DOT/FAA/AM-97/20

Office of Aviation Medicine
Washington, D.C. 20591

# Use of Object-Oriented Programming to Simulate Human Behavior in Emergency Evacuation of an Aircraft's Passenger Cabin

Mary C. Court
University of Oklahoma
Norman, OK 73019

Jeffrey H. Marcus
Civil Aeromedical Institute
Federal Aviation Administration
Oklahoma City, OK 73125

August 1997

Final Report

U.S. Department of Transportation
Federal Aviation Administration

Technical Report Documentation Page

| | | |
|---|---|---|
| 1. Report No. DOT/FAA/AM-97/20 | 2. Government Accession No. | 3. Recipient's Catalog No. |
| 4. Title and Subtitle Use of Object-Oriented Programming to Simulate Human Behavior in Emergency Evacuation of an Aircraft's Passenger Cabin | | 5. Report Date August 1997 |
| | | 6. Performing Organization Code |
| 7. Author(s) Court, M.C.[1] and Marcus, J.H.[2] | | 8. Performing Organization Report No. |
| 9. Performing Organization Name and Address [1]University of Oklahoma School of Industrial Engineering Norman, OK 73109 | [2]FAA Civil Aeromedical Institute P. O. Box 25082 Oklahoma City, OK 73125 | 10. Work Unit No. (TRAIS) |
| | | 11. Contract or Grant No. |
| 12. Sponsoring Agency name and Address Office of Aviation Medicine Federal Aviation Administration 800 Independence Avenue, S.W. Washington, DC 20591 | | 13. Type of Report and Period Covered |
| | | 14. Sponsoring Agency Code |
| 15. Supplemental Notes | | |

16. Abstract

Progress in the development of a computerized aircraft cabin evacuation model is described. The model has a two-fold purpose (i) to supplement current certification tests that use human subjects, and (ii) to serve in the investigation of aircraft accidents as a reconstruction tool and identify factors influencing survival of passengers. For the model to be a valid predictive tool when simulating aircraft accidents, the toxic and debilitating effects on passenger behavior of fire and smoke must be modeled. Other aircraft cabin evacuation models use an expert system/rule-based approach to simulate these effects. The work described here presents an object-oriented approach to modeling human behavior in aircraft cabin evacuations. Object-oriented programming (OOP) lends itself to the modeling of complex systems. OOP's foundation is modularity. OOP allows a one-to-one correspondence with the physical world, and thus, eases the burden of model validation. Validation is critical to the successful use of a model as a predictive tool and involves testing to ensure that the model accurately reflects the behavior of a real system. Easing model validation is of particular importance since the real system's environment is hazardous, and performing any tests on the real system is either impossible or not repeatable. The result of this work will help to expand the simulation's capabilities in improved passenger queuing analysis by allowing the incorporation of human behavior into class objects.

| | | | |
|---|---|---|---|
| 17. Key Words Evacuation Model; Cabin Safety; Object-Oriented Programming; Human Behavior | | 18. Distribution Statement Document is available to the public through the National Technical Information Service Springfield, Virginia 2261 | |
| 19. Security Classif. (of this report) Unclassified | 20. Security Classif. (of this page) Unclassified | 21. No. of Pages 11 | 22. Price |

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

# Use of Object-Oriented Programming to Simulate Human Behavior in Emergency Evacuation of an Aircraft's Passenger Cabin

## 1. SUMMARY

The paper presents an object-oriented framework to model human behavior under both certification and accident evacuations. The framework opens up a new area of analysis by proposing a paradigm for predicting human behavior. Object-oriented programming lends itself to the modeling of complex systems by supporting a one-to-one correspondence with the physical world, and thus, eases the burden of model validation. Easing model validation is of particular importance when the real-system's environment is hazardous, and performing tests on the real-system is either impossible or not repeatable.

## 2. INTRODUCTION

New designs of passenger aircraft are required to show compliance with 14 Code of Federal Regulations, Part 25, Section 803, *Emergency Evacuation.* This requirement is frequently referred to as *The 90 Second Rule.* The manufacturer must show that with half of the available exits blocked, a full load of passengers can safely evacuate the aircraft into a darkened hanger in 90 second or less. This requirement provides a performance based test of the emergency evacuation system of an aircraft. It has been found that in many accidents, the passengers survive the impact but perish because they are overcome by smoke and fire while trying to evacuate (Marcus 1994).

