SIMULATION OF PROCESSES IN BUILDINGS AS A FACTOR IN THE OBJECT REPRESENTATION OF BUILT ENVIRONMENTS

Filiz Ozel, Asst. Prof.
New Jersey Institute of Technology
School of Architecture
323 M.L. King Blvd. Newark, N.J. 07102

ABSTRACT

This paper explores the implications of object oriented representation of buildings in the simulation of dynamic processes in architectural environments. Two main groups of objects are represented in such simulations: architectural objects, and active agents that drive the events in a simulation. To accommodate the data needs of a wide range of simulations and expert systems, architectural objects must be represented as a single data model. on the other hand, active agents are usually application specific, i.e. they are only used in a specific simulation. Furthermore, the properties of objects, and the relationship of objects to each other must be clarified within the framework of the behavior of agents in the simulation. Emergency egress behavior model, BGRAF (Ozel, 1985) is used to illustrate some of the issues discussed in the paper. Interface to CAD systems and the need for a standardized object-graphic entity relationship is also addressed.

INTRODUCTION

The management of several discrete, complex and overlapping areas of knowledge in architectural design has led to the need for a central constraint management protocol and to the efforts to develop a single model of architectural environments. While this is an emerging field in knowledge based systems, the use of object oriented representations in simulating processes in buildings has often been overlooked.

Traditional CAD systems store all designs in the same way, in terms of graphic components such as points, lines, surfaces etc. The representation of buildings is solely in terms of the drawing of the object, rather than the object itself. Whereas the model of a building needs to include a semantic representation and data about the relationship of parts to each other and to the whole. This is important for simulations that require knowledge about the building as an object.

The simulation of dynamic processes in architectural settings require the representation of objects that have spatial and temporal qualities. Such objects are usually highly structured and are composed of other objects, many of which have properties that vary in time and space. Computer simulations in architectural settings must have means of representing not only buildings, but also other objects such as humans that are located in them.

On the other hand, the relationship between objects and their behavior can be expressed in a rule base, which allows reasoning and inferencing. Such a reasoning system represents the hypothesis of the researcher about the behavior of intelligent objects in a simulation.

EXPERT SIMULATION SYSTEMS

Among the advantages of expert simulation approach cited by Robertson (1987) are a) the ease of describing intelligent agents (objects in a simulation), b) the creation of the simulation by the application of 'small chunks of knowledge' through an inferencing system, which means that the structure of the simulation is dynamically generated by the inferencing mechanisms and not explicitly by the programmer.

Robertson refers to the ability of an expert simulation to appeal to one's intuitions about the real world so that the rules that define the real world can be used in the simulation and vice versa as "Referential Transparency". Traditional simulation languages tend to create and use operational mechanisms that do not necessarily have real world counterparts. This leads to the problem of using mechanisms that do not appeal to our intuitions about the real world. In fact referential transparency is one of the major advantages of using an expert system approach in the simulation of dynamic processes in architectural environments.

The representation of architectural knowledge solely as rules is often found to be inadequate, since reasoning about architectural artifacts requires "method calls", i.e. procedures that extract knowledge from an existing data base. Thus, architectural expert systems usually call for a mixture of rule-based and procedure-based reasoning mechanisms. On the other hand, simulation languages are traditionally procedure-based, and they tend to process real world objects in a highly abstract manner, whereas architectural environments do not usually lend themselves to the level of abstraction expected and supported by traditional simulation languages such as GPSS (Schriber, 1974), etc. The configurational aspects of such environments must usually become constraints in spatial processes through geometric reasoning, thus abstraction limits the accuracy and the realistic

representation of the real world event under study. Therefore, a programming platform that simultaneously allows accurate descriptions of architectural settings and the simulation of dynamic processes in such settings is needed. An object oriented approach can achieve this by interacting with a computer aided design system that can create a graphic data base and a front end, i.e a user interface for discrete event simulations.

REPRESENTING COMPLEX OBJECTS

Researchers from a variety of disciplines are beginning to explore the implications of object-oriented data structuring methods in simulating dynamic processes (Adelsberger, 1986). While expert systems aim at reasoning about the objects represented in a system, object oriented simulation environments aim to define intelligent objects that can reason about their environment, their relationship to each other and to the changing conditions in the simulated process.

