PMAPM:
An Object Oriented Project Model for A/E/C Process with Multiple Views

by
Kenji Ito, Kincho H. Law and Raymond E. Levitt

TECHNICAL REPORT
Number 34

July, 1990

Stanford University
PMAFM:
An Object Oriented Project Model
for A/E/C Process with Multiple Views

Copyright © 1990 by
Kenji Ito, Kincho H. Law and Raymond E. Levitt

If you would like to contact the authors please write to:
c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020
PMAPM: An Object-Oriented Project Model for A/E/C Process with Multiple Views

Kenji Ito\(^1\), Kincho H. Law\(^2\) and Raymond E. Levitt\(^3\)

Abstract

Generating, sharing and maintaining project data among multiple disciplines and throughout a project life cycle is a difficult task in the highly fragmented Architecture/Engineering/Construction (A/E/C) industry. Presently, the A/E/C industry lacks a common standard for exchanging project data to allow applications to share graphical and non-graphical information produced and consumed at the various stages of a project cycle. The object-oriented paradigm provides a useful mechanism to organize and structure complex information and offers a powerful approach for sharing information and knowledge. This paper describes an attempt to develop an object-oriented project model to facilitate information sharing among multiple disciplines and across project stages from planning, design, construction to management of a facility. The feasibility of employing this model for information transfer from CAD to various knowledge-based expert systems will be discussed.

Key word


---

\(^1\) Senior Research Engineer, Shimizu Corporation and Visiting Research Fellow, Center for Integrated Facility Engineering, Stanford University

\(^2\) Assistant Professor of Civil Engineering, Stanford University

\(^3\) Professor of Civil Engineering and Associate Director, Center for Integrated Facility Engineering, Stanford University
1. Introduction

Large amounts of information are generated and consumed during the various phases of a project life cycle from planning to design, construction and management of the facility. Sharing and maintaining these project data among multiple disciplines and throughout a project life cycle is a complex and difficult task. The project data needs to be stored, retrieved, manipulated, and updated by many participants, each has his/her own view of the information.

A number of computer-based systems such as CAD, analysis and simulation software have been developed for the A/E/C industry. However many of these systems can only be utilized within narrow application domains. Efforts have been attempted to integrate these application software by linking systems and providing data transfer interfaces so that a richer communication among applications of different domains can be realized. Efforts to establish data exchange standards for CAD applications have also been initiated [Howard 89, Warthen 88]. Most of these efforts are still in the early stages of development. We believe that a project model that can properly describe a facility and is accessible by multiple participants of different disciplines is a very important ingredient for integration.

Object-oriented programming paradigm offers many useful features to facilitate the description of a constructed facility. Some of the benefits include:

- To define data, programs, or data and programs in a unified manner as objects;
- To store an atom, symbol, list, character strings, method or function in the object as the value of each slot;
- To inherit information from the abstract level to the detail level;
- To define a relationship, such as supported_by, consists_of or others, using the slots of a related object.

With these features, we can define several kinds of objects. In our model, we define four kinds of objects -- data-object, scope-view-object, eval-view-object and create-view-object. These four kinds of objects are collectively termed multiple purpose objects. In addition, the relationships among the objects are defined not only for inheritance, demons or methods of an object but also to facilitate the search for information.
The scope of our research is to establish an object-oriented project model supporting multiple views that are shared by the various participants of an A/E/C project. The objective is an eventual development of an integrated system which includes activities from project planning through design, estimation, construction and facility management. The object-oriented project model is intended to link CAD systems, relational databases, knowledge-based systems and other conventional application software; these applications can easily be treated as a view-function embedded in a view-object.

2. Related Research

In building design, we can benefit from using databases for storing a large amount of independent but interrelated data about design elements and their relationships. The objects observed in a building project are best modeled by an object-oriented model. Elements (such as columns and beams) can be defined as individual object types, and their properties (such as length and weight) are defined as attributes. Operations on the objects can be defined using methods. Many researchers have attempted to develop product models or project models using the object-oriented methodology. In this section, we briefly review some of the relevant work related to product and process modeling of a constructed facility.

To manage the vast volume of data inherent in building design, a formal representation is needed to organize the data. The concepts of aggregation and generalization are useful tools for describing design information [Eastman 78, Law86, Garrett89]. The database integrity issue of this abstraction concept for engineering modeling has been studied and an implementation using the object-oriented programming paradigm has been proposed [Law 87]. Recently, a formal approach based on a semantic structural data model to manage engineering design objects using a relational framework has been introduced [Law90]. The benefit of this data modeling approach is that information can be shared among different applications through multiple views while maintaining data integrity in the database.

Recently, product models for building design have attracted considerable interest. An object-oriented data model for steel frame structures using a top-down approach according to the concept of PDES (Product Data Exchange Specification) has been proposed by Howard and Lavakare [Lavakare 89]. A significant effort is currently underway at the Technical Research Centre of Finland to develop a building product model (RATAS) using relational databases and hypermedia [Pentilla 89]. Their main target is to describe the architectural elements as the product model and to store this model into a relational database with relationships among elements.
Recently, a formal approach for product modeling of building design information has been proposed by Eastman et.al. [Eastman 90].

Despite all the attention that has been devoted to modeling product design information, additional research is needed to establish a product model that not only represents the physical elements but also the processes involved in an A/E/C project. In structural engineering, Sause and Powell proposed a design process model based on a problem decomposition approach [Sause 90]. A top-down approach for the integration of the building process has been suggested by Sanvido et. al. [Sanvido 89]. In this paper, we describe an object-oriented concept for modeling the product design information as well as the processes, using a multiple view approach. We believe that this object-oriented multiple-view approach is sufficiently flexible to support the development of a project model for an integrated environment for planning, design, construction and management of a constructed facility.

