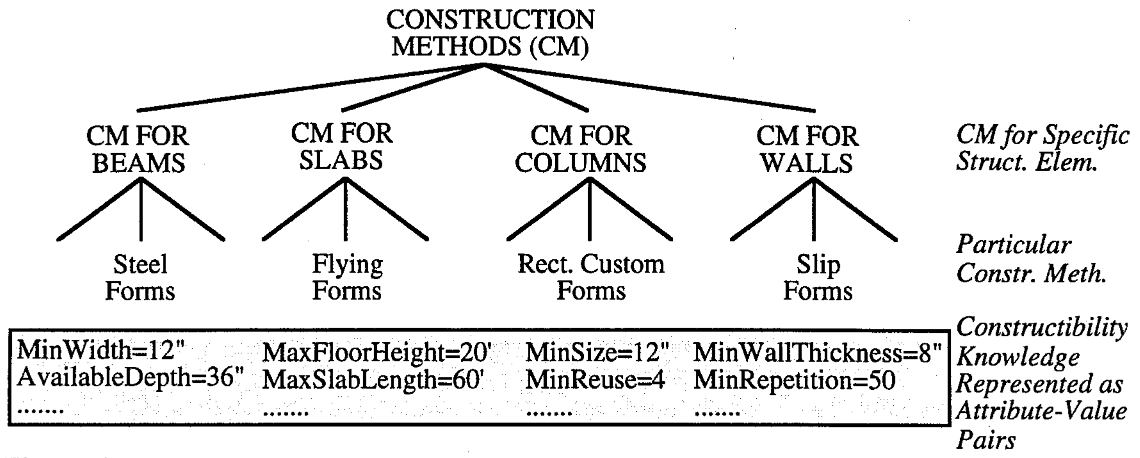

Figure 4. Representation of Constructibility Knowledge

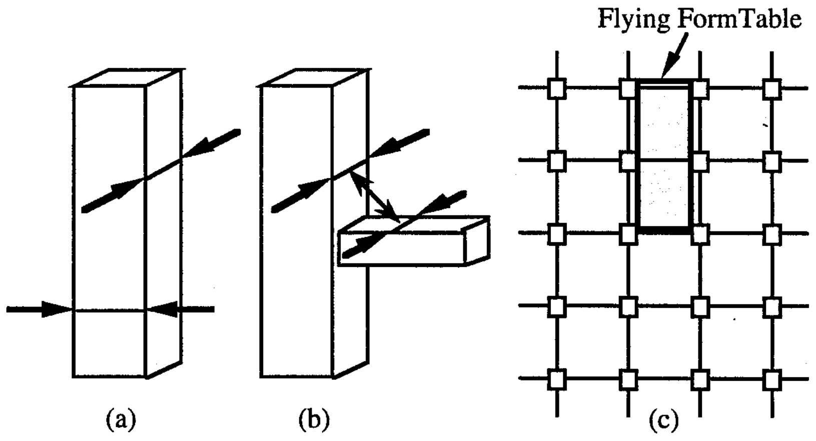

Figure 5. Examples of the Three Types of Reasoning

(c) *Spatial reasoning:* This is the most challenging type of reasoning because the amount of data and the number of objects involved are often very large. Humans can easily reason spatially by looking at a plan and recognizing patterns and relationships among objects. However, formal models that describe how to implement spatial reasoning in a software system are not readily available. An example of spatial reasoning is the task of figuring out whether a flying form table will fit into a column layout (Figure 5c). In COKE, constructibility knowledge that requires spatial reasoning is not represented as attribute-value pairs in the construction method hierarchy (Figure 8). Each knowledge item is implemented as a function that first derives the data necessary for spatial reasoning from the geometrical and topological model and then deduces the constructibility of the structure. The question remains whether this type of knowledge can be represented at all as attribute-value pairs.