This certification testing has proven to be quite stressful and costly to the manufacturer. Today's certification test costs an average of $2.3 million, involves over 4000 people, and requires three years of planning (Shook 1995). Adding to the cost is the risk of injuries to the test subjects. Consequently, there has been increasing pressure to improve certification tests' safety, even if the resulting tests give up some realism (Marcus 1994). There is one realism that certification tests have never incorporated: the dynamic environment of an aircraft cabin during an accident (i.e., fire and smoke).

Since requiring manufacturers to perform certification tests under actual accident conditions is ethically unacceptable, the need then exists to develop an evacuation model capable of simulating (i) various cabin configurations, (ii) the dynamic environment of fire, and (iii) passenger behavior. At issue is the ability to accurately predict human behavior. More specifically, the ability to simulate the physical and psychological effects fire and smoke have on human behavior and decision making.

Past evacuation models have taken an expert system/ rule-based approach to model human decision making and behavior during aircraft evacuations (Galea and Galsarsoro 1993, Schroeder and Tuttle 1991). The authors present a new approach to modeling human behavior: object-oriented programming.

Object-oriented programming (OOP) has two inherent features:

(1) OOP lends itself to the modeling of complex systems by supporting modular construction.
(2) OOP yields a one-to-one correspondence with the physical world and thus, eases the burden of validation.

Validation is critical to the successful use of a model as a predictive tool and involves testing to ensure that the model accurately reflects the behavior of the real system. Easing model validation is of particular importance when the real-system's environment is hazardous.

Other benefits of an object-oriented approach to predicting human behavior are: (i) human behavior modeling will support and enhance certification tests, currently conducted with human subjects, since simulated tests lend themselves to statistical and predictive analysis, and (ii) human behavior modeling will assist in the investigation of aircraft accidents by providing a means to analyze how behavior influences passenger survivability. For example, human behavior modeling will allow investigations into the impact flight attendants and their behavior have on directing passengers.

The paper presents an object-oriented framework for modeling human behavior and decision making during aircraft evacuations. The remainder of the paper is organized as follows:

(1) Section 3, *Objectives*, outlines the goals for the human behavior model.
(2) Section 4, *Object oriented Programming*, presents a brief overview of the object-oriented approach to programming, corresponding terminology, and recent results in modeling cognitive processes via an object-oriented approach.
(3) Section 5, *Proposed Framework*, is the proposed object-oriented framework for modeling human behavior during aircraft evacuations.
(4) Section 6, *Conclusions and Future Research*, provides an overview of the construct and outlines future work.

## 3. OBJECTIVES

The object-oriented framework of Section 5 is proposed to support the development of human behavior models for analyzing aircraft cabin evacuations, with the following objectives:

(1) The model must be capable of analyzing various aircraft cabin configurations without requiring changes to its source code.
(2) The model must run in real-time or near real-time.
(3) The model must be able to conduct simulations of both certification tests and accident evacuations.
(4) The model must consider relationships among passengers. For instance, the impact on the evacuation behavior of a mother traveling with an infant versus a passenger traveling alone, must be incorporated.
(5) The model must consider the impact a flight attendant's behavior has on passengers. This feature will allow *passenger management* to be explored, such as determining the optimal number of flight attendants per passenger load.
(6) The model must offer dynamic behavior as opposed to behavior that is fixed at the time of model execution. That is, the model must allow the behavioral characteristics of the passengers to change over time.
(7) The model must take into account the dynamic, toxic environment of fire and consider the physical as well as psychological effect of fire and smoke on human behavior.
(8) The model must support simulation output analysis, designs of experiments, and sensitivity analysis.
(9) The model must provide animation of the evacuations to support model validation and presentations.

## 4. OBJECT-ORIENTED PROGRAMMING

The advantages of object-oriented programming over traditional (procedural) programming are well documented (Cox 1986, Meyer 1987), as are the advantages of object-oriented design and development (Jacobson 1991, Kamath et al. 1993, Wang and Fulton 1994, Nof 1994, Fishwick 1995). The major advantage of this approach to modeling is the preservation and reusability of source code. Traditional design uses the functions the systems performs as its basis for software development; while the object-oriented approach uses objects the system manipulates as its foundation. Since functions are likely to fluctuate and objects tend to be stable, the object-oriented approach allows for modularity in design and maintains reusability of software. Reductions in software development cycles have been realized as a direct result of code reusability (Kamath et al. 1993).