Fenves and his colleagues [Fenves 88] are developing an integrated environment for building design and construction (IBDE) using a number of AI techniques to model the process and information flow from architecture design to construction planning. Presently, IBDE consists of seven knowledge-based modules: ARCHIPLAN (Architectural Planning), CORE (Spatial Layout), STRYPES (Structural System), STANLAY (Preliminary Analysis), SPEX (Structure Component Design), FOOTER (Preliminary Design of Foundation), and PLANEX (Construction Planning System). They have successfully developed a partial integration of the facility engineering process, combining knowledge-based expert systems and conventional systems used from conceptual design through construction planning. However, presently this integration is a batch serial process and does not use the concept of a product model.

Another related effort on object-oriented CAD is the BUILDER system which generates a symbolic model as a graphic drawing is created on a LISP machine [Cherneff 88, Logcher 89]. The data generated through the manipulation of the KEE-Pictures interface becomes an object-oriented model within the KEE environment. The object model then serves as the input to a prototype system for knowledge-based planning and scheduling implemented in KEE. Moreover, the concept of this object-oriented model has been incorporated in an integrated engineering environment (DICE) as part of its global model [Sriram89]. In DICE, an architect and a structural engineer each has his/her own object hierarchy to access the global data model, installed in the GEMSTONE database system. Using multiple object hierarchies and a blackboard architecture as the control mechanism, DICE provides a constraint management environment to resolve potential conflicts among several participants. This concept is very similar to ours in defining multiple
views for project participants. However, we primarily focus on the development of a project model that supports an integrated system environment.

3. An Object-Oriented Project Model Supporting Multiple Views (PMAPM)

The object-oriented project model (PMAPM) described herein is an extension of a project model (PMAP) initially developed as a vehicle to share project information for a mid-rise building project between a CAD system and two knowledge-based construction planning systems [Ito 89, Ito 90]. While PMAP was implemented using Framekit [Nyberg 88], PMAPM has been implemented using Allegro Common Lisp, Parmenides and FRulekit [Shell 88, Shell 89] on a Macintosh II computer. The system supports not only graphic information but also the non-graphic or attribute information including the relationships among the building elements. In this section, we first review the basic capability of PMAP. We then describe the extension of the model to support multiple views and the implementation of multiple purpose objects.

3.1 PMAP - An Object-Oriented Project Model

In the PMAP model, an object represents a building element or a non-building element. Each building element consists of eight basic attributes: material, finish, size, position, rotation angle, offset and relation. The attributes of a non-building element include: project code, project name, user id, client name and site location. Included in each building element is its relationship with other elements. Some examples of the relationships between the building elements are:

- Girder is supported by Columns.
- Beam is supported by Columns or Girders.
- Exterior Wall is connected with Columns or Exterior Walls.
- Interior Wall is connected with Columns, Exterior or Interior Walls.
- Slab is supported by Girders or Beams.
- Door and Window is attached on Exterior or Interior Walls.
- Space (Room) is consisted by Exterior or Interior Walls.
- Floor is consisted by Spaces.

Four basic access mechanisms are available for the user to extract object data:
Using the frame or slot access functions in FRulekit and Parmenides with methods embedded in the object. For example, the size of an object can be determined with the following (Lisp) function:

```lisp
(defun get-size (frame-name)
  (get-value frame-name 'size))
```

Figure-3.1.1 shows a function to access the size of a specific column by providing the column's name as the value of the "frame-name" argument.

Using the relationship among the building elements. Figure-3.1.2 illustrates the extraction of information about an object via the relationship definitions. In this example, a room is said to be enclosed (consisted) by a series of walls (see Figure-3.1.2a). In Figure-3.1.2b, an opening (such as a window or a door) is defined as attached to a wall. The composition of these relationships then yields the information about the openings of a room (see Figure-3.1.2c). In fact, we can issue a query to request the information about the size of a room's opening.

Generating a file as an interface to an external application program. For example, if a construction planning system needs information about a specific relationship between the building elements, the planner can issue the relationship such as supported-by and specify an output file:

```lisp
(get-relationship-inf 'supported_by 'list-form "temp.txt")
```

The system will look for the "supported-by" relationship from all object instances in a product model and return the values in terms of a series of list-expressions like the following

(Supported_by Girder-1 Column-1 Column-2)

These lists will be put it into the file "temp.txt". As will be seen in the next section this is the mechanism that has been used in integrating PMAP with AutoCAD via a CIFECAD interface program. If the user does not specify the file name, the system will show the value on the screen window.

Using the graphic interface of PMAP as shown in Figure-3.1.3. PMAP has been used as an interface to various knowledge-based systems, a CAD system, and a relational database system (ORACLE). The basic configuration of PMAP as an interface to multiple application programs is schematically depicted as shown in Figure-3.1.4.
3.2 PMAPM: An Object-Oriented Project Model with Multiple Views

Each participant of an A/E/C project has his/her own view of the information about a constructed facility. In general, for each project, there is one facility and the object representing the facility has only one physical value but many functional values depending on the various views of the information by the participants of different disciplines.

In the PMAP model, we define an object hierarchy describing the physical elements used in the design and construction of a facility. Only three views -- architect's view, structural engineer's view and construction engineer's view -- about the physical objects are currently defined. As shown in Figure-3.2.1, in PMAPM, a project can be viewed in terms of a global view (concerned with the social and labor types of information), a project view (defining the perspective of various phases of a project) and an object view (defining the physical building elements of the facility). While the object view representing the physical elements is similar to the object hierarchy in PMAP, we introduce a global view and a project view in PMAPM to represent the functional views corresponding to the processes of an A/E/C project. In an object oriented programming environment, we can treat the views as objects defined with data and methods.