The type of reasoning required for a given application depends on the knowledge and on the representation of project data (i.e., the data model of each object). On one hand, a rich data model (one that associates much data with each object) simplifies reasoning functions but might require a significant effort for data input. On the other hand, a plain data model requires only minimal data input but might require complex reasoning functions (or make reasoning about certain knowledge items impossible). Ultimately, the inference engine compares a constraint value with a project value. This is why reasoning types (b) and (c) are more difficult. Before the inference engine can conclude whether a certain constraint is violated or not, it needs to derive the data required for reasoning from the available project data. For example, the alignment of columns can be an important fact for spatial reasoning, but is not stored explicitly with each column. Alignment information is implicit in the CAD drawing. One could imagine objects with data models that already contain all the data ever needed for reasoning. This would reduce even spatial reasoning to a straightforward comparison of two values. However, the creation of such extremely rich data models seems prohibitive given the sheer size of each class of objects. Therefore, one faces the tradeoff between the amount of data and knowledge contained in a data model and the complexity of the reasoning processes [Fischer 91b].

User Interface

When working with COKE, a designer must draw the structure in CIFECAD. Whenever the designer wishes to check the constructibility of the structure, he/she gives a command in CIFECAD that creates an ASCII file from the CAD database. As described above, this file contains all the data necessary for constructibility reasoning. KAPPA-PC then builds a symbolic model of the project from this ASCII file. This symbolic model represents the structure of the project with all the data required to perform constructibility reasoning.

At the beginning of constructibility reasoning, the designer can specify the construction methods for which he/she would like to receive feedback. If the designer does not specify any construction methods, the system uses general project information and application heuristics to rule out certain construction methods that are not applicable. For example, if there are no walls in the CAD database, all construction methods for walls are excluded, or if the building is only five stories high, feedback for flying and slip forming is omitted. However, the designer has the option to get detailed constructibility feedback even if the application heuristics for this method are not satisfied.

After this, the system compares the data in the symbolic model with the constructibility knowledge for the applicable construction methods (layout and dimensioning knowledge) and gives the designer feedback about the constructibility of the structure whenever it spots a mismatch between constructibility requirements and the structure at hand. The designer can then disregard the constructibility advice if other constraints (such as usability constraints) make the consideration of the advice impossible, or he/she can incorporate the necessary changes in the CAD model.

OPIS

Overview of the OPIS System

This section describes the *Object-model-based Project Information System (OPIS).* OPIS has been developed as part of a research project conducted at Stanford University to investigate integration in Computer-Aided Project Management (CAPM) through shared object-oriented models [Froese 92a, Froese 92b]. The OPIS system provides a project manager with a set of integrated planning tools, thereby improving the data flow between applications, allowing interaction among applications, and adding CAD and knowledge-based processing to traditional project management functions, such as estimating and scheduling. This research investigated object-oriented data model standards (i.e., formal languages for representing data in general), defined a standard general-purpose project model or schema for project management and construction, and implemented the prototype OPIS system. OPIS is implemented in Objective-C on a NeXT computer and consists primarily of integrated application modules that operate on a shared object-oriented database. Figure 6 shows the prototype's overall architecture, which consists of the following major components:

1. *System-Level Components* in the integrated system include a small number of system-level control and shared utility objects. These include, for example, a user interface panel that allows the user to switch between the integrated system's different applications.

2. *An Object-Oriented Database* is the core of the system and is shared by all the integrated applications. The database stores project information using a standard object-oriented project model. The applications use this database to store all information that may be relevant to other applications; only data that are completely application-specific are stored within the applications themselves. The database uses *SOL (Shared Object Libraries)*, an object-oriented database management system written in Objective-C as part of this research.

3. *Application Modules* that perform the data input, processing, and reporting reside within the system and use the shared database for primary data storage. These application modules consist of links into the shared database objects and of application-specific objects, such as user interface or calculation objects relevant to the particular application only. Applications can also make use of libraries or tool kits of general-purpose objects, such as generic user-interface objects. The specific application modules included in our prototype are a general database browser (for inspecting and editing objects in the database), an expert system that generates construction plans, an estimating system that assigns costs to construction activities to produce project estimates, and a scheduling system that calculates schedule dates for activities and provides a graphical network interface for working with construction plans.

