J. M. Bishop
8/7/84

Quaestiones Informaticae

Vol. 3 No. 2 July, 1984

Quaestiones Informaticae

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

'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Editor: Prof G. Wiechers
Department of Computer Science and Information Systems
University of South Africa
P.O. Box 392, Pretoria 0001

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R6	$7	£3,0
Institutions	R12	$14	£6,0

Circulation manager
Mr E Anderssen
Department of Computer Science
Rand Afrikaanse Universiteit
P O Box 524
Johannesburg 2000
Tel.: (011) 726-5000

Quaestiones Informaticae is prepared for publication by Thomson Publications South Africa (Pty) Ltd for the Computer Society of South Africa and the South African Institute of Computer Scientists.

Editorial Note

Regrettably this is the last issue of *Questiones Informatica* to appear in its current format. There are a number of reasons for terminating the production of *QI* in printed form. Firstly, the cost of publication has tripled over the last couple of years. Secondly, the interest in the journal has dwindled to the point where one may question the need for a South Arican publication on the more academic aspects of computing: the readership has not expanded but, more seriously, despite many appeals no contributions are submitted. At present I have only one paper in the pipeline. It appears that the only supply of papers comes from the two-yearly Computer Science Symposium, and this is not sufficient to justify the expense of publishing *QI* in its present form.

Nevertheless, both the Computer Society and the Institute of Computer Scientists feel that a vehicle for publishing the results of research in the field of information technology is required for this country. We propose to continue *QI* in a format similar to that used by *Questions Informatica*. In other words we shall use a photo-reproduction process and a printed cover to continue the publication of *QI*, but at greatly reduced costs. This puts the onus of delivering reproducible copy of the authors, but relieves the printers from the problems experienced with unusual symbols, diagrams, computer printouts, etc.

The journal will continue to accept only papers which have been referred. In fact, we shall try to satisfy all the requirements for being regarded as a publication acceptable under the rules for obtaining university subsidies. It is also intended to guarantee quick publication, say not more than three months after acceptance of a paper. However, the journal can only continue to exist if it receives enough contributions of sufficiently high standard.

Again, I would like to appeal to all engaged in research and development in computer related areas, to consider publishing your results in *QI*.

Finally, it is my pleasure to thank Dick White, of Thomson Publications, for all he has done to produce this journal in its current form.

G WIECHERS,
Editor

Specification and Performance Prediction of Fourth Generation Language Run Units

S. Wulf
Comcon (Pty) Limited, Johannesburg

Abstract
Fourth generation languages offer substantial productivity benefits to data processing departments. Major problems associated with their use include the inability of conventional project management systems to control systems development when they are used and possible future performance problems when large volumes are implemented. A process graphing technique is presented for specifying run units in this environment. It can be used in a prototyping environment and also provides the necessary data for building models to predict eventual performance.

1. Introduction

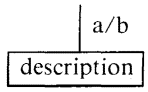
During the structured system design phase [1], all required business processes and computer run units are identified. The term "run unit" is used to describe stand-alone programs or transactions which must be implemented as part of the required system. In traditional systems development, the next phase includes program specification, design and coding. Fourth generation languages (eg [2]) have been developed to enhance both specification and programming productivity. The major problem associated with their use is that if the controls of traditional project management systems are adopted, techniques such as user prototyping cannot be used. (Project management systems require signed off user specifications). Alternatively, organisations which abandon conventional project control find that they lose control of quality during programming. The ADM methodology has been developed using fourth generation languages in an integrated database environment. It includes the creation of process graphs for every required run unit. These process graphs become run unit specifications for later implementation and also provide the necessary parameters for building prediction models [3]. The predictions of performance when full volumes are implemented on the computer system become the primary control tool for arriving at a physical database design which not only satisfies performance constraints but also models as closely as possible the underlying conceptual schema [4] of the enterprise. In this paper, process graph construction is described and the required steps to build a prediction model when direct access files [5] are used for database management are defined.

2. Process graph construction

The process graph structure is a functional description of the run unit using top-down decomposition techniques. The structure is hierarchical with the root node describing the function of the run unit. We call the lines between parent and child nodes, arcs. The root node is sub-divided into its logical sub-functions and these in turn are decomposed in sufficient detail to be able to specify database and other device access requests. Every resultant arc is assigned arc-values which specify probability of selection and frequency of execution. Thereafter, database and device accesses are filled in on those nodes where they occur. A logic narrative is produced for every node on the process graph.