In order to fully realize the model description of an A/E/C project, we need to consider not only a product-based description but also a process-based description. From the process point of view, a project consists of the following stages corresponding to the view definitions in our PMAPM model:

- Project Management View
- General View
- Project Planning View
- Sales View
- Design View
- Estimation View
- Construction View
- Facility Management View
PMAPM: An Object-Oriented Project Model for A/E/C Process with Multiple Views

• Maintenance View

The project management view (for the project manager) and the general view (to store the general information about the project) provide the high level control and support of project information. The other views about a project follow the various phases of an A/E/C process. For each view object, we can further define object subclasses. For example, the design view can be divided into six sub-views:

• Conceptual View
• Architecture View
• Structural View
• Mechanical View
• Electrical View
• Coordination View

The coordination view is intended to support negotiation when conflicts occur among the various design views. Figure-3.2.2 shows the decomposition of views into subviews and the definition of slots for each view. Details of the PMAPM hierarchy that have been implemented are summarized in Appendix-A.

PMAPM includes all the access mechanisms of PMAP as described in the earlier section. The major difference between PMAPM and PMAP is the inclusion of the view definition for each participant in the A/E/C process. A view consists of not only the data but also many kinds of methods, functions and rules in order to extract or update the information according to each participant's needs. For each view, PMAPM needs to assign to the user the mode of access (i.e. update or read only) as illustrated in Figure-3.2.3. Most users (except for the project manager and PMAPM designer) have only limited access to the various views. For example, a sales person can access the global-view, management-view, general-view, planning-view and sales-view with the read-only mode but is allowed to access sales-view and general-view with the privilege of updating the information. Similarly, a client can access to the global-view, management-view, general-view and facility-management-view with the read only mode but is allowed to access the facility-management-view with the update mode. This view facility can facilitate the management and presentation of information to the users.

3.3 An Illustrative Scenario for Multiple View Definition
PMAPM: An Object-Oriented Project Model for A/E/C Process with Multiple Views

The multiple view definition and the data access mechanisms described in the previous section may best be illustrated using a scenario for a simple project. At the beginning of the project, the project manager was assigned and the project code and the project name are entered into the PMAPM model. The sales person obtains information about the clients (number of clients and their names) and the site location and stores this information into PMAPM through the general-view using the update mode as shown in Figure-3.3.1.

From the project site information, the project manager can now evaluate the applicability of various structural and construction methods, extract information from the “Structural and Construction Methods Database” and store this information into the structural-technology-view and construction-technology-view under the technical-strategic-view of the management-view (See Figure-3.3.2). The state-of-the-art information about the material and facility (such as smart building or automated factory) are stored in the new-material-view and the new-facility-view. Some of these evaluation functions are implemented in terms of production rules which are defined as methods imbedded in an object and can be accessed from the multiple views through messages. For example, a construction method selection expert systems may be used by the project manager, the structural engineer, the construction manager or construction engineer; this expert system may use the information created at every stage of the AEC process. As the project proceeds, the system will provide increasingly reliable and appropriate selection as the amount of unknown and missing information decreases and as the quality of the information improves (see Figure-3.3.3). This illustrates the importance of the process-based view description of a project model.

With the information about the available technology, the sales person can extract the information (such as the name of the technology, applicable domain and the characteristics of the technology) from the sales-technology-view of the sales-view so that they can explain the technology to the client. After the decision on the technologies to be used in the project has been made, the sales person stores this information as the sales-design-construction-constraints in the project-construction-views under the general-view. These constraint information will then propagate to the other views as depicted in Figure-3.3.4.

At the structural design stage, when the structural engineer recalls the structural-constraint-view, a method would be initiated to check upper-level constraints that would affect the structural design process. The structural engineer can further re-evaluate the structure and construction methods from the structural-technology-view via the structural-building-view. It is worth noting that the sales person and the structural engineer both have access to the same object but obtain different information according to their own view.
3.4 Multiple Purpose Objects (MPO)

In order to realize the multiple views that are needed to capture or update the model information, we use the notion of multiple purpose object (MPO) in our project model. A multiple purpose object can be one of the following four basic object types (see Figure-3.4.1):

1. Data-Object: This object contains the basic data, such as numeric, symbol, character strings or list expression (data list). The following objects are defined as data object:

   1.1 Class or subclass object which does not have any functions or rules defined as its slot value. For example, "project" is a data object and has 5 slots for `project code, project name, start date, end date` and `instance of`; each slot carries a data item as its value.

   1.2 Most of the object instances defined under object view are data objects. For example, the object instance "Columns-1" has many slot entries including `size, position, material`, etc.; each slot carries a single datum as its value.

2. Scope-View-Object: This object type contains both data and functions written in Lisp or some other language. With the functions, the object can extract data from other objects or update the slot value of its own or other objects. In addition, some of these functions can create or remove a data object. For example, the column-object has not only the default value to be inherited by its instances but also a method to create, upon request, an instance automatically.

3. Eval-View-Object: This object type contains not only data or functions but also rules written in Lisp or another language. Production rules are embedded in an object as slot values. The difference between the functions in the scope-view-object and the rules in the eval-view-object is their purpose and use. The purpose of a function is to extract or to update the slot value with some calculation, while the purpose of a rule is to evaluate not only the slot value but also the object, group of objects or relationships of objects. For example, the progress-view is an eval-view-object consisting of six slots:

   - `Start_Date` (Date1 Date2)
   - `Required_Duration` (Duration1 : pre-if-set 'ask-duration1)
   - `Temporal_Duration1` (Duration2 : pre-if-set 'ask-duration2)
   - `Temporal_Duration2` (Duration3 : pre-if-set 'ask-duration2)
• Fixed_Duration (Duration4: pre-if-set '(ask-duration3))
• Instance-of (Management-View)

In the above definition, "Date1" is a start date of the project and its value is stored by the project manager using the Start_Date value of the project-view. "Date2" is a start date of the project after contract agreement with the client and its value is stored by the sales person or the project manager. "Duration1" is a required duration by the client; when the user asks for the Required_Duration, the method named "ask-duration1" sends a message to the required-duration slot of the sales-view, extracts the data from the other object and stores the data into its own slot. This procedure is a common way of using a method or demon in an object-oriented environment. For "Duration2", "Duration3" and "Duration4", however, we define the functions in terms of rules. These functions evaluate the preliminary duration which in turn depends on the total area of the building, the ground condition of the site, number of the stories and usages of the building etc.. Such information may at times not available depending on the current phase of the A/E/C process. This function must be able to respond to known and missing information using heuristic knowledge acquired from a human expert, in this case, to estimate the duration based on the information available.

