Developments in Business Simulation & Experiential Exercises, Volume 15, 1988

EVENT-EXTENDED ENTITY-RELATIONSHIP DIAGRAMS FOR UNDERSTANDING SIMULATION MODEL STRUCTURE AND FUNCTION

F. Paul Fuhs, Virginia Commonwealth University

ABSTRACT

Because of the similarities in the underlying modeling constructs of database design and simulation modeling proven effective database design methods, if enhanced, can be adapted to simulation modeling. This paper presents the Event-Extended Entity-relationship Diagram that enables simulation model designers and users to better communicate among themselves concerning the entities, attributes, relationships, and events that exist within simulation models.

This modeling method is an extension of the highly successful database modeling tool called the Entity-relationship (E-R) diagram, developed by Chen (1976) and clearly discussed by Grant (1987). The E-R diagram expresses the logical associations among data.

This paper demonstrates the need for such a communication method for designers and users, It briefly introduces some of the concepts in E-R modeling and shows the similarity of model constricts between simulation models and database models as the basis for beginning with the E-R diagram, It then sets down the semantic constructs required to enhance E-R diagrams to fore Event-Extended Entity-relationship diagrams. The paper illustrates The new constructs with an example and with discussion of its communication potential. The paper points cit the relationship between the Event-Extended Entity- relationship diagram and the design of either file structures or database structures needed to support a simulation model, Lastly, it sketches oat the design steps leading to the development of simulation modeling computer code.

INTRODUCTION

In developing simulation models the designers must first identify the entities, attributes, events, triggers, and functional relationships in a sub-set of the real world. Secondly, they must be able to abstract those components and relationships that pertain to the objectives of the model. Thirdly, they must express the model to a symbology that is understandable by other designers and the users of the model. Fourthly, the designers must implement the design on a computer. This implementation includes the design of file structures for the simulation data and the computer modules to run the model. The design of the file structures is especially important when the input or output data are database data and not merely a loose physical association of variables.

Designers must be able to communicate among themselves and to users the components of their models and the relationships among these components. These components and relationships are called the semantics of the model.

Communication problems exist among designers. Fritzsche (1985) points out the fact that "...a great deal of effort is required to modify a program developed on one system to run on another system.' Teach (1986) adds that "Frequently, the ability to alter parameters is not reported in the documentation and one must know how to read the programming code in order to sake changes."

The communication problems are worse between designers and users. The dc-briefing process too often demonstrates that users missed critical semantics within the model not only during the introductory phase before exploration with the model, but also after repeated exposure to the model. This situation becomes more acute as the complexity of the inter-relationships in the model are increased and as more emphasis is placed on sensitivity analysis among the attributes of the model.

The causes are too frequently incorrectly attributed either to the inexperience of the users or in a spirit of arrogance to their diminished mentality. Most users of a model are not proficient in these languages, nor do they have the time nor inclination to get involved with the computer code. The underlying cause is that designers and users do not have a common means of communication to express the logical associations among the entities, attributes, and events of a model.

Since diagrams are one of the best communication tools, it seems appropriate that a diagrammatic means of clearly elucidating the concepts of entities, attributes, events, and relationships is needed.

DATABASE DESIGN AND E-R DIAGRAMS

Peter P. S. Chen (1976) is credited with the invention of the Entity-relationship Diagram. It is widely used in database design to express the logical relationships among data and their semantics. This diagram has recently been modified by Martin (1981) in an attempt to standardize the symbols used across many different diagrammatic tools and to provide the basis for automated database design. Figure 1 defines a few of the entity-relationship symbols proposed by Mart in to express the semantics between entities.

While it goes beyond this paper to explain in detail all of the nuances, a few examples may help the reader understand the use of the symbols and later the relationship between such symbols and their use in simulation modeling. The symbols and text within Figure I show different ways in which two entities can be logically associated. Each entity is represented by a rectangle and the existence of a relationship is made by a line joining them. One can superimpose upon the line additional semantics pertaining to the cardinality of the association. For example, in the third relationship of Figure 1 the crow's feet in front of entity B tell us that each entity A is associated with one or more B entities. Therefore, the relationship between two entities need not be symmetrical.

Entities that are related within the same entity type are modeled is looped relationships, shown in Figure 2.

