Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

SIMULATION LANGUAGES AND DATABASE THEORY: SOME CONSIDERATIONS FROM THE ENTITY-RELATIONSHIP MODEL

Robert S. Roberts
New Mexico State University
Las Cruces, NM 88003

ABSTRACT

The Entity-Relationship (ER) model is an increasingly important database model for analyzing data requirements. This model has become a popular tool for use in computer systems analysis and design, because it provides a systems analyst with a way to model and design efficient data storage for a complex system. The major result of this type of analysis is the specification of data files which are readily implemented with, and accessed by, relational database software.

This paper applies the ER-database modeling techniques to discrete event simulation. The ER model provides a logical way to organize the data from a simulation run. When the output data are stored using this organization, a relational database can be used to make queries in a database language that should answer virtually any question about the simulation performance and facilitate the statistical analysis the data. Also, the ER modeling approach is contrasted with the models which are used in the simulation language SIMAN. This comparison provides insight into both approaches and suggests some modifications and additions to the SIMAN language which should improve its flexibility and make it more consistent with the ER theory.

1 INTRODUCTION

The Entity-Relationship (ER) model is one of the most popular modeling tools now used for database design. The ER model is used to design a relational database to store data for large and complex business systems. The key element of the ER-modeling process is the construction of ER-diagrams which are used to show relationships between key entities in a system (Litton, 1987).

Of course, simulations of large systems have complex data requirements. When discrete event simulations are used to model complex systems, a great deal of interesting data is often generated and collected for analysis. Data from simulations is collected for several reasons; the data may be used in exploratory and statistical analysis, hypothesis testing, and graphical presentation.

Although a number of graphical modeling techniques have developed for the construction of discrete event simulation, these tools are used primarily to model the physical system or design the computer program. For example, the state transition diagrams or modeling diagrams used in SIMAN, SLAM and GPSS, are mainly used to specify program logic (Pritsker, 1986; Pegden, et. al. 1990; Pegden, 1986; and Schriber, 1991).

Application of the ER-modeling techniques allows another way to conceptualize simulations. In this instance, the term entity as used in the ER model is not defined or used in the same way as it would be in simulation languages such as SIMAN, SLAM or GPSS. Comparing one of these languages with the ER modeling to simulations can provide insight into each approach. Therefore, this paper explains how the ER-model can be applied to the design of discrete event simulations and looks at the implications of this approach.

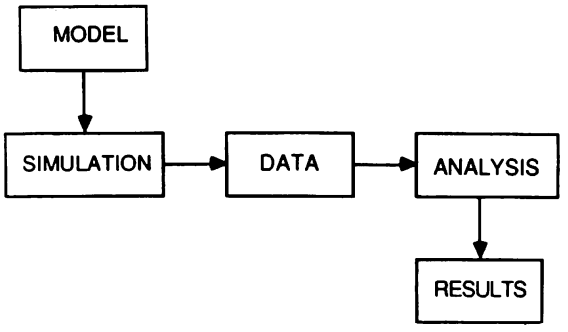

Figure 1. The Process of Simulation and Analysis.

2 BACKGROUND

Data collection is often viewed as one of the major parts of computer simulation. Figure 1, adapted from Pegden (1986), shows a typical conception of the organization of a computer simulation. In a first step, a model is used to specify the system to be simulated. Next, the model, expressed in a simulation language, is compiled into a simulation program. As the program runs, a file of data is collected and some preliminary reports are generated. Finally, the collected data is processed to produce summary reports, do statistical testing, and construct graphical presentations as a final result.

From a statistical point of view, there are two main reasons for data collection: exploratory data analysis and hypothesis testing. In exploratory work, the simulation may be run under a variety of conditions just to get a basic familiarity with the response of the target system and the accuracy of the simulation model. For these cases, it is probably desirable to collect as much data as possible from each simulation run. After the exploratory stage, hypotheses and conjectures will develop and then so more formal experimental design approaches are desirable. When the data collection is focused on an experimental design, data collection may be limited to the specific variables of interest.

