Journal of Information Technology and Engineering
Vol. 2 No. 2 (December, 2017)

Guidelines and Notations towards a Simulation Temporal Data Base Management System

Abdelhak BOUBETRA

Computer Science Department, University of Bordj Bou ArreridjBordj Bou Arreridj, 34000, Algeria
E-mail: boubetraabd@yahoo.fr

Received: 23rd July 2017 Revised: 14th September 2017 Accepted: 15th November 2017

Abstract: *The approach to make connection between simulation and temporal data bases helps in dividing the simulation activities into independent stages. This connection deals with the temporal aspect in simulation and data bases. Guidelines and notations for a design approach of a data base management system for simulation based on temporal aspects are presented.*

Keywords: *simulation, data base, time, DBMS, temporal map*

1. INTRODUCTION

Beside its popularity and advantages, simulation still has limitations since simulation users are always seeking for easy simulation software with more features, and there is an urgent need for user-friendly software for all stages of simulation including the definition, development, and validation of simulation models. In order to reduce the complexity to implement a desired simulation software, several authors [5,6,7] recognize the need for a model development and analysis environment in which tools can be used to support modelling and analysis, thus separating data management from the simulation language itself, and this environment should be integrated. The recommendations argue that a data base management system (DBMS) provides a very effective means of implementing this integration. This may be the reason that one of the main activities in simulation [2,3], today, is the management of data generated by large scale simulation studies. We can think of a logical view of the main activities for building, analysing and utilising a simulation model with the aid of a data base management system as it is shown in Figure 1.

First, the analyst must study the real world system under investigation to specify a conceptual model (its static and dynamic structure), which he believes is the map of his real world system, within some language. The resulting description can be loaded into the data base. The simulation system can run the simulation, retrieve parameters from the data base and store outputs in the data base to obtain answers to queries or information for simulation purposes.

Within this logical view, the simulation user will have an integrated environment which has the ability to maintain model descriptions and store results of simulation runs. However, in most of the existing DBMS the running facility of the simulation does not exist. In other words, when most of these DBMS were first designed they did not include a simulation system. Parallel to that, in the design of the so-called simulation languages or simulation 'packages' simulation designers did not include the possibility of having a data base.

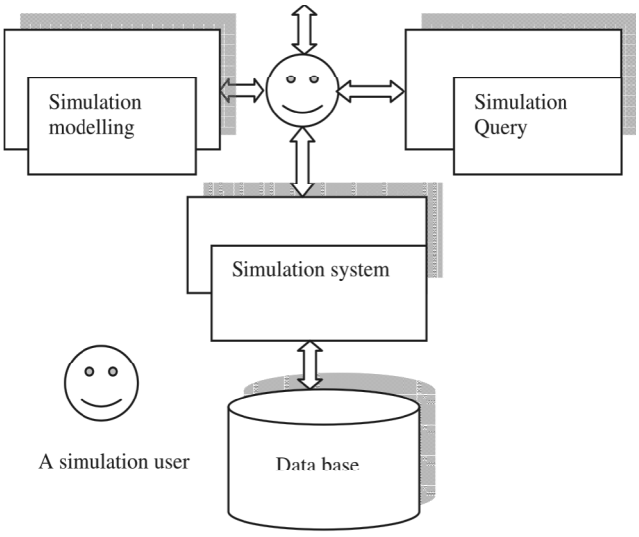

Figure 1: A Simulation System Using a Data Base

2. TIME, SIMULATION AND DATA BASES

The problem with most data base modelling techniques is that they concentrate almost exclusively on static views of the system being modelled. What is modelled is a time slice of a given system's existence or a time slice of the real world, the system dynamics are only modelled implicitly and obscurely in the data base transactions rules. However,

simulation modelling techniques have concentrated almost on dynamic views of the system being modelled. What is modelled is the system behaviour or the dynamism of the real world.

Here the aim of our suggestion can be seen in the need to have something in the DBMS that represents the entity behaviour just as a collection of events drives an entity through its behaviour in the real world. On the other hand, in specifying the dynamic behaviour of a system, time forms the core of any simulation study. Thus, temporal data bases that "capture what is known about events and their effects occurring over time" can offer a good support to meet the requirement of a DBMS for the simulation.

