South African Computer Journal
Number 18
December 1996

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 18
Desember 1996

# Computer Science and Information Systems

Special Edition: Computer Security

# Rekenaarwetenskap en Inligtingstelsels

**The South African Computer Journal**
*An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists*

**Die Suid-Afrikaanse Rekenaartydskrif**
*'n Amptelike publikasie van die Rekenaarverenging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes*



**Subscriptions**

| | Annual | Single copy |
|---|---|---|
| Southern Africa: | R50,00 | R25,00 |
| Elsewhere: | \$30,00 | \$15,00 |

An additional \$15 per year is charged for airmail outside Southern Africa

to be sent to:
*Computer Society of South Africa*
*Box 1714 Halfway House 1685*

Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

UP Drukkers Printers

# A Pragmatic Approach to Development Information to Provide Service on a Wide Scale

Stephen S Mncube, PhD
*Divisional Manager: Development Information Services, Centre for Policy and Information, Development Bank of Southern Africa*

## 1 Introduction

The rapid technological growth in our time has produced an explosion of information. This, in turn, has spawned information systems based on the use of computers and automated systems. These mechanised devices with their seemingly infinite capacity to store and retrieve knowledge on command have myriad applications. But the use of computer and automated information devices pose serious problems to individuals, groups and societies on an international scale in disseminating the available information. This is even more true in information flow between regions with high information capabilities than in those that have little or none. The information flow between these regions has been varied and frequently haphazard whenever it has existed.

In this context the philosophy for development information speaks to interdependence and humanitarian concern in information flow to various regions of the earth. Information must be perceived as a universal entity. The initial failure of one region and the ability of another to acquire information should not dictate a permanent global demarcation into 'developed and less developed' categories.

## 2 Objectives

The objective of development information is for it to function as an interlinking mechanism between a complex industry of information resources and the users of information. In addition, development information should be a catalytic agent that aims at providing objective clarification in information needs between regions with high information capabilities and those with little.

## 3 Mission

The mission is to strive for a move in all regional, national and international agencies and organisations concerned with information to give full co-operation and assistance in setting criteria and standards, formulating policy, and assessing the information needs of a given region of the world. Development information is capable of realistically assessing information related to socio-economic development in the light of the unique requirements of world regions in need of information and information technology.

## 4 The Dynamics of Information Processing

The proliferation of information technology production has resulted in the growth and development of an ever expanding information packaging industry. This area has become so large in size and scope that it is necessary to discuss some of the important developments that are taking place in the area.

The newest phrases now being used in information packaging range from electronic archives, compact discs (CDs), computer tapes, microfiche, teletexts, video discs, magnetic tapes and interactive imaging systems (optical systems) to word processing and the use of laser technology. Developing countries will not escape this new wave of information packaging. It soon will be bombarded with vendors of these products, to a point where some adaptation will be inevitable.

Assuring the quality of technical processes and the accuracy of packaging information is becoming an increasingly difficult task. Rapid increases in the volume of information, the sophistication of information uses, and the complexity of material flows and processes are characteristic of most modern technical environments. As complexity increases, the risk of introducing significant errors into material processes increases. The very complexity of such systems makes the detection of error itself a complex task. With increasing frequency, public and private organisations are seeking help from corporations with experience in quality control and information validation to ensure that technical process and information packaging meet performance and accuracy standards. The problems association with quality control and validation can be minimised by following these guidelines:

- Establishing ways of aiding in planning, organisation and control of software purchasing and development

through
  - creating a directory of software suppliers
  - evaluating the quality of software supplied
  - keeping abreast of the state-of-the-art in software production
- Providing leadership in the innovative use of software materials and the utilisation of extensive market research on software before making a major purchase
- Establishing cost-efficient ways of packaging and designing your own software by learning how to design and evaluate software for your own use

Preparation for these new technologies for packaging information ought to be made in institutions of higher learning. Perhaps it would be timely to introduce some of these concepts in technical institutes in order that future demands imposed by the new information technology may be met.

The successful growth of developing countries information technology will ultimately depend upon the commitment of substantial resources, especially financial resources. The successful application of this information technology will require more than the mere receiving and storing of it. In addition to the tasks of acquiring and organising informational materials, channels must be established to analyse incoming information. Too much of the information technology that does get transferred out of the industrialised nations is never utilised because it is unsuited for the consumption of users in developing countries. A great deal more effort must be made to analyse, package and disseminate materials on existing and forthcoming information in all vital areas of work and study so that these technologies will be accessible to the developing countries' information-user communities.

## 5 The Need to Establish a Consortium

Information technology specialists need to establish a consortium of regional, national and international information networks and associations. The consortium could be an open structure inviting any institutions, organisations and agencies existing for the purpose of forming a network or documentation clearinghouse and of providing information technology not as an end-product but as a means for human change.

The element common to all membership is an interest in and dedication to providing useful and accurate information that can bring about humanistic change. Equally important is a commitment to the development of relevant information resources to meet the needs of regions with low information capacity.

The philosophical outlook of the consortium would therefore be to crystalise and emphasise broad knowledge, deep understanding, and imaginative efforts, including a dedication to great ideas in providing accurate solutions to the information needs of various regions on an international scale.

## 6 The Organisation of Services for Members

The ideas constituting a conceptual framework for a service-oriented consortium are as multitudinous as the Kalahari sands but in this instance the consortium could function to:

- support creative change within its membership
- facilitate and support new educational enterprises and programs addressed to meet the needs of previously disenfranchised persons
- develop and implement co-operative programs and projects among it members
- provide a meeting ground for a diversity of persons, institutions, and agencies with common values and purposes
- provide a forum for the exchange of ideas among its member associates
- encourage methods of solving social problems
- influence public policy to be consistent with its mission and purpose.