Conceptually, the development of object data systems helps to build a model where simulated objects directly relate to their real world counterparts. This allows ease at tracing the performance of the model, the nature of the objects in the model and the states of the model within a given time period.

Several programming languages support object oriented programming and data structuring. Smalltalk, Simula, Ross, KBS and Orient84/K are only some of these languages (Adelsberger, 1986). For example Ross, which stands for Rule Oriented Simulation System (Khlar, Faught & Martins, 1980) was developed specifically for war game simulations and military air battles. It represents real world behavior/knowledge by rules and by incorporating knowledge in hierarchially arranged objects. Objects in the system have names, parents, properties and behaviors (as messages) associated with them. Smalltalk (Goldberg et.al, 1983) is considered to be the hallmark of object oriented programming languages. Its philosophy is to define abstract objects and classes, and operations that may take place on those objects.

Object: In terms of data structuring, objects are data systems which request, send, and receive data via sub-routines called messages. In simulations, classes of objects represent real world counterparts. The class of an object defines what properties its objects can have and their associated operations. Thus when an object is instantiated, it inherits all the properties and operations of its class. Objects can be simulated entities such as people, or architectural components such as walls, doors, windows etc.

Agent : An agent is an instance of a particular object specification in a simulation. In architectural settings, a distinction must be made between active and passive objects. While the former actually generates the simulated dynamic process, the latter usually acts as constraints (such as architectural elements acting as barriers), or wait for a scheduled time to become active (such as the arrival of fire fighters in an emergency egress simulation). The term "agent" usually refers to active objects in a simulation.

Behavior: This refers to the history of actions taken by an agent. In discrete event simulations, since behaviors are sequences of actions, they can be subdivided into smaller behavior chains. Behavior can also be described as the actions taken by an agent between two simulated times.

In an intelligent simulation system, each object needs to be associated with:

a) its initial state in the simulation,
b) logic rules that govern its operation,
c) how it can change state.

These rules include all of the other objects or processes that affect its operation. As the simulation proceeds, each object is associated with new states of the system, where a time stamp is used as a reference. The simulation logic rules express propositions about objects, since the researcher aims to test hypothesis about their behavior in a simulation. As a result, all objects in the system are logically interconnected with each other and with the processes that could influence their behavior.

Object Representation of Buildings

Approaches taken in the area of object representation of buildings range from developing parallel data structures, using traditional and relational data base systems to using object oriented languages (Coyne, et al.1990; Turner et al, 1984; MacKellar & Ozel, 1990).

Some of the important issues in the object representation of architectural environments can be listed as follows:

a. The relationship of architectural objects to each other,
b. Functional views: Extracting a subset from a single data model to fulfill the data requirements of a specific simulation.
c. Architectural objects as space definers
d. Active object (agent) - space relationship,
e. Behaviors of agents: object relationship,
f. Graphic interface
   - Geometric reasoning
   - User Interface

a. A fundamental characteristic of design objects is that they are hierarchical in nature (Fig.1). For example, a building floor may be composed of rooms; each room has a set

of walls, a floor and a ceiling as components; and each of these components in turn may have doors, windows etc. as components. Such a hierarchial system can be extended as far as the simulation under consideration necessiates. The fundamental relationship is a "has_part" relationship. At the lowest end of the hierarchy graphic entities that define each object reside.

