

ENHANCING OBJECT-ORIENTED SIMULATION WITH RULE-BASED EXPERT SYSTEM

Otto Lee & Hon Wai Chun

City University of Hong Kong
Department of Electronic Engineering
Tat Chee Avenue
Kowloon, Hong Kong
email: eehwchun@cityu.edu.hk

ABSTRACT

This paper describes our research in developing an intelligent simulation system by enhancing an object-oriented simulator with a rule-based expert system. Designing and implementing a simulation model for a particular problem requires extensive expertise in simulation techniques, computer languages and queuing theory. However, those who would like to use simulation to assist their decision making might not always be technically knowledgeable enough to develop simulation programs. The primary motivation for our research is to shift the burden of simulation software development to an intelligent artificial intelligence system. The test domain for our research is simulating the queuing situation at airport check-in counters. The main objective for the simulation is to determine how many counters are needed in order to check-in all passengers given a certain level of service standard.

KEYWORDS

knowledge-based simulation, resource allocation, scheduling, expert system

1. INTRODUCTION

This paper describes an intelligent domain-specific simulator called RBOOS. This intelligent simulator was developed as an experiment in integrating an object-oriented simulator with a standard OPS5 rule-based expert system to provide a more accurate and user friendly approach to performing simulation experiments. RBOOS is an airport simulation system which can predict the number of check-in counters needed to adequately service the passengers of one departing flight. The graphic user interface was designed so that a novice user can easily use the simulator without technical training in simulation techniques.

RBOOS will analyse the current problem (defined as a set of flight information), determine appropriate simulation parameter values using the rule-based expert system, build the simulation model, run the simulation, compare the simulation output with the desired goals, make changes to the model parameters if necessary, and continue until the simulation goals are achieved. The simulation goals define the level of service required at this airport and is measured in terms of queue length and waiting time.

RBOOS consists of four main components (see Fig. 1) - the Object-Oriented Simulation Module, the Animated Graphics Module, the Intelligent Front-End Module, and the Intelligent Back-End Module.

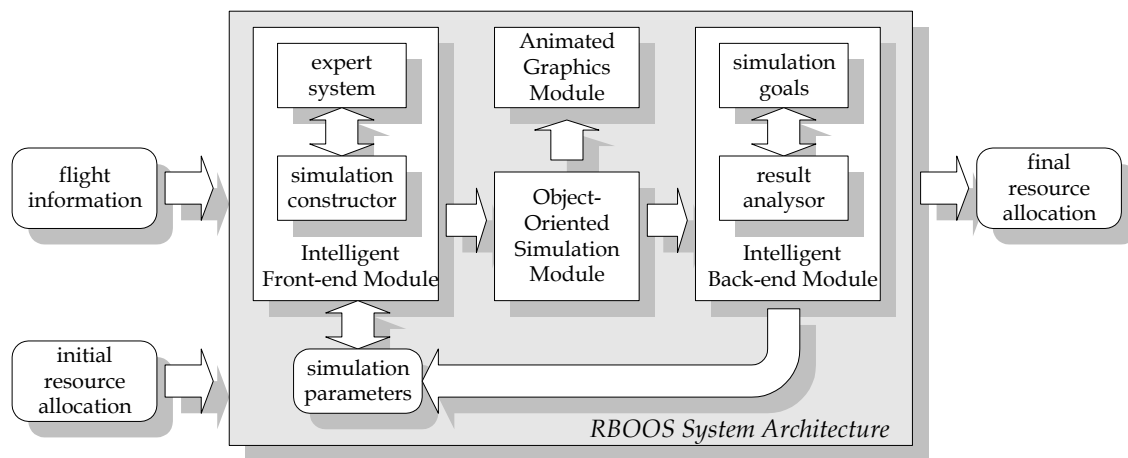


Figure 1. Overview of the RBOOS system.

The simulator [7, 8] was implemented using object-oriented techniques since it is a more natural way of representing a system than is possible in procedural languages. In the Object-Oriented Simulation Module, we have implemented objects that correspond to real world entities such as flights, passengers, check-in counters, queues, etc.

The Animated Graphics Module produces an animation of all the queues and passengers. Also displayed are running statistical information such as the next arrival time, next departure time, number of customer in the counter queue, total number of customer in the queue, accumulated server utilisation, and accumulated queue length.

