THE PROJECT MODEL OF AN AUTOMATED BUILDING SYSTEM

Abraham Warszawski[1] and Rafael Sacks[2]

The aim of an Automated, Computer Integrated Building Realization System is to automatically generate all of the information required for the design, planning and execution of a building project. A central computer project data base forms the core of the System. The project data includes all the relevant information about the facility and the resources required through the various realization stages. Program modules function collectively to advance the project through each stage. The model presented is designed to facilitate both change through the project life-cycle and the use of functional and technical solution libraries. We describe various considerations in the design of the model and parametric assembly based solutions.

INTRODUCTION

Computer Integrated Construction

The quantity of information generated and used for the average modern construction project is vast. This is due to the complexity of the building product, the multitude of organizations involved and the fact that the majority of construction projects are prototypical. While computers are currently used to perform diverse design and management tasks, the construction industry has yet to adapt its work practices to the Information Age. Fully Computer Integrated Construction (CIC) will fundamentally change the way project participants perform and communicate (Teicholz & Fischer 1994).

The realization of the potential benefits of CIC has spurred research and standardization efforts aimed at providing a common project data base, for use by all the project participants, throughout the project life cycle (e.g. ISO 10303 - STEP). Projects such as DICE (Ahmed et al 1992) and CIFEWORLD (Khedro et al 1994) seek to define ways for architects and engineers to collaborate in a 'paperless' computer network environment. However, we believe that our vision of CIC in the future can be expanded in two important ways: comprehensively automated building process must be considered, and such processes, and their project models, should be tailored to a specific building type.

The project model outlined here is appropriate for an automated process for the design and construction of orthogonal, multi-story buildings with single level floors of uniform floor plan; the range of building technologies is not limited. The model includes resource and activity data as well as product data: the term Project Model is more comprehensive than Product Model, and is used to express this approach.

[1]Professor, Faculty of Civil Eng. Technion Israel Institute of Technology, Haifa 32000, ISRAEL
e-mail: cvabwa@techunix.technion.ac.il
[2]Doctoral Student, Faculty of Civil Eng. Technion Israel Institute of Technology, Haifa 32000, ISRAEL
e-mail: cvsacks@techunix.technion.ac.il

Automated Building Process

The concept of an Automated Building Realization System (Warszawski 1990), is based on the proposition that it is possible to advance through the various stages of conception, design and construction of a building project in a fully computerized process. The first step in defining the process is to delineate the stages, including their inputs and outputs. The stages are defined in figure 1 (an automated process for a building type other than that considered here may, of course, have different stages). The controlling factor is that after each of the design stages, and intermittently during construction, the project is presented to the owner/developer (or their professional consultants) for approval, change or rejection. Although not shown explicitly in the figure, backtracking is also possible. Construction may be fully automated (robotics), conventional (controlled by the system) or both.

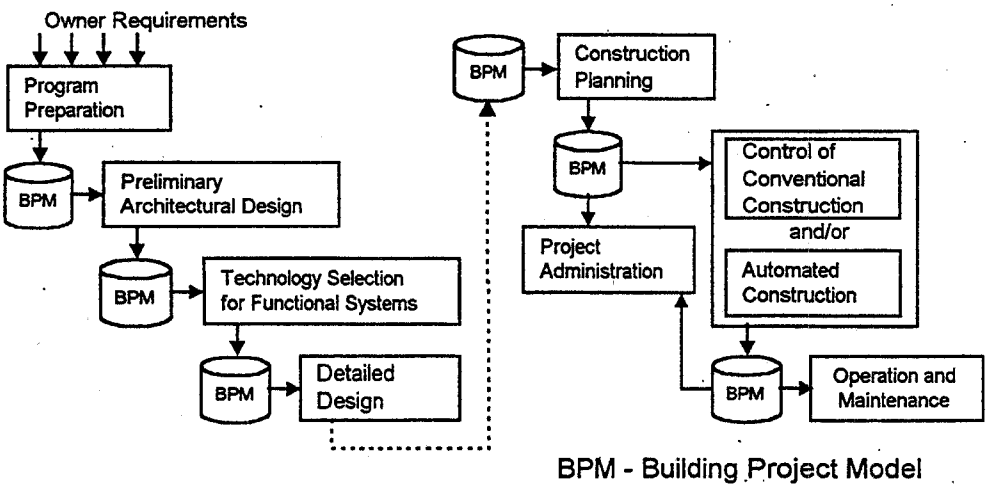


Figure 1.Automated Building Process Stages.

Program modules function collectively at each stage to advance the project through each stage (figure 2). These software 'black boxes' contain Knowledge-Based Expert Systems and include rules, procedures and algorithms. The programs also have access to external data bases containing all the information relating to the technological, business and physical environment (product catalogs, design codes, town planning requirements, building regulations, economic and financial data and physical site data).

The system is founded on a Building Project Model data base which grows in detail as the project progresses: the project *instance* has a new *state* at the end of each stage. The system also includes user interfaces which enable all the actors involved (owner, designers, contractors, suppliers) to access the central project data base. They can therefore generate, modify and share data on-line. Most importantly, they can automatically produce drawings and documents in professionally acceptable formats, so that they can monitor progress and make changes where necessary.

Various components of the automated system have been investigated at the Israel National Building Research Institute. These include preliminary design (Wiesel & Warszawski 1993), detailed design of prefabricated structures (Retik & Warszawski 1994), scheduling (Warszawski & Shaked 1994), planning of robot work (Warszawski et al 1995) and robot construction (Navon 1995),(Warszawski & Rosenfeld 1994).