## 7 An Appraisal of Internet

While casual observers have the leisure to observe unobtrusively the growth and development in Internet to be a world-wide phenomenon in information sharing, they do so at no cost. On the other hand, information specialists have to judge and weigh the work of an ongoing Internet program and estimate its usefulness as a network or networks to their daily operations. Information specialists are, therefore, still more sceptical about the scope and magnitude of the Internet. They alone are facing challenges of adding another performance task of being evaluation researchers of Internet in order to provide objective clarification of incorporating Internet as an integral component of their information system. This can be a tedious undertaking because it entails not only knowing how to navigate the Internet network but also cognisance of the following key factors:

- how appropriate the Internet is to your information environments
- to what extent the databanks provided through Internet are relevant to the mission and objective of your environment
- what the relationship is between costs and benefits of having Internet at your disposal

All these factors need to be addressed to determine the effectiveness of Internet in any given information environment, be it in a government setting or in other work environments.

## 8 Training in Information Networks

A seminar for the network should be designed to launch the co-operative exchange of knowledge and experience with information accessibility and utility of the participant's respective information holdings. To succeed in this effort, the organisers will marshal appropriate interdisciplinary experts and technical resources from within the regions involved. The content of the seminar will consist of information related to formalising and establishing a plan for information networking. It is important that information ministries achieve a high level of knowledge and sensitivity to the information needs of their individual country, region, and ultimately the world, in order to assess, prescribe, design, manage and evaluate the most appropriate uses of information technology for enhancing the advancement of their world countries.

Although the organisers will determine who will be invited to participate in this seminar, special effort will be made to ensure that representatives come from a diversity of backgrounds, and have some knowledge or experience relating to information systems. Additionally, consideration will be given to the level of information technology which is currently utilised by the representative's country.

The content of the seminar will be tailored to meet the specific needs and issues designated by the participants through a pre-seminar survey and needs assessment, which will be administered by the organisers.

## 9 Summary

The above aims at presenting some possible scenario and does not pretend to be exhaustive. The issues, however vital to development information, are given cursory treatment here. It remains important for the luminaries in this area to expand on some of the thoughts contained above.

## Bibliography

1. A D Burnett. *Technology for information in development: proceedings of the sixth conference of the International and Comparative Librarianship Group of the Library Association.* International and Comparative Librarianship Group of the Library Association, Brighton Polytechnic, Falmer, Sussex, August 21–23 1987.
2. M Cook. Guidelines on curriculum development in information technology for librarians, documentalists, and archivists. Paris Unesco, 1986.
3. O Fairfax, J Durham, and W W. Audio-visual materials: development of a national cataloguing and information service: Report of a joint feasability study for the british library and the council for educational technology. Distributed by Councils and Education Press, 1976.
4. M Fransman. *1948 – The market and beyond: Co-operation and competition in information technology development in the Japanese system.* Cambridge University Press, New York, 1990.
5. N K Hanna. The information technology revolution and economic development. World Bank Discussion Papers, World Bank Washington DC., 1991.
6. N Heaton and L W MacDonald. *Human factors in information technology product design and development.* Ellis Horwood, 1991.
7. T Koizumi and G E Lasker, eds. *Advances in education and human development: new trends in learning and teaching evolution of human systems and institutions, individual freedom and sociopolitical controls in the high technology environment of the advanced information society.* International Institute for Advanced Studies, Baden-Baden, West Germany, 1990.
8. V Sethi. The development of measures to assess the extent to which an information technology application provides competitive advantage. Ann Arbor: MI, University Microfilm International, University of Pittsburgh, 1988.

SACJ is produced with kind support from
Mosaic Software (Pty) Ltd.

# Dynamic Deadlock Avoidance: A Prototype

C N Blewett* G J Erwin†
*Business Information Systems Section, Dept. of Accounting and Finance, University of Natal, King George V Ave., Durban, 4001, South Africa. blewett@bis.und.ac.za
†Information Systems Section, Dept. of Accountancy, University of Durban-Westville, Private Bag X54001, Durban, 4000, South Africa. erwin@is.udw.ac.za

## Abstract

*In this paper, we describe a prototype of an expert system software advisor for the deadlock treatment of a lock manager in a database system. The software advisor, called EAGLE (**E**xternal **A**dvisor for **G**ranting **L**ocks **E**xpertly), maintains a record of lock requests and lock status within a simulated database management system environment processing transactions. The sequence of lock information is held as a script(s). EAGLE matches lock request sequences against previously stored scripts. As EAGLE gains experience of lock event sequences leading to deadlock, it recognises patterns which have led to deadlock, and dynamically objects to the further granting of locks which would repeat a previous deadlock-inducing sequence of locks. This paper describes the design of EAGLE, EAGLE's dynamic deadlock avoidance (DDA) technique, and presents preliminary results from the EAGLE prototype compared with a simulated Deadlock Detection and Resolution technique (DLD).*

**Keywords:** *deadlock, database, expert system, scripts, knowledge representation*
**Computing Review Categories:** *H.2.2, I.2.4*

## 1 Definitions and Preliminaries

"Deadlock is a situation in a resource allocation system in which two or more processes are in a simultaneous wait state, each one waiting for one of the others to release a resource before it can proceed." Deadlock may occur within database, communication and distributed systems using locking as a concurrency control strategy [3].

The necessary conditions for deadlock [12] are; *lock-out*: exclusive control of a resource by a transaction (process); *concurrency*: competition for exclusive control of a resource(s); *completion*: exclusive resource control until transaction completion; *additional request*: request for exclusive control over a growing number of resources; *circular wait*: a circular chain (cycle) in which a transaction has exclusive control over a resource requested by the next transaction in the chain. Deadlock detection (usually) notices that there is a circular wait (cycle). The deadlock state and its treatment are well-described [1, 5, 8, 10, 12, 22].