No matter what data need to be collected, size and volume of the output becomes an issue. Concerns about storage requirements and memory size may require some limits on the data to be collected. For this paper, these concerns are largely ignored; for now, it is better to first take a look at desired performance without worry about technical constraints.

Many simulation models avoid saving large collections of 'raw data'. In these cases, the calculation of statistical measures is incorporated into the simulation program. For example, in the simulation of a single-server-queue, it may be very easy to collect the data necessary to calculate the mean and variance of the time-in-system by simply keeping track of three variables. On the other hand, if data on the time-in system for every job was saved in a file, then not only could the mean and variance be calculated, but correlations, plots, histograms, and many other statistical measures and techniques could be applied after the simulation run is complete.

Most of the commonly used discrete event simulation languages have built in provisions for both the calculation of limited statics and saving files of raw data. For example, in SIMAN a TALLY block in the model can be used to collect data on time-in-system. When a TALLY is used, certain statistics such as mean, variance, minimum, and maximum are readily available in a report. In addition, SIMAN also has a provision so that, each of the system-times used by the TALLY block can be saved in a file for later analysis by the SIMAN Output Processor.

Unfortunately, there are some situations in which these features are not sufficient. If the data on time-in-system is tallied and stored in a simple sequential file, a number of statistical tests and graphical procedures can be performed. However, certain questions might be asked such as "What was the average queue length which was seen by customers who had system times greater than X minutes?" or "What is the correlation between queue length and time-in-system?" Unfortunately, because the system-time data are stored in a simple file it would not be easy to answer these questions without modifying the model and rerunning the simulation. These complex queries could be answered if the data were appropriately stored with database software.

3 DATABASE MODELS

In a well designed database system, information is stored on records in files. In the case of the single server queue, a natural way to store the information would be to create records for each customer. In database terminology, each record corresponds to an entity (in this case, the customer) and the records contain fields to specify information about the various attributes of each entity. If a single server queue

simulation created a file to store the experience of each customer, appropriate processing of that file with a database query language could extract virtually any information about system performance or at least about customer history. The data from a customer file could also be directly processed as the input to statistical analysis software such as SAS (1985).

A database for a more complex system would have information stored in several related files with a database query language designed to extract information from such a set of related files. Database theory can be used to determine how data for various systems should be stored for most efficient data retrieval and processing. A database has the advantage of storing data so that any query can be expressed simply by using the database query language. Ad-hoc questions and reports can be handled easily with the database query language.

4 RELATIONAL DATABASE MODELS

Historically, the three most popular database models have been the network model, the hierarchical model, and the relational model. The older database models, including the hierarchical model and the network model, required complex and sophisticated storage techniques. The newer and more popular relational model assumes that all data are stored in a set of tables (or relations) which are usually implemented as so-called 'flat files'. Each record in the files corresponds to a row of the table and each field corresponds to a column. Queries are performed using a query language which manipulates the files. A theory has been developed to prove that all queries on the database can be accomplished with some combination of three operations: the *join, project* and *select*. The *join* operation is used to merge tables (files), the *project* reduces a table to include only specific columns, and the *select* reduces a table to include only specific rows. In addition, data in related tables can be referenced by using key values. For example, in a banking application, a customer id-number can be used to find the appropriate record in a customer file and also look up transactions in a transaction file.

Most database software includes extensions to the simple relational language which include statistical functions. These features do some data processing so that the mean, variance, or simply a count of select values is easily accomplished. For example, with a database containing class registration information, not only is it relatively easy to list all the students enrolled in the simulation class but the query might just ask about the total enrollment, the average enrollment in several classes.