Figure 1
E-R Diagram Constructs Proposed by James Martin
(basic associations)

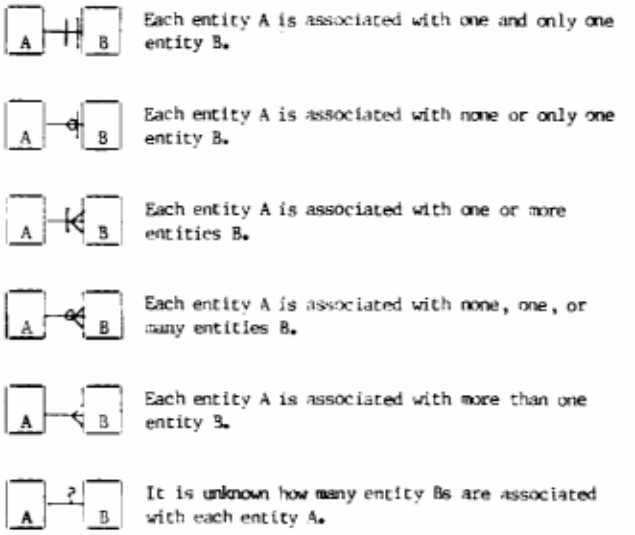
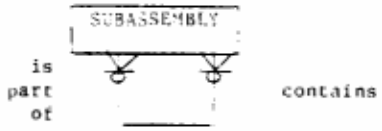
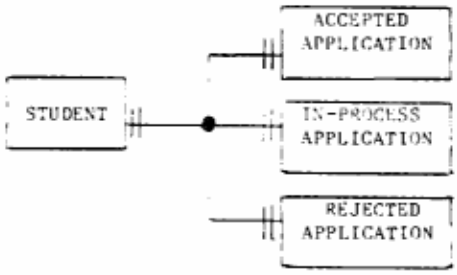


FIGURE 2
Other E-R Constructs

Looped Relationships



Mutually Exclusive Relationships

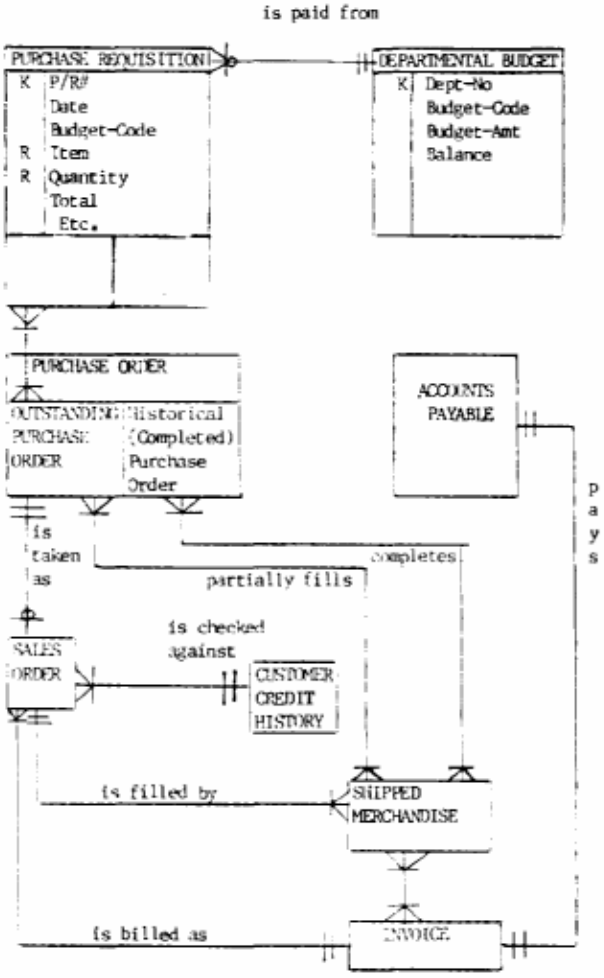


For example, a sub-assembly may be a part of none, one, or more higher level sub-assemblies. The same sub-assembly in turn may be composed of none, one, or more sub-assemblies. Figure 2 shows the situation in which the semantics captured tells us that a student entity can be related to any one of three entities; an accepted application, an in-process application, or a rejected application. The solid dot at the intersection of the lines expresses a mutually exclusive relationship.

Figure 3 shows an example of part of a simulation model expressed as an E-R diagram.

This Figure shows the entities of two interdependent companies. It also shows the attributes within two of these entities, i.e., within the entities called purchase requisition and departmental budget. In the model the company at the too company. The state of the accounts payable entity the first company is dependent upon the other entities in the model. The Figure includes a hierarchical construct called an entity sub-type. The Purchase Requisition entity type has two entity subtypes; "Approved Purchase Requisition" and "Rejected Purchase Requisition". In the Figure each box with the larger lettering represents an entity type. Within each entity type there are a number of entity occurrences. For example, the entity type, "Sales Order" contains a number of individual sales orders.

FIGURE 3
AN ENTITY-RELATIONSHIP DIAGRAM



The attributes of an entity are listed in an Entity- relationship Diagram below their respective entities. For example, "P/R#" and "Date are attributes of a 'Purchase Requisition". The column in front of the attributes captures additional semantics. For example. "K" represents primary key, while "R" represents a repeating data group. purchase requisition may have note than one Item quantity pair.

