# 11. An Intelligent Simulation Approach in Simulating Dynamic Processes in Architectural Environments

*Filiz Ozel*

New Jersey Institute of Technology
School of Architecture
323 M.L.King Blvd.
Newark, NJ. 07102

*The implications of object-oriented data models and rule-based reasoning systems is being researched in a wide variety of application areas ranging from VLSI circuit design (Afsarmanesh et.al., 1990) to architectural environments (Coyne, et.al., 1990). The potential of this approach in the development of discrete event simulations is also being scrutinized (Birtwistle, et.al., 1986). Such computer models are usually called "expert simulations" or "intelligent simulations". Typically rule-basing in such models allows the definition of intelligent-objects that can reason about the simulated dynamic processes through an inferencing system. The major advantage of this approach over traditional simulation languages is its ability to provide direct reference to real world objects and processes.*

*The simulation of dynamic processes in architectural environments poses an additional problem of resolving the interaction of architectural objects with other objects such as humans, water, smoke etc., depending on the process simulated. Object-oriented approach promises potential in solving this specific problem. The first part of this paper addresses expert simulation approach within the context of architectural settings, then the second part summarizes work done in the application of such an approach to an emergency egress simulation.*

## Introduction

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. Simulations in architectural settings must deal with the physical environment as well as other objects such as humans that are located in such environments.

In representing complex objects, a number of studies have proposed extensions to conventional record-oriented data models (Lorie,R.A. and Plouffe,W., 1983), but a more widely implemented approach has been the development of entity- or object-oriented data models (Lochovsky, 1986; Coyne, et.al., 1990). Researchers have also begun to explore the implications of object-oriented data structuring methods in simulating dynamic processes (Adelsberger, 1986), where knowledge about dynamic processes are represented as rules that allow inferencing. 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. This paper focuses on the implications of object oriented data structures and rule-based simulation environments on the simulation of dynamic processes in architectural settings. An emergency egress model will be used as an example in the second half of the paper.

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. Simulation languages are traditionally procedure-based, and they tend to process real world objects in a highly abstract manner. On the other hand, 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, thus abstraction limits the accuracy and the realistic representation of the real world event under study. Therefore, a programming platform is needed that simultaneously allows accurate descriptions of architectural settings and the simulation of dynamic processes in such settings.

## 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 afforded by Expert Simulation Systems is one of the major advantages of using expert system approaches in simulating dynamic processes in architectural environments.

Rule based systems are relatively high level tools for capturing the descriptions of the behavior of systems. In addition they can also encompass computational interpretation. For the purposes of rule based simulation languages, it is necessary that the system under study allow structuring of data and operations into object and relation descriptions. Forward chaining systems additionally provide the means to couple operations with the objects that make up the domain of the operations.

Object oriented programming refers to a data structuring approach that is based on real world objects and their properties. Several programming languages support such data structuring environments. 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. The class of an object defines what properties its objects can have and their associated operations. Thus when a class object is instantiated, it inherits all the properties and operations of its class. KBS, Knowledge Based Simulation, was developed at Carnegie Mellon Univeristy for the Intelligent Management System project. The underlying database is similar to Ross, and was specifically designed as a knowledge based simulation environment.

There is some consensus in the use of the following terminology among researchers interested in the development of intelligent simulations:

*Object* : This refers to a class of objects in the simulation that represents a real world counterpart. Objects can be simulated entities such as people, or architectural components such as walls, doors, windows etc. Specific instances of an object in a simulation are generated from its type (or class description). In terms of data structuring, objects are data systems which request, send, and receive data via sub-routines called messages.

*Simulation Knowledge Base* : However, it is not enough to define and create the objects in a simulation, one must also describe the inter-connectivity of objects.The interconnectivity of components is defined as the interdependence of objects to each other within the context of the dynamic process simulated and can be represented by Artificial Intelligence software engineering concept called rule basing. Oren (1986) defines knowledge in the domain of computer simulations as facts and rules, procedures, methods about objects (projected to the instances of objects) and their relationships with each other and with their environment. A rule can reside in an architectural data model along with actual components and can be executed by an inference engine. Function of a component can also be stored as an object and get associated with a component.

Each object needs to be associated with a) its initial state in the simulation, b) logic rules that govern its operation, and 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.

*Agent* : This refers to an instance of a particular Object specification. These are the active objects in a simulation. In the simulation of dynamic processes 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).

*Behavior* : This refers to the history of actions taken by an agent. Since these 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.