Studies in data base applications [8] have revealed the need to have new functions for new requirements. Among these additional requirements is temporal (or time domain) support, so as to allow the data base to model the time dimension. For example, in the storage of legislation, historical information is just as important as current data. As a result of the discovery that time domain support is needed in many applications, research work has been started. In some cases, the temporal domain is handled in a very restricted manner [4] and other studies [1] have concentrated on the user's perspective. Here the researchers [9, 10] attempted to see what kind of logic, interface, and language or query facility would be needed to support the different usages involving the time domain. Little serious effort has been expended to implement a system with general function for time support as well as the normal data base functions.

In a simulation, it is impossible to answer question such as 'what is the state of a system at a specified time' without access to history data. Thus, it is of immense value to have access to the notion of time in simulation through a data base or exactly to have access to the simulation data evolution through the time dimension. The development of tools and techniques required for managing data generated by large and complex simulated problems gives the simulation a new dimension. To fulfil this requirement, we are suggesting a specification of a tool to deal with that matter. We shall refer to such a tool as a simulation temporal data base management system (Sim-TDBMS). Beyond the conceptual complexity involved, attempts to implement such system can support the simulation with a powerful tool for building models and for following and keeping the step by step evolution of the simulated system.

3. METHODOLOGY

Dean T. L. and Mc Dermott D. [12] proposed a map for a real world system with the name of "a temporal map" which he defines it as a graph in which the nodes are snapshots of time associated with events, and the arcs connecting these nodes describe relations pairs of snapshots. The relations between the pairs of snapshots are the necessary conditions leading to the occurrence of events at these snapshots. Reasoning about simulation and time in this framework, we can imagine a simulation temporal map as a means of viewing time in simulation and provides us with a convenient reference to design a simulation temporal data base that captures the outcomes of the snapshots in the simulation temporal map and their effects over time. The information in the data base specifies the different situations and the relationships between the different entities of the simulated system at different points of time. In planning and decision making, it might be interesting to verify if certain conditions ('C' events: Conditional events) or certain facts of certain situations ('B' events: Bound events) had occurred or have been satisfied.

From above we can say that a simulation temporal data base management system (Fig. 2) supporting the functionality of the simulation temporal map might be seen as a powerful tool for the simulation user and for simulation in general. Such system might consist of a simulation temporal data definition system (Sim-TDDS), a simulation temporal data manipulation system (Sim-TDMS), a simulation temporal data query system (Sim-TDQS) and a simulation temporal documentation system (Sim-TDS).

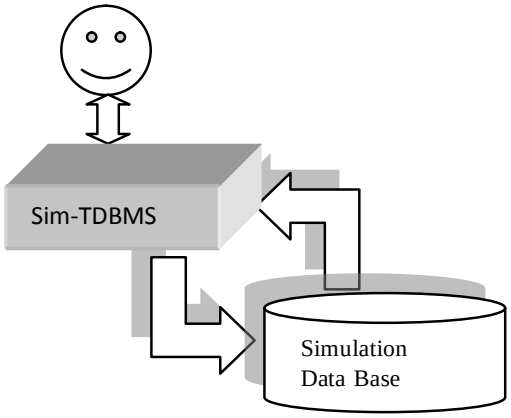

Figure 2: The Sim-TDBMS tool.

4. THE NEEDED SIMULATION TEMPORAL DATA BASE

To store and record the behavior generated by the simulation reflecting the state changes at the different snapshots of the time evolution, we propose a temporal data structure in the form of a relation that can be viewed as a 3-dimensionnal structure where the first two dimensions represent the objects and their attributes and the third dimension represents the time aspect. With the cubic representation applied to the simulation; each horizontal slice will indicate the changes at a specific snapshot of a simulation object. The upper horizontal slice represents in general the most current view of the simulated system object.

For example, let's consider a system consisting of a set of classes of objects. Each class is composed of objects having the same characteristics called instances in an object oriented approach. Among these classes we have a class consisting of three objects having for example three attributes notated:

ClassObject( ): Attributes(atr1,atr2,atr3) where an attribute can be also an object.