4. Create-View-Object: The main purpose of this object type is to create a new object or slot according to the user's requirements. Although PMAPM has its own object hierarchy which is designed to support a wide variety of building projects, this hierarchy cannot exhaustively include all the possible views required by each project participant. Therefore, PMAPM provides a method to define a new Scope-View-Object. By turning on a trace mode, PMAPM will make a trace of the user's operations to extract the data from other objects and will store the set of operations as a function in the new object.

This concept of a "multiple purpose object" provides the flexibility to accommodate unforeseen situations and to expand PMAPM's applications in the future.

4. Automatic Model Generation with CAD Application

PMAPM, as an extended version of PMAP, includes a high level user interface, CIFECAD, for defining building elements and their attributes. CIFECAD is a customized AutoCAD system written in AutoLisp [Ito 89, Ito 90]. AutoCAD is basically a computer-aided drafting system rather than a computer-aided design system. Although several third party vendors have developed interfaces to facilitate drafting in specific domains, there is no high level way to input building elements, such as "Column", "Exterior Wall", "Beam" etc., even in graphical form via the original AutoCAD interface. In order to define non-graphics or attribute information to the object...
model, we need an interface to input these building elements. In our approach, we first define the building elements using the BLOCK data storage facility of AutoCAD with attributes. CIFECAD has its own object hierarchy definitions (See Figure-4.1.1) and its own high level interface to carry out the A/E/C design (See Figure-4.1.2). AutoCAD data are stored in a binary file termed a .DWG file. To extract data out from the .DWG file, a set of AutoLisp functions have been implemented to read the .DWG files and write the information in ASCII format onto an external file. With CIFECAD, a user can input the building elements with non-graphic information and obtain the drawing information not only in the DXF format for AutoCAD but also in customized ASCII formatted list-expression for other application programs.

Currently, CIFECAD supports the definition of the following elements and their attributes:

- **Columnline**
  Number of x-Columnline, Span size of x-Columnline, Number of y-Columnline, Span size of y-Columnline, Coordination.

- **Foundation**
  Center position, Size, Material, Intersection of Columnline.

- **Foundation Beam**
  Size (W x D), Material, Start and End position, Connection Information, Offset, Rotation Angle and so on.

- **Column**
  Size (X x Y), Material, Center position, Finish code, Offset, Connection Information, Rotation Angle, Floor name and so on.

- **Round Column**
  Same as Column.

- **Exterior Wall**
  Thickness, Material, Start and End position, Finish code, Offset, Connection Information, Rotation Angle, Floor name and so on.

- **Interior Wall**
  Same as Exterior Wall.

- **Girder**
  Size (W x D), Material, Start and End position, Finish code, Offset, Connection Information, Rotation Angle, Floor name and so on.

- **Beam**
  Same as Girder.

- **Slab**
  Thickness, Material, Finish code, Floor level, Connection Information, Area, Floor name and so on.

- **Roof**
  Same as Slab.

- **Window**
  Width, Height, Floor Level, Type, Connection Information, Floor name, Center position and so on.

- **Door**
  Same as Window.
PMAPM: An Object-Oriented Project Model for A/E/C Process with Multiple Views

- **Space**
  Name, Finish Code, Uses, Connection Information, Floor name, Area, Floor Level, Ceiling Height and so on.

- **Floor**
  Connection Information, Area, Floor Height, Floor Level and so on.

- **Desk**
  Size (X x Y), Center position, Connection Information, Material, Type, Floor name and so on.

- **Table**
  Same as Desk.

Although CIFECAD has many limitations, it does have sufficient functions and capabilities to be used as a computer aided design system with object data structure and high level user interface suitable for a prototyping environment.

PMAPM obtains most of the physical object information about a building from CIFECAD. The major building elements of the architectural and structural design can be defined using CIFECAD and the information about these objects is stored under the object-view in PMAPM (see Figure-4.1.3). The procedure for model generation from the CIFECAD to PMAPM can be summarized as follows:

1. Create the drawing using the CIFECAD and its high level user interface. (See Figure-4.1.4)

2. Create a temporary file to store list-expressions derived from the drawing data using AutoLisp functions.

3. PMAPM reads the list-expressions and stores the information into the object model.
   3.1 Read the general description and check if the value exists in the target slot. If there is no value in the model, the system stores the data into model. However, if there is already a value in the target slot, the system would request whether the previous information should be replaced.

   3.2 Read the default value of each element. The process of checking existing value is the same as above.

   3.3 Read the number of each building element, and create an instance for each element.

   3.4 Store the relationship information into the instances.

Hence, the user does not need to worry about transferring design data from CIFECAD to the PMAPM. This process is automatically done by a file interface between CIFECAD and PMAPM.
(see Figure-4.1.5). In future efforts, we will try to obtain design data from other CAD systems into the PMAPM object model.