*Graphic interface* : This refers not only to an iconic representation of the agents, but also to the configurational description of architectural settings. When the behavior of agents heavily rely on the interaction with the environment (as it is in spatial behavior of people, cars, robots etc.), the interaction with a graphic database becomes a major component in a simulation. Usually traditional simulation languages do not support graphic interfaces or provide very limited support.

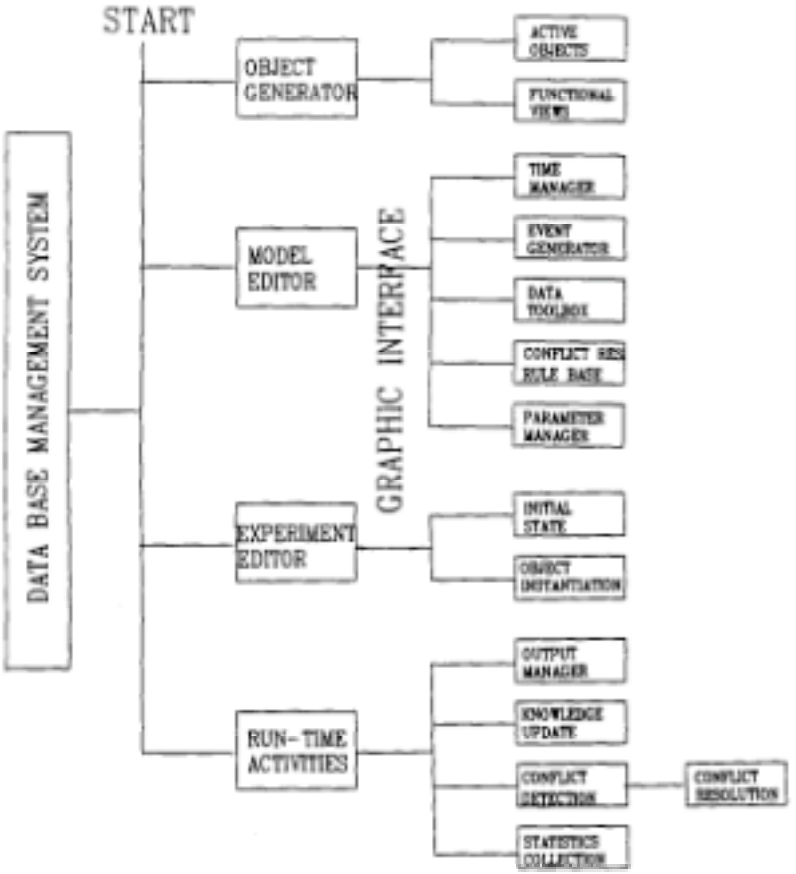


**Figure 1.** The Structure of Intelligent Simulations in Architectural Settings.

While existing CAD systems can be the basis for a graphic interface, other graphic tools such as virtual reality software can be very powerful. In fact, the simulation of dynamic processes in architectural environments can be reflected to the user within a very realistic virtual setting. For example, the graphic interface of a fire spread simulation can show smoke and fire spread in a virtual reality environment.

## The Structure of Expert Simulations

Functions in an expert simulation environment can be divided into those that are related to the description of real world events and objects, and those that are related to experimental setup. Simulation of dynamic processes is actually an experimentation technique, where the parameters of a real world event are explored by means of the output data produced by the simulation. In essence, setting the experimentation is the instantiation of the system developed. In non-deterministic simulations (i.e. stochastic ones), each instance of the system generates a different set of output data which need to be further analyzed. The parts of an object oriented and rule based simulation environment can be seen in Fig.1

### a) Object Generator Module

This allows one to generate the objects in a simulation independent of the context they will be used. The nature of the objects in architectural settings and the architectural environment itself allow a hierarchial structure. Parts (or components) are related to each other in a hierarchic manner.

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. The process of extracting functional views from an architectural object data base for simulation purposes is not any different than the process of retrieving functional views for expert system purposes. (A functional view is defined as a data model that is extracted from a more extensive object data base to meet the specific requirements of an expert system application). On the other hand, simulation specific objects need not only be defined within their own type hierarchy, but their relationship to the extracted functional view of the architectural setting also need to be established. 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, this module must encompass a type generator as well as an instance generator.

The Object generator must have an interface which could at times be graphics that would allow one:

- to define the configuration and location of active objects (Possibly through a Computer Aided Design system, i.e. a graphic data base to define the configuration of 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.