4. *Stand-Alone Applications* can be used by the integrated system by creating interface modules. In OPIS, an example of this is an interface module to an intelligent CAD system that creates project component objects in the database. The section on Integration using OPIS gives an example of how all the applications operate collaboratively on the shared database.

This approach can be extended to integrate a large number of construction-related applications, for example: construction planning, estimating and bid preparation, risk analysis, time and cost control, document control, contract management, materials management, project accounting and personnel, detailed activity planning, job-site CAD, 3-D construction simulation, legal analysis, construction method selection, and so on. The approach can also be extended to include all phases of the project life cycle from conception and feasibility analysis, through design and construction, and on to facility management. While all of these phases would not necessarily share a single central database, they could share a standard project model for constructed facilities and share much of the actual data for a project—thus the idea of integration through standard project models would remain.

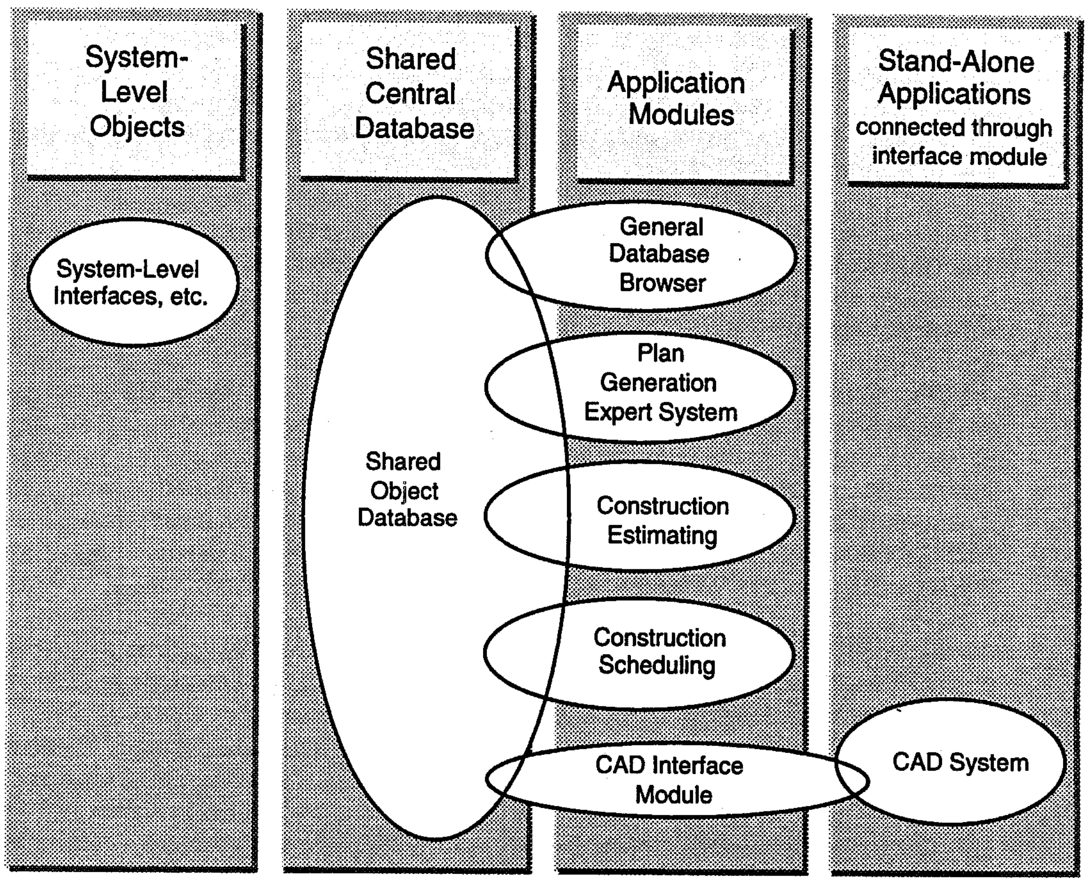

Figure 6. Overview of the major components of OPIS

Integration using OPIS

Figure 7 is an example of how OPIS would assist project planning. Typical user interfaces employed by the various applications also appear in the figure. Users work with traditional project management software interfaces and need no direct interaction with the shared database. The sequence of possible operations follows:

1. *CIFECAD*, the prototype intelligent CAD system described in the examples, is used to create a 3-D CAD model of the project. The internal representation of the CAD model identifies elements as specific project components rather than as geometric shapes only. CIFECAD is not an integrated application module within the OPIS system; but *CADLink*, an OPIS interface module, combines the CIFECAD project model with general component category information from the shared database to create a model of the facility within the shared database.

2. *AutoPlan*, an expert system plan-generation application within OPIS, creates a construction plan within the database using knowledge that identifies the activity types required to create each component type (stored in the component category objects), and knowledge that describes what activities must precede or follow each activity type (stored in the activity category objects).

3. *InCost*, an integrated estimating application module, assigns costs to the activities to generate an estimate for the project (estimate line items appear as specific types of activities; these may or may not be the same set of activities that represent the construction schedule, for example). The user may add quantity take-off and estimated cost values, or may allow the system to calculate costs automatically from the database's component quantity data and the activity category's typical resource productivity and unit-cost data.

4. Finally, a scheduling application module, *InTime*, calculates schedule dates for activities. Again, data about specific project activities and general activity categories aid in determining durations, precedence constraints, etc.

Note that the system does not require that applications execute in this sequence. For example, the estimating system could be used first. The process of describing estimate line items and linking them to unit-cost data from the activity category objects would lead to the creation of project-specific product, process, and resource objects in the database. Alternatively, the scheduling system could initially help in defining the project objects in the database. Because of the shared database, any operations performed by an application immediately and automatically becomes available to all other applications.