Construction of the tables or files used for data storage is a prime concern in relational database theory. A process called normalization is used to reduce larger tables into a set of more compact and efficient related tables which can store the same data. In the case of a student registration system, it would be possible to store all data in one file. If this were done, the file would be excessively large and contain much redundant information. In such a file, each record would contain information about a student registered in a class and would have to include all student information, course information, and the course grade. Obviously, this is not an efficient scheme. File normalization is used to break a large file into a more efficient and logical set of related files. For example, in the case of student registration, the information should be saved in three files, a student file, a class file, and a registration file. Fortunately, the ER-model techniques automatically stores data into appropriate files which are appropriately normalized into their efficient form.

The principles of normalization provide important guidelines on how the most efficient data storage is accomplished. For example, normalization requires that non-key functional dependencies be removed from a table. For example, in the case of a single server queue, if each customer record contained arrival-time and departure-time, then total time-in-system would not be included in the record since it is simply a functionally dependant on the arrival and departure times.

A complete discussion of the theory of relational databases and ER-models is beyond the scope of this paper, but it is important to indicate that a large body of knowledge exists and should be studied if someone wants to apply these techniques to simulation model design. This discussion only points out the importance and relevance of these approaches.

5 ENTITY-RELATIONSHIP MODEL

The E-R model is an extension of the relational model and provides a straightforward way to graphically analyze and model data for a system. ER analysis automatically accomplishes most file normalization. The ER model indicates the required data files which are then implemented with a relational database.

The main feature of the model is the construction of E-R Diagrams which show entities and relationships between them. Figure 2 shows a typical entity-relationship diagram. The final product of this diagramming technique suggests the set of files which could be used to place all the data in a relational database.

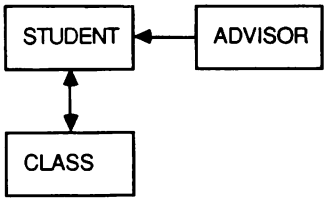

Figure 2. A typical entity-relationship diagram.

These ER diagrams contain important information about the system and the data. In Figure 2, the relationships between student, class, and advisor are shown. Since each class has several students and each student has several classes, this is a many-to-many relationship. There is also a one to many relationship between advisor and student. In addition, each entity has a number of attributes. In some versions of the ER diagrams, attributes are shown somehow in the diagram. Although the definition varies a little from author to author, an entity usually represents a person, place, thing, or event. Events include transactions, such as in a banking system, or an event like a machine breakdown in a machine shop. Each entity in the diagram will be represented directly as records in a file (or table) with records (or rows) for each entity and fields (or columns) for each attribute.

Relationships between entities are shown as arcs in this diagram. There are three basic types of relationship: the one-to-one, the one-to-many, and the many-to-many relationship. Each of the arcs is labeled and the type of relationship is indicated; in this diagram, the arrows are used. Some of the relationships are conditional to imply that the relationship may or may not exist. An example is a relationship between advisor and student. Although this should be a one-to-many relationship, some students may not have an assigned advisor.

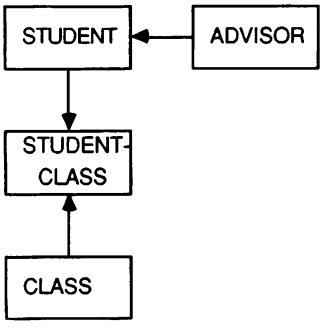

Figure 3. An ER-Diagram with Intersection Entity.

The one-to-one and one-to-many relationships are easily expressed by adding appropriate fields (attributes) to the records (entities), but the many-to-many relationship can cause problems. To deal with the one-to-many relationship, an "intersection entity" is usually inserted in the diagram. Figure 3 shows how the many-to-many relationship between class and student can be expressed with an intersection entity. The intersection-entity between the student and class entities will change the many-to-many relationship into two one-to-many relationships. The inserted entity eventually becomes a registration file with related student-id numbers to class-id numbers in each record. Information on the entire system is then appropriately stored with advisor information in an advisor file, student information in a student file, class information in a class file, and registration information, including grades and date enrolled in a registration file.