Building Project Model Requirements

It is widely accepted that Object-Oriented technologies are most appropriate for Project Model development. This is the approach adopted in this work. Object-Oriented terminology, (such as *object, class, inheritance* and *instance*) is used as defined by Rumbaugh et al (1991).

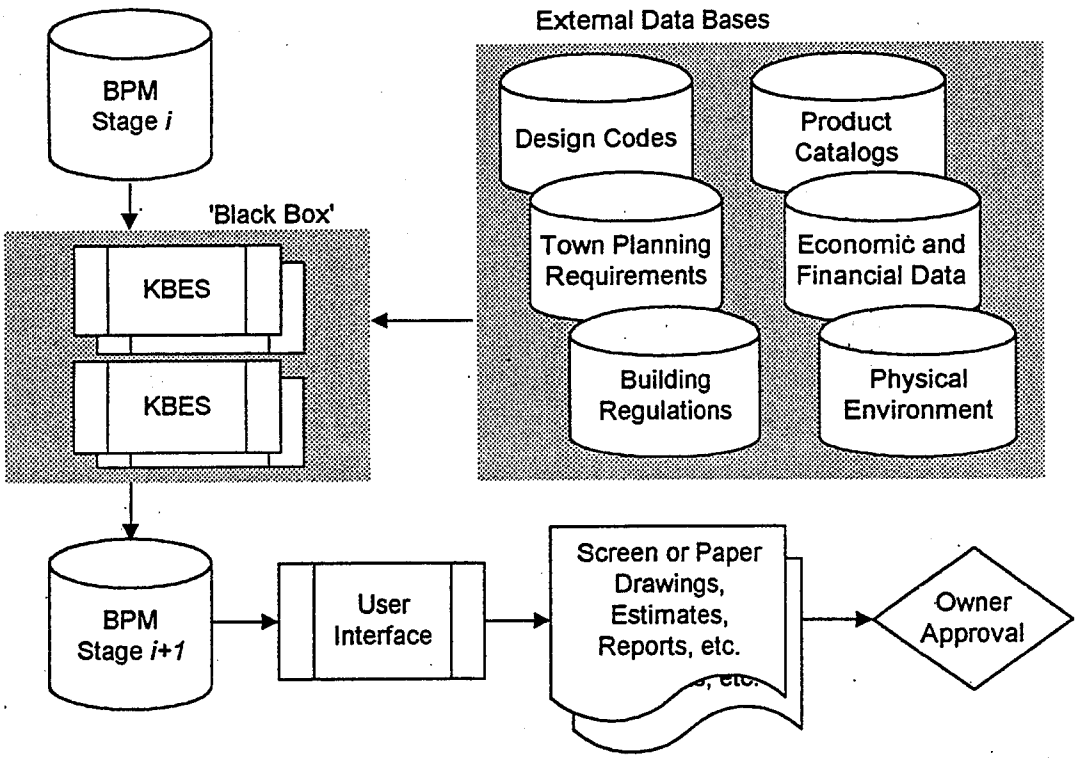


Figure 2. Automated Building Process Architecture.

GROWTH IN DETAIL OVER TIME

The automated system described above functions by adding detail to the Project Model instance at each stage. This is implemented in three ways: by assigning values to instance attributes, adding new object instances, or replacing object instances. This is greatly simplified if the object classes are defined appropriately for each stage. In order to formulate specific requirements of the Building Project Model, its content at the conclusion of each stage must be defined. This is dictated by the reporting required by the system controller(s) for effective decision making; thus the model content at each stage is defined by its outputs. The design and planning stages are summarized in Table 1.

The major project model classes can be now be defined. At the outset of the Preliminary Design stage, the building *spaces* are laid out together with their *functional system requirements* (figure 3). The floors are *primary spaces* and they are sub-divided into rooms, which are *secondary spaces.* This can be done most effectively by assigning values to the key attributes of predefined parametric templates. Templates of complete floor arrangements can be stored as aggregations of space and functional requirement objects (the template approach is discussed in more detail in the next section).

Table 1. Automated Building Process Outputs.

STAGE	OUTPUTS
Brief / Program	List of building spaces, their required functions and minimum area requirements, budget, site description and other constraints.
Preliminary Design	Plan view layouts of building floors, elevations, sections and preliminary cost estimates.
Technology Selection for Functional Systems	Technical specifications, design and construction directives.
Detailed Design	Complete drawings, sections and details of the building. Detailed cost estimates.
Construction Planning	Construction schedule, cost estimates, procurement schedules.
Control of Conventional Construction	Activity reports and deviation analyses.
Automated Construction	Materials handling directives, robot work plan, robot control.

For those functional system requirements whose fulfillment may influence spatial layout, *work assemblies* and their *elements* must be selected and detailed at this stage (as in figure 4). For example, a building floor (primary space) which will be used for offices will have functional requirements such as 'Vertical Load Carrying', 'Space division', 'Acoustic insulation' and 'HVAC'. A structural work assembly, such as a 'Slab on Columns', may be selected to fulfill the 'Vertical Load Carrying' requirement: the columns influence the spatial layout and must therefore be integrated with it at this stage. An 'acoustical insulation' requirement may have no influence on the layout: the selection of a specific work assembly can be postponed.