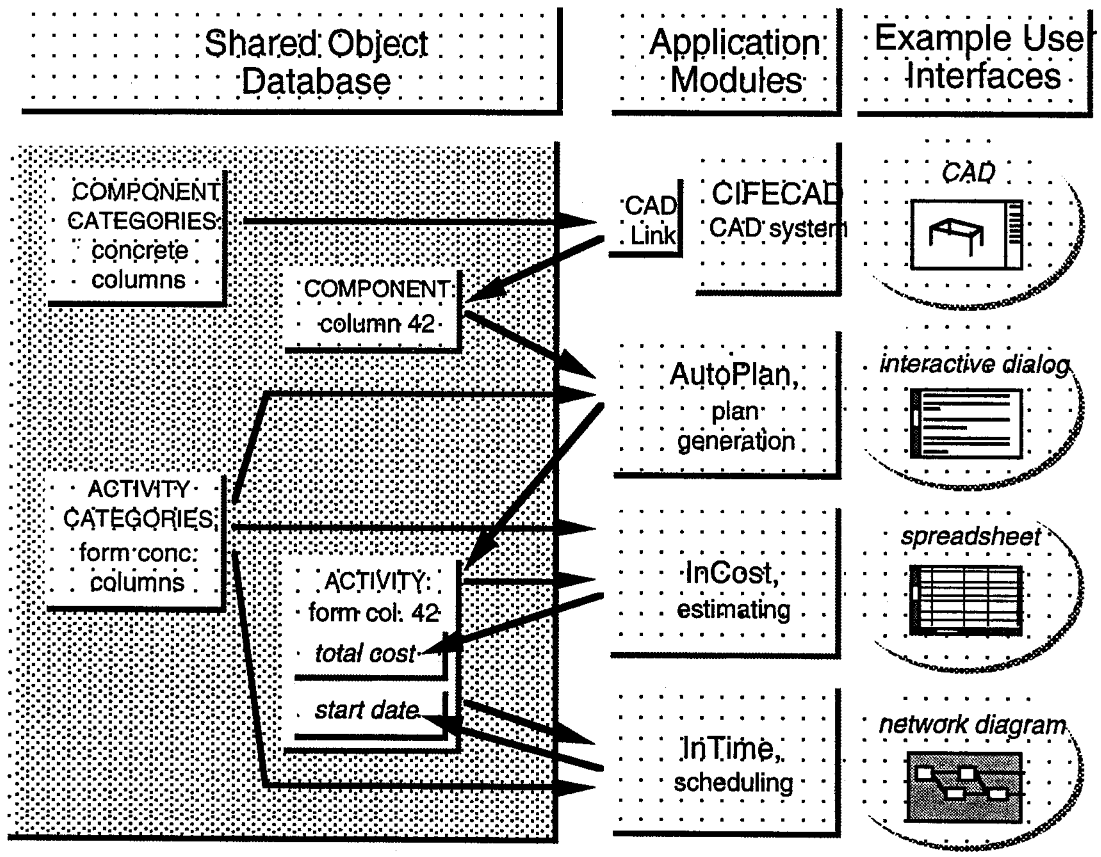

Figure 7. An example of operations on the shared object-oriented database

Summary of Examples

At this point we would like to summarize how these examples illustrate the theme of the paper. Both examples show how standard project models support very different forms of AEC computer integration.

COKE relies only on data available from a CAD system. Reasoning about constructibility requires the availability of specific project data at the appropriate level of detail. Therefore, a project model schema was developed as the integrating mechanism. This schema allows the user to enter project information in the CAD system and to transfer this model to the constructibility expert system. This illustrates:

- integration by transferring a model between applications using a standard schema
- integration in early phases of design of a building project
- integration between CAD and knowledge-based systems
- integration of high-level knowledge and low-level design data
- integration of construction knowledge into preliminary structural design.

OPIS imports the same CAD project model, though the project model can also be created through other mechanisms (during the estimating process, for example). New or updated project data are added to the project model as they are generated by individual applications or entered by the user. Thus, various construction planning applications share this project model that resides in a common database. In addition to demonstrating some of the same concepts as COKE, the OPIS system illustrates:

- integration by sharing a single database among different applications
- integration at the construction planning phase of the project
- integration among several applications using a shared model
- integration at evolving levels of detail.
- integration in a general-purpose system (rather than a system designed to address a specific problem).

These examples demonstrate that a shared project model can provide the common "language" required to achieve computer-supported integration. Such models must be general to allow several applications to communicate with each other and to share the same data. In the next section, we derive specific characteristics that contribute to the generality of such project models.

Characteristics of a Project Model to Support Integration

Project models are intended to provide a high level of integration across AEC computer applications: a role that begs a solution that is "all things to all people." This requires intelligent project models that are able to interpret themselves actively. As the examples above illustrate, this places the following general requirements on project models:

Explicit, Rich, and Accessible

An example of the integration capabilities of current commercial AEC software is the Timberline Software package that can be linked to a graphical model in AutoCAD to identify project components and quantities. It performs estimating in its own precision estimating package, and can export data to Microsoft project or Primavera scheduling systems [Timberline 90]. A very complete model of the project exists between these applications. However the model is not directly accessible as a whole. Graphical data is stored in one application, work package and cost data in a second, temporal and precedence data in a third, and quantity data is stored in two or more different applications. It would be very difficult to add an additional application such as an expert system that required access to all of these data.

In contrast, both systems described in the examples store their complete project models in one location and make their data generally available. We suggest that a project model for supporting integration must be explicit and easily accessible by all applications. That is, the model and all of its elements must be available as separate entities in a system, and it must be clear where each piece of data resides in the system and what form it takes. The models in both example systems are knowledge-rich representations of a project, i.e., project data is captured at the project component level (and not at the raw data level only). The model associates meaning with each data item and contains associations between data items. Clearly, the integrating project model must include the underlying project data at levels of detail that are appropriate for the level and breadth of integration required.