2.1 Arc values

Every process graph element consists of a node and the arc to its parent node as follows:



The arc values a/b are associated with each arc. The first value a is such at $o < a \leq 1$ and represents the probability that the node and its sub-tree is selected during processing. The second value $b > o$ represents the number of times that the node is executed when selected. Since most arc values on a typical process graph are such that $a = 1$, $b = 1$, we assume a default arc-value of 1/1 if the parameters are omitted.

2.2 Database access primitives

Access to the conceptual schema is specified using access primitives. These have the following format:

operation (file name – 1 [D | VIA file-name-2 | S])

where operation is
- R (read)
- O (read owner)
- I (insert)
- D (delete)
- U (update)

and D is direct or random access

VIA file-name-2 is access via the owning file-name-2

S is physically sequential

The following are examples of access primitives:

1 "Get customer record for customer number = 1234"
R(customer D)

2 "Add the order to the set for the current customer"
I(order VIA customer)

Using these access primitives, the required database access operations are specified for each process graph node next to the relevant nodes.

2.3 Access to non-database disks and other devices

Similar access primitives are used to specify accesses to non-database files and devices. These have the format:

A(file/device [no])

where no is the number of physical IOs incurred. The following are examples of access primitives:

1 "Write 10 lines on the printer"
A(printer10)

2 "Access the VDU"
A(VDU 1)

3 "Write 6 IOs to the CICS status file"
A(CICS 6)

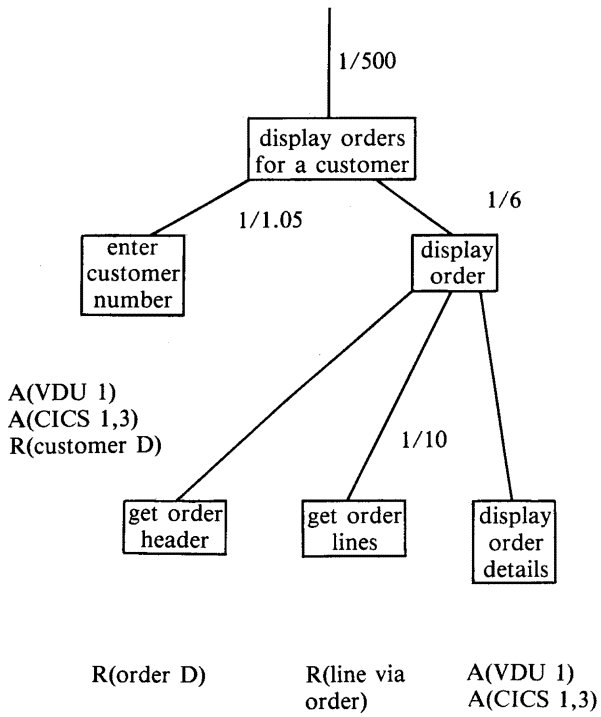
These access primitives are specified alongside the affected node.

2.4 Logic narratives

For each node on the process graph, a logic narrative or specification containing processing rules (eg. validation rules) is completed.

Completion of the above components for each process graph constitutes process graph specification.

2.5 A process graph example



We assume a 5 % error rate on entering customer numbers, 6 orders per customer and 10 lines per order. This information is contained on the conceptual schema. Software-dependent parameters are derived separately for each environment. Note that all terminal conversations (terminal in-process-terminal out) are defined. This is the basic prerequisite for fourth generation run unit specification.

3. Generating physical IOs

In [4] algorithms are presented to convert database access primitives to physical IOs for TOTAL, IMS and VSAM database management systems. In [6], automated facilities are provided to perform this operation for the above-mentioned systems as well as ADABAS and IDMS. For direct access files, [3]presents the following conversion algorithms:

read random insert and delete	$A = 1 + (R/B)$
read sequential	$A = R/0.8B$
update	$A = 1$

where R is second size, B is block size, packing density is 80 % and A is the number of physical IOs corresponding to each logical IO.

The hierarchical path to every node consists of the arcs of the parent node, grandparent node etc., to the root node of the process graph. For each node, we multiply together the a and b values of all arcs on its hierarchical path go give the factored execution of the node.