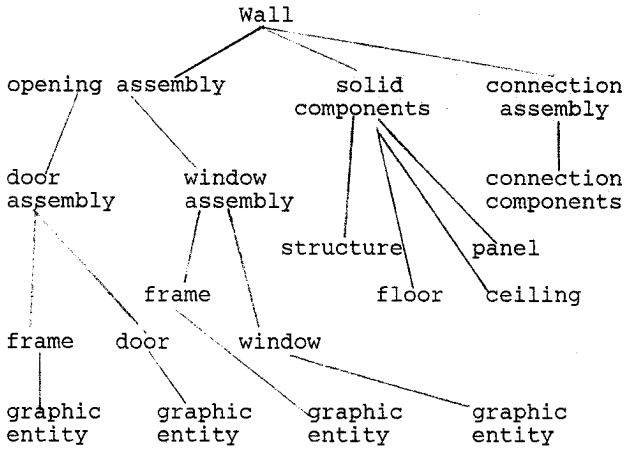

Fig.1 Component hierarchy

b. Functional views : Another important issue is the capability to extract a version of the basic model that will fulfill the requirements dictated by a particular simulation. For example, in an emergency egress simulation, rooms, stairs, doors, windows, extinquishers and alarms may be the only components needed, which means smaller components such as studs in a wall are not significant in the object representation of buildings for emergency egress purposes. On the other hand, alarms and extinquishers are probably not needed by any other type of building performance simulation. Therefore, an object representation of buildings needs to be able to accommodate different levels of detail and part relationships.

In some cases, regrouping of objects must be done based on a functional view of the data model. For example, a protected egress route which is composed of a sequence of spaces, which in turn are made of a variety of architectural components such as doors, walls etc. must be defined as a new type. This can be achieved by creating a functional hierarchy. Some of the architectural objects will appear in the functional hierarchy, whereas others will not. For example if a door is defined within the egress_route_functional hierarchy, it will be assumed to have the appropriate fire ratings and will be defined as a legal "exit door". On the other hand, another door will not be considered as a legal exit since it is not in the egress_route_functional hierarchy.

c. Architectural objects as space definers: Architectural environments are composed of elements whose main purpose is to define spaces and the links between these spaces. This process requires a rule base and geometric reasoning. In some cases the distinction between adjacent spaces may not be as clear, i.e. there might be overlaps. The rule base determines those properties of architectural elements that define them as boundary elements.

Since agents are usually located in spaces, the boundaries of spaces can constrain spatial movement. Geometric reasoning and space-boundary relationship must be used to extract this interaction. Furthermore, the definition of the boundaries of spaces are context dependant. For example, a wall is not permeable for human movement, but may be considered permeable for heat transfer or humidity. Therefore, the extraction of functional views not only requires the identification of objects that participate in a simulated event, but also requires a contexual definition of the properties of such objects.

d. Active object (agent) - space relationship: Furthermore, certain types of simulations may require components that are not necessarily an inherent part of the building. For example in a behavioral simulation, people need to be located in spaces. Usually object-related-methods identify the relationship between the components of the simulation and the objects (Fig.2). For example, the relationship of a person to a space can be an "in_room" relationship, which has a procedure linked to it to determine if a person is located in a particular room or not. In a heat transfer simulation, there might also be a need to establish similar object-space relationships.

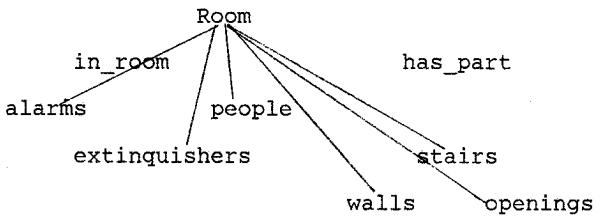

Fig.2 Space-object relationships

e. Behaviors of agents as relationship definers: Agents and architectural objects may also interact due to a behavior of an agent. This relationship can be established by specifying affecting and affected agents. A procedure and a set of rules determine the nature of the relationship between the affected and affecting agents.

f. Graphic Base: This refers not only to a graphic user interface, but also to the configurational description of architectural settings. Therefore, graphics is not only a

matter of how to display, but also a matter of geometric reasoning and graphic data structure.

In simulations of dynamic processes in architectural environments, the behavior of agents heavily relies on the interaction of the agent with the environment (as it is in spatial behavior of people, cars, robots etc.), therefore avalibility of a graphic data base is very important. Traditional simulation languages typically do not support graphics, or only provide a graphic tool box to build user interfaces. While existing CAD systems can be the basis for a graphic interface, more importantly they can help establish the graphic data base that is needed to define architectural and simulated objects spatially.

Linking such a system to a CAD system requires the definition of components in terms of graphic components, such as surfaces in surface modeling or polyhedras in solid modeling. Geometric relationships must also be modeled. For example in a building not only a space-boundary relationship, but also other relations such as adjacency or connectivity need to be represented.