Flexible and Comprehensive

The model will be used by various applications for the communication of a wide range of information. Most of its specific uses have yet to be determined. Thus the model must have great flexibility. That is, it must support integrated systems that have few or many applications, that are used in a single organizational unit or in many different industry segments, and that require and produce vague conceptual information or detailed as-built information. The model must also be able to represent voluminous data, high-level heuristic knowledge, and complex graphical models. The project model must be comprehensive, i.e., it must cover a broad range of data. For example, the model used in the COKE system "knows" only about geometrical and topological project data, whereas the model in the OPIS system includes resource and organizational data.

Application-Independent and Inter-Disciplinary

The project model should cover a broad range of information about the project and should support the many different disciplines that collaborate on AEC projects. For example, it must represent the elements of the facility to be constructed, but it must also represent the construction process used to create the facility, the project's organizational make-up, etc. The OPIS system arranges the project model into several major categories (see Figure 8):

1. The *Product Model* consists of those objects that represent the actual product, i.e., the constructed facility and its physical components. This portion of the project model is shared throughout the project life cycle. For example, 3-D CAD models or models used by architectural and structural design programs focus on the product alone [Lavakare 89], [Phan 90].

2. The *Process Model* consists of objects representing the *construction process* used to create the facility described by the product model. Unlike the product model, the process model is mainly relevant to the construction phase of the project life cycle. We have adopted the "*activity*" as a unit of construction effort. This is a more general concept than a scheduling activity, which is primarily a logically constrained time interval, and it implies no specific level of detail (i.e., tasks and work packages would both be specific types of the general concept "activity").

3. *Resource Model* objects represent anything that can be employed in performance of the construction process.

4. *Organization Model* objects represent projects' organizational elements; for example, the companies participating in the project, the contracts that exist between the participants, etc.