For each node, we multiply the factored execution of the node by the calculated number of physical IOs by file and device and accumulate for all files and devices. This produces the total IOs by file or device for the process graph.

Based on the total IOs for each file across all process graphs, files are allocated to disk devices so as to equalise IO traffic. The resultant specification of IOs to each device in the system is the information needed to specify a discrete class of the queueing network model used for performance prediction [7].

3.1 Calculating physical IOs of the example

Assume record sizes of 100 characters and block sizes of 1 000 characters.

File/Device	*Phycial IOs*
VDU	3525.0
CICS	4582.5
Customer	577.5
Order	3300.0
Line	33000.0

If customer, order and line are all on the same disk and CICS is on a second disk then disk 1 has 36877,5 IOs and disk 2 has 4582.5 IOs.

4. Physical database design

The prediction model produces response time predictions for the run units assuming the ultimate workload planned and that the computer system will contain other system workloads which run concurrently. These run units which are predicted to run unsatisfactorily are then used to determine the parts of the database which must be modified to improve performance. The iteration continues until a satisfactory design is achieved.

5. Summary

A technique to specify the functions of run units has been described. It is first used to perform performance predictions and subsequent physical database design. Thereafter, the programmer can take the specification directly to the terminal for implementation using a fourth generation language.

References

[1] The ADM Methodology, Comcon (Pty) Limited, 1981.
[2] Application Development System/Online, Cullinane Corporation, 1981.
[3] S. Wulf, Performance Prediction During Database System Development, Ph.D. Dissertation, Unisa, 1982.
[4] ANSI/SPARC, Report on Database Management Systems, 1975.
[5] G. Wiederhold, Database Design, McGraw-Hill, 1977.
[6] AUTO DESIGNER/PREDICTER Reference Manual, Comcon International, 1983.
[7] AUTO-CONFIGURATER Reference Manual, Comcon International, 1983.

New reading

Mathematical Foundations of Programming **by Frank S Beckman. Published by Addison-Wesley, Reading, Massachusetts, 1980. 443 pages, exercises, chapter references, index.**

THE phrase 'mathematical maturity' is used to describe a way of thinking that is required not only of computer scientists but also of programmers, not only of systems analysts, but also of managers of programming projects and thus also of dp managers. Very few people would like to attempt a definition of this desirable quality, but one may indicate a relatively painless way of acquiring it: Peruse the book under review.

In its breadth of coverage (with one omission noted below), in the topics dealt with, in the way all topics are related to actual or potential computing practice, the book contributes to the acquisition of the vitally important quality of mathematical maturity and also a quality that can only be termed algorithmic maturity. In other words, this book contributes to the appreciation of what can but also of what cannot be computed, and the various ways in which a computation may be characterised, realised, criticised, and analysed.

The author covers the area variously known as 'theory of computation' or 'meta-theory of computing', in a descriptive, almost intuitive, but mathematically sound fashion. The topics covered include: The concept 'effective'; functions and sets; recursive functions; computability and its limitations; automata and languages; and computational complexity.

In every chapter the author introduces the main topics by examples (so necessary for the development of the 'feel' for a subject) and then progresses through mathematical descriptions to applications. In the process many sidelines are touched upon which in textbooks are mentioned but then tantalisingly ignored. This is thus a book not only to be read but to be dipped into.

The practitioner in computing will find the book worth reading, not only to acquire the aforementioned maturity, but also to gain a background knowledge of the many developments in progress all over the world which go by the name 'fifth generation'. It is rather unfortunate, however, that the predicate logic receives but scant attention in the book since the Japanese fifth-generation R and D is based on Prolog-programming in logic. The research thrust in America, by the MCC, is to be based on a Lisp foundation and various aspects of this functional style are covered in the book.

For students taking a formal course at honours level in one or more topics covered by the author, the book should be recommended reading in order to put their field of study in perspective. For honours students not taking such topics, the book should be required reading.

STEF W POSTMA

Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to: Prof. G. Wiechers at:

Department of Computer Science
University of South Africa
P.O. Box 392
Pretoria 0001
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9**, 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of context-free Languages,* McGraw Hill, New York.

Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

Quaestiones Informaticae

Contents/Inhoud

*Presented at the Third South African Computer Symposium held on 14th to 16th September, 1983.