Graphic representation of architectural environments also depends on the requirements of different simulations. For example, an emergency egress simulation may only require a plan description of the building. On the other hand an acoustical simulation must use a 3-dimensional surface or solid modeler. A heat and mass transfer simulation might also require a solid model representation of a building. While all of these representations can be derived from a single model, the object-graphic entity relationship needs to be standardized.

In designing graphic user interfaces, an additional issue is the degree of detail with which each object is displayed. At different scales, people can perceive different sets of detail. Similarly, very detailed graphic representations of objects at large scales create crowded imagery. On the other hand, at small scales where more detail can readily be seen, there is a need to display subcomponents of an object. To reduce processing time, an iconic representation may also be considered during run-time. These discussions indicate a need for multiple graphic representations of objects which is managed by a version control mechanism.

STRUCTURE OF INTELLIGENT SIMULATIONS

Intelligent simulations must accommodate the creation and use of a range of objects and their instances. Since simulations are fundamentally experimental tools that aim to help the researcher to test his/her hypothesis about the behavior of simulated objects, not only an object generator but also an experiment generator module must be provided.

A simulation object generator must have an interface which could at times be graphics that would allow one:

- to define the configuration of architectural objects,
- to define the configuration and location of active objects,
- to position objects on the screen,
- to modify their location, size, and configuration,
- to specify the properties of the objects
- to specify the relationship between objects,
- to specify the constraints on the objects,
- to establish class-instance relationships where instances inherit the characteristics of the class they belong to.

One must also make a clear distinction between the object representation of architectural environments and the representation of active objects. While the former can be part of a more general object description and data base system, the latter needs to be more specific. Furthermore, in a multi purpose simulation environment, one needs to provide facilities to generate new types as well as the instances of these types. Thus, intelligent simulations must encompass a type generator as well as an instance generator.

Since simulations are highly procedure-based, there is also a clear need to create a system where "methods" as well as properties are inherited by the instances of objects. Manola and Dayal (1990, p.211) solve this problem by treating stored as well as computed properties as functions, i.e. the system they have developed uses a fuction-inheritance system.

Experiment Editor

While model building environment refers to the creation of the hypothesis of the researcher about the real world event, experimentation refers to the instantiation of the model built. Intelligent agents inherit the properties, relationships and method associations from their type. Therefore, while the model building step establishes the types of objects, experimentation step instantiates these objects. Furthermore, simulations aim to generate knowledge about the states of a given system, thus there is a need to describe the initial state of the system. Thus the parts of an experiment generator can be identified as follows:

- Initial state generator for each experiment,
- Object instantiation,
- Specification of termination conditions
- Statistical output manager
- Definition of interrupt conditions
- Specification of the parameters of the simulation state manager which keeps track of the states of the simulation.

Time Factor

Simulation of dynamic processes in architectural settings usually require a method of representing the passage of time. Since both active and passive objects can go through change during the duration of a simulation, one needs to keep track of this change. Therefore, multiple representations are not only required to solve graphic interfacing problems, but also needed to keep a record of the states of the agents in a simulation.
Temporal properties of objects, such as the first_action time, or the current_action_time can be part of the object data structure. An additional method is to provide a version control facility, which keeps track of different versions of an object at different time frames. This can be achieved either by storing only those properties that have changed over time with a time stamp, or by keeping a record of the full properties and relationships of objects with a time stamp. Although the former method is preferable in terms of memory requirements, it might not be always possible to implement it, since relationships between objects may also change over time or with the change in their properties. Ideally an inferencing system should allow one to generate new rules that determine the relationships between objects due to their new properties.

OBJECT BASE OF BGRAF-II

Some of the early examples of simulations in architectural environments have emerged in the fire safety field, where smoke and fire spread models based on building physics have been developed (Jones, 1986). Other simulations range from behavioral simulations (Stahl, 1979; Ozel, 1985) to acoustical performance simulations (Stettner and Greenberg,1989; Turner, 1990).
These models usually rely on procedural languages such as Fortran and are based on a variety of representations of architectural environments, none of which are object oriented. For example, in BFIRES (Stahl, 1979) walls, doors and people are only allowed to occupy grid locations on a building plan, and BGRAF uses the data base of a CAD system called CAEADS (Turner, et.al. 1984). In BGRAF, Fortran was preferred over more specific simulation languages such as GPSS (Schriber, 1974), due to the need to access a CAD data base for spatial reasoning.
Typically, passive objects in BGRAF-II are the architectural elements in the system, and active ones are those that participate and create the dynamic process. Therefore, while passive ones do not necessarily have a "behavior" associated with them, active ones must have one or more behaviors associated with them, some of which may include relationships with passive objects.