The Intelligent Front-End Module has two main components - the rule-based expert system and the simulation constructor. The rule-based expert system is implemented using OPS5 [9]. Encoded in the expert system is a set of rules that describes how the statistical parameters might be affected by the given information on the current flight. For example, one rule might be "Travellers heading to China will usually bring more luggage." This translates to a high value for the service time during simulation. A production system was selected because it allowed us to improve the intelligence of the simulator by incrementally adding more rules. Input to the expert system are flight information such as the destination of the flight, the airline, the total number of passengers boarding locally, etc. The constructor module takes the results from the rule-based expert system and constructs a simulation model. This simulation is then executed by our object-oriented simulator.

The Intelligent Back-End Module then analyses results from the simulation to determine whether the simulation goals have been met. If not, it will adjust the simulation parameters and re-executes the simulation until the goals are satisfied. The result of the whole simulation process is the total number of check-in counters that are needed over the time period that check-in is opened for a flight such that all the simulation goals are satisfied.

Fig. 2 displays the RBOOS graphic user interface. The main display is the lower left-hand corner which animates the results from check-in counter simulation. The GUI and the four system modules will be explained in detail in the following sections.

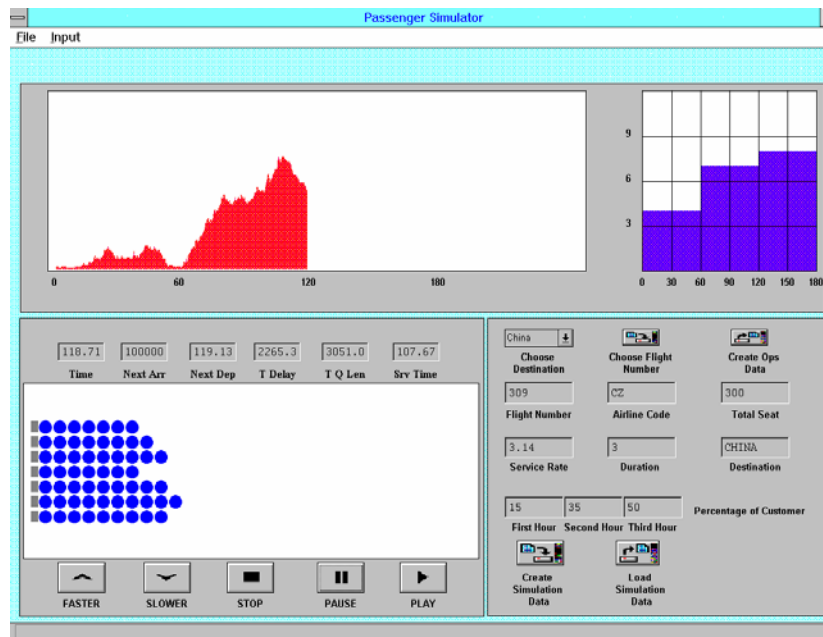


Figure 2. The graphic user interface to the RBOOS system.

2. INTELLIGENT FRONT-END MODULE

The “Intelligent Front-End Module” (IFEM) of the RBOOS system consists of two major components - the Rule-Based Expert System and the Simulation Constructor [5, 6]. The main purpose of the IFEM is to use encoded knowledge, stored as rules, to automatically generate a simulation model to pass to the object-oriented simulator.

2.1 Rule-Based Expert System

The Rule-Based Expert System in the IFEM was developed using OPS5 - a language to implement production systems [4]. A production system allows knowledge, in the form of “If-Then” rules, and control structure to be separated. This improves maintainability and enables the system to be incrementally enhanced by simply adding new rules or modifying existing ones. The version of OPS5 used in RBOOS was implemented in Common Lisp, hence integrating the expert system and CLOS-based simulator was made easier.

Information on a flight, such as airline, destination, departure time, etc. are stored as attributes of a fact. In addition, the IFEM fact base contains several sets of facts. There is a set of facts to relate the aircraft type to seating capacity, which depends on the aircraft configuration and will be different from airline to airline. To supplement this, there is a set of facts that encodes the average loading for each flight based on past flight statistics. There is also a set of facts to relate the destination to the haulage of that flight. The haulage is used to determine the duration that check-in counters are to be opened.