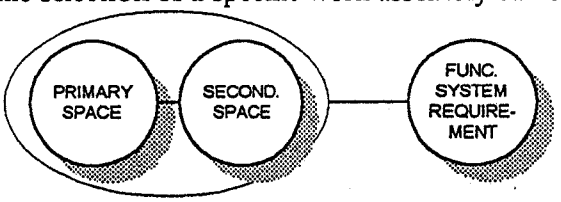


Figure 3. Preliminary Design Stage Project Model Classes.

The first step in detailed design is to select *technological solutions* for each functional system required (figure 4a). An 'Internal Space Division' functional system may be realized with a 'gypsum board partition' technology. The choice of technology defines the building *work assembly* and its *elements* (figure 4b): thus, for example, a work assembly consisting of gypsum board partitions with wooden doors could be added to the model. A work assembly concentrates all of the elements which will be installed in an activity.

The relation of work assemblies and their elements to functional system requirements provides an effective mechanism for backtracking by removal of a previously selected assembly and its replacement with an alternative. However, only one level of selection is provided - the assembly and its elements are selected as a complete technological unit. This reflects the emphasis the authors place on the integral nature of the link between an assembly and the work activities which will be required for its installation.

For certain building elements, individual *components* must be detailed (figure 5). For example, reinforcing bars must be calculated and detailed. This part of the design work is relegated to this phase as it is, by definition, of purely algorithmic nature and of no direct interest to the

user(s). For most common building elements this is unnecessary as the parts from which they are built can be considered as resources.

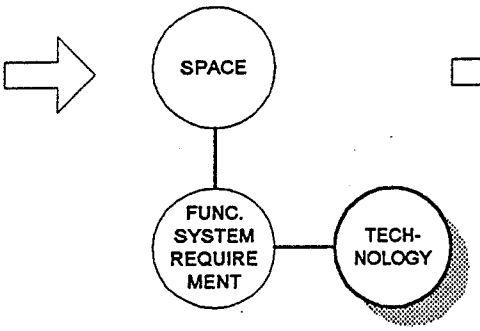


Figure 4(a)
Technology Selection

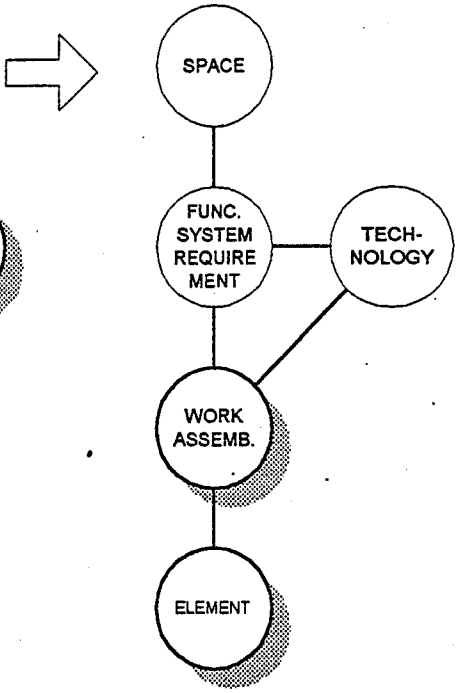


Figure 4(b)
Detailed Design Stage Project Model Classes

Construction planning begins with the addition of *basic activities* (which install elements) and *activities* which collect basic activities and install assemblies. Basic activities are performed by *labor* and/or *equipment*, which is organized into work *teams*. In addition, each activity consumes *resources* (these are detailed further in the full model). Activities are defined to install/build one work assembly. They are collected in *tasks* which are defined to install one kind of assembly (a *building assembly*) in one primary space (as each assembly fulfills a requirement in only one space) - this ensures minimum interference and interdependence between activities (Warszawski & Shaked 1994). While productivity and efficiency considerations require that all the Elements of an Assembly should be installed in one uninterrupted Activity by one Team, there will be exceptions. For example, the glazing of windows is usually carried out at a late stage of the work. The glass and frame belong to the same element, but including both the basic activities 'install frame' and 'install glass' in the same Activity is not feasible. Object relationships must therefore also allow separate links between basic activities and activities, distinct from the element/assembly relationship.

RESOLUTION AND CHANGE OF ELEMENTS

Within each stage more than one program/expert system will perform design synthesis each for different functional requirements. Duplication of elements may occur; for example, a structural program may place a shear wall (structural) in the same physical space occupied by a partition (space division). Such elements must be resolved (combined), and hence will serve two (or more) functions simultaneously. In the case of building elements, the resolved element object would then belong to two or more assemblies. In a purely statically typed Object-Oriented system, this would require complete a priori definition of such elements through multiple

inheritance. However, since a) the full scope of such combinations cannot be predicted at the time of system design, and b) the number of potential combinations would be huge, hardwiring of multi-purpose elements is not practical. Also, effecting changes to a BPM object instance may require changing the instance's class inheritance. This implies an object-centered rather than a class-centered approach (Hakim & Garret 1994). Dynamic typing is therefore a minimal requirement for such functionality in an object-oriented system.