The following are definitions and features of object-oriented programming supporting the modular decomposition of the software and code reusability:

(1) *Inheritance* allows for base code reusability and the implementation of new objects from existing objects. All objects belong to a class, where classes are defined in a hierarchical tree structure with subclasses inheriting the procedures and data storage structures of their superclasses. For a biological system, man is a subclass of mammals, mammals is then a sub-class of vertebrate, etc.

(2) Items within a system are called *objects*. This classification allows for the separation of physical items from functionality. Objects are treated either as a *class* or an *instance* of a class. A class is the software module providing the complete definition (capabilities) of the members within a particular class. These definitions are obtained either by procedures and data stored directly within the class definition, or are inherited from other related classes. An instance of a class is a realization of the class having all of the capabilities provided in its class definition. That is, an instance represents the execution of a class. Continuing the example of a biological system, a man is an instance of the class mammals. Man then inherits the features of other related classes, such as a vertebrate and animals. Man is distinguished from other mammal instances, like whales, through man's class definition of reasoning capabilities.

(3) *Late binding* allows for delaying the process of joining procedures to the data on which they will operate until execution of the model. Traditional programming uses early binding where the procedures and their data are joined (hard coded) at the time of code construction. Delaying the binding allows data types to change during execution and, again, supports code reusability.

(4) *Encapsulation* allows internal class implementations to be modified without changing the relationships of the instances of classes to other objects. Encapsulation ensures that an object's definition is within an impenetrable boundary. That is, the data stored within an object is only accessible by the procedures of its defined class. Message passing, as defined below, is a direct consequence of encapsulation.

(5) For one object to interact or affect the internal condition of another object, the requesting object must send a message asking the second object's procedures to execute the request. This is called *message passing*. Sending a message to an object invokes the same-named *method* (execution of code/routine/procedure) to be carried out by that object. Methods may either be inherited or are within an instance. Groups of similar methods are called *protocols*.

(6) The capability of different objects to respond to the same message in the appropriate manner is called *polymorphism*. That is, the message initiates different behaviors in various objects, even though the same message is sent.

*Behavior* is defined as the action an object takes when a message is received and when the object's behavior is influenced by its past behavior, then the object is said to have a *state* (Bourne 1992). For an object to achieve a state, both class and instance (internal) variables must be defined. Class variables are common to all instances of a class, while internal variables define the state for a particular instance.

It was not until the 1980s when the works of King and Fisher (1986), Thomasma and Ulgen (1988), Adiga (1989), Mize et al. (1989), and Ulgen et al. (1989) showed that the underlying concepts of object-oriented programming could be extended to simulation modeling.

Recent work has involved designing and developing an object-oriented simulation environment for manufacturing systems (Tretheway and Court 1995). The approach was to model the physical and information/decision components of the manufacturing environment separately by using a five-level structured hierarchy of subsystems with the Smalltalk-80 language (Goldberg and Robson 1989). A set theoretic formalism, first proposed by Karacal (1990), was used to support this separation.

There are many similarities between modeling manufacturing environments and modeling human behavior, such as:

(1) both systems are physical, yet information/decision based environments,
(2) both systems use complex decision-making heuristics and structures to control various operations (behavior),
(3) both are subject to the same problems associated with decision making: decisions can only be based on information that is currently available, although it may be incomplete or inaccurate,
(4) a passenger (like a decision maker within a manufacturing environment) will use heuristics, personal experience, and rules to arrive at control (behavior) decisions, and,

(5) as manufacturing systems are designed to achieve different goals and objectives yielding different levels of performance; humans are motivated and driven by different preferences and motivations, yielding different behaviors.

However, if the formalism for simulating manufacturing environments is followed, the objective of having the model run at real-time or near real-time will not be met.

In 1988, Burns and Morgenson (1988) published a construct for simulating systems involving endogenous decision making. Their work proposes describing the system in terms of a *suite of actor classes* (collection of object classes) whose endogenous decisions impact the performance and behavior of the system. They suggest a model where all actors, including pseudo-actors (environment), follow an actor-centered description (Figure 1). Each actor class requires data structures (assets, attributes and vulnerabilities) and methods (cognitive and physical capabilities), described as follows:

(1) *Assets* are discernible characteristics and *attributes* are descriptive characteristics. The actor's own assets and attributes comprise the *actual state*, while the *perceived state* is the actor's perception of its surroundings (the environment and other actors). The state of the actor is the combined data structures of the actual and perceived state. This state data is the input to the *cognitive inference engine* of the actor.