The ER-diagramming technique is a very good way to organize data into logical data files. Data collection for this system must recognize this structure and store data in appropriate files. Once the data storage design is accomplished, a relational database and query language can be used to retrieve and process data in numerous ways. Custom reports and ad-hoc queries should be easy to create. In the example of students and classes, when the data are stored as indicated, it is easy to produce class lists

showing who is enrolled in a particular class, or produce student grade reports showing the grades for each class that a particular student enrolled in.

6 EXAMPLES

To show how these techniques can be applied, two typical examples of simple queueing systems and ER-models for each are provided in Figures 4, and 5.

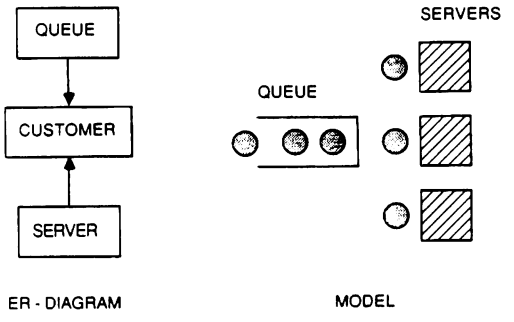

Figure 4. A Multiserver Queue and ER-Diagram.

Figure 4 shows a multiserver queue situation. Here there are a number of customers, several servers, and a single queue. In the related ER-diagram, the three basic entities are shown. Although there are two one-to-many relationships, there are no many-to-many relationships. Data collection for this system should recognize the three entities identified in the ER diagram. In this case, a small table could be used to store information about the several servers and most of the data would be collected in a customer file.

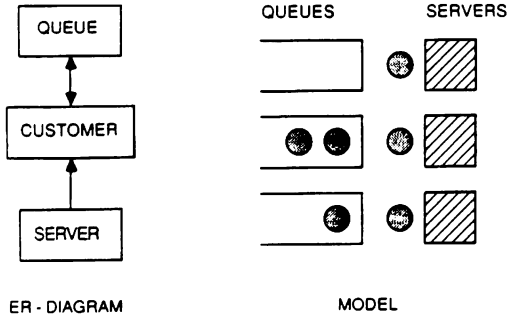

Figure 5. Multiple Single Server Queues with line jumping.

Figure 5 depicts a slightly more complex queueing situation where there are multiple single server queues and line changing can occur. This situation is similar to a grocery store. In this case, one customer can get in a queue, but later move to another queue. In this assumption, the queue to customer relationship becomes a many-to-many relationship and it is necessary to insert an intersection entity called "customer-queue". This customer-queue entity will become a file containing information about when a customer enters and leaves a particular queue. This is similar to the registration file which expresses the student-class relationship.

A most important point to remember in using this model is that the ER-diagramming techniques provide a method for designing a complete and efficient set of files in which to store simulation data. In addition, the files created by these methods can be used directly by standard relational database software to facilitate data exploration and analysis.

7 OTHER MODELING TECHNIQUES

A number of other graphical modeling techniques are used in the design of discrete event simulations. The state transition diagram and the flow chart are commonly found in simulation texts. While these techniques are useful in showing the process logic and flow of the event processing, they say little about data organization and collection. These diagrams are most helpful for programming event driven simulation simulations.

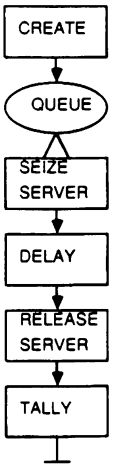

Figure 6. A SIMAN Model of a Multiserver Queue.

The modeling languages of SIMAN, SLAM, and GPSS are based primarily on a graphical representation of the physical system. Figure 6 show a simplified SIMAN model of a multiserver queue. To understand how the approach here contrasts with a pure ER view, it is important to define the important parts of the system using SIMAN and contrast this with the approach shown in Figure 4. In SIMAN terminology, the customer entering the system is an entity. Further, in the SIMAN model, the queue is a file where entities are stored and the servers are described as resources. Even though it is easy to provide the model with multiple servers, in this case it is much more difficult to distinguish the individual servers.