The lock manager, LOMAN, of a database management system controls resource access by arranging exclusive control (lock) over a resource. A request for an exclusive lock can be refused, causing a wait, because another transaction currently has exclusive control over that resource. Or, the lock request can be granted, possibly contributing to a future deadlock. The sequence of lock activities by transactions using a database management system is a *locking event sequence*.

There are three methods for dealing with the deadlock problem: prevention, avoidance and detection with recovery. Prevention ensures that a deadlock will never happen. Avoidance should also ensure no deadlocks, but, can only do that by using *a priori* data about the locking activities of transactions (resource consumers). Detection with recovery allows the system to enter a deadlock state and then recover from it [3].

## 2 Introduction

Blewett and Erwin [7] describe *four* categories of deadlock treatment: *viz. ignore deadlock, deadlock detection and resolution, deadlock prevention* and *deadlock avoidance*. Belik [3] describes a previous deadlock avoidance approach, *based on pre-stored knowledge* of the content and order of locking activities. The dynamic deadlock avoidance (DDA) approach described in this paper does *not* require pre-stored knowledge of lock request sequences, but attempts to notice potential deadlocks based on experience accumulated from observations of the previous locking event sequences which led to deadlock. DDA allows deadlock to occur, records the conditions which led to the deadlock, then watches for similar, deadlock-producing conditions in future.

DDA treats the deadlock problem as a plan *recognition issue* [4], rather than as a *problem resolution issue*. The DDA approach attempts to identify whether the "goal" of the running "plan" of current locking event sequences is deadlock. When this goal is recognised, DDA objects to further locking activities which could lead to a future deadlock.

In the following sections we first describe the overall design of our DDA-based prototype, called EAGLE (**E**xternal **A**dvisor for **G**ranting **L**ocks **E**xpertly), incorporating script-based knowledge representation [18, 20, 19], in a centralised resource allocation (database) system. We then discuss the implementation of the EAGLE prototype, EAGLE's components, and test runs under simulated conditions within a database system. We describe the sets of test data used to assess the impact of EAGLE on the occurrence of deadlock under those simulated conditions and present some preliminary results from those test runs.

## 3 Script-based knowledge representation

A script is a knowledge representation scheme for representing time-based events. "A script is a finite set of events of some duration and importance oriented towards a goal." [17]. Schank and Abelson [18] state "... a script, ... is a stereotypical representation of a sequence of actions oriented toward attaining some goal." A script-based knowledge representation in an expert system stores data about sequences of events with the following characteristics; *duration*: a start and a finish; *dependency*: certain events occur before/after others, and *stereotypical aspects*: event sequences are often predictable and well-understood.

The script-based knowledge representation is a specific application of the frame knowledge representation. Each frame's slot definitions are constructed to apply to the specific application under consideration [21].

Scripts were initially formulated, [18, 20, 19], to provide structures and manipulative ability for the handling of time and dependency-based events in the domain of "understanding" natural language. This work established conceptual dependency operators within natural language as a formal methodology for representing the semantic content of sentences and stories. Scripts are suitable for describing, storing, and searching for sequences of actions with attendant states. Script structures and manipulations have been applied to a variety of contexts. [4, 9, 11, 14–16].

## 4 Representing locking event sequences as scripts

Locking event sequences exhibit, *inter alia*, the same characteristics as scripts. Each locking event (activity) has starting and finishing times; each event (except the first and last events) has a predecessor(s) and a successor(s); and, the combination of events in sequence usually forms a well-ordered, stereotypical and understood series of events. In the EAGLE context, we regard the "goal" of these locking events sequences as the attainment of deadlock. EAGLE formats locking event sequences as scripts, and stores those scripts which resulted in deadlock.

Each frame (script) in DDA contains slots for: script name, process roles, resource props, expected event, critical event, critical event response, sequence description, response, activation level, utilisation level, minimum activation level and maximum activation level.

The *script name* uniquely identifies each script. The *process roles* and *resource props* identify the number of transactions and resources involved in the deadlock. The *expected event* is used to look for the next expected event for a script. Any objection to the granting of a lock request must take place before the *critical event*. The *sequence description* describes a stereotypical event sequence and the *response* is the result of a match against this sequence description. The various *activation* and *utilisation levels* monitor and control the activation of the script.

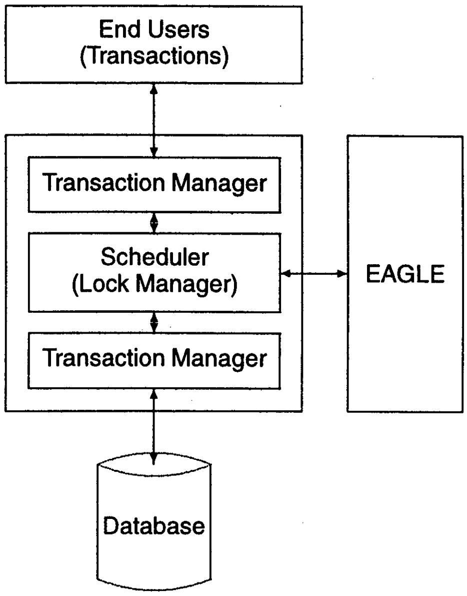


*Figure 1. The EAGLE operating environment*

## 5 EAGLE Implementation

The EAGLE system was developed in the CA-Clipper programming language / database system and Common LISP. CA-Clipper was used for the simulation of database activities, and Common LISP was used for the Matcher/Analyzer which generates stereotypical event sequences and compares locking event sequences against stored stereotypical event sequences. EAGLE is the script-based implementation of the DDA scheme. Incoming lock requests (as part of locking event sequences) are passed from LOMAN to EAGLE, where they are matched against stored stereotypical locking event sequences, representing various patterns of lock events which lead to deadlock situations. Should an event (lock or unlock request) in a locking event sequence cause a match to occur with a stored stereotype script, the lock or unlock request is rejected. If no match occurs, the lock event is approved. Figure 1 shows an overview of the EAGLE operating environment.