(2) The actor can physically move (*transfer*) or change (*transform*). Transformation takes place by modifying the actor's assets/attributes. *Vulnerabilities* represent degradation to the actor's capabilities via the reduction or destruction of the actor's assets.

(3) An action space for cognitive capabilities and activities describes the decision set and state of each actor. A *cognitive event* (decision) is capable of (i) delaying decisions, (ii) invoking physical activity, and (iii) changing the action space. By delaying decisions and changing the action space, an actor then has the ability to "change its mind" (non-monotonic reasoning).

Notice that the actor-centered description proposed by Burns and Morgenson (1988) is not a pure object-oriented paradigm, since knowledge bases (production rules and heuristics) and inference engines are utilized for achieving cognitive activity. Then adopting this approach would equate to developing data, knowledge, and method structures for each actor (passenger) and pseudo-actor (environment). Thus the objective of supporting real-time simulation may not be met.

The proposed framework of Section 5 modifies the actor-centered description by avoiding the incorporation of knowledge bases and inferences via a pure object-oriented paradigm. By maintaining a pure object-oriented approach, it is expected that all of the objectives for the human behavior model (outlined in Section 3) will be achieved.

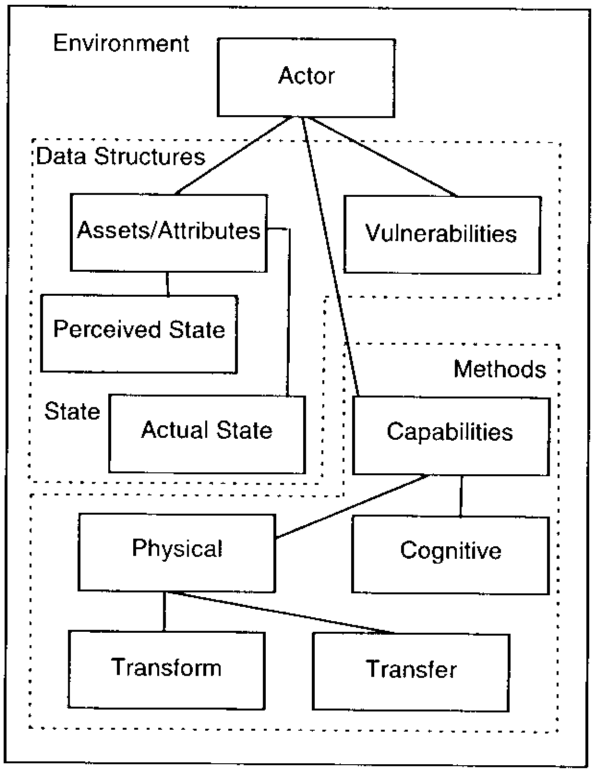

**Figure 1**. Actor-Centered Description
Source: Burns and Morgenson [1988] {modified}

## 5. FRAMEWORK

The proposed framework for modeling human behavior is to adopt the actor-centered description of Burns and Morgenson (1988) but avoid the incorporation of knowledge bases and inference engines. This is achieved by allowing the actors to obtain their data and functions by copying other objects or parts of objects. This construct supports the need to have (i) a varied passenger and crew profile, (ii) a wide variety of aircraft cabin configurations, and (iii) the capability of simulating various hazardous environmental conditions.

An overview of the system is presented in Figure 2. The system is described below:

(1) The user is responsible for constructing a *passenger scenario module,* or choosing a system generated scenario. The passenger scenario includes all of the physical characteristics of the passengers and crew