5. Overview of the Data Storage Mechanism in PMAPM

Including the interface with CIFECAD, there are altogether five different ways that one can store the data into PMAPM:

- Enter data using the textual interface of PMAPM. (see Figure-5.1.1)
- Create and store data using the graphical interface of PMAPM.
- Transfer data from the relational database system; presently, we are using ORACLE for storing cost, actual result, personnel and client information in the PMAPM environment.
- Transfer data from a CAD system, such as CIFECAD.
- Create and store data using a method or rule.

These interface facilities allow PMAPM to capture project information from the participants of different disciplines who are involved in an A/E/C project. PMAPM can be used in the planning and design phases in the following manner:

At the beginning of a project:

- The project manager defines the project using textual interface of PMAPM;
- The sales person inputs the client and site information using textual interface of PMAPM as well as the relational database interface;
- The project evaluates and stores the duration information using rules and methods of PMAPM;
- The sales person stores the client's requirements for usages of the building using the textual interface of PMAPM;
- The sales person evaluates and stores the information about the building usages and the building cost using rules, methods or other application programs.

During the preliminary design phase:

- The architect evaluates the site and stores the information using a CAD system or other application programs.
- The architect considers the layout and the grade of the building using CAD or application programs.
The structural engineer evaluates and stores the above-ground as well as under-ground information using the relational database interface, application program and the textual interface of PMAPM.

The mechanical engineer evaluates the concept of the heating system using the relational database interface, rules and other application program, etc...

During the conceptual design and design development phase:

- The architect and engineers store the graphic information using the CAD interface.
- Engineers evaluate and store data using the rule facility of PMAPM, its relational database interface and other application programs.
- The project manager evaluates the duration information using PMAPM's rule facility.

In this process, there will be information conflicts among the project participants. These conflicts are stored as constraints in views and propagated and resolved by the constraint management system in PMAPM.

6. An Example of Using PMAPM

In this section, we describe the use of PMAPM using a small room in an office building as an example. The process from the planning phase to the construction phase will be described to demonstrate the various views of the project model. Furthermore, the information sharing among the multiple participants and the mechanism for information retrieval using views will be illustrated. In this example, the participants are an architect and a structural engineer, and the shared object is a room of an office building. The shape of the room is shown in Figure-6.1.1 and the basic data information are stored in PMAPM as shown in Figure-6.1.2.

Various disciplines require different types of information about a room. For architectural design, an architect may be interested in attribute information about the room (such as area, uses, grade, color, ceiling height and other spatial information), its elements (such as the enclosing walls, openings, size, finish, color, etc.) and cost information (about the materials and finishes). On the other hand, a structural engineer may be interested in the overall system of the room (including the basic structure of the room, span size, live loads, etc.), its structural elements (including size of each element, material, bearing wall if any, opening information, etc.). The architect and the structural engineer can access these data directly from PMAPM through each view, (such as architectural-space-view, structural-element-view and so on).
As an example, Figure-6.1.3 shows how each participant can extract or update the PMAPM information through his or her own view. From the architectural-space-view, the user can access information as follows:

- area of the room  “38400 square inches” via path-A.
- uses of the room  “OFFICE” via path-A.
- basic color of the room obtained from the material of finish via path-A, path-C and others
- ceiling height of the room  “120 inches” via path-A.

From the architectural-element-view, the following information can be accessed as follows:

- the walls enclosing the room  “E_Walls-11, E_Walls-12, I_Walls-6, I_Walls-7” via path-A.
- openings on the enclosing walls  “Windows-10, Windows-11, Doors-20” via path-A, path-C and path-E.
- size of each element  via path-A, path-C, path-D, path-E and others.
- finish of each element  via path-A, path-C, path-D, path-E and others.

From the architecture-cost-view, the architect can obtain the cost information in the following manner:

- cost of material  obtain the cost of each material via path-A, path-C, path-D, path-E and others.
- cost of finish  obtain the material of finish and cost of finish via path-A, path-C, path-D, path-E and others.

Through the structural-space-view, the structural engineer can acquire the following information:

- basic structure of the room  “Reinforced-concrete” via path-B, path-C, path-D, path-E and others.
- basic span size  obtain the columnline information by via path-B path-C, path-D and others.

From the structural-element-view, the following information can be obtained:

- size of each structure element  “30.0x30.0” via path-B, path-C, path-D and others.
- material of each structure element  “Reinforced-concrete” via path-B, path-C, path-D and others.
That is, multiple participants of a project are sharing the same information of the design and they can extract or update the information of PMAPM according to his/her own view.

7. Other Application Programs Interfacing with PMAPM

OARPLAN: A Knowledge-Based construction planning system.[Darwiche 89]

OARPLAN is a prototype knowledge-based construction planning system based on the notion that activities in a construction project plan can be viewed as intersections of objects, actions and resources at several different levels of abstraction. OARPLAN generates the needed activities in a project plan by elaborating a high level activity such as <build> <building> into a set of project activities at one or more finer levels of detail, guided by activity scale reduction and activity sub-plans. Therefore, OARPLAN needs design data to generate the activities in a project, and to generate the needed sequencing constraints among the activities. In general, OARPLAN is used mainly at the end of a design stage or at the construction planning phase. That is, OARPLAN has its construction-planning-view for extracting information from PMAPM. In fact, a user can use OARPLAN from the construction-planning-view of PMAPM through the system menu. The result of an activity network generated by OARPLAN is illustrated in Figure-7.1.1.

SIPEC: Knowledge-Based construction planning system.[Kartam 89, Kartam 90]

SIPEC is a prototype knowledge-based construction planning system using SIPE (System for Interactive Planning and Execution) which developed by SRI International. SIPE is a domain-independent AI planner, to hierarchically planning projects involving multiple agent tasks. Although SIPEC has the same construction-planning-view as OARPLAN, the needs of information format is different from OARPLAN. PMAPM can support these different needs from different applications because of its multiple views function and object hierarchy.

FCOST: Finish cost Estimation and Counseling expert system.