A slice of the state of one of the three objects at time t_i is notated as follows:

Slice :ClassObject():Object( )(time t_i)

So the slice of the state the three objects can be at time t_i like it is shown in Figure 3:

Slice :ClassObject(1):Object(1)(time t_i) =

Object 1	Ψ	χ	Д

Slice :ClassObject(1):Object(2)(time t_i) =

Object 2	Ф	ж	Қ

Slice :ClassObject(1):Object(3)(time t_i)=

Object 3	Ж	Ѳ	ьнь

Figure 3: A Slice of the Objects' Attributes at Time t_i.

In simulation, when time advances entities change their states as a result of the appearance of a particular event or a set of events. However, the cubic representation reveals some difficulties. When some attributes of an object remain unchanged but others change their states at a snapshot, the object can be assumed to have logically taken the last state at the most current snapshot which is not true like it is shown in Figure 4.

Slice: ClassObject(1):Object(1)(time t_{i+1}) =

Object 1	Ψ	Ч	Д

Slice: ClassObject(1):Object(2)(time t_{i+1}) =

Object 2	Ф	ж	ф

Slice: ClassObject(1):Object(3)(time t_{i+1})=

Object 3	Ж	ю	нь

Figure 4: A Slice of the Objects' Attributes at Time t_{i+1}.

To overcome this problem, we extend the concept to include the history behavior of the simulated system. Instead of building one table for each class of entities with additional time attributes to contain all the tuples and their changes since their creation, we shall have a table that contains only the current tuples as is done in the current view systems. However, to capture the time aspect we will have its value stored with the tuples of the current view. This time will not behave as in the case where the temporal domain is supported by adding the time attributes to the tables. Included with each tuple will be a pointer that points to the first history tuple when there is.

The method for creating and storing history simulation data works as follows: all history information belonging to one tuple representing an entity is chained in time order. The beginning of the chain is the current view of the entity and with this structure we can process all the current and history view of the entities. Now we go into discussion of our approach and focus on the performance and the design of the historical view. We organize another table for sequential accessing of the time aspect which will be referred to as the simulation historical table and it consists of two fields. The first one represents the time and the second one is a pointer to the attributes values of an entity. The historical pointer of the entities class points to the simulation historical table which acts as a pointer to the attributes values of the entities with the time as a key access in the first field. The second field of the simulation historical table is a pointer to the attributes which have undergone a change.

Within such an implementation framework, the simulation can gain in all its stages starting from initialization until the investigation of the simulation outcomes. The initialization phase generate the slices of the objects' attributes with their corresponding initial states. The simulation run generates all the next slices corresponding to the different changes of the objects' attributes.

5. THE SIM-TDBMS COMPONENTS

So far we introduced within the simulation temporal map the notions of snapshots and the outcome of a snapshot. To deal with these notions the Sim-TDBMS should use a data base to capture the outcomes of each of the snapshots and the different snapshot-to-snapshot paths relating them. Adding to that the Sim-TDBMS should provide means to walk within the simulation temporal map in order to access and interrogate the simulation temporal data base.

On the other hand from a software design point of view, Ian Sommerville [11] stated that a more methodical approach to software design is offered by so-called 'structured methods', which are basically sets of notations and guidelines about how to create a software design. In that direction, the Sim-TDBMS components design might be expressed by the following forms.

5.1 The Sim-TDDS Component

There are a number of modelling methodology and issues in simulation. These issues consist in chosing the level of detail of the system representation, and what is to be represented of the real world under study. It is obvious that defining a system is defining its static and dynamic structures and in some cases is defining some pictorial structures representing some formalism of the system under study. For such a purpose, we introduce in the next sections the modelling formalism in the Sim-TDDS to define the static, dynamic and pictorial structures of the system under investigation.

Static descriptions in the Sim-TDDS: A static structure of a real world system is for the major part the set of entities

with their attributes. These attributes are supposed to represent an abstract image of the real system and its parts. As a consequence, their choice is highly influenced by the modeller of the system. Based on these considerations, it can be seen that there are two fundamental concepts involved in specifying the static structure of a system. These are the concepts of entities (objects) and of the entities classes (objects classes) where an entity has a name and a set of attributes. The name is a distinct identifier not possessed by any other entity. The set of attributes is the set of images and values that the entity can take. The identification of entities in the Sim-TDBMS is specified by records with fields representing the entity inter-relationships in the simulation temporal data base.