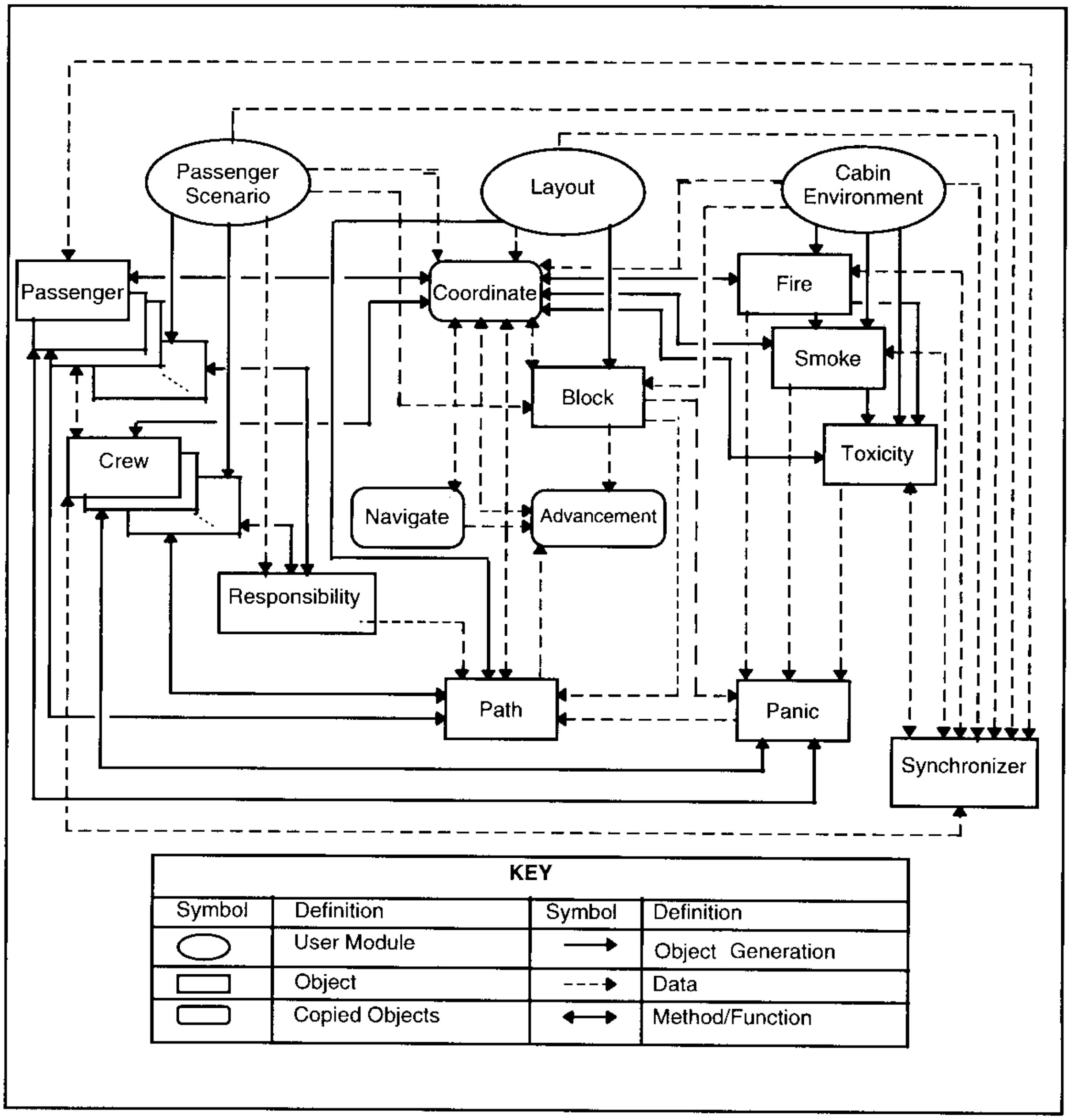

**Figure 2.** Object-Oriented Framework for Human Behavior Model

members (age, sex, height, weight, etc.), relationships between passengers (husband, wife, father, etc.) and those passengers identified as traveling together.

(2) The *layout module* is used to assign passengers and crew members to their seats and generate cabin configurations (number of rows, number and types of exits, locations of exits, aisle widths, etc.). The user has the option of using pre-existing cabin configurations, as well as the capability of generating new configurations. The user also has the option of pre-assigning the exits passengers and crew members use to evacuate the aircraft. Operable and blocked exits are also identified in this module.

(3) The *cabin environment module* is the vehicle to simulate accident/hazardous conditions. For example, the user has the option of invoking a pre-set fire with a known location and any ensuing toxicity or smoke. To run a certification test simulation the user would choose not to initiate this module.

(4) When the input to the three modules is complete, the system compiles the data and generates the various *actors* and *pseudo-actors.* For each passenger and crew member an actor object is created. The *fire, smoke,* and *toxicity* objects are pseudo-actors. For certification simulations these objects are not generated. Pre-existing objects are the *coordinate, navigate, advancement, synchronizer, panic,* and *responsibility* objects. The *synchronizer* object coordinates the objects and runs the simulation.