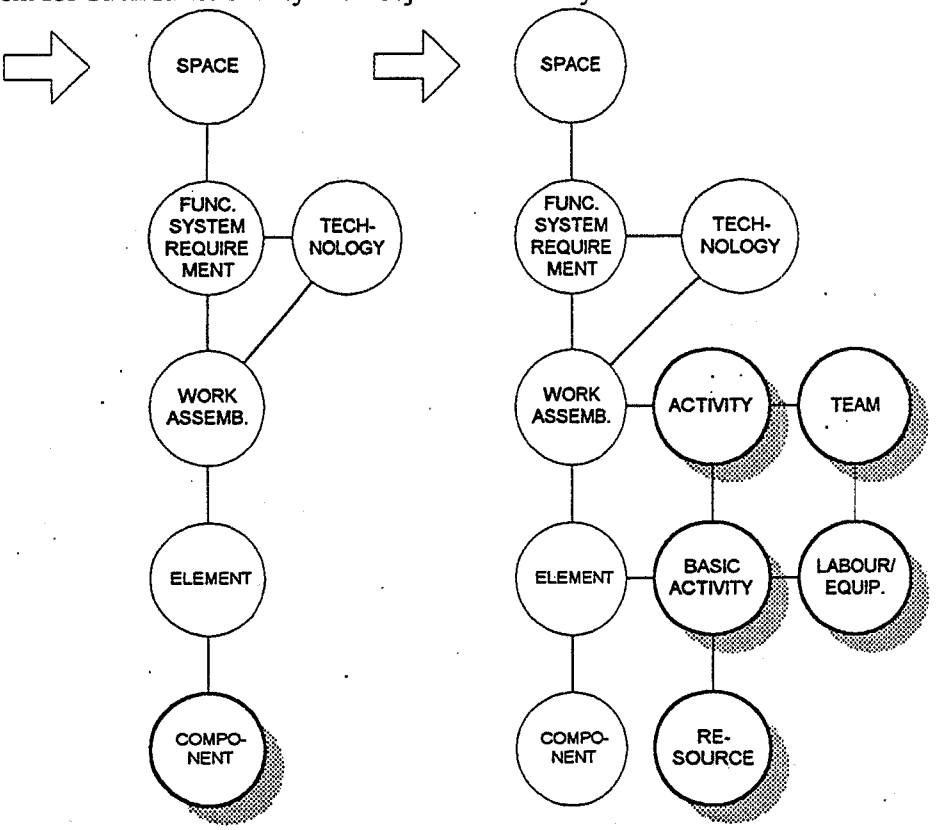


Figure 5. Construction Planning Stage Project Model Classes.

SUPPORT FOR CONSTRAINTS

In order for an automated system to produce acceptable results, it must allow the project developer and/or professional consultants to stipulate their requirements at the start of the project or at the start of each stage. This includes the expression of constraints which limit the scope of feasible solutions. Most Expert Systems which perform design synthesis (e.g. KTISMA (Tsakalias 1994), EIDOCC (Sacks & Buyukozturk 1987)) allow formulation of such constraints. However, an Automated Building System will incorporate a number of Expert Systems within its modules. Therefore, a generalized method for storing constraints which are accessible and interpretable to all program modules is required.

DESIGN, STORAGE AND USE OF PARAMETRIC ASSEMBLIES

Different computing paradigms have been proposed for automated building design systems. These include algorithmic methods, Rule-Based Reasoning and Case-Based Reasoning. For the case of automated design and construction of multi-story orthogonal buildings, a **'parametric assembly'** solution is proposed. Parametric Assemblies can be thought of as complete work assembly solution templates. They are implemented as collections of linked object classes,

based on the Building Project Model schema classes. They include complete definitions of the methods for 'detailing' themselves and their constituents. Their methods can employ any or all of the strategies listed above. They are stored in libraries outside of any calling program, and it is therefore possible to add or adapt assemblies as construction technologies change and develop. The key to their definition and use within the Automated System is that they conform to the basic BPM schema - i.e., they and their elements are defined as sub-classes of BPM classes.

Parametric Assemblies can serve design synthesis at any stage. For example, at the preliminary design stage, a program could select a 'Slab on Columns' work assembly to fulfill a 'Vertical Load Carrying' functional system requirement. The 'Slab on Columns' parametric assembly would include an 'R/C Flat Slab' work assembly object, whose constructor would generate the 'Column' and 'Slab Section' element objects. A simplified example of such an assembly is shown schematically in figure 6. The classes are shown in block diagrams including name, attributes and methods. Super classes and inherited attributes are shown in italics. Figure 7 shows the instanced parametric assembly as it might appear in a specific building model.