FCOST consists of two basic modules, FINISHES and FINISHCST. FINISHES is a knowledge-based counseling system which evaluates the finishing material based on the usage of the room and the grade of the building. This information can be directly accessed through the architectural-view of PMAPM (See Figure-7.1.2). FINISHCST is a cost estimation system which calculates the preliminary finish cost using the information from the detail-cost-view. The information required by FINISHCST can be obtained via the architectural-view and the estimation-
view. Both FINISHES and FINISHCST can be directly accessed from the system menu of PMAPM.

**DURATION: Knowledge-Based Construction Duration Evaluation system.**

DURATION consists of a set of rules to evaluate the construction duration from the total area, floor area, building structure, ground condition, etc. After the evaluation, the following duration information is given to the user:

- Under Ground Construction Duration.
- Super Structure Construction Duration.
- Finish Work Duration
- Total Duration.

A user can run this system from many views or directly from the PMAPM's system menu.

8. Conclusion and Discussion

In this paper, we have described an object-oriented project model that supports multiple views for building projects. We have implemented the basic model hierarchy of PMAPM as described in Appendix-A, and integrated it with various applications as described in Sections 4 through 7. Various interfaces to store and access information from PMAPM have also been developed and illustrated. The overall integrated system is schematically depicted in Figure-8.1.1.

During the course of developing PMAPM, we have found many benefits of using an object-oriented paradigm to describe project model and to provide multiple views of the project model. Particularly, with the definition of multiple views, we can analyze and recognize the various requirements throughout the A/E/C process. Furthermore, this process-based approach is very useful to discover the information flow among the participants from different disciplines. In summary, we have demonstrated two important concepts:

- The usefulness of an object-oriented project model for system integration in an A/E/C project -- we need not only a product model but also a project model.
- The usefulness of the multiple view concept for the project model -- we need not only a product-based model description but also a process-based description of a constructed facility.
Finally, our purpose in this work is to realize the plausibility of a project model that can be referenced by multiple disciplines throughout the A/E/C process. In order to achieve this purpose, continuing effort is needed to evaluate the requirements of each discipline and their needs from PMAPM so that the high level project model with multiple views can be established. Furthermore, investigation of the most appropriate view for each domain and a detailed description of the objects are needed to complete the model. Nevertheless, we believe that we have shown the usefulness of our new concept of an object-oriented project model supporting multiple views for an integrated system. We plan to extend the building element types, the information about each element and the views. Furthermore, we plan to continue to examine the feasibility of using PMAPM as an integration tool for other projects, including IPCM (an Intelligent Project and Construction Manager), a Knowledge-based Constructibility Evaluation System [Fischer 89] and PENGUIN (an object interface with relational database system) [Law 90].

Acknowledgment

This work has been supported in part by the Center for Integrated Facility Engineering of Stanford University. At CIFE, we continue to investigate and develop new theory and system for the integration of facility engineering and management.

References


<table>
<thead>
<tr>
<th>Listener</th>
</tr>
</thead>
<tbody>
<tr>
<td>? (get-size 'Columns-1)</td>
</tr>
<tr>
<td>? (35 35)</td>
</tr>
</tbody>
</table>

User can define the menu which call this function

Figure-3.1.1 Example of the Slot Access Function
In this network → means "contains".

**Figure-3.1.2a** Relationship between Space and Wall

**Figure-3.1.2b** Relationship between Wall and Opening

**Figure-3.1.2c** Relationship among Space, Wall and Opening

**Figure-3.1.2** Example of the relationship.
Figure-3.1.3 Example of the Graphic Interface
Figure-3.1.4 System Integration using PMAP
Figure-3.2.1 High Level View Hierarchy in PMAPM
Figure-3.2.2 Example of Subviews
Figure-3.2.3 Occupation Definition in PMAPM
Figure-3.3.1 Initial Data stored into the General View
Management View → Strategic View → Technical Strategic View → Structural Technology View
                        |                                    | Construction Technology View
                        |                                    | New Material View
                        |                                    | New Facility View
                        |                                    | Robotics Technology View
                        |                                    | Computer Technology View
Management Strategic View → Management Tools View
                        |                                    | Quality View

Progress View
- Start Date
- Required Duration
- Temporal Duration
- Fixed Duration

Project Cost View
- Total Budget
- Land Cost
- Building Cost
- Design Cost
- Construction Cost
- Material Cost
- Other Cost (For Risk)

Project Organization View
- Project Manager
- Design Team
- Estimator Team
- Construction Team

Project Risk View
- Financial Risk
- Construction Risk
- Technical Risk
- Other Risk

Constraints View

Figure-3.3.2 View for Structural and Construction Methods
Figure-3.3.3 Example of Sharing the Function
Figure-3.3.4 Example of the constraints propagation. Each slot of Structural-Constraints-View has method which access to the upper level constraints-view.
Figure-3.4.1 MPO Hierarchy and Example of Object.
CIFECAD has the following information:

- Project
  - Project Description
    - Project Code
    - Project Name
    - Site Location
    - Client Name
    - Uses
    - Designer Name
  - Floor Information
    - Each Floor Information as the layer
      - Number of Floor
      - Floor Level
      - Floor Height
      - Ceiling Height
      - From To Floor Name
  - Graphic Information
    - Size of Element
    - Position of Element
    - Material
    - Finish Code
    - Floor Name
    - Rotation Angle
    - Other
  - Connection Information
    - Relation between Elements

Figure-4.1.1 CIFECAD Data Hierarchy
Figure-4.1.2 CIFECAD Interface
Figure-4.1.3 Object Hierarchy under Object-View
Figure-4.1.4 Drawing Example using CIFECAD
<table>
<thead>
<tr>
<th>Defining class COLUMN-21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining class COLUMN-22</td>
</tr>
<tr>
<td>Defining class COLUMN-23</td>
</tr>
<tr>
<td>Defining class COLUMN-24</td>
</tr>
</tbody>
</table>