The coordinate, navigate, and advancement objects are copied into each of the actors and pseudo-actors. This construct avoids the need for developing knowledge bases and inference engines for each type of actor or pseudo-actor generated. The coordinate object receives the data input by the user and generates a map of the aircraft cabin. The passenger, crew, fire, smoke, and toxicity objects copy the coordinate functions of the coordinate object, and thus, are able to store and update their positions and distances from other objects. The navigate object is also copied into each actor object; and if applicable, the fire, smoke, and toxicity objects. The navigate functions allow the actors and pseudo-actors the capability of choosing headings (direction) for movement. The path object is called upon to generate possible paths for the actors and pseudo-actors, (fire, smoke, and toxicity) based on their positions, headings, and cognitive abilities to access the environment. The advancement object functions are used to move the actors and pseudo-actors to their requested positions. Data from the block object is used to keep actors and pseudo-actors from moving into inaccessible positions. The block data types consist of architectural (seats, walls, etc.), human (passenger and crew), and environmental (fire, smoke, and toxicity) obstacles.

How "capable" the actor is at using the coordinate, navigate, and advancement objects depends on its physical and cognitive capability objects. For example, the possible paths an object can construct and how many times a new path is generated, depends not only on how often the path object is called upon, but is a function of the actor's actual and perceived states. Thus, path generation is a function of the actor's (i) immediate environment (fire, smoke and toxicity levels), (ii) ability to access its current path and blockages, (iii) time spent in hazardous environments, and (iv) the type of evacuation being performed (certification or emergency).

The panic object directly influences the actor's ability to reason and react. Again, the call to the panic object will depend on the actual and perceived state of the actor and thus, is a function of the actor's capabilities.

The responsibility object is used to bind objects together. This is one of the vehicles used to establish a psychological profile for each actor, as well as a means to distinguish flight attendants and crew members from passengers. A flight attendant is expected to assist and direct passengers during an evacuation. In this paradigm, a flight attendant actor has access to the internal data of other objects. This is achieved in object-oriented programming by designating objects as *friends* to other objects. Friend-type objects also include passengers traveling together. The amount of internal data sharing depends on the relationship type and the amount of responsibility an actor has toward another actor.

The construct supports pre-defined biological hierarchies but allows distinction between objects within the same biological class. For example, although females have many similar physical characteristics, they do not have the same physical capabilities; likewise, they do not have the same cognitive reasoning abilities. Thus a distinction based on technical knowledge can be made between a female passenger and a female flight attendant. That distinction is incorporated through the ability to copy the coordinate, navigate, and advancement objects. The flight attendant is expected to have knowledge of the aircraft's configuration and therefore, has more access to the functions and data of the aforementioned objects then an average passenger. Also consider a female passenger traveling alone versus one traveling with an infant. The mother is bound to her child and therefore, would be expected to ensure that the child is evacuated safely. In the construct, when the mother actor is generated a copy of the bonding function from the responsibility object is copied into the *mother actor.* The mother actor is now tied to her *child actor* where the child's state is input to the cognitive object of the mother actor.

## 6. CONCLUSIONS AND FUTURE RESEARCH

The construct presented is an object-oriented approach to modeling human behavior and decision making during aircraft evacuations. The construct alters the actor-centered description of Burns and Morgenson (1988) by allowing objects to copy other objects for function execution. The ability to copy objects or parts of objects is the mechanism for actors to carry out cognitive and physical activities and thus, avoids the need for inference engines and knowledge bases.

Actors are capable of carrying out non-monotonic reasoning by repeatedly copying other object functions when deemed necessary by the actor. The number of times the copying can be carried out, the function that is actually copied, and the action that is taken by the actor (including no action at all), are all dependent on the actor's own cognitive and physical capabilities.

The construct also provides a means to study (i) the debilitating effects fire, smoke, and toxicity have aircraft evacuations, (ii) passenger management issues, and (iii) bonding.

The construct is currently being implemented at the University of Oklahoma's School of Industrial Engineering. The human behavior study is at the object and user module development stage, with validation being performed against certification test data. The study is also identifying the physical and psychological parameters most influential to passenger survivability (Jayarama and Court 1995). A preliminary set of parameters is currently being incorporated into the model.

## 7. REFERENCES

Adiga, S. (1989), "Software Modelling of Manufacturing Systems: A Case for an Object-Oriented Programming Approach," In *Analysis, Modelling and Design of Modern Production Systems,* A. Kusiak and W.E. Wilhelm, Eds.,

Bourne, J.R. (1992), Object-Oriented Engineering: Building Engineering Systems Using Smalltalk-80, Aksen Associates Incorporated Publishers, Homewood, IL.