Since simulations are highly procedure-based, there is a 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 function-inheritance system.

### b) Model Editor Module

This module helps to establish the rule base of the processes simulated, define the relationship between objects in time and space, and define the functional relationship between them. Among the tools that are needed to build the model are:

#### i. Time manager

Two traditional approaches to time management in simulations are the time-increment method and the event scheduling method. Time managers in expert simulations will have to provide both options. Furthermore, tagging of objects with a time stamp can help with statistical data collection. Temporal properties of the objects in a simulation, such as first_action time, current_action_time etc. need to be part of the object data structure. An additional issue in time management is the condition of an object at different times during the simulation. This can be resolved either with a version control facility, where different versions of an object at different time frames are stored, or those properties of objects that have changed over time are time stamped and a record of these are saved.

#### ii. Event generator and scheduler

This is the process of creating the rule base of the events in the system. Major transactions in this section are :

a) Establishing the relationship between objects and events;
b) Determining the parametric relationship between objects;
c) Defining the method associated with the execution of each event.

Several researchers propose a "goal driven" event-execution system within the intelligent-simulation environments. Adelsberger and Shannon (1986, pp.110) use the term "goal-directed simulation", and Robertson (1986, p.13) refer to "goals" as he discusses a bank-queue simulation where people have goals but detailed actions are molded by the environment. BGRAF, the emergency egress simulation developed by this researcher also uses a "goal driven" event driver. 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 event 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 the interaction of different objects needs to be more specific.

Subdividing the objects into affecting and affected agents can start defining the relationship between objects through a method. Rather than embedding the object_relationship (except for spatial relationship) into object_type hierarchy, it can be made part of the event description and the associated method that will act upon these agents. This helps to identify a wide variety of relationships specific to an expert_simulation without necessarily providing this data in the object data base. In a sense, events are also treated as objects through this structure.

### iii. Parameter-base manager

Determines the parametric relationship between objects and events, which is established by factors such as size, proximity, density etc. Event_selection process can rely on such parametric relationships. This is usually a general rule base that is checked by objects prior to the selection of new action strategies.

### iv. A numerical data tool box that can generate random numbers and different distribution functions

Random number generators are used in simulating the chance parameters in stochastic simulations. In the development of stochastic expert simulations, Robertson (1986) proposes an "uncertainty factor", where a percentage of certainty is embedded into the rule base. This allows the intelligent-object to select among a number of alternatives and to reason about the immediate circumstances. For example in an emergency egress simulation:

IF smoke_density greater than 10mg/cubic feet*
THEN fire is located at current_floor (75%)*
IF smoke_density greater than 10mg/cubic feet*
THEN turn back (60%)*
(* The numbers are given for demonstration purposes)

Among other tools needed during model building are the interface between the rule base managing language and the object data base generating language, an on-line knowledge update facility, and on-line documentation.

- Conditions of conflict resolution

Model generator must also provide a rule base to resolve conflicts.

### c) 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. This requires the instantiation of the objects in the system and the specification of different parameters. 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.

### d) Run Time Activities

Among the run-time activities are display and output management, collection of data about the states of the objects in the simulation and hardcopy generation. Error and conflict detection need to be done during run-time, so that they can be resolved using the rule base of conflict resolution. Forward and backward chaining processes help to drive the simulation. Updating of the knowledge base during run-time represents the learning process. Therefore, intelligent agents in the simulation can learn during the course of the simulation by updating the rule-base.

## Object Base of the Emergency Egress Simulation

In the past, reseachers have studied dynamic processes in architectural environments by the development of discrete event simulations, such as fire spread models (Jones, 1986) and emergency egress models (Stahl,1979; Ozel, 1987; etc.). 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 incorporate graphical interface and the need to access a CAD data base.

Conceptually, the development of object data systems helps to build a simulation 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. Furthermore, object oriented data structures help to establish the relationship of architectural and non-architectural objects much more clearly.

In defining the structure of the objects, it is important to differentiate between those objects that are passive in the system and those that are active. Typically, passive objects 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. Within the scope of this paper, the data structure of passive objects, i.e. the architectural environment will not be addressed, since it has already been addressed by this researcher in a jointly authored paper (Ozel & MacKellar, 1991). The extraction of functional views from a single data model, and the details of the relationship between passive and active objects will be reserved for a future paper. 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 actions |

| | | |
|---|---|---|
| name | - | Alarm |
| parent | - | Fire safety system |
| properties | - | Loudness, location, sounding_time, current_time, alarm_switch |
| behavior | - | IF sounding_time EQUAL 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. |

## The Event Base of the Emergency Egress Conceptual Model

### a) Event Generator of the Model

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.