LOMAN receives transactions from the Transaction Manager. LOMAN checks its lock table (if a lock request is received) to determine if the requested resource is available, i.e. the resource is not currently locked by another transaction. If the lock request is approved by LOMAN, the lock request details (transaction and resource(s) involved) are passed to EAGLE for further inspection. EAGLE adds the current lock request to the script for that transaction. EAGLE then matches the current content of the locking event sequence against stereotypical scripts to estimate the probability of a future deadlock occurring.

EAGLE calculates a similarity metric (SM) or belief [17] for each instantiated script to determine the degree of relevance of a script to an observed locking event sequence. This is similar to Benoit et al's [4] SCAN system, where an overall likelihood is computed for each of the scripts. The SM is the ratio of the number of bindings made in the matching process to the total number of variables in the script. The SM (estimated probability of deadlock occurring) is the basis of EAGLE's decision to approve or object to a lock request. If EAGLE approves granting of the lock, then LOMAN grants the lock to the transaction. If denied, the transaction is forced to re-request the lock, at which time, if LOMAN approves the request, EAGLE will re-evaluate the lock request. The EAGLE system initially has no stored scripts describing deadlock situations stored. As a result, all "early" lock requests are granted. When a deadlock occurs, the contents of that locking event sequence are passed to the Learner sub-system of EAGLE. Here, the current locking event sequence is converted into both a *specific* stereotypical event sequence (one which contains lock, unlock, and wait tasks) and into a *generic* event sequence (one which contains lock and wait tasks of only the transactions involved in the deadlock). As further deadlocks occur, so the number of stored scripts increases, building a Script Base. The Matcher inside EAGLE continues to match fresh locking event sequences against the Script Base. If a match is made, EAGLE OBJECTs to the current request, and returns that OBJECTION to LOMAN. EAGLE has **four main components**. Considering each in turn:

- **Rule Component.** The rule component has two subcomponents, namely the Communication Interface and the Activation System. The Communication Interface is responsible for passing messages and responses between LOMAN, the Matcher/Analyzer and the Script Base. The Activation System is responsible for activating the Matcher/Analyzer when the Precondition Header (at least 2 concurrent processes in operation) and the Instrumental Header (at least one lock request for a resource required by another transaction) are satisfied.
- **Matcher/Analyzer Component.** The second component of EAGLE is the Matcher/Analyzer, based on the IPS architecture of [4]. The Matcher/Analyzer is activated by the Rule Component. The Matcher/Analyzer receives the current locking event sequence from the Communication Interface of the Rule Component. The Matcher/Analyzer compares the current locking event sequence against stereotypical locking event sequences stored in the Script Base. If the conditions for an OBJECTion are met, the Matcher/Analyzer will return an OBJECT response to the Rule Component, and the transaction lock request is not granted. Figure 2 depicts an OBJECT being made based on a successful script instantiation. If no other transaction has made any requests, the same transaction may attempt to request the resource. This component also converts actual locking event sequences into stereotypical format.
- **Script Base Component.** The third component of EAGLE is the Script Base. The Script Base holds the stereotypical locking event sequences which have been encountered previously. There are two types of stereotypes, namely, a *specific* stereotype and a *generic* stereotype. A *Specific Stereotype* describes a stereotypical event sequence in the context of the current operational environment, i.e. a specific stereotype is not applicable in another context. A specific stereotype records all lock, unlock, and wait events of deadlock participants and non-deadlock participants. A *Generic Stereotype* describes scripts in a domain-independent context. No details of non-participating transactions' events or unlock requests are recorded in the generic stereotype. Generic stereotypes are useful in situations where specific stereotypes have not yet been defined.
- **Learner Component.** The Learner component is activated by the Rule Component whenever LOMAN's deadlock detector detects a deadlock. To detect deadlocks, LOMAN constructs wait-for-graphs which are analysed for cycles showing the presence of deadlock. The deadlock detection technique is similar to Topological Sorting [3], using a graph reduction approach with an adjacency list representation of the graph. The entire list is scanned for cycles. The details of the event sequence leading to the detected deadlock are converted into specific and generic stereotypical forms and stored in the Script Base. The utilisation of the scripts recorded in the Script Base is continually monitored. The Activation Level of the script is adjusted to tune the script to an effective level of performance. Over-utilisation of a script can lead to unnecessary system interference, and under-utilisation of a script wastes system resources. Scripts failing to perform within the defined parameters, after adjustments have been made, are removed from the Script Base.

## 6 Test runs

A simulated transaction environment was developed in which EAGLE operated. A series of simulated transactions was run in order to test and evaluate EAGLE's impact on the occurrence of deadlock. Simulated transactions occurred in two situations, *viz*, (i) EAGLE active, i.e. Dynamic Deadlock Avoidance (DDA), and (ii) EAGLE inactive, i.e. Deadlock Detection (DLD). [5] identified the following assumptions for evaluating a concurrency control algorithm, *viz*, "All transactions require the same number of locks, all data items are accessed with equal probability, and all locks are write locks. The transactions use Strict 2PL: data items are locked before they are accessed, and locks are released only after all transactions commit (or abort). The database is centralised.....". Considering each assumption in turn in EAGLE:

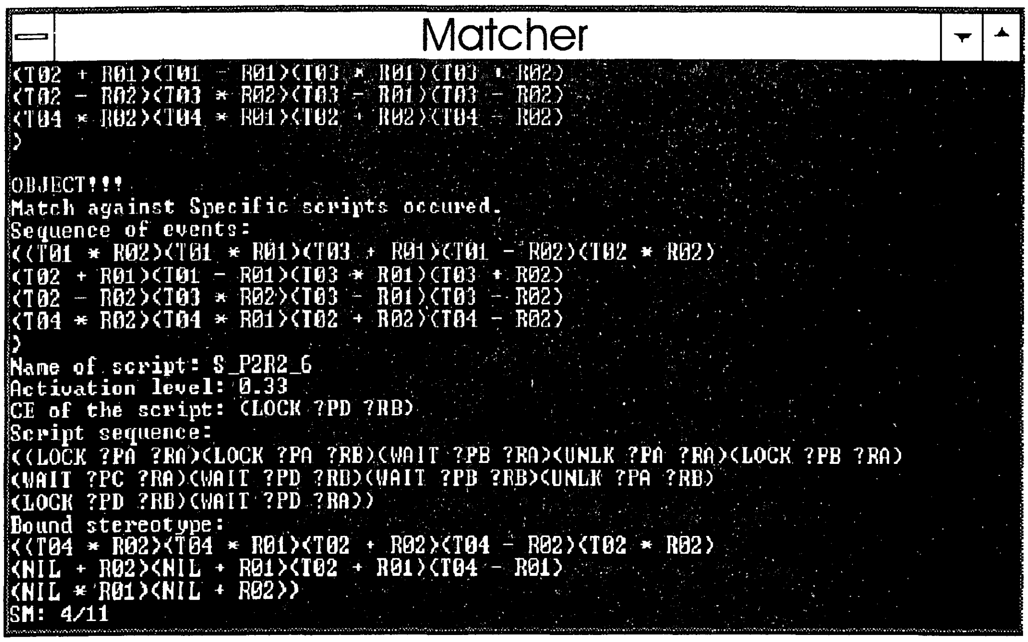


*Figure 2. EAGLE OBJECTing to a lock request*

- *"All transactions require the same number of locks".* Although this assumption was largely adhered to, it was relaxed for one simulation run (Run 8) in order to create a low conflict situation (mean number of deadlocks below 1) with four concurrent transactions.
- *"All data items are accessed with equal probability".* This assumption was also relaxed in Run 8.
- *"All locks are write locks".* All locks were treated as if they were write locks.
- *"The transactions use strict 2PL".* Strict 2PL (2 phase locking) was used in the simulator as this is the most commonly implemented and best performing form of locking [1].
- *"Centralised Database".* The simulated model is based on the closed queueing model of a single-site database used in [1].

**Simulation parameters**

Several parameters were used within the transaction simulations. Considering each parameter in turn:

- **Transaction Size.** This is the number of resources processed by transactions. Except for run 8, all transactions accessed the same number of resources. Varying the transaction size alters the number of possible deadlock situations, and hence the number of scripts required to represent the stereotypical deadlock situations.
- **Multiprogramming Level (MPL).** The MPL is the number of transactions that are processed concurrently in a run. The higher the MPL the higher the potential resource conflict, and consequently the higher the probability of deadlock. The MPL parameter varies from 2 to 5 transactions.
- **Resource Contention.** This is the proportion of the resources used by all participating transactions. A value of 1.0 means that all transactions require all resources, resulting in higher resource contention. A value of 0.5 means that only half of the resources required are common to all transactions, so resulting in lower resource contention. This parameter allows for larger transaction sizes with lower resource contention. This parameter was set to 1.0 except in Run 8 and Run 12. In Run 8 it was set at 0.5 and in Run 12 to 0.2. This allowed for a low conflict situation between 4 concurrent transactions.
- **Activation Level.** This is the degree of similarity required (measured between 0, no similarity, and 1, exact match) between an observed locking event sequence and a stored script to activate the script. The activation level of all new scripts is initially at 0.5. This level is adjusted as scripts are utilised.
- **Clean Level.** This parameter determines when un/under-utilised scripts are removed from the Script Base. The clean level is the number of activations of the Matcher/Analyzer after which un/under-utilised scripts are removed. The clean level was set at 200. Decreasing the level below 200 caused many scripts to be removed from the database that had not yet been given sufficient opportunity to be activated. Increasing the clean level above 200 caused the number of scripts in the Script Base to grow too large, thus resulting in long matching times and multiple OBJECT ceilings (see below). The clean level was set at 200.
- **OBJECT Ceiling.** In order to avoid an EAGLE-lock (deadlock-like) situation arising, a limit was set for the number of consecutive OBJECTions. If no such limit is set, it is possible that no transactions will proceed, because all transactions match a script in the Script Base. The OBJECT ceiling limits the number of consecutive OBJECTions that will be permitted. Once this ceiling value is reached, any further lock request

is automatically granted. Setting the OBJECT ceiling too low e.g. 2, means that many valid OBJECTions are ignored. However, setting the OBJECT ceiling too high e.g. 10, slows the forward progress of transactions. An OBJECT ceiling of 5 was found to be an effective value, providing a trade-off between ignoring valid OBJECTions and impeding transaction progress.

The following is a summary of the main simulation parameters used.