In many systems these differences are not important. However, in this one, let's assume we want to model three servers at the station. One server is a slow worker, and two are fast. Also, one of the fast servers is constantly interrupted by phone calls. If the servers are not treated as distinct entities with distinguishable attributes, this model is not as easy to construct.

There are other places where the SIMAN treatment of the entities causes trouble. Consider the grocery store checkout example where there are eight checkout stations and all eight of the stations open and close in response to traffic through the store. In the SIMAN language it is not easy to have the arriving customer select shortest open line by a direct technique. Again, because the queues or stations are not treated as entities, we can't easily and directly assign attributes such as the open/closed status. SIMAN does have the built-in functionality to keep data on queue-length and server-utilization but if you want to go beyond what has been anticipated, such as assign the queue status as open or closed, the limitations are constraining.

This is not a problem unique to the SIMAN language. Virtually all of the discrete simulation packages of this type have the same problems. The following example illustrates the weaknesses in more detail.

8 EXAMPLE - A JOB SHOP WITH BREAKDOWNS

This example is provided to compare a simulation from the point of view of the ER diagrams and SIMAN language. Figure 7 shows a simplified SIMAN block diagram for a job shop operation. In this case, there are multiple work stations which are similar. Each station has a queue and a machine which is used for a specified amount of time for each job. This example is typical of the type of application for which SIMAN is well suited. One notable feature of this model is the use of timing entities used to trigger the breakdown of machines; this technique is recommended and fairly commonly used in SIMAN (Pegden et. al. 1990).

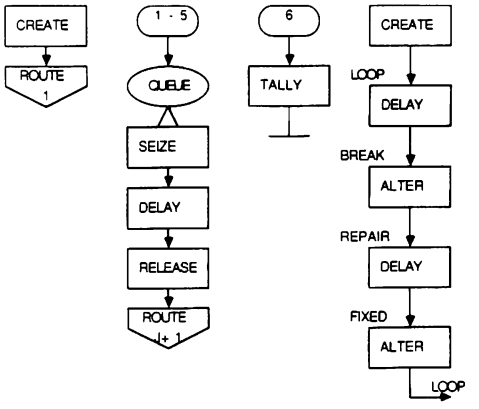

Figure 7. A SIMAN Job Shop Model with Breakdowns

The timing entity is used to trigger the random machine breakdown and repair of a machine. It is however, important to note that the timing entity is used only to trigger the event but is not an entity which represents the event itself. The breakdown event is treated separately in the ER diagram and has a one-to-many relationship with machines or stations. This example suggests that data collected on machine breakdowns should be collected in a separate file.

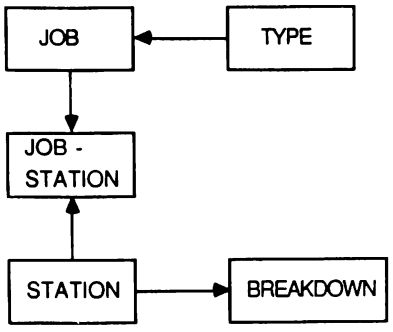

Figure 8. An ER-Diagram for the Job Shop Example.

A corresponding ER diagram is shown in Figure 8. This example has a classic many-to-many relationship between stations and jobs which is interesting from the ER point of view. This relationship requires an intersection entity (and file) which contains information about the job-station interactions. Assuming that each job is assigned an id-number, and each station has a station-number, the intersection file could simply contain the information about job-id-number, station-number, enter-queue-time, start-of-processing, and the time that processing-ended. Also, we assume that there are several distinct job types and thus the time at each step is functionally determined by the job type. Because this is true, normalization requires that information about the characteristics of each job type be stored in a separate table or file.