A decomposition of the system into classes represents a simple decomposition of a model into components where each component or a class is a set of entities having the same attributes and the same functions in the system. Thus, entities can be categorised as belonging to various classes and that by knowing the class of an entity we can determine some of its properties. Each such class in the Sim-TDDS is reprented by a set of records.

Dynamic descriptions in the Sim-TDDS: In addition to structural static descriptions in terms of entities and entities classes, the simulation user may additionally define the dynamic structure of the system components and how the entities of the classes are connected together. Connection may be described by the set of entities taking part in an activity. Thus, The Sim-TDDS provides means for the behavioural description of a system which is the definition of the type of entities starting or ending an activity. In the three phase simulation approach, we distinguish two types of events, the B and C events which form the system's behaviour. B events are those occurrences which correspond to ending an activity. C events are those occurrences indicating the starting of an activity. Wether an event is B or C, the type of entities taking part in it is known to the modeller before simulation. However, the exact identification of the entities is unknown. In the Sim-TDDS, the most elementary method for defining B and C events is to indicate simply class type and the number of entities from each class to take part in an event. The data structure of the records to support this description should be defined.

Pictorial descriptions in the Sim-TDDS: In addition to structural static and dynamic descriptions, a simulation user can define pictures to his system. The Sim-TDDS provides a set of images among which the user can choose to define his system for the purpose of learning the simulated behaviour generated in a pictorial form. These pictures will form a communication language between the simulation user and the Sim-TDBMS.

System functioning of the Sim-TDDS: To accomplish its functions, the Sim-TDDS interface is one of screen menus and keyboard functions providing the simulation user with facilities to describe the static and dynamic structure of his system. The Sim-TDDS takes this information in the form of question/answer from the simulation user ans stores it in the simulation temporal data base.

5.2 The Sim-TDMS Component

The Sim-TDMS is the set of software components responsible for running the simulation, accessing the simulation temporal data base to extract information about the system under study, storing the generated behaviour and forming the interface between the simulation user program and the simulation temporal data base.

The main actions that occurs when the simulation is running by means of the Sim-TDMS are as shown in Figure 5. and they consist of :

(1) the simulation user program issues a call to the Sim-TDMS to read a current view of an entity in the simulation temporal data base by stating the name of the entity type.
(2) The Sim-TDMS gives the value of the key of the required entity based on a hash coding function.
(3) The Sim-TDMS examines the physical simulation temporal data base descriptions and determines what physical record or records to read and looks up the entity in question.
(4) The Sim-TDMS issues a command to the computer operating system, instructing it to interact with the storage media containing the simulation temporal data base and read the requisite record.
(5) The Sim-TDMS transfers the data from the simulation temporal data base to the work area of the simulation program.

If the simulation program updates a record, the sequene of actions is similar. The simulation program will normally read and issue an instruction to the Sim-TDMS to write back the modified data.

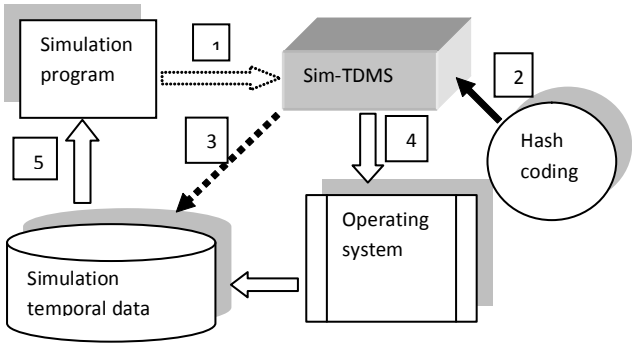

Figure 5: Main Actions of the Sim-TDMS Functioning.

System functioning of the Sim-TDMS: As mentioned above, the Sim-TDMS reflects the functioning of the simulation temporal map based on the three phase simulation approach. Thus, the Sim-TDMS is responsible for advancing time from snapshot to snapshot in the simulation temporal map and provide the three phase simulation executive with

the necessary informations to run the simulation model, such as some of the model static and dynamic descriptions strored in the simulation temporal data base with the help of the Sim-TDDS.

In addition to that, the Sim-TDMS is responsible for storing in the simulation temporal data base the outcomes of each snapshot or the results of the execution of the B and C events representing the simulation model. From that, we understand that the three phase simulation executive will form the heart of the Sim-TDMS which has the ability to make transactions on the simulation temporal data base to extract and store informations.