There are several types of rules in the production system. One type of rule determines the service rate from destination, haulage, and/or airline. If no specific service rate rule was fired, a default service rate will be used. In our experiments, the default was set to 2.8 minutes per passenger. For example, one rule may be “If destination is to country X, then the service rate will be higher than normal due to more than normal amount of baggage to be checked in.” Another type of rule determines the simulation duration from the destination and/or haulage.

A set of rules is used to predict the passenger arrival profile. The interarrival times differ for different time period within the check-in counter opening duration. In addition, the passenger arrival profile may be different from different time of the day or day of the week. For example, one rule may be "If flight is early morning, then passengers will usually arrive late."

2.2 Simulation Constructor

The duty of the Simulation Constructor is to convert the output of the rule-based expert system to simulation parameters that will be used to construct the simulation model. After all relevant information about a flight has been entered, the rule-based expert system can be called. The flight information is inserted into the *working memory* and relevant rules will then be triggered and fired by the *inference engine*. The final result is a set of simulation parameters, such as the duration of simulation, passenger arrival profile, number of passengers boarding locally, and the service rate. These parameters are used to construct the simulation model [2] for use by the object-oriented simulator.

3. OBJECT-ORIENTED SIMULATION MODULE

The "Object-Oriented Simulation Module" (OOSM) of the RBOOS system was implemented in CLOS using a *three-phase discrete event simulation* [3] algorithm. The simulator was implemented as two main layers. The first layer consists of a general object-oriented discrete event simulator with classes defined to represent customers, servers, queues, etc. The second layer is domain-specific to the airport check-in counter problem. Using subclassing capabilities offered by object-oriented programming, classes for passengers and check-in counter service agents were defined.

For airport check-in counter simulation, the simulation clock starts at time 0 indicating the time when check-in counters begin to service passengers for the flight. The event-driven simulation is triggered by passenger arrival and passenger departure from check-in area. The simulation will begin in the "empty-and-idle" state. i.e., no passengers are present and the servers are idle. The simulation ends at the time that check-in counters are to be closed for that flight. The duration that check-in counters are to be opened, i.e. the duration of simulation, depends on the haulage and/or destination of the flight. This is one of the parameters generated by the rule-based expert system. Usually, check-in processing will stop a certain number of minutes before the departure of a flight.

The OOSM uses the next-event time technique to advance the simulation clock, i.e. the simulation clock is advanced to the time of occurrence of the most imminent future event. The interarrival times of passengers are taken from the passenger arrival profile produced by the expert system. A passenger who arrives after an interarrival time and finds one of the check-in counter server idle enters service immediately. A passenger who arrives and finds all the servers busy joins the end of the shortest queue.

Among the various components of the OOSM program, the most important are the event handling routines. There are three main types of events: arrival of a passengers into the system, the departure of a passenger from the system after check-in, and data file generation. A separate routine was implemented to handle "arrival event" and "departure event." The result of the OOSM simulation is placed into a data file which contains information for each discrete point of time. The type of information includes the simulation clock time, next event time and type, number of queue, queue number, number of passengers in queue, server status, number delayed, total delayed, accumulated queue length, accumulated server utilisation, etc.

4. ANIMATED GRAPHICS MODULE

The "Animated Graphics Module" (AGM) of the RBOOS system provides an animated simulation of the queuing condition at a set of check-in counters assigned to service one single flight over the check-in duration for that flight. The object-oriented simulator simulates the check-in activity over the whole check-in duration and stores the simulation results into an ASCII data file. The AGM then reads this data file to perform the graphic animation.

The graphic animation is separated from the object-oriented simulator for several reasons. Firstly, the animation speed will be faster since simulation has already been performed in batch. Secondly, the data file format is such that data files generated by other simulation languages, e.g. SIMAN, can easily be used without any adjustment. In addition, previously generated simulation data files can be displayed by the AGM for comparison.

The lower left-hand corner of the RBOOS graphic user interface (see Fig. 2) displays the queuing animation. Each circle represents a customer in the check-in counter queue. Circles appear and disappear as passengers arrive and depart from check-in queues. The rectangles represent check-in counters. The state of the simulator is displayed in the rectangular boxes above the animation. In order to make the simulator easier to use, a tape player-like interface is borrowed with buttons for PLAY, PAUSE, and STOP. Two additional buttons, FAST and SLOW, control the speed of animation.