**TRAN_SIZE.** The resource size of a transaction, varies from 2 to 5.

**MPL.** Multi-programming level, varies from 2 to 5 transactions.

**RESOURCE CONTENTION.** Proportion of resources used by all participating transactions, varies from 0.2 to 1.0.

**ACTLEVEL.** Activation Level – The degree of similarity necessary to activate a script. The initial ActLevel is 0.50.

**OBJCEIL.** The OBJECT ceiling is the maximum number of consecutive OBJECTions allowed. Set at 5.

**Test data**

Twelve sets of test data were used, with varying MPLs and TRAN_SIZEs. As performance was not being measured, it was not necessary for transactions to perform any actual processing/updating of a real database. Simulated transactions acquire locks and then, on completion, release all acquired locks. The MPL varies between 2 and 5 concurrent transactions, and the TRAN_SIZE between 2 and 5 resources. These values were chosen for the following reasons:

- **Simulation Time.** The runs were constructed to test the algorithm under varying degrees of resource contention. The MPL and TRAN_SIZEs were therefore "selected so as to jointly yield a region of operation which allows the interesting performance effects to be observed without necessitating impossibly long simulation times" [1]. This is particularly true in the simulated environment, when DDA is active, as all lock requests have to be passed to the Matcher/Analyzer for evaluation. As the Matcher/Analyzer was implemented in interpreted LISP, increasing the MPL or TRAN_SIZE results in more complex matches and extremely long elapsed times for simulation runs.
- **Rarity of Conflict.** Besides the problem of long simulation run times, with high MPLs and large TRAN_SIZEs, "conflicts become very rare; and when conflicts are rare, all concurrency control algorithms perform alike" [1].
- **Heavy Conflict Workloads.** The chosen MPLs and TRAN_SIZEs yield a relatively high mean number of deadlocks as a result of high levels of conflict (measured by mean number of waits). Bachman [2] indicates that a high deadlock rate is 100 per hour, and the simulation runs can exceed that figure. For example, run 4 resulted in approx. 300 deadlocks in one hour. These "non-negligible conflict levels ... facilitate understanding how the algorithms will perform under heavy conflict workloads, or when hot spots' exist in the database." [1].
- **Realistic levels.** A MPL of 5 is comparable to the general workload of a conventional order entry system [13].

## 7 Simulation results

There were 12 sets (runs) of transaction data constructed with various combinations of MPL, TRAN_SIZE and RESOURCE CONTENTION. Each of the 12 runs was executed (simulated) 200 (usually) times (cycles), with random selection of the sequence in which transactions were selected for forward progress. Three of the runs were not done for 200 cycles, because of the infeasible (long) simulation elapsed times involved. Each set of data was processed both with EAGLE on (DDA) and with EAGLE off (DLD). The number of locks granted, waits (denied locks), unlock requests, deadlock occurrences etc. were recorded in a report file. Results from the twelve simulation runs are summarised in Table 1. The first four rows (RUN NO., MPL, TRAN_SIZE, RESOURCE CONTENTION) describe the run. The SAMPLE SIZE is the number of times (cycles) the same set of transactions was processed. The last 4 rows contain the MEAN number of DEADLOCKS and the MEAN number of WAITS per cycle of transactions for both the EAGLE runs (DDA) and the runs without EAGLE (DLD).

As an illustration, in Runs 2 and 3, MEAN WAITS using DDA is *less* than MEAN WAITS using DLD. In Runs 4 and 11, MEAN WAITS using DDA is *more* than MEAN WAITS using DLD. In Runs 2 and 3, MEAN DEADLOCKS occurring with DDA is less than that occurring with DLD. However, in Runs 6 and 11 the MEAN DEADLOCKS is more with DDA than when DLD is used. A moving average provides a visual indication of the effect of introducing DDA into simulated locking activity. A moving average of the number of occurrences of deadlock per transaction cycle was calculated for each run. The moving average period was set to 20. The moving average chart for *one* of the Runs (Run 3) is shown in Figure 3. The impact of EAGLE with DDA, relative to the "normal" Deadlock Detection and Resolution (DLD) without EAGLE, is clearly (visually) apparent. RUN 3 began with DLD processing and no "advice" or learning from EAGLE, with an average number of deadlocks per transaction cycle of 0.87. Then, beginning at transaction cycle 201, EAGLE began to "advise" LOMAN on the granting of locks, and to learn about deadlock-inducing sequences of lock activities. The average number of deadlocks quickly decreased to 0.61.

The simulation runs results indicate that, in certain situations, EAGLE reduces the number of deadlocks and waits occurring.

**Table 1. Results of EAGLE simulation runs**

| Run no. | No. trans. (MPL) | No. resources (TRAN_SIZE) | Resource contention | Sample size | MEAN WAITS (DLD) | MEAN WAITS (DDA) | MEAN DEADLOCKS (DLD) | MEAN DEADLOCKS (DDA) |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 1.0 | 200 | 1.56 | 1.76 | 0.52 | 0.57 |
| 2 | 2 | 3 | 1.0 | 200 | 2.13 | 1.93 | 0.73 | 0.58 |
| 3 | 2 | 4 | 1.0 | 200 | 2.84 | 2.35 | 0.87 | 0.61 |
| 4 | 3 | 3 | 1.0 | 200 | 5.82 | 6.05 | 1.65 | 1.65 |
| 5 | 5 | 3 | 1.0 | 40 | 20.2 | 19.4 | 6.73 | 6.23 |
| 6 | 3 | 2 | 1.0 | 200 | 4.36 | 4.56 | 1.19 | 1.23 |
| 7 | 4 | 2 | 1.0 | 200 | 10.8 | 11.0 | 3.64 | 3.62 |
| 8 | 4 | 1-3 | 0.5 | 200 | 3.08 | 2.55 | 0.52 | 0.33 |
| 9 | 4 | 3 | 1.0 | 110 | 12.6 | 11.9 | 3.91 | 3.63 |
| 10 | 3 | 5 | 1.0 | 200 | 8.23 | 7.83 | 2.33 | 2.09 |
| 11 | 5 | 5 | 1.0 | 50 | 24.3 | 25.3 | 7.04 | 7.40 |
| 12 | 4 | 3 | 0.2 | 200 | 6.23 | 6.05 | 1.36 | 1.26 |