The mentioned transactions are mainly the events stated in the previous section and we will discuss them in the next section in terms of the three phase (A,B,C) simulation approach. So the the three phase simulation executive will be embedded in the Sim-TDMS.

The Sim-TDMS executive: within the Sim-TDMS, the formalism to implement the three phase approach is the same as described in the different literatures, but with more functions which consists of the management of the differing transactions on the simulation temporal data base to extract and store information. This implementation will be as follows:

(a) The Sim-TDMS 'A Phase': a time scan which determined the next time at which one or more B events are scheduled to be executed and forms a list of all such actions that must take place at that time.

(b) The Sim-TDMS 'B Phase': which consists of the actual execution of the bound events found necessary in the previous phase. Within the execution of the bound events, the Sim-TDMS makes a transaction on the simulation temporal data base. This transaction is the creation of the new historical view of each entity corresponding to the new image of its attributes. Within the execution of each of the B events, the Sim-TDBMS changes the linkages belonging to the entities taking part in these events. Thus, creating the current view of the entities having undergone a change at this snapshot.

(c) The Sim-TDMS 'C Phase': Which consists of testing and, where the tests succeed the execution of the action part, of each conditional event. In the Sim-TDMS, some of the testing consists of making transactions on the simulation temporal data base to read the current view of some of the entities and consult their states. Where the tests succeed, the execution of the actions part consists of making a transaction on the simulation temporal data base to create the new historical view of the entities having go through a change and modifying the lnikage belonging to these entities within the simulation temporal data base.

The executive program cycles round these three phases until a preset duration or a predetermined termination condition is reached.

5.3 The Sim-TDQS Component

So far we introduced within the simulation temporal map the notions of snapshots and the outcome of a snapshot. To deal with these notions the Sim-TDBMS should use a data base to capture the outcomes of each of the snapshots and the different snapshot-to-snapshot paths relating them. Adding to that the Sim-TDBMS should provide means to walk within the simulation temporal map in order to access and interrogate the simulation temporal data base. In that direction, the Sim-TDQS which is of great value to understand a simulation run will be built on the basis of the notions described in the simulation temporal map cited above and will take its semantics from answering questions denoted by:

? A(St) which means what snapshot-to-snapshot paths lead to the appearance of the snapshot St or the identification of all the paths related to that snapshot.

? Outcome(St) which means what is the outcome of a snapshot St or the identification of all the beginning and ending events at that snapshot.

? B-event(St) which means identify all the ending events at the snapshot St.

? C-event(St) which means identify all the beginning events at the snapshot St.

In general, the appearance of events within the snapshots takes actions on elements or entities of the simulated system. The investigation of the events' effect, on the attributes of the entities, is usually a primary target in simulation. Such investigation can take the following forms denoted by:

? (Entity (B-event :evtb)) (St) Which means identify the entity which takes part in the ending event evtb at the snapshot St.

? (Entity (C-event :evtc)) (St) Which means identify the entities which take part in the beginning event evtc at the snapshot St.

?Attribute:atr1>Entity:ent1>B-event:evtb)(St) which means identify the value of the attribute atr1 of the entity ent1 which takes part in the ending event evtb at the snapshot St.

?(Attribute:atr1>Entity:ent1>C-event:evtb)(St) which means identify the value of the attribute atr1 of the entity ent1 which takes part in the beginning event evtc at the snapshot St.

One of the important characteristics of the simulation temporal map that it allows one to walk backward and forward from snapshot to snapshot. We denote such actions by:

$(St_{i-1}) > (St_i)$ Which means jump forward from snapshot St_{i-1} to snapshot St_i

$St_{i-1}) < (St_i)$ Which means jump Backward from snapshot St_i to snapshot St_{i-1}.

In addition to that, a dynamic pictorial facility within the Sim-TDBMS is of a great value. This makes is possible to access visually the time dimension in the simulation and create pictorial images of the snapshots. The design inspiration of such facility can be based on the following semantics denoted by:

View(?A(St)) which means visualize the snapshot St.

View(?O(St)) which means visualize the outcome of the snapshot St.

View(?B-event(St)) which means visualize the ending events of the snapshot St.