Burns, J.R. and J.D. Morgenson (1988), "An Object-Oriented World-View for Intelligent, Discrete, Next-Event Simulation," *Management Science 34,* 12, 1425-1440.

Cox, B.J. (1986), Object-Oriented Programming: An Evolutionary Approach, Addison Wesley, New York.

Fishwick, P.A. (1995), *Simulation Model Design and Execution: Building Digital Worlds,* Prentice-Hall, Inc., Engelwood Cliffs, NJ.

Galea, E.R. and J.M. Perez Galsarsoro (1993), "EXODUS: An Evacuation Model for Mass Transport Vehicles," *Technical Report: CAA Paper 93006,* Civil Aviation Authority of the United Kingdom, London,

Goldberg, A. and D. Robson (1989), *Smalltalk-80 The Language,* Addison-Wesley, Reading, MA.

Jacobson, I. (1991), *Object-Oriented Software Engineering,* ACM Press, New York.

Jayarama, S. and M.C. Court, "Evacuation Model Parameter Sensitivity Study: Project Progress," Technical Report, School of Industrial Engineering, University of Oklahoma, June, 1-34.

Karacal, S.C. (1990), "The Development of an Integrative Structure for Discrete event Simulation, Object-oriented Programming, and Imbedded Decision Processes," Ph.D. Dissertation, School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK.

Kamath, Y.H., R.E. Smilan, and J.G. Smith (1993), "Reaping Benefits with Object-Oriented Technology," *AT&T Technical Journal*, September/October, 14-21.

King, C.U. and E.L. Fisher (1986), "Object-Oriented Shop-Floor Design, Simulation, and Evaluation," In *Proceedings of the 1986 Fall Industrial Engineering Conference*, Institute of Industrial Engineers, Norcross, GA, 131-137.

Marcus, J. (1994), "A Review of Computer Evacuation Models and Their Data Needs," Technical Report Number DOT/FAA/AM-94/11, Department of Transportation, Federal Aviation Administration, May 1994. NTIS ordering no. ADA280707.

Meyer, B. (1987), "Reusability: The Case for Object-Oriented Design," *IEEE Software*, 4, 2, 50-64.

Mize, J.H., T.G. Beaumariage, and S.C. Karacal (1989), "Systems Modelling Using Object-Oriented Programming," In *Proceedings of the 1989 Spring Industrial Engineering Conference*, Institute of Industrial Engineers, Norcross, GA, 13-18.

Nof, S.Y. (1994), "Critiquing the potential of object orientation in manufacturing," *International Journal of Computer Integrated Manufacturing*, 7,1,3-16.

Schroeder, J.E. and M.L. Tuttle (1991), "Development of an Aircraft Evacuation (AIREVAC) Computer Model, Phase I: Front End Analysis and Data Collection," *Technical Report: SwRI Project Number 12-4099*, Southwest Research Institute, San Antonio, TX.

Shook, William (1995), *Technical Meeting on the Evacuation Model Parameter Sensitivity Study*, University of Oklahoma, School of Industrial Engineering, March 6, 1995.

Tretheway, S. and M.C. Court (1995),"The Experimental Frame: A Conceptual Structure and an Outline of Research Issues for Developing a Comprehensive Manufacturing Analysis Tool," *The 1996 Industrial Engineering Research Conference*, May 1996.

Thomasma, T. and O.M. Ulgen (1988), "Hierarchical, Modular Simulation Modeling in Icon-based Simulation Program Generators for Manufacturing," In *Proceedings of the 1988 Winter Simulation Conference*, M. Abrams P. Haigh, and J. Comfort, Eds. IEEE, Piscataway, NJ, 254-262.

Ulgen, O.M., T. Thomasma, and Y. Mao (1989), "Object Oriented Toolkits for Simulation Program Generators," In *Proceedings of the 1989 Winter Simulation Conference*, E.A. MacNair, K.J. Musselman, and P. Heifelberger, Eds. IEEE, Piscataway, NJ, 593-600.

Wang, C.Y. and R.E. Fulton (1994), "Information system design for optical fiber manufacturing using an object-oriented approach," *International Journal of Computer Integrated Manufacturing*, 7, 1, 61-73.

☆U.S. GOVERNMENT PRINTING OFFICE: 1997 - 559-079/60027