The number of counters opened to service a flight in each half-hour period is displayed in the upper right-hand corner of the RBOOS graphic user interface. For example, the particular simulation shown in Fig. 2 indicates that four counters will be operational in the first hour, then seven in the second, and finally eight counters in the last hour.

5. INTELLIGENT BACK-END MODULE

The "Intelligent Back-End Module" (IBEM) of the RBOOS system is used to analyse results from the simulation to determine whether the simulation goals have been met. If not, it will adjust the simulation parameters and re-executes the simulation until the goals are satisfied. The result of the whole simulation process is the total number of check-in counters that are needed over the time period that check-in is opened for a flight such that all the simulation goals are satisfied.

For example, the object-oriented simulator keeps a running tab of the queue length at particular points in time. The IBEM will analyse this information to check if the queue length exceeds a given threshold. This threshold is one of the simulation goals - maximum tolerable queue length. If the goal is violated, the simulation parameters of the number of the counters during a certain period is increased and the whole set of the model is simulated again until the goal is satisfied.

In the upper left-hand corner of the RBOOS graphic user interface (see Fig. 2), the height of the chart defines the "ceiling" - maximum tolerable value of a simulation goal. If the graph reaches the ceiling, the maximum tolerable value has been exceeded and a simulation goal has been violated.

6. SYSTEM IMPLEMENTATION

The RBOOS system presented in this paper was implemented using Allegro Common Lisp on a IBM-compatible PC running Microsoft Windows. The rule-based was implemented using a public domain Common Lisp version of OPS5. This version was based on a implementation done by Charles L. Forgy at Carnegie-Mellon University. The object-oriented simulator was developed from ground up using CLOS. The graphic user interface was implemented using the Interface Builder (a GUI source code generator) provided by Allegro Common Lisp.

7. CONCLUSION

This paper documents work in developing an object-oriented simulator and enhancing it with a rule-based expert system. The key advantage offered by this domain-specific simulator is that a novice user can easily obtain accurate simulation results without prior knowledge of simulation techniques or queuing theory. The knowledge required to analyse a problem and to convert the problem into a set of simulation parameters is already encoded into the OPS5 rules. By using a rule-based approach, the results of the simulation can be gradually improved by incrementally adding new rules.

The work described in this paper is part of an undergraduate thesis research. Further development is planned for graduate work. This includes extending the simulation to cover the whole airport, include passenger traffic flow into the simulation system, and exploring the possible interaction between an object-oriented simulator and a knowledge-based scheduling system [1].

REFERENCES

1. Sanjay Jain, Karon Barrer, David Osterfeid, 1990, October, "Expert simulation for on-line scheduling", *Communication of the ACM*, p.54-60.
2. Averill M. Law, W. David Kelton, *Simulation Modeling & Analysis*, 2nd Edition, McGraw-Hill.
3. James T. Lin, Chia-Chu Lee, 1993, June, "A three-phase discrete event simulation with EPNSim graphs", *Simulation*, p.382-385.
4. George F. Luger, William A. Stubblefield, 1993. *Artificial Intelligence Structures and Strategies for Complex Problem Solving*, 2nd Edition, The Benjamin/Cummings Publishing Company, Inc.
5. Galina V. Merkuryeva, Yury A. Merkuryev, 1994, February, "Knowledge Based Simulation Systems - A Review", *Simulation*, p.74 - 89.
6. Norman R. Nielsen, "The impact of using AI-based techniques in a control system simulator", *Simulator and AI*, SCS, p.72-77.
7. Subramanian Prakash, Robert E. Shannon, 1989, "Goal Directed Simulation Environment: A Prototype", *Proceedings of the 1989 Summer Computer Simulation Conference*, SCS, p. 545-549.
8. Robert E. Shannon, Richard Mayer, Heimo H. Adelsberger, 1985, June, "Expert systems and simulation", *Simulation*, SCS, p. 275-285.
9. Porter D. Sherman, John C. Martin, *An OPS5 Primer Introduction to Rule-based Expert Systems*, Prentice Hall.