BGRAF can function as a general purpose emergency egress simulation by means of a rule based system, since model users can identify their hypothesis about the behavior of people during fires by accessing and modifying this system. The dialogue with the model user can aid in the modification of the rule base. Data from real world fire incidents indicate that people show different response patterns in different types of occupancies, therefore the ability of emergency egress simulations to allow change in the rule base is very critical in processing a wide variety of theories about human behavior in fires.

The decision process concerning the relationship of the individual with the environment, with other people and with the event-specific elements (i.e. fire and smoke spread) establishes the rule base of the model. This is in fact the reflection of the relationship of active and

passive objects to the rule base. In the original version of BGRAF, these were built into the procedural knowledge base, and did not allow forward and backward chaining.

| | | |
|---|---|---|
| Affecting agent | - | Person |
| Affected agent | - | Door |
| Constraints | - | Affected and affecting agents be within 2 ft. of each other<br>Affected agent be closed<br>Smoke level at the adjacent space not exceed the tolerance level of the affecting agent<br>Affecting agent not_incapacitated |
| Associated method | - | Set door status to open and update the graphic display. |

| | | |
|---|---|---|
| Action name | - | Wake person |
| Affecting agent | - | Person |
| Affected agent | - | Person |
| Constraints | - | Affected and affecting agents must be in the same space.<br>Affected agent asleep and not_incapacitated<br>Affecting agent not_asleep<br>Affecting agent not_incapacitated |
| Associated method | - | Set sleep_status of the affected agent to "awake" |

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. Furthermore, actions can be more systematically defined, thus new actions can be defined consistent with the existing ones, and the role of each action can be traced easier during run time for statistical purposes.

For example, in the original version of BGRAF, rescue was not an action defined in the event library. To add this action, one needed to know the data and programming structure of the model, and there was no readily available method to upgrade the library. Whereas in an object oriented environment, since the structure of each action is clear and there is referential transparency, one can easily add new actions. A "rescue" action will be as follows within the structure identified above:

| | | |
|---|---|---|
| Action name | - | Rescue |
| Affecting agent | - | Person |
| Affected agent | - | Person |
| Constraints | - | Affecting and affected agents must be within the same space<br>Affecting and affected agents must be within 1 ft. of each other.<br>The mobility of the affected agent must be nil (i.e. either handicapped, ill, drugged etc.) OR the safety_status must be nil (i.e. the person is incapacitated due to smoke etc.) |
| Associated method | - | Move the affecting and affected agents together towards exits. |

### b) Goal Generator

As discussed earlier, among the properties of rule based object oriented simulations, Adelsberger and Shannon (1986, p.110) cite "Goal directedness" as a method that can augment the experimentor and provide automatic modification to the simulation model. BGRAF also uses a goal-driven simulation approach. Within this structure, the model user was able to identify several deterministic alternative action strategies that will lead to the same goal.

Objects are related to goals within a Has_goal relationship, and the behavior of active agents are identified with this relationship. Has_goal relationship of each object has an associated uncertainty factor.

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.

### c) Goal Modifiers

Goal modifiers act as conditions that will affect a goal change, i.e. change the behavior of an agent. Some of the approaches to the management of knowledge base of goal modification are:

a) Providing a rule base embedded into the behavior of the object, which controls the response to method calls,

b) Creating a local knowledge base independent of the methods that define the behavior of objects as in Orient84/K object oriented language (Adelsberger, et.al. 1986).

c) Providing a general knowledge base, which is checked by every object-type, whenever a method call is activated.

The second approach has the potential to reduce processing time, since only those rules related to a specific object are checked. On the other hand it may result in the repetition of certain rules for different objects. The decision between the second and the third options depends on the number of different objects needed in a simulation and the amount of redundancy in the rule base, therefore must be decided upon on a case by case basis while developing intelligent simulations. At this point in this study, rules exist in a general rule base, since emergency egress simulations do not have a wide variety of active objects.

Rules for goal modifiers can be time-dependent or local-condition-dependent. While the former requires a scheduling process expressed within the bounds of the simulated time, the latter requires reasoning about objects and/or checking the existence of certain conditions. These can best be expressed in terms of IF-THEN rules, and can use pre-stored or calculated parameters.