To help the understanding and communication of the semantics between the entities, a relationship is also labeled. this labeling is done correctly, the relationship can be read as a sentence. For example, the entity "Outstanding Purchase Order" of the first company is taken as a "Sale Order" entity by the second company. Shipped Merchandise completes a purchase order or partially fills an outstanding purchase order.

The diagrammatic simplicity and communication clarity of the symbolic conventions in E- R diagrams should not be underrated. There is a rich amount of semantics captured even in this simple example. single sales order may result in one or more shipments, but the semantics expressed also tell that in the relationship between the two companies the second company anticipates that each shipment can be related back to one and only one sales order. This E-R diagram communicates clearly the fact that there is some management constraint which specifies that combining shipments from different sales orders is prohibited.

SIMULATION MODELING AND DATABASE MODELING

Simulation modeling and database modeling share all of the same fundamental semantic constricts, although this sharing is not immediately evident, it is easy to see the common use of entities and attributes in simulation modeling and database modeling. Simulation languages such as Simscript and E-R diagrams both employ entities and attributes for expressing the nature of the data.

Simulation languages use the "events" and "trigger" constructs in modeling. Processing is organized into modules called events that are activated along a time continuum by changes in rue state of entities or attributes within the events. An event can create or destroy entities; can increase, decrease, or nullify attribute values depending upon the state(s) of other entities and attributes.

One apparent difference between simulation models and database models is that the “event’ construct and the "trigger” construct that sequences events are explicit in many simulation model, yet are not explicit in E-R diagrams. The reason is that E-R diagrams are meant to model primarily the logical association or structure of data, not the processing made upon it. Other diagrammatic techniques such as Data Flow Diagrams popularized by Game as Sarson (1977) are used in database development to express the processing aspects of database modeling.

Events, however, can be easily modeled in E-R diagrams as relationships between entities. An event is modeled simply as a process relationship between entities. E-R diagrams can express triggers as constraints placed upon the relationships among entities and/or attributes. For example, a trigger may be associated with the creation of an approved chase requisition (an entity). The completion of a batch of 20 purchase requisitions may be the trigger to initiate an event that causes a purchase requisition to be approved or rejected. This is not shown in Figure 4.

E-R diagrams and simulation models handle relationships among events and entities in an inverse manner. Simulation models embed the entities within the events. C-R diagrams embed the events between one or more entities.

Al though both types of models can handle events and triggers, the lion's share of the semantics is captured by simulation models, although most of the semantics become buried in unintelligible computer code. Traditional E-R diagrams model events and triggers only in the most generic and rudimentary fashion • We therefore turn now to Event-extended E-R diagrams as a means of diagrammatically expressing more of The event semantics or simulation models.

THE EVENT-EXTENDED E-R DIAGRAM

Since events are closely associated with changes in attribute values, we must first enhance E-R diagrams so that they go beyond the expression of relationships between entities to the relationships between attributes. That is, oust be able to clearly show which attributes are affected when an event occurs. Secondly, we must be able to communicate through the diagram the changes in attribute values when an event is triggered. In developing the modeling structure of the Event-Extended Entity- relationship Diagram we must ensure that the diagram communicates the following.

- The differentiation between entities and events
- An identification of the inputs and outputs of an event
- An identifications of the direction of flow inputs to outputs)
- The differentiation between changes in entity states (creation and destruction) and changes in attribute values caused by the events.
- An expression of the processing performed within the events

We introduce the following symbolic constructs to convey the above semantics. First, a rounded rectangle denotes in event, while a regular rectangle represents an entity or an entity with its attributes. This symbology is consistent with that used by Gane and Sarson (1977) in Data Flow Diagrams. Secondly, attributes that are involved as inputs or outputs are denoted with traditional arrows; the lines showing the attributes involved dud the arrow head; showing flow direction. If one or tore attributes change the attribute values of other entities when an event occurs, this is symbolized by having arrows extend from inside the input entity boxes outward toward the event box. Then one or fore r rows tend from the event box into the entity boxes of the changed attributes. To better differentiate between input and output attributes arrows formed with dashed lines designate about attributes.

If an event changes an entity state (creation or destruction), the arrow extends to that entity box is a whole. If an event transforms an entity into entity sub-types the placement of the arrow lines should clearly denote this by pointing to the subtypes.

The processing that - an event performs is specified within the event box. La its simplest form this expression is a function of the form $y = f(x,y,z)$. More complicated expressions would be sets of simultaneous equations. Still more complicated expressions would be algorithmic expressions. If the event box is not large enough to contain all of the semantics of the event, a high order expression of the semantics can be made with further decomposition on supplementary diagrams. This is consistent with such diagrams as structure charts as treated by Martin (1937) and allows the designer to develop a model in a top-down fashion.