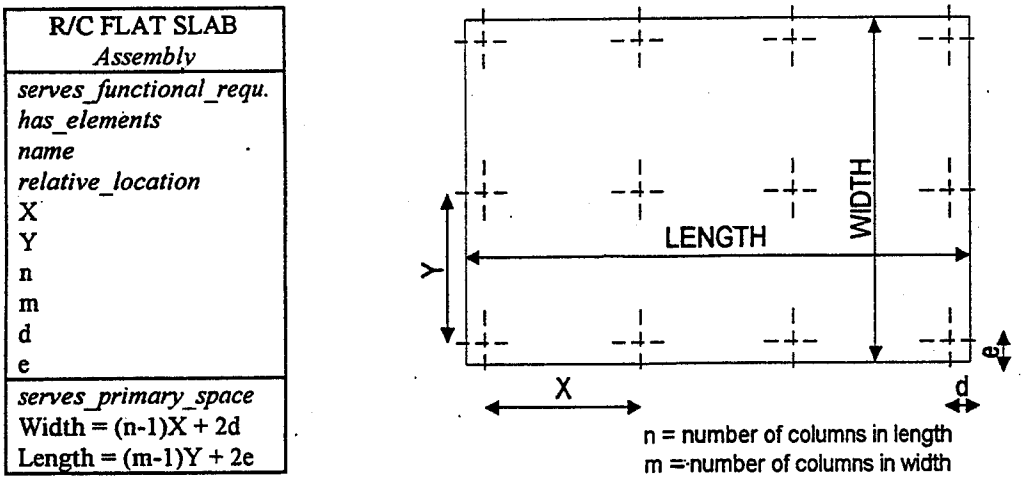
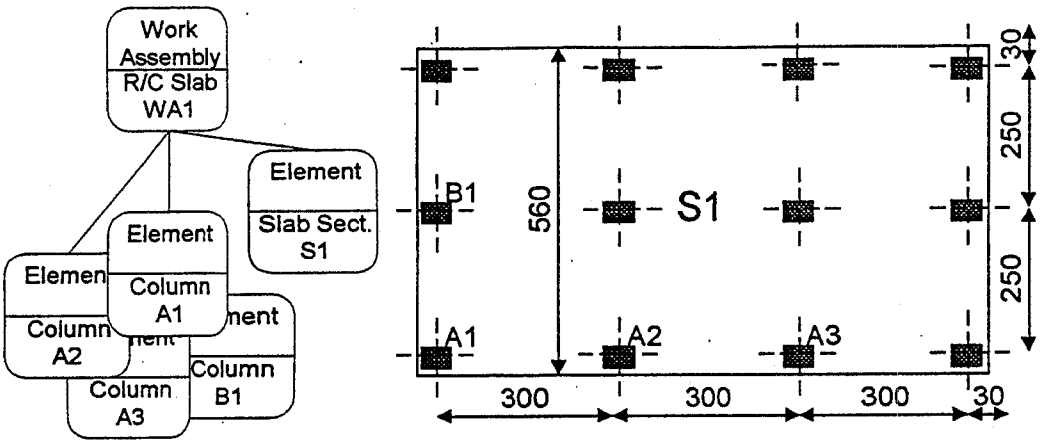


Figure 6. R/C Flat Slab Parametric Assembly.



Note: For sake of clarity, some columns are not labeled.

R/C FLAT SLAB *Assembly*	
name	WA1
serves_functional_requ.	FS1 (Vert. Load Supp)
has_elements	S1, A1, A2, A3, B1....
relative_location	0,20
X	300
Y	300
n	4
m	3
d	30
e	30

SLAB SECTION *Element*	
name	S1
part_of	WA1
has_reinforcement	
relative_location	0,0
thickness	16
length	960
width	560
has_hole_elements	

R.C. COLUMN *Element*	
name	A1
part_of	WA1
has_reinforcement	
relative_location	30,30
width	40
depth	20

Figure 7. R/C Flat Slab Work Assembly Instance.

At the detailed design stage, any parametric work assembly which contains building elements which can collectively satisfy the functional system requirements can be inserted. An assembly of "steel columns, girders and precast concrete slab sections" could be selected; similarly, a "cast-in-situ concrete assembly of columns, beams and one-way spanning slabs" or a "cast-in-situ concrete flat slab on columns" are alternatives. Each is a technologically complete solution, and could be inserted complete with activity and resource elements. All the Assembly and Element behavior is embedded in the class methods, including the mechanisms for parametric specification (instantiation) of the complete assembly. In this way, addition of new parametric assemblies can be done without altering system programs. Of, course, the work involved in adding new elements and assemblies to the libraries is greatly simplified by the fact that they are based on existing hierarchies of objects.

PROPOSED PROJECT MODEL SCHEMA

The proposed schema is shown in figures 8(a) and 8(b). The schema is designed to accommodate the requirements of growth in detail over time, resolution and change of elements, support for constraints and the use of parametric templates. An important feature of the schema is that it clearly reflects the project on three clear levels of detail (the *building*, *primary space* and *secondary space* horizontal axes). Certain entity relationships have yet to be fully tested: for example, for the specific building type considered, a recursive part-of relationship which enables sub-division of *secondary spaces* may be necessary (shown dashed). The relationship *functional assembly* serves *building* may be unnecessary if all the assemblies can be detailed at the *primary space* (floor) level. Similarly, elevator shafts, stairwells and complete building cores could be modeled as *vertical primary spaces* (Warszawski & Shaked 1994): this aspect is not yet fully developed in the proposed model.

The resources required during a *basic activity* for installing an *element* can be derived automatically, since the quantities and the technology employed are known. A special case arises when a *resource* must be directly associated with an *element*. An example is a reinforcing bar which is detailed for inclusion in a specific concrete *element*. In this case, the resource is called a *component*, and associated as a part of the *element* (this relationship is shown dashed in figure 8(b)). This enables the material handling system to deliver it to the appropriate place at the required time.

A typical multi-story office building has been detailed in terms of the Project Model schema in order to demonstrate the objects and the relationships between them. Figures 9 (a) to (e) show selected details from the complete Project Model Instance. In figure 9(a), the building project and its primary and secondary spaces are set out. Figure 9(b) shows the specification of functional systems for each space.

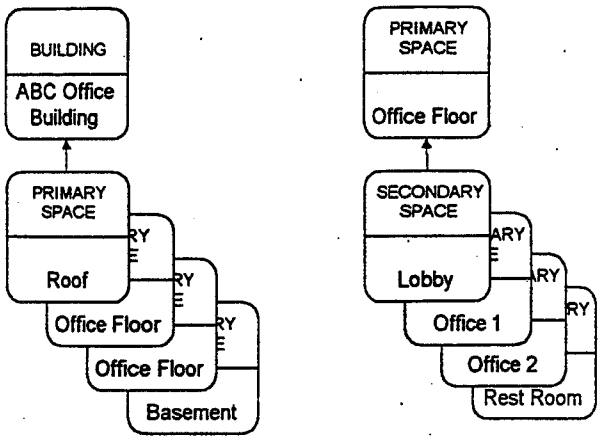


Figure 9(a).
Building Project With Defined Spaces.