Some of the time dependant modifiers are:

1. IF alarms go_off, THEN activate agent and choose goal.
2. IF Smoke_detectors sound, THEN activate agent and choose goal.
3. IF fire_fighters arrive, THEN rescue action,
4. IF fire_fighters arrive, THEN alert action. etc.

Examples of condition dependent modifier rules are:

1. IF noise-level .GT. noise_tolerance, THEN wake person.
   constraint : limit on the proximity of the person to the source of the noise
2. IF density_of_smoke .GT. smoke_tolerance of individual, THEN inactivate person.
   constraint : consider only the smoke_density of the current location of the person.
3. IF fire exists in the current_room, THEN add onto the information buildup factor (that may modify the current_action or goal).
   constraint: current_goal must not be fight fire.
4. IF current_room ventilated, THEN increase tolerance_to_stress.
   constraint :person must not be already incapacitated.
   person must be in the ventilated room.
5. IF current_room_ventilated, THEN reduce the smoke_level of current_room
6. IF fire extinquished, THEN increase tolerance_to_stress.
   constraint : person must not be already incapacitated.
7. IF tolerance_to_stress .GT. information_build_up_factor, THEN activate action change.
8. IF asleep, THEN no_action
9. IF mobility nil, THEN no_action etc.

Currently, the project summarized above is under development. There are plans to access an object-oriented data model that is being developed using INGRES data base system with an interface to AutoCAD as part of a code analysis expert system (Ozel & Mackellar, 1990).

## 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 distinguishing 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.

The distinction between architectural expert systems and expert simulations lie in the fact that the former aims at aiding the architect in reasoning about architectural objects, whereas

the latter tries to simulate how intelligent objects reason about processes in such environments. Due to this 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

Afsarmanesh, H., Knapp,D.,McLeod,D., & Parker,A., "An extensible Object Oriented Approach to Databases for VLSI/CAD", in Object Oriented Data Base Systems, Zdonik, S. & Maier, (Eds.), D.,Morgan Kaufmann Publishers, San Mateo, CA., 1990, pp.607-618.

Balci, Osman, (Ed.) *Methodology and Validation,* Proc. of the conference on Simulations, The Society for Computer Simulation, San Diego, CA. 1987

Birtwistle, Graham & Kendall, John, "A View of LISP", in Intelligent Simulation Environments, Simulation Councils Inc., SanDiego, CA. 1986, pp.157-161.

Coyne,R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., & Gero, J.S., *Knowledge-Based Design Systems*, Addison-Wesley Co., NewYork, 1990.

Goldberg, Adele & Robson, David, *Smalltalk-80: The language and its implementation*, Addison Wesley, 1983.

Jones, W., "FAST: A Fire and Smoke Spread Model", NBS, Center for Fire Research Publication, 1986.

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

Lochovshy, F. (ed.) *Database Engineering*, Vol.8, No.4, Special Issue on Object oriented Systems, 1986.

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

Luker, Paul, & Adelsberger, Heimo, (Eds.) *Intelligent Simulation Environments*, Proceedings of the Conference on Intelligent Simulation Environments, Simulation Councils Inc., San Diego, CA. 1986.

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.

Oren, Tuncer, "Knowledge bases for an advanced simulation environment", in Intelligent Simulation Environments, Simulation Councils Inc., SanDiego, CA. 1986, pp.16-22.

Ozel, Filiz, "The Computer Model BGRAF: A Cognitive Approach to Emergency Egress Simulation", Un. of Michigan, Doctoral Program in Arch. Dissertation, Ann Arbor, Mi., 1987.

Ozel, F., & MacKellar, B., "An Integrity Constraint Approach to Verifying Building Designs", Research Paper accepted for presentation and publication at the First International Conference on Artificial Intelligence in Design, Edinburgh, Uk. 1991.

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

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

Stahl, F.I., "Final Report on the BFIRES/Version 1, Computer Simulation of Emergency Egress Behavior During Fires: Calibration and Analysis, NBSIR 79-1713", National Bureau of Standards, Fire and Life Safety Program, Washington DC., March, 1979.

Turner, J., Hall, Ted, & Borema, Lynn, "CAEADS: User's manual", Un. of Michigan, Architecture and Urban Planning Research Lab., Ann Arbor, Mi., 1984.

Zdonik, Stanley, & Maier, David, (Eds.), *Object Oriented Data Base Systems*, Morgan Kaufmann Publishers, San Mateo, CA., 1990.