Triggers in the form of large arrows can be appended adjacent the data flows in the Event-Extended Entity-relationship Diagram. The shape of this symbol follows the recommendation of Martin in (1987).

An Example of the Event-Extended E-R Diagram

Figure 4 illustrates an Event-Extended E-R diagram. It shows the superimposition of two events upon the E-R diagram of Figure 3 and the constructs needed to support the semantics of the events. When a batch of 20 purchase requisitions has been collected, an authorization check is made that transforms the entity "purchase requisition" into the sub-types "approved" and "rejected". The event called "authorization check" has two inputs for each purchase requisition. The first is the attribute "Total" from the entity "purchase requisition". The second is the attribute called "Balance" (the amount remaining in the budget) that is associated with a given department. The event compares the attribute 'Balance" with the total dollar value requested on a purchase requisition. Based upon whether there is enough money left in the departmental budget, the purchase requisition becomes either approved or rejected.

THE EVENT-EXTENDED E-R DIAGRAM AS A COMMUNICATION TOOL

The Event-Extended E-R diagram aids both the simulation model designer and the model users by providing a common set o easily understood symbols by which the semantics of the model can be viewed and discussed. For example, answers to the following set of typical generic questions are easily provided by this diagram.

What are the entities in this simulation model?
Do any of the entities have entity sub-types?
What are the attributes for a given entity?
What are the events?
What causes specific events to occur?
What entities are affected by each event?
How does the flow of data and changes in that data move throughout the entire model?

Answers to the following typical specific questions are easily provided by this diagram.

How is the tax calculated on the invoice? What entity or entities are involved in the tax calculation?
Within the sales order entity which attributes affect the tax calculation?

The structural constructs coming from the E-R part of the Figure 4 allow answers to the following questions

Is there only one invoice associated with a given sales order?
Can a purchase requisition be traced back to more than one department in this simulation model, that is, can more than one department pool their budgetary resourced together to make a single purchase? [Answer: no.]

Physical File Structures and Program Code

In cases where the attribute values of a simulation model are riot required to be physically associated on secondary storage, there is no need to implement physical file structures. In other cases the model designer has within the Event-Extended E-R diagram the basis for developing the physical files or databases to support a simulation model. The entities along with their attributes are first translated into a set of unnormalized relations, then normalized as per Kroenke (1988). If formal databases are to be implemented, after an appropriate Database Management System (DBMS) has been chosen, the physical databases ire constructed as discussed by Haweryszkiewycz (1984).

The program code of a simulation model can be developed from the Event-Extended E-R diagram in a top-down manner by decomposition of the event boxes. Each event box circumscribes one or more computer modules that need to be developed. By inspecting the flow arrows of the diagram in conjunction with the relations or file structures formed later in the simulation model's design the designer can create navigation diagrams and action diagrams with pseudocode as discussed in Martin (1937). The pseudocode can then be decomposed into some computer language.

SUMMARY

The semantics of simulation models need no longer be hidden within a simulation model's computer code. When trying to communicate the meaning of a model among designers or users the Event-Extended Entity- relationship Diagram can be used to clearly show the relationship between entities, attributes, events, and triggers. The diagram also unifies the two processes needed to develop a simulation model, namely the development of the file structures to support the model and the development of the program code.

REFERENCES

Fritzsche, David J. and Richard V. Cotter, " Guide to Writing Microcomputer Simulations," in Developments in Business Simulation & Experiential Exercises, James W. Gentry and A. C. Burns, eds., 1985, p.[69].

Teach, Richard D., "Building Microcomputer Business Simulations," in Development in Business Simulation & Experiential Exercises, A. C. Burns and Lane Kelley, eds., 1986, p. 239.

Grant, John, Logical Introduction, to Databases, New York: Harcourt Brace Jovanovich, Inc., 1987, pp. 174-192.

Haweryszkiewycz, I. T., Database Analysis and Design, Chicago: SRA, 1984.

Chen, P. P. s., "The Entity-Relationship Model --Towards a Unified View of Data," in ACM Trans. on Database Systems, 1976, 1(1), 9-36.

Martin, James, Recommended Diagramming Standards for Analysts and Programmers, New Jersey: Prentice-Hall, 1987.

Gane, Chris and Trish Sarson, Structured System Analysis and Design, New Jersey: Prentice-Hall, 1977.

Kroenke, David, Database Processing, Chicago: SRA, 1988, Chapter 5.

FIGURE 4
AN EVENT-EXTENDED ENTITY RELATIONSHIP DIAGRAM