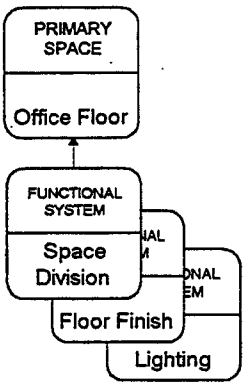
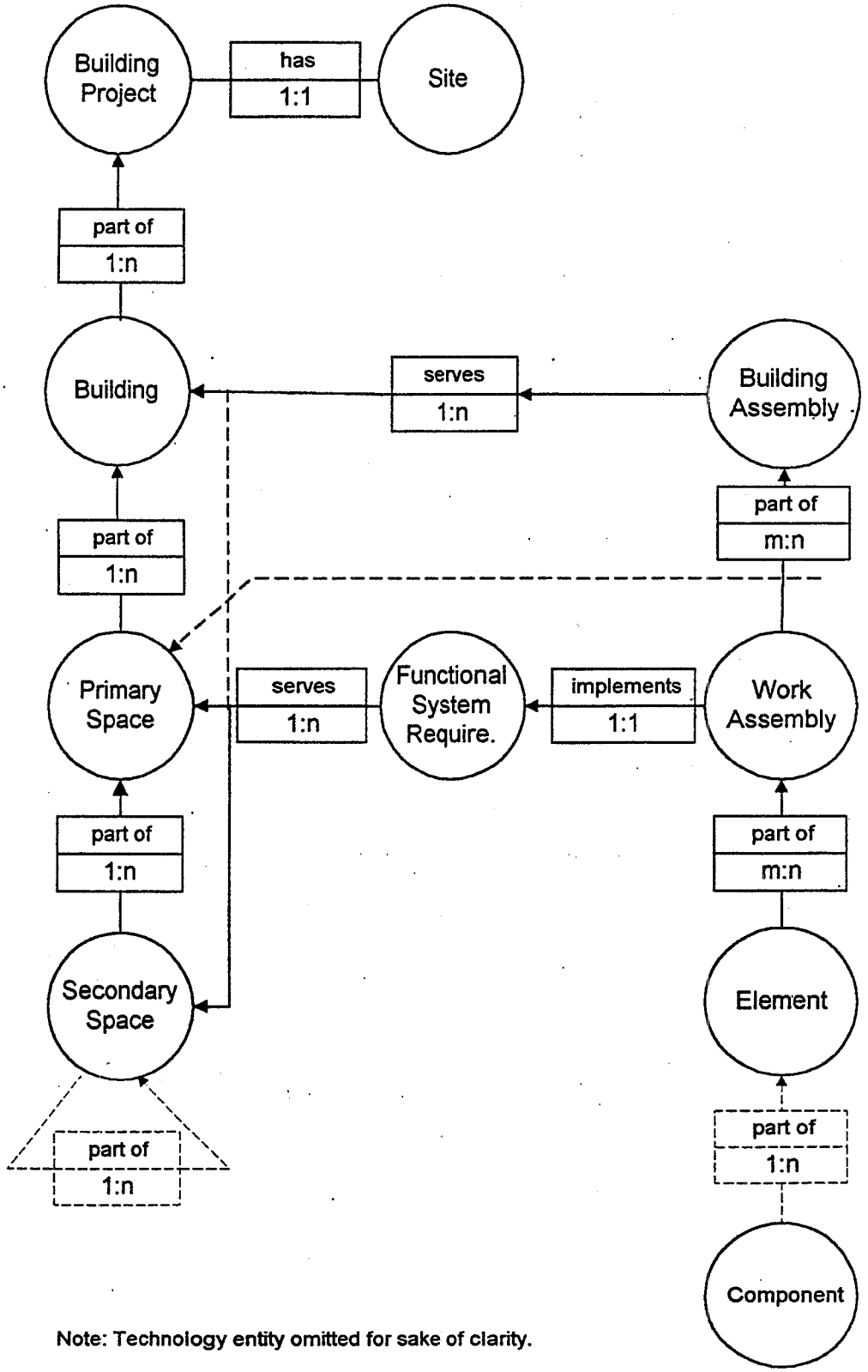


Figure 9(b).Functional System
Requirements Defined.

In figure 9(c), the expansion of one functional system is shown. The system is designed by instancing a work assembly and its elements. In this case the 'Space Division' functional system requirement is fulfilled by selecting a technological solution: this includes defining a 'work assembly' and instancing its elements in such a way that all of the functional

requirements are fulfilled. Once all of the Building elements have been detailed, it is possible to assign basic activities and activities (figure 9d).



Note: Technology entity omitted for sake of clarity.