"END DATA" Defining class E_WALLS-1
Defining class E_WALLS-2
Defining class E_WALLS-3
Defining class E_WALLS-4
Defining class E_WALLS-5
Defining class E_WALLS-6
Defining class E_WALLS-7
Defining class E_WALLS-8
Defining class E_WALLS-9
Defining class E_WALLS-10
Defining class E_WALLS-11
Defining class E_WALLS-12
Defining class E_WALLS-13
Defining class E_WALLS-14
Defining class E_WALLS-15
Defining class E_WALLS-16
Defining class E_WALLS-17
Defining class E_WALLS-18
Defining class E_WALLS-19
Defining class E_WALLS-20

"END DATA" Defining class J_WALLS-1
Defining class J_WALLS-2
Defining class J_WALLS-3
Defining class J_WALLS-4
Defining class J_WALLS-5
Defining class J_WALLS-6
Defining class J_WALLS-7

"END DATA" Defining class GIRDER-1
Defining class GIRDER-2
Defining class GIRDER-3
Defining class GIRDER-4

USER | New size #8(1364 431)

Figure-4.1.5 Object Creation using CAD Data
Figure-5.1.1 Example of the textual interface
Figure-6.1.1 Shape of the Example Room
General Description of Building

Usage of Building: Office Building
Location: Palo Alto, California
Client: American Research Inc.
Structure: SRC
Super Structure: RC
Penthouse: RC

Number of Floor: Under Ground
1
Super Structure: 4
Penthouse: 1

Ground Condition: Good

General Description of Floor

Floor Name: 2
Usage of Floor: Office
Area of Floor: 1400 square ft.
Structure: RC

General Description of Room

Consists_of: 2 Exterior Walls
Material RC, Thickness 10.0
2 Interior Walls
Material RC, Thickness 8.0
4 Columns
Material RC, Size 30.0x30.0
4 Girders
Material RC, Size 20.0x30.0
1 Slab
Material RC, Thickness 12.0
2 Windows
Width 40.0, Height 40.0
1 Door
Width 120.0, Height 80.0

(Unit Inch)

Figure-6.1.2 Description of the Example
Figure-6.1.3 Example of the Data Search Path for every View
Figure-7.1.1 OARPLAN Operation Screen
Figure-7.1.2 Example of FINISHES
Figure-8.1.1 Current System Image
Appendix-A: PMAPM Object Hierarchy:

In PMAPM, we define many kinds of object, collectively termed multiple purpose objects (MPO). We cannot exhaustively define all the view objects shared by all the participants in an A/E/C project. Some view-objects described in this Appendix only have object-name, and some have objects and slot names (but no values). However, we believe that these examples of object information will provide some useful ideas to other researchers who are interested in object oriented project or product models, or object-oriented databases for CAD systems. The basic hierarchies of PMAPM are summarized in the following figures and the examples of various descriptions of views and objects are given in Appendix B.

A-1  Global Hierarchy of PMAPM
A-2  Hierarchy of Social-View
A-3  Hierarchy of Cost View
A-4  Hierarchy of Labor View
A-5  Hierarchy of Management View
A-6  Hierarchy of General View
A-7  Hierarchy of Planning View and Sales View
A-8  Hierarchy of Design View
A-9  Hierarchy of Estimation View
A-10 Hierarchy of Construction View
A-11 Hierarchy of Facility Management View
A-12 Hierarchy of Maintenance View
A-13 Hierarchy of Building_1
A-2 Hierarchy of Social View

These views describe the trends of social needs to the Facility and General Contractor
A-3 Hierarchy of Cost View
A-4 Hierarchy of Labor View
A-5 Hierarchy of Management View
A-6 Hierarchy of General View
Planning View

Project Planning View

Planning Constraints View

Sales View

Sales Strategy View

Sales Technological View

Sales Constraints View

A-7 Hierarchy of Planning View and Sales View
A-8 Hierarchy of Design View
A-9 Hierarchy of Estimation View
A-10 Hierarchy of Construction View

Construction view
  Construction Planning View
    Construction Methods View
    Construction Work Site View
    Temporary Facility Plan View
    Material Procurement Plan View
    Construction Equipment Plan View
    Construction Organization Plan View
    Cost Management Plan View
    Quality Management View
    Sub-Contractor Selection View
    Transportation Plan View
    Safety Plan View

  Construction Management View
    Construction Work Schedule Control View
    Labor Control View
    Sub-Contractor Control View
    Material Control View
  Construction Equipment Control View
  Construction Cost Control View
  Construction Quality Control View
  Construction Safety Control View
  Value Engineering View

Construction Constraints View
A-12 Hierarchy of Maintenance View
A-13 Hierarchy of Building 1
Appendix-B Detail Object Description

In this section, we list some examples of object definition and the description of various functional and physical objects.