Active objects in BGRAF-II are structured as follows:

name - object name
parents - names of the superclasses to which this object belongs
properties - description of object characteristics
behaviors - rules describing the object's function and behavior characteristics

Examples of BGRAF-II objects:

NAME : Occupant group
PARENT : People
PROPERTIES : Location, walking_speed, mobility, sleep_status, smoke_tolerance, tolerance to stress, noise_tolerance, current_action, current_goal, safety_status, initial_action_time, safe_exit_time, current_time.
BEHAVIOR : Goal structure of the object and the methods associated with action

NAME : Alarm
PARENT : Fire safety system
PROPERTIES : Loudness, location, sounding_time, current_time, alarm_switch
BEHAVIOR : IF sounding_time EQUAL or GT current_time, THEN increase alarm_switch by one.
IF alarm_switch EQUAL one, THEN increase information_buildup factor. etc.

NAME : Extinquisher
PARENT : Fire safety system
PROPERTIES : Location, type, assigned_person, in_use, etc.
BEHAVIOR : IF in_use ON, THEN decrease fire spread rate.
IF assigned THEN set in_use ON.

Definition of Behaviors

Originally, BGRAF was visualized as a tool where different conceptual models of emergency egress behavior can be tested. This was achieved by differentiating "goals" of the simulated people from their actual "actions". An "action" had the potential to be associated with a variety of "goals". The stochastic component was incorporated into the model at the "next goal" selection process. Actions were associated with a goal by the model user.

In fact, several researchers propose a "goal driven" event-execution system within intelligent-simulation environments. Adelsberger and Shannon (1986, pp.110) use the term "goal-directed simulation", and Robertson (1986, p.13) refers to "goals" as he discusses a bank-queue simulation where people have goals but detailed actions are molded by the environment. In defining intelligent-objects, goal-oriented behavior can allow the objects in the system to reason about their environment, and choose an action strategy.

The mechanics of each action needs to be defined as a method. This can be irrespective of the objects it will act upon, such as moving an object at a given speed irrespective of whether it is a car or a human being. Most spatial events can be defined as generic methods, while other types of interactions must be more specific. Most actions in BGRAF-II require spatial movement, therefore this is provided as a generic method. Other actions change the properties of simulated objects, such as waking_up a sleeping person. Therefore properties of objects must be updated as a result of different actions, such as setting the sleep status of a person to "awake". While spatial movement, at times require interaction with other objects, such as going through a door, at other times it only requires the updating of the current location of an object. Therefore, the rule base of each action must identify the objects an affecting agent must or might interact during the execution of the action and the nature of this interaction. Actions are defined within the following structure in BGRAF-II:

Action name : Name
Affecting agent : An active object in the hierarchy
Affected agent : Any object in the simulation
Constraints : Restrictions on the properties of objects, or on the relationship between objects.
Associated method: Procedure associated with the execution of the event

On the other hand constraints are grouped into the following categories:

a. Spatial constraints
   - Location of objects
   - Spatial relationship of objects
b. Those related to the properties of active agents
c. Those related to the properties of passive agents

An example of an action in BGRAF-II is as follows:

Action name : Go to door
Affecting agent : Person
Affected agent : Door
Constraints : Existence of a door Not to go through walls, Walking speed of the simulated person, Smoke density of currently occupied space, Mobility status of a person, Smoke tolerance of a person, Person not outside the building, and Person not incapacitated
Associated method : Move the person towards the doorway.

This can be extended to as many actions as needed. The advantage of object-oriented and rule-based approach is the ease with which new actions can be added to the simulation, since actions need not be built into the code as it is in the case of other programming approaches.

Goal Generator

Based on previous discussions, BGRAF-II implements a goal-driven simulation approach. Objects are related to goals within a Has_goal relationship, and the behavior of active agents are identified with this relationship. There is also an uncertainity factor that is associated with each Has_goal relationship. Each goal has a likelihood of being chosen by an active agent.