9 RESULTS AND CONCLUSIONS

When simulation data are stored for analysis, the ER model is applicable. The main outcome of this approach is the specification of a data model which efficiently stores data from the simulation in a set of files and tables which are compatible with existing relational database systems. However, using the ER model not only helps describe an efficient way to store data from a simulation but it also provides insight into the interrelationships between players in the simulation.

The ER theory and the related relational database theory have been developed over several years and have direct application for systems modeling and simulation. One of the most obvious conclusions is that data can be efficiently collected in a set of related files. Allowing a simulation to build flat files for relational database has value. Certain statistical and graphical features would obviously be facilitated by this approach. Data that is collected in this fashion would be easily exported into statistical analysis software environments like SAS, which includes a full featured relational database capability. The theory of relational databases explains why data stored in this manner is efficient and can be used to answer to any query.

Most simulation languages were not specifically designed to collect data in this fashion. Although many of the interesting statistics can be calculated without resorting to this approach, more detailed analysis of simulation data may require storage of large amounts of data. Simulation languages such as SIMAN provide for data collection and statistical analysis in the simulation model but the data collection features are not ideally designed to store data as an ER model might indicate. Analysis through the perspective of ER modeling suggests that information about the attributes of an entity should be collected as whole records of information. Fortunately, the latest release of SIMAN does provide a way to write whole records into files during the simulation. ER theory provides a motivation for the use of this feature.

Obviously, the ER modeling approach has direct relevance to the design of simulation languages. The theory not only has direct application to data analysis but it also provides insight into the important definition and treatment of entities in the system.

Contrasting the ER model approach to the graphical models used in SIMAN points out some important differences. The ER analysis shows that some of the important players in simulation models written with SIMAN, such as queues and resources, might better be treated as entities. Specifically, if treated as entities, these elements of the model could be assigned attributes including an ID-number and be more readily distinguished in the model and the data analysis.

Considerations of file space may moderate direct application of this approach. The most obvious drawback to the idea of collecting all of the data from a simulation into several files is the concern about memory, storage size, and cost, plus the extra time required for writing records to secondary storage such as disk. However, the practical considerations of storage requirements should not inhibit consideration of the ER modeling and diagram techniques. The insights gained by contrasting the

usual block simulation models and the ER diagrams are worth consideration even if the data is ultimately collected in a much more limited and traditional manner. The dramatic improvements in cost and performance of storage over the past few years should at least indicate that this will all be practical in the near future.

REFERENCES

Pritsker, A.A.B. 1984. *Introduction to simulation and SLAM II*, Second Edition. New York: Halstead Press.

Litton, Gerry M. 1987. *Introduction to Database Management: A Practical Approach* Dubuque, IA: Wm. C. Brown Company.

Shannon, R.E., 1975. *Systems Simulation - the art and science* Englewood Cliffs, NJ: Prentice Hall, Inc.

Hoover, S.V. and Perry, R.F., *Simulation - a problem-solving approach*, Reading, MA: Addison-Wesley Publishing Company.

Payne, James A. 1982. *Introduction to Simulation: Programming Techniques and Methods of Analysis*, New York, NY: McGraw-Hill Company.

Pritsker, A.A.B., 1986. *Introduction to Simulation and SLAM II*, New York, NY: Halstead Press.

Pegden, C.D., Shannon, R.E., Sadowski, R.P., 1990. *Introduction to Simulation using SIMAN*, New York, NY: McGraw-Hill Company.

Pegden, C.D., 1987. *Introduction to SIMAN* State College, PA: Systems Modeling Corporation.

SAS Institute Inc., 1985. *SAS User's Guide: Basics, Version 5 Edition* Cary, NC: SAS Institute Inc.

Schriber, Thomas J., 1991. *An Introduction to Simulation using GPSS/H* New York, NY: John Wiley and Sons.

AUTHOR BIOGRAPHY

ROBERT S. ROBERTS is an assistant professor of Computer Information Systems at New Mexico State University. He received his Ph.D. from Oregon State University in 1980. Current research interests include data communications, interconnection network design, computer simulations, and curriculum for information systems.