Example of the Functional object definition

<table>
<thead>
<tr>
<th>Project</th>
<th>Class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project_name</td>
<td>Strings</td>
</tr>
<tr>
<td>Project_code</td>
<td>Strings</td>
</tr>
<tr>
<td>Start_date</td>
<td>Strings</td>
</tr>
<tr>
<td>End_date</td>
<td>Strings</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total Cost View</th>
<th>Class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concreting Cost</td>
<td>List</td>
</tr>
<tr>
<td>Forming Cost</td>
<td>List</td>
</tr>
<tr>
<td>Reinforcement Cost</td>
<td>List</td>
</tr>
<tr>
<td>Steel Cost</td>
<td>List</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Progress View</th>
<th>Instance Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start-date</td>
<td>List</td>
</tr>
<tr>
<td>Required-Duration</td>
<td>List</td>
</tr>
<tr>
<td>Temporal-Duration</td>
<td>Demon (Rule)</td>
</tr>
<tr>
<td>Fixed-Duration</td>
<td>Demon (Rule)</td>
</tr>
<tr>
<td>Dummy</td>
<td>List</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Cost View</th>
<th>Instance Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total-budget</td>
<td>List</td>
</tr>
<tr>
<td>Land-cost</td>
<td>List</td>
</tr>
<tr>
<td>Building-cost</td>
<td>List</td>
</tr>
<tr>
<td>Design-cost</td>
<td>List</td>
</tr>
<tr>
<td>Construction-cost</td>
<td>List</td>
</tr>
<tr>
<td>Material-cost</td>
<td>List</td>
</tr>
<tr>
<td>Other-cost</td>
<td>List</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Organization View</th>
<th>Class Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project-Manager</td>
<td>Strings</td>
</tr>
<tr>
<td>Design-team</td>
<td>Demon(Function)</td>
</tr>
<tr>
<td>Estimator-team</td>
<td>Demon(Function)</td>
</tr>
<tr>
<td>Construction-team</td>
<td>Demon(Function)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Risk View</th>
<th>Instance Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Financial-risk</td>
<td>List</td>
</tr>
<tr>
<td>Construction-risk</td>
<td>List</td>
</tr>
<tr>
<td>Technical-risk</td>
<td>List</td>
</tr>
<tr>
<td>Other-risk</td>
<td>List</td>
</tr>
</tbody>
</table>
Constraints View  (Class Object)
  Requirement-of-usage  List
  Init-physical-constraints  List
  Init-functional-constraints  List
  Init-legal-constraints  Demon(Rule)
preset-constraints  Demon(Rule)
postset-constraints  Demon(Rule)
unconfirmed-constraints  Demon(Rule)
constraints-propagate-1  Demon(Rule)

Project Client View  (Class Object)
  Number-of-client  List

Clients-1  (Instance Object)
  Client-name  Strings
  Client-Address  Strings
  Client-business  List
  Fund  List
  Company-group  List
  Bank-group  List
  Company-level  List
  Client-note  List

Project Site View  (Class Object)
  Street-name  Strings
  City-name  Strings
  State-name  Strings
  Site-ground  List
  Soil-condition  Demon(Function)
  Neighbor-information  List
  Site-note  List

Design View  (Class Object)
  Project-leader  Strings

Architectural Space View  (Class Object)
  Space-volume  Demon(Function)
  Finish-check  Demon(Rule)
  Usage-check  Demon(Function)
  Color-check  Demon(Rule)

Example of the Physical object definition

Columns  (Class Object)
  Size  List
  Offset  List
  Material  List
  Finish  List
Number-of-column

Columns
  ID
  Size
  Offset
  Material
  Finish
  Position
  Angle
  Floor
  Support
  Supported_by

E_Walls
  Size
  Offset
  Material
  Finish
  Number-of-e-wall

E_Walls-1
  ID
  Size
  Offset
  Material
  Finish
  Position
  Angle
  Floor
  Connected_with

Windows
  Width
  Height
  Floor_level
  Number-of-window

Windows-1
  ID
  Type_id
  Width
  Height
  Position
  Floor
  Floor_Level

List(Demon)

(Instance Object)
  List
  List(Inherit)
  List(Inherit)
  List(Inherit)
  List(Inherit)
  List
  List
  List
  List

(Class Object)
  List
  List
  List
  List
  List
  List
  List
  List

(Instance Object)
  List
  List(Inherit)
  List(Inherit)
  List(Inherit)
  List(Inherit)
  List
  List
  List

(Class Object)
  List
  List
  List
  List
  List
  List

(Instance Object)
  List
  List
  List(Inherit)
  List(Inherit)
  List
  List
<table>
<thead>
<tr>
<th>Attached_on</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Spaces</strong></td>
<td>(Class Object)</td>
</tr>
<tr>
<td>Floor_level</td>
<td>List</td>
</tr>
<tr>
<td>Ceiling_height</td>
<td>List</td>
</tr>
<tr>
<td>Floor_finish</td>
<td>List</td>
</tr>
<tr>
<td>Ceiling_finish</td>
<td>List(Demon)</td>
</tr>
<tr>
<td>Wall_finish</td>
<td>List(Demon)</td>
</tr>
<tr>
<td>Number-of-space</td>
<td>List(Demon)</td>
</tr>
<tr>
<td><strong>Spaces-1</strong></td>
<td>(Instance Object)</td>
</tr>
<tr>
<td>ID</td>
<td>List</td>
</tr>
<tr>
<td>Room_code</td>
<td>List</td>
</tr>
<tr>
<td>Area</td>
<td>List</td>
</tr>
<tr>
<td>Room_name</td>
<td>List</td>
</tr>
<tr>
<td>Floor</td>
<td>List</td>
</tr>
<tr>
<td>Floor_Level</td>
<td>List(Inherit)</td>
</tr>
<tr>
<td>Ceiling_height</td>
<td>List(Inherit)</td>
</tr>
<tr>
<td>Floor_finish</td>
<td>List(Inherit)</td>
</tr>
<tr>
<td>Ceiling_finish</td>
<td>List(Inherit)</td>
</tr>
<tr>
<td>Wall_finish</td>
<td>List(Inherit)</td>
</tr>
<tr>
<td>Live_load</td>
<td>List(Demon)</td>
</tr>
<tr>
<td>Grade</td>
<td>List</td>
</tr>
<tr>
<td>Color</td>
<td>List(Demon)</td>
</tr>
<tr>
<td>Usage_zone</td>
<td>List</td>
</tr>
<tr>
<td>Fire_protection_zone</td>
<td>List</td>
</tr>
<tr>
<td>Consisted_by</td>
<td>List</td>
</tr>
</tbody>
</table>