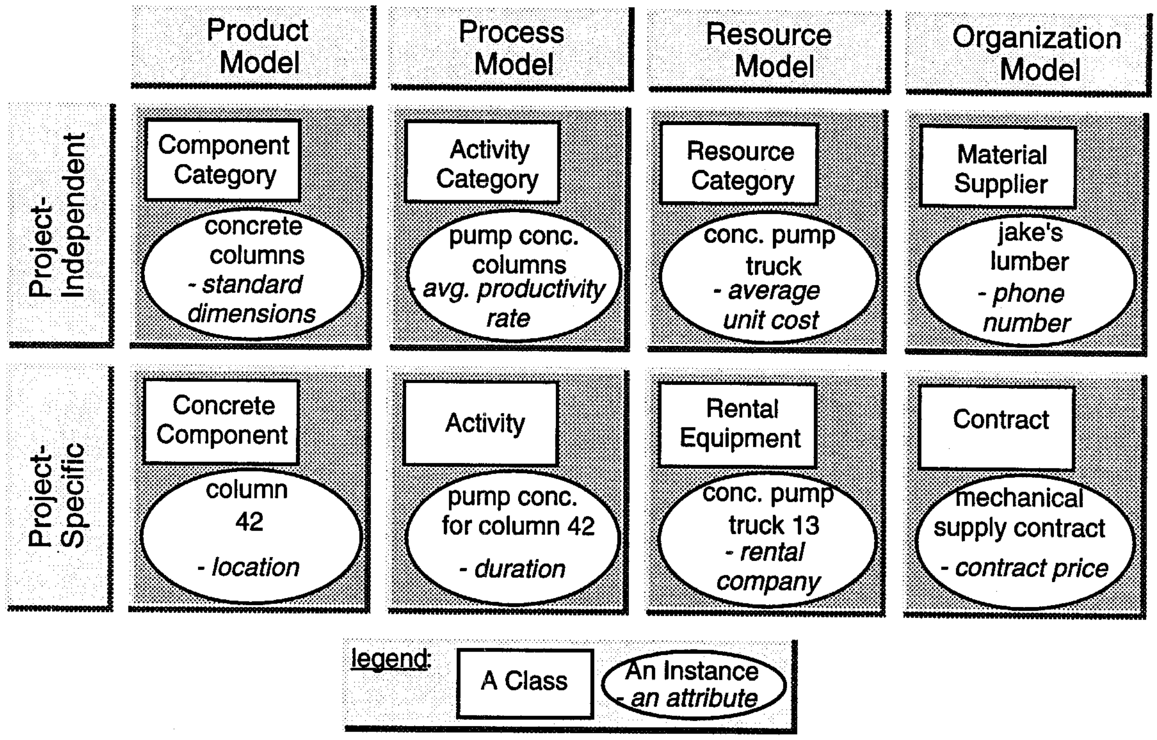

Figure 8. Major Categories of Objects in OPIS' Project Model

Project-Specific vs. Project-Independent

In addition to the above four categories, most objects fall into one of the two following designations:

1. *Project-Independent* objects are those that do not refer to a specific project. These objects often, though not necessarily, represent collections or categories of things, such as a "form concrete activity" object that represents information applicable to all concrete forming operations (e.g., average productivity rates).

2. *Project-Specific* objects relate to one specific project. These objects often represent individual things that are *members of* project-independent category objects. For example, a "pump concrete for column 42" object would represent a specific construction activity on a job, and would have a "member of" relationship to the project-independent "pump-concrete-columns activity" object.

Dynamic

Different views of a project model and its underlying objects are necessary not only because various applications require different data to work with, but also to support the several levels of abstraction a project passes through from inception to completion. For example, design progresses through fairly predictable levels of abstraction. Schematic design is used to capture the basic requirements of a project and to provide sufficient information to communicate these concepts to the client. For this purpose, architects use visualization tools, such as 3D design models, "paste-in" pictures, sketches, etc. Engineers of a process plant, for example, develop layout and piping and instrumentation diagrams (P&ID) that will carry out the chemical and energy processes defined in the process flow diagram for the plant. Such early design tasks are followed by more detailed design phases. The next stages of design complete the design concept sufficient for client review and approval. The succeeding stage permits construction bidding and the last stage (working drawings) is adequate for actual construction work. Each of these phases requires a representation of the project at different levels of abstraction, each of which must be supported by the shared project model.

Implementation Technology

Object-oriented systems provide many of the underlying concepts required to implement product models with the characteristics described above. Their support of abstract data types, object identity, encapsulation, active data, inheritance, and so on makes it possible to model a project explicitly and comprehensively in a computer (Ahmed et al. 1991). These concepts also support the dynamic and inter-disciplinary use of such models throughout the project life cycle.

Conclusions and Future Outlook

This paper discussed two research projects that were recently completed at Stanford University. Both projects developed and used shared project models to integrate various engineering applications. We then outlined the characteristics of such shared project models. We presented these characteristics as an initial list to foster discussion of their usefulness and their implementation with available or future technology. We believe that all these characteristics are essential, though we have not yet formalized how each of them would be implemented and put to practical use. Implementation may be unimportant from a pure research perspective, but we have found that it leads to new ideas, it allows us to communicate with the practitioners who will ultimately use these advanced tools, and it helps us to build bridges to computer science researchers since we can demonstrate our application of their concepts and tools.

We are convinced that shared project models can contribute significantly to increased productivity and quality in the project delivery process. However, the discussion and examples presented above show that international and inter-disciplinary collaboration is necessary to further the development and use of shared project models. We must address the need for modeling and data standards and the differences between traditional application developers and data modeling experts. We must investigate parallel work such as the product modeling efforts of PDES/STEP [Gielingh 88] or enterprise-integration efforts in other industries. Finally, we should step back and assess how computer integration supports overall integration among a project team's individual members, within a company, and throughout the AEC industry.

Acknowledgments

Research funded by Stanford University's Center for Integrated Facility Engineering (CIFE) and National Science Foundation grant #MSM-91-14095. Portions of this paper have been based on [Fischer 91a] and [Froese 92].