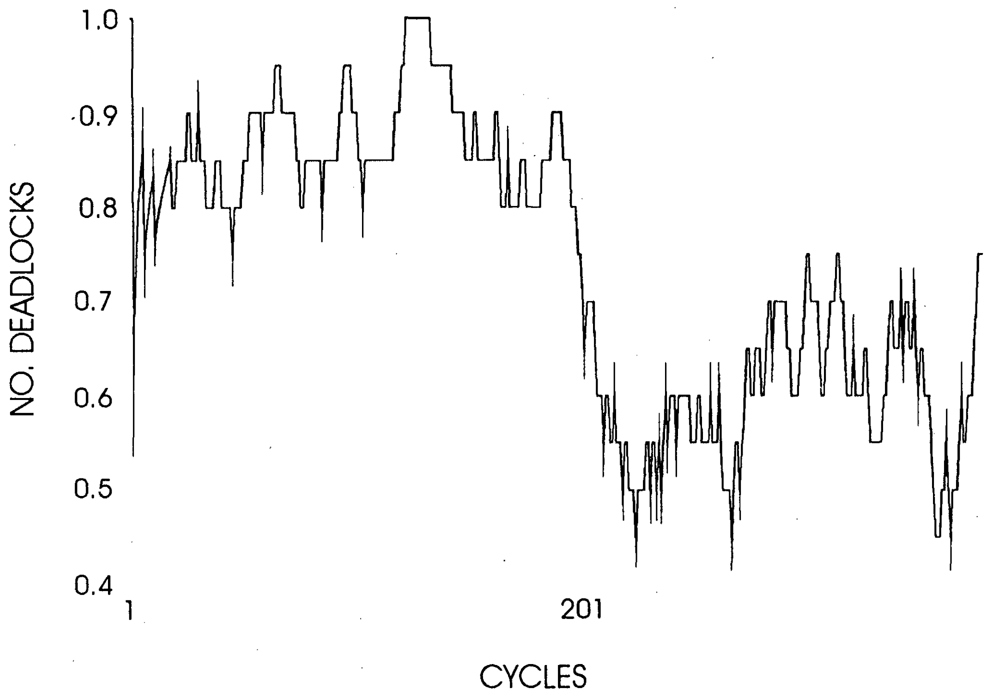


*Figure 3. Run 3: Moving average of deadlock occurrences without EAGLE, then with EAGLE.*

## 8 Evaluation of DDA vs DLD

Two sets of transactions were processed with the different deadlock treatment techniques (DDA and DLD). Other deadlock treatments exist [1, 3, 5, 10, 12, 13, 22], but this study was restricted to a direct comparison between DDA and DLD. The effectiveness of DDA compared with DLD was measured with two of the "most important performance factors for locking" viz, "deadlock rate" (number of deadlocks) and "lock wait time" (number of waits). The third factor (lock conflict probability) is the primary measure of resource contention, which was not part of our investigation. All three factors "are interrelated" [13].

The null hypothesis was established as:

$H_0$: *There is no decrease in the criteria measures (number of deadlocks and number of waits) occurring with DDA compared with DLD.*

Alternate hypotheses were also tested, but are not reported in this paper. The Mann-Whitney U-Test was selected for (most of the) testing for significant difference [6]. Because of the nature of some runs, it was not meaningful to draw any statistical conclusion. For example, Run 1 was a 2 transaction, 2 resource simulation, so DDA could not avoid deadlock dynamically. All statistical tests were evaluated initially at a 95% level of confidence.

For many of the runs, the null hypothesis was not rejected. Further examination showed that DDA was not associated with a significant decrease in the number of deadlocks when MEAN DEADLOCKS was greater than 1, and, was not associated with a significant decrease in the number of waits when MEAN WAITS was above 3.08. Runs 2, 3 and 8 all have a significantly decreased number of deadlocks and waits associated with the use of DDA. The number of deadlocks is significantly decreased with confidence intervals in excess of 99.998%, and the number of waits is significantly decreased with confidence intervals in excess of 99.994%. DDA reduces the number of deadlocks and waits, compared with DLD, in low conflict situations, typified by a mean number of deadlocks below 1 and a mean number of waits below 3.08.

See [6] for a fuller discussion of the simulation run results for various combinations of transactions, parameters and resources.

## 9 Conclusion

Script-based knowledge representation has proved to be appropriate for treatment of potential deadlock situations based on learning from previous experience. EAGLE (DDA) significantly decreases the number of deadlocks and the number of waits in low conflict situations. However, DDA (currently) does not significantly decrease the number of deadlocks or the number of waits in a high conflict situation. Further work is needed in dealing with high conflict situations. There are two possible reasons why DDA is effective in low conflict situations:

- The OBJECTion ceiling is too low for high conflict situations. Too many situations are identified by EAGLE where deadlock seems imminent, resulting in multiple OBJECTions and many occasions in which OBJECTions reach the OBJECT ceiling. The result is that EAGLE is unable to avoid many deadlocks.
- High conflict situations may result in a large number of combinations of lock processing for transactions with multiple (different) deadlock situations. The number of stored stereotypical scripts increases quickly as does the likelihood of an unfolding locking event sequence not matching any of the stored scripts. The Activation Level and Clean Level also contribute.

The authors continue to develop EAGLE and produce empirical data from a large variety of simulation runs. Future work will look at the impact of EAGLE on throughput and response time inside a database management system, other forms of matching (especially to improve EAGLE's performance in high conflict situations), such as neural networks, and evaluation of conditions in which EAGLE reduces deadlock occurrences with related reductions in system resource usage. As with many deadlock treatment schemes, EAGLE can also reduce the amount of interleaving of concurrent transactions. The parameterisation of many aspects of EAGLE will allow future work to explore the trade-off between interleaving and deadlock treatment. Experimental comparisons of various deadlock and lock methods may also be possible.