On the other hand, each goal is associated with a set of actions with a Has_action relationship, which also entails an uncertainty factor, i.e. each action has a likelihood of leading to a selected goal. The weight of this factor may change from one condition to another. The constraints in the rule base of each action determine whether the conditions are appropriate for the execution of a selected action. Originally in BGRAF, action selection was deterministic, but the goal selection process was stochastic, and environmental constraints contributed only to the goal change process. Whereas when intelligent agents respond to changing conditions in a dynamic process, they should select an action strategy that would best fit the given conditions. Therefore, deterministic action strategies can be replaced by inferencing mechanisms.

SUMMARY

Object-oriented, rule based simulation environments provide the possibility of simulating dynamic processes in architectural environments with direct reference to real-world counterparts. The process which is sometimes termed as "referential transparency", has several advantages, among which are the ease with which the model user can trace the processes in the model, the clear identification of the logic of the system as a set of logical rules, the potential to describe new objects and add new rules to the simulation knowledge base, and the possibility of distinquishing architectural objects from objects located in such environments which resolves the problem of representing the interaction of dynamic objects with environment. Furthermore, such simulation systems have the potential to access the data base of more general purpose architectural object data models, and extract functional views as required by the specifics of the simulated process.

Due to the simulated reasoning mechanism, intelligent objects can modify their behavior and choose alternate strategies that will lead to the same goal. While, traditional simulation environments were able to provide such alternatives only by strictly defining available strategies, theoretically intelligent agents can generate alternate

strategies from the rule base by means of an inferencing system.

The potential of intelligent simulations is only beginning to be tapped by researchers from a variety of disciplines. Architectural researchers need to focus more on this potential, since they tend to deal with issues related to computer simulations very often, due to the fuzziness of the line between architectural expert systems and architectural performance simulations.

REFERENCES:

Adelsberger, Helmo H., & Shannon Robert, E., "Rule based object oriented simulation systems", in Intelligent Simulation Environments, Simulation Councils Inc., SanDiego, CA. 1986, pp. 107-112

Coyne, R.D., Rosenman, M.A. Radford, A.D, Blachandrian,M, and Gero,J.S., Knowledge based design systems. NewYork: ADDISON-WESLEY Pub. Co. 1990.

Klahr, P., Faught W.S., & Martins, G.R., "Rule Oriented Simulation", Preoc. of International Conference on Cybernatics and Society, 1980, pp.350-354.

Lorie,R.A., and Plouffe, W., "Complex Objects and their use in design transactions", Proc. of ACM Engineering Design Applications, San Jose, CA. 1983.

MacKellar, B., & Ozel, F., "An Integrity Constraint Approach to Verifying Building Designs", Research Paper, First International Conference on Artificial Intelligence in Design, UK, 1991.

Manola,F. & Dayal,U., "An Object Oriented Data Model", in Object Oriented Database Systems, Zdonik,S. & Maier, D.,(Eds.) Morgan Kaufman Publ.,San Mateo, CA. 1990., pp.209-215.

Jones, W., "FAST: fire and smoke spread model", Technical Report, NIST, Center for Fire Research, Gaithersburg, MD. 1986.

Ozel, F., "A Stochastic Computer Simulation to model the Behavior of People in Fires: An Environmental Cognitive Approach." Proceedings of the Building Technology and Safety Conference, National Institute of Building Sciences, Los Angeles, CA. March 1985, pp.91-95.

Robertson, P., "A rule based expert simulation environment", in Intelligent Simulation Environments«MDNM», Simulation Councils Inc., SanDiego, CA. 1986, pp.9-15.

Schriber, Thomas J., "Simulation Using GPSS", John Wiley and Sons, NewYork, 1974.

Stahl, F., "Calibration of Computer Model BFIRES to simulate human behavior in fires", Technical Report, NIST, Center for Fire Research Technical Report, Gaithersburg, MD. 1979.

Stettner, A. and D.P.Greenberg (1989) "Computer Graphics Visualization for Acoustic Simulation", SIGGRAPH'89 Conference Proceedings (pp.195-206)

Turner, J., "An application of geometric modeling and ray tracing to the visual and acoustical analysis of a municipal open-air auditorium", ACADIA 1990 proceedings, pp.173-185.