References

[Ahmed 91] Ahmed, S., Wong, A., Sriram, D., and Logcher, R. "A Comparison of Object-Oriented Database Management Systems for Engineering Applications." *IESL Research Report*, R91-12, MIT., 1991

[Björk 89] Björk, B. C., and Penttilä, H. ."A Scenario for the Development and Implementation of a Building Product Model Standard." *Adv. Eng. Software*, **11**(4), 176-186, 1989

[Fischer 91a] Fischer, M., "Constructibility Input to Preliminary Design of Reinforced Concrete Structures." *CIFE Technical Report*, No. 64, Stanford, 1991a

[Fischer 91b] Fischer. M. , "Reasoning about CAD Data," in L. M. Chang (ed.), *Preparing for Construction in the 21st Century*, Proceedings of the ASCE Construction Congress '91, Cambridge, April 13-16, 1991, 318-323, 1991b

[Froese 92a] Froese, T. M. ., "Integrating Project Management Software through Object-Oriented Project Models." *Ph.D. Thesis*, Dept. of Civ. Engrg., Stanford University, In preparation, 1992

[Froese 92b] Froese, T. M., and Paulson, B. C. ,"Integrating Project Management Systems Through Shared Object-Oriented Project Models." *Proceedings of AIENG 92, the Seventh International Conference on Applications of Artificial Intelligence in Engineering,* Waterloo, Canada, July 14-17, 1992.

[Froese 91] Froese, T. M., and Waugh, L. M.. "Project Management and Computers in the Year 2010." *Proceedings of the 1991 Annual Conference of the Canadian Society for Civil Engineering,* Vancouver, Canada, May 29-31, 1991, 435-444. Also published as *CIFE Technical Report*, No. 46, Stanford.1991

[Gielingh 88] Gielingh, W. , *General AEC Reference Model (GARM): an aid for the integration of application specific product definition models.* Unpubl. PDES/STEP working document, 1988

[Howard 89] Howard, H. C., Levitt, R. E., Paulson, B. C., Pohl, J. G., and Tatum, C. B. , "Computer-Integrated Design and Construction: Reducing Fragmentation in the AEC Industry." *J. of Computing in Civ. Engrg.*, ASCE, **3**(1), 18-32, 1989

[Howard 89b] Howard, H. C., and Rehak, H. C., "KADBASE: A Prototype Expert System-Database Interface for Engineering Systems," *IEEE Expert*, **4**(3), 65-76, 1989

[Ito 89] Ito, K., Ueno, Y., Levitt, R. E., and Darwiche, A. , "Linking Knowledge-Based Systems to CAD Design Data with Object-Oriented Building Product Model." *CIFE Technical Report*, No. 17, Stanford, 1989

[Lavakare 89]	Lavakare, A., and Howard, H. C. "Structural Steel Framing Data Model." *CIFE Technical Report*, No. 12, Stanford, 1989
[Phan 90]	Phan, D. H., and Howard, H. C..,"Evaluation of the Structural Steel Framing Data Model." *CIFE Technical Report*, No. 41, Stanford, 1990
[Tarricone 92]	Tarricone, P. "Howdy, Partner," *Civ. Engrg.*, **62**(3), 72-74, 1992
[Tatum 87a]	Tatum, C. B. "Improving Constructibility During Conceptual Planning." *J. of Constr. Engrg. and Mgt.*, ASCE, **113**(2), 191-207., 1987a
[Tatum 87b]	Tatum, C. B., "The Project Manager's Role in Integrating Design and Construction." *Proj. Mgt. J.*, **13**(2), 96-107, 1987b.
[Tatum 89]	Tatum, C. B., "Management Challenges of Integrating Construction Methods and Design Approaches." *J. of Mgt. in Engrg.*, ASCE, **5**(2), 139-154., 1989
[Timberline 90]	Timberline *Precision Estimating Plus.* User Manual, 1990