View(?C-event(St)) which means visualize the beginning events of the snapshot St.

View(?(Entity>B-event:evtb)(St)) which means visualize the entities having taken part in the B-event: evtb at the snapshot St.

View(?(Entity>C-event:evtc)(St)) which means visualize the entities having taken part in the C-event: evtc at the snapshot St.

View(?(Attribute:atr>Entity:ent>Bevent:evtb)(St)) which means visualize the attribute :atr of the entity :ent of the B event :evtb at the snapshot St.

View(?(Attribute:atr>Entity:ent>C-event:evtc)(St)) which means visualize the attribute :atr of the entity :ent of the C event :evtc at the snapshot St.

View($(St_{i-1}) > (St_i)$) Which means jump visually forward from snapshot St_{i-1} to snapshot St_i.

View($(St_{i-1}) > (St_i)$) Which means jump visually Backward from snapshot St_i to snapshot $St_{i-1.}$

6. CONCLUSIONS

In today's simulation, the simulation modelling community is seeking from the simulation software community for software to solve simulation problems by formal means, and to deal with systems complexity at all levels. There is much scope for future development of the Sim-TDBMS if the simulation software community could particularly transform the notions of the Sim-TDBMS into a simulation temporal data base management language (Sim-TDBML) and a simulation temporal data query system (Sim-TDQS).

REFERENCES

[1] H. S. Arons, "Knowledge-based Modeling of Discrete-event Simulation Systems", in *Proceedings of the 1999 Winter Simulation Conference*, ed. P.A. Farrington, H.B. Nembhard, D. T Sturrock, and G. W. Evans, Society of Computer Simulationm, 1999, pp. 591-597.

[2] T. McLean, L. Mark, M. Loper and D. Rosenbaum D., "Applying Temporal Databases to HLA Data Collection and Analysis", *In Proccedings of the 1998 Winter Simulation Conference*, ed. D.J. Medeiros, E.F. Watson, J. S. Carson and M. S. Mannivannan, Society for Computer Simulation, 1998, pp. 827-833.

[3] L. G. Randell and G. S. Bolmsj, "Database Driven Factory Simulation: A Proof-of-Concept Demonstrator", *In Proceedings of the 2001 Winter Simulation*, ed. B.A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer, Society for Computer Simulation, 2001, pp. 977-983.

[4] N. H. Robertson and T. Perera, "Feasibility for Automatic Data Collection", *In Proceedings of the 2001 Winter Simulation Conference*, ed. B.A. Peters, J. S. Smith, D. J. Medeiros and M. W. Rohrer, Society for Computer Simulation, 2001, pp. 984-990.

[5] C. R. Standridge and D. B. Wortman, "The Simulation Data Language (SDL), A Database Management Systems for Modelers", *Simulation Journal*; August 1981, pp. 55-88.

[6] C. R. Standridge, "Modular Simulation Environments. An Object Manager Based Architecture", *In Proceedings of the 1999 Winter Simulation Conference*, ed. P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, Society of Computer Simulation, 1999, pp. 598-602.

[7] T. Wiedmann, "Database Oriented Modeling with Simulation Microfunctions", *In Proceedings of the 1999 Winter Simulation Conference*, ed. P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, Society of Computer Simulation, 1999, pp. 586-590.

[8] J. L. Camolesi Jr, "Survivability and Applicability in Database Constraints: Temporal Boundary to Data Integrity Scenarios", *5th IEEE International Conference on Information Technology: Coding and Computing*, 2004, pp. 518-522.

[9] A. K. Mok, C. Lee and H. Woo, "The Monitoring of Timing Constraints on TimeIntervals", *Proc. IEEE Real-Time Systems Symposium*, 2002, pp. 1-10.

[10] G. Câmara, R. Souza, B. Pedrosa, L. Vinhas, A. Monteiro, J. Paiva, M. Carvalho, and M. Gatass, 2000, "TerraLib: Technology in Support of GIS Innovation", *II Brazilian Workshop of GeoInformatics*, São Paulo(ed), June 2000.

[11] I. Somerville, Software Engineering, *Addison-Wesley* Publishing Company, Inc., 1989.

[12] T. L. Dean, D.V. Mcdermott, "Temporal Data Base Management", *Artificial Intelligence Journal*, n° 32, 1987, pp. 1-5.