## References


1. R Agrawal, M J Carey, and L W McVoy. 'The performance of alternative strategies for dealing with deadlock in database management systems'. *IEEE Transactions on Software Engineering*, **SE–13**(12):1348–1363, (1987).
2. C W Bachman. 'The programmer as a navigator'. *Communications of the ACM*, **16**(11):653–658, (1973).
3. F Belik. 'An efficient deadlock avoidance technique'. *IEEE Transactions on Computers*, **39**(7):882–888, (July 1990).
4. J W Benoit, J R Davidson, E J Hofman, S J Laskowski, and R R Leighton. 'Integrating plans and scripts: an expert system for plan recognition'. In *Proceedings of the Third Annual Expert Systems in Government Conference*, pp. 201–207. IEEE Computer Society Press, (1987).
5. P A Bernstein, V Hadzilacos, and N Goodman. *Concurrency Control and Recovery in Database Systems.* Addison-Wesley Publishers Ltd, Reading, Massachusetts, 1987.
6. C N Blewett. *A Script-based Dynamic Deadlock Avoidance Technique*. University of Natal, Durban, South Africa, 1996. M. Comm. Thesis, Business Information Systems Section, Faculty of Economics and Management.

7. C N Blewett and G J Erwin. 'The application of scripts to deadlock avoidance'. In *Proceedings of SAICSIT95, Research & Development Symposium, Pretoria, South Africa*, pp. 33–47. (May 1995).
8. C N Blewett and G J Erwin. 'EAGLE: A script-based embedded expert system for deadlock avoidance'. In *Proceedings of Third World Congress of Expert Systems (WCES-3), 5-9 February, Seoul, South Korea.* (1996). accepted for publication.
9. D C Chen. 'Progress in knowledge-based flight monitoring'. In *Proceedings of the Second Conference on Artificial Intelligence Applications, The Engineering of Knowledge-Based Systems*, pp. 441–446. IEEE Computer Society Press, Washington, (December 1985).
10. C J Date. *An Introduction to Database Systems*, volume 1. Addison-Wesley Publishing Coy., USA, fifth edition, 1990.
11. G J Erwin and P A Bowen. 'Describing the building procurement process using script-based knowledge representation'. *South African Journal of Science*, **90**(10):543–546, (October 1994).
12. G C Everest. *Database Management: Objectives, System Functions and Administration*. McGraw-Hill, USA, 1986.
13. B Jenq, B Twichell, and T Keller. 'Locking performance in a shared nothing parallel database machine'. *IEEE Transactions on Knowledge and Data Engineering*, **1**(4).
14. J E Larsson and P Persson. 'Knowledge representation by scripts in an expert interface'. In *Proceedings of the American Control Conference*, pp. 1159–1162. (1986).
15. S J Laskowski and C N Hoffman. 'Script-based reasoning for situation monitoring'. In *Proceedings of AAAI-87*, pp. 819–823. (July 1987).
16. L Z Markosian, A J Rockmore, K J Keller, and M R Gaan. 'An expert system for air-to-air combat management'. In *Proceedings of the Eighteenth Asimolar Conference on Circuits, Systems, and Computers*, pp. 112–116. (1985).
17. C N Mutchler and S J Laskowski. 'Script matching and belief propagation'. In *Proceedings of the 5th IEEE International Symposium on 'Intelligent Control'*. MITRE Corporation, (September 1991).
18. R C Schank and R P Abelson. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977.
19. R C Schank and P Childers. *The Cognitive Computer: On language, learning and artificial intelligence.* Addison-Wesley, Reading, Massachusetts, 1984.
20. R C Schank and C K Riesbeck, eds. *Inside Computer Understanding: Five programs plus miniatures.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1981.
21. D A Waterman. *A Guide to Expert Systems*. Addison-Wesley, Reading, Massachusetts, 1986.
22. G W Wiederhold. *Database Design*. McGraw-Hill Book Company, New York, second edition, 1983.


**Acknowledgements:** Support work for EAGLE was done by Karl Fischer, a Business Information Systems (Honours) student in the Department of Accounting and Finance, Business Information Systems Section, University of Natal, King George V Ave., Durban, South Africa, 4001. LISP code for the Matcher/Analyzer was developed by Anna Richter c/o Mr A. Richter, Dept. of Mechanical Engineering, University of Natal, King George V Ave., Durban, South Africa, 4001.

Received: 2/96, Accepted: 7/96, Final copy: 10/96.

# Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

### Form of Manuscript

Manuscripts for *review* should be prepared according to the following guidelines.

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets [1–3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) LaTeX file(s), either on a diskette, or via e-mail/ftp – a LaTeX style file is available from the production editor;
2. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be original line drawings/printouts, (not photocopies) on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

   Contact the production editor for markup instructions.
3. In exceptional cases camera-ready format may be accepted – a detailed page specification is available from the production editor;

Authors of accepted papers will be required to sign a copyright transfer form.

### Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30-00 per final page for LaTeX or camera-ready contributions that require no further attention. The maximum is R120-00 per page (charges include VAT).

These charges may be waived upon request of the author and at the discretion of the editor.

### Proofs

Proofs of accepted papers in category 2 above may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. Camera-ready submissions will only be accepted if they are in strict accordance with the detailed guidelines.

### Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

### Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

### Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

## References


1. E Ashcroft and Z Manna. 'The translation of 'goto' programs to 'while' programs'. In *Proceedings of IFIP Congress 71*, pp. 250–255, Amsterdam, (1972). North-Holland.
2. C Bohm and G Jacopini. 'Flow diagrams, turing machines and languages with only two formation rules'. *Communications of the ACM*, **9**:366–371, (1966).
3. S Ginsburg. *Mathematical theory of context free languages*. McGraw Hill, New York, 1966.

**South African Computer Journal**

Number 18, December 1996
ISSN 1015-7999

**Suid-Afrikaanse Rekenaar-tydskrif**

Nommer 18, Desember 1996
ISSN 1015-7999

# Contents

## GUEST CONTRIBUTION



## RESEARCH CONTRIBUTIONS



## COMMUNICATIONS AND VIEWPOINTS