Figure 8a. Proposed Project Model Schema

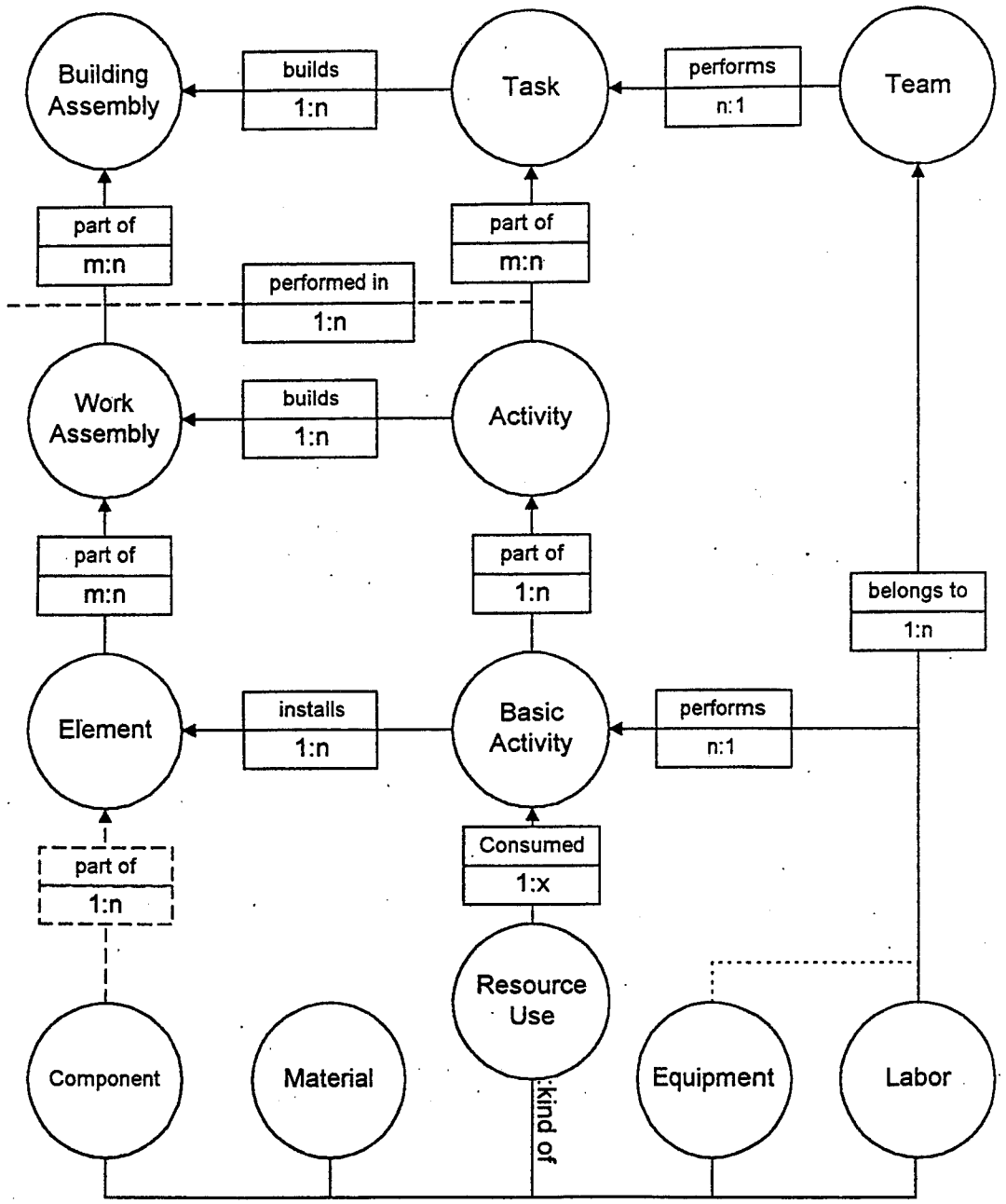


Figure 8b. Proposed Project Model Schema

An Activity, by definition, builds an entire Assembly. It is performed in one Primary Space. In order for this to be possible, the Team performing the collective Activity must incorporate all of the labor and/or equipment needed to complete the Assembly. The parallel levels of Assembly *built by* Activity *performed by* Team and Element *built by* Basic Activity *performed by* Labor/Equipment are highlighted in figure 9(e).

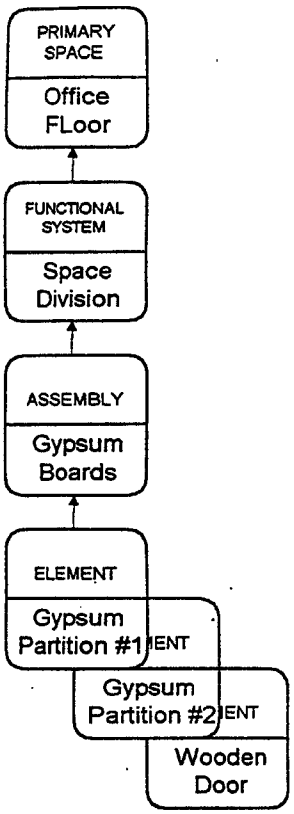


Figure 9(c). Instantiation of Assemblies and Elements.

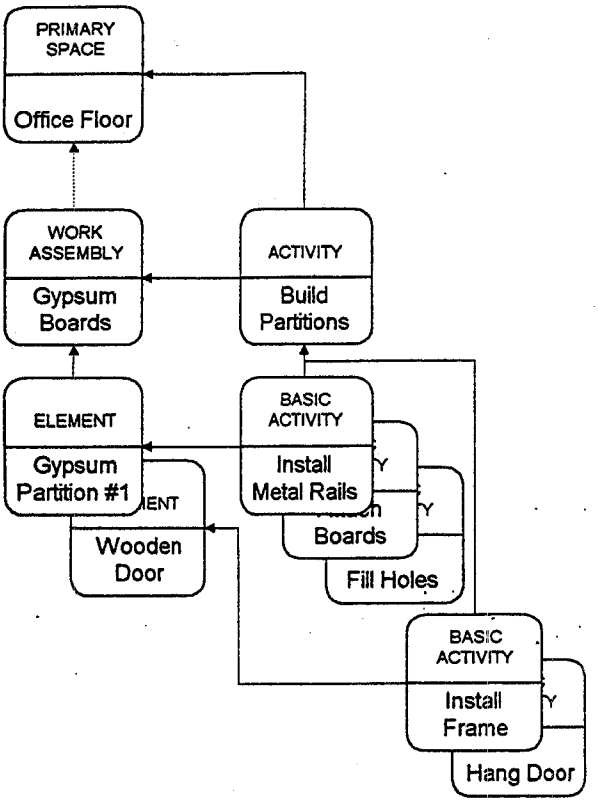


Figure 9(d). Detailing Activities and Basic Activities.

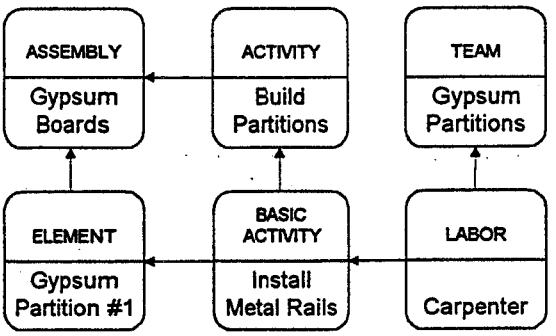


Figure 9(e). Addition of Teams and Labor.

CONCLUSIONS

The project data model forms the foundation of the automated system: it must therefore also fully support the automated process. Thus four specific requirements can be detailed:

The structure of the project model must directly facilitate the addition of detail through the project life-cycle. Definition of object attribute values is not sufficient for description of the changes in object states which occur through the design and planning stages. Schema

design must include separate objects which are added and detailed as design and planning progresses.

Its major space, product and activity classes should parallel one another. This reflects the automated construction objectives of minimum interference and interdependence between work teams.

It must facilitate changes to the project data objects during the life of a project. This requires a software platform which supports dynamic typing and multiple inheritance.

It must allow definition of constraints of general or specific nature. In order for an automated system to produce acceptable results, all known constraints should be detailed at the start of each stage. This may require the placement of detailed building object instances at earlier stages than would normally be the case. In addition, such constraints may not be explicitly specified, but rather stated as a range of acceptable alternatives.

It must support the storage of libraries of template style solutions at all levels. The software modules which perform design synthesis at various stages may use parametric templates, case-based solutions or standard library items. Each of these should be fully expressible in terms of the product model classes.

REFERENCES

Ahmed S., Sriram D., Logcher R. (1992). "Transaction Management issues in Collaborative Engineering." *ASCE Journal of Computing in Civil Engineering*, Vol 6 No. 1

Hakim M. Maher & Garrett J.H.Jr.(1994) "Modeling Engineering Design Information: An Object-Centered Approach", Khozeimeh K. (Ed.) *ASCE Conference on Computing in Civil Engineering*, Washington, June 1994.

Khedro T., Genesereth M.R. & Teicholz P.M.(1994) "A Framework for Collaborative Distributed Facility Engineering" Khozeimeh K. (Ed.) *ASCE Conference on Computing in Civil Engineering*, Washington, June 1994.

Navon R.(1995) "COCSY - CAM Oriented CAD System for Tile-Setting", *ASCE Journal of Computing in Civil Engineering*.

Retik A. & Warszawski A.(1994) "Automated Design of Prefabricated Building" *Building and Environment* Vol 29. No. 4., Pergamon Press Ltd.

Rumbaugh J., Blaha M., Premerlani W., Eddy F. & Lorensen W. (1991) *Object Oriented Modeling and Design* Prentice-Hall, New Jersey, USA.

Sacks R. & Buyukozturk O. (1987) "Expert Interactive Design of R/C Columns under Biaxial Bending." *ASCE Journal of Computing in Civil Engineering*, Vol 1 No. 2.

Teicholz P. & Fischer M.(1994) "Strategy for Computer Integrated Construction Technology" *ASCE Journal of Construction Engineering and Management*, Vol 120. No. 1, March 1994.

Tsakalias G.E.(1994) "KTISMA: A Blackboard System for Structural Model Synthesis of Asymmetrical Skeletal Reinforced Concrete Buildings" *Artificial Intelligence and Object-Oriented Approaches for Structural Engineering*, Topping B.H.V. & Papadrakakis M. (Editors) CIVIL-COMP Press Edinburgh, Scotland.

Warszawski A. & Rosenfeld Y. (1994) "Robot for Interior-Finishing Works in Building: Feasibility Analsysis" *ASCE Journal of Construction Engineering and Management* Vol 120. No. 1

Warszawski A. & Shaked O. (1994) "Intelligent Construction Planning" Khozeimeh K. (Ed.) *ASCE Conference on Computing in Civil Engineering*, Washington, June 1994.

Warszawski A.(1990) *Industrialization and Robotics in Building*, Harper and Row, New York.

Warszawski A., Rosenfeld Y. & Shohat Y. (1995) "Autonomous Mapping System for an Interior Finishing Robot" *ASCE Journal of Computing in Civil Engineering*" Vol No .

Wiesel A. & Warszawski A. (1993) *Preliminary Design of a Structure Using an Automated System* Israel National Building Research Institute, Technion IIT, Haifa (in Hebrew).