ISSN 0254–2757

QI QUÆSTIONES INFORMATICÆ

Volume 6 • Number 2 September 1988

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

QUÆSTIONES INFORMATICÆ

The official journal of the Computer Society of South Africa and of the South African Institute of Computer Scientists

Die amptelike vaktydskrif van die Rekenaarvereniging van Suid-Afrika en van die Suid-Afrikaanse Instituut van Rekenaarwetenskaplikes

Subscriptions

The annual subscription is

	SA	US	UK
Individuals	R20	$ 7	£ 5
Institutions	R30	$14	£10

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

Quæstiones Informaticæ is prepared by the Computer Science Department of the University of the Witwatersrand and printed by Printed Matter, for the Computer Society of South Africa and the South African Institute of Computer Scientists.

A Classification of Partitioning Rules for Information System Design

J Mende
Department of Accounting, University of the Witwatersrand, P O Box 1176, Johannesburg, 2000

Abstract

In designing a computer-based information system one can partition the transformation process in many alternative ways. To find the optimal partition, a designer needs explicit rules which predict the most successful grouping of functions to form modules. Three criteria for success and four types of grouping are distinguishable. This suggests a three-by-four classification scheme. When published rules are classified according to this scheme, it emerges that:

- *many categories are empty, or near-empty, and therefore constitute research problem areas*
- *apparent contradictions between rules can be resolved by explicitly stating the success criteria addressed*
- *rules which have been proposed for use in physical design can also be applied in logical design.*

Received September 1987, Accepted January 1988

Need for Partitioning Rules

The typical computer-based information system today is required to carry out a complex transformation of data into information. This transformation usually involves so many different data items and processing operations that it cannot easily be programmed as a whole. It first has to be divided into a set of smaller, more manageable *segments* [1]. However, a large system can be partitioned in very many alternative ways; so finding the best partition is no trivial problem. Therefore the designer needs *formal partitioning rules* to assist him. Many such rules have been published since 1962: for instance *flexibility* [2], *information hiding* [12], *cohesiveness* [16], *packaging* [18], *concurrency* [4] and *subsystems* [3]. However it is doubtful whether *all* the needed rules have been established to date [13]. Many more may still be discoverable. To aid the search, this article proposes a classification scheme, categorises existing segmentation rules, and identifies gaps that should be filled by further research.

Structure of the Needed Rules

What form of rule does a system designer need as an aid in making successful partitioning decisions? Several alternative answers may eventually be found; however at present only one is offered. Rules should take the form "A > B if C", where A and B represent alternative groupings of functions, > indicates preference with respect to some success criterion, and C represents the type of situation in which the rule applies. This answer emerges from the train of reasoning set out in the following four paragraphs.

With a few exceptions such as prototyping and object-oriented development, most system development methodologies today are functionally oriented. They are based upon the notion that an information system's major building blocks are *functions* – i.e. processing components whose purpose can be stated in a simple sentence and which can be coded as a block of instructions with only one entry and one exit. A majority of the functional methodologies call for two distinct design phases – logical (i.e. conceptual) and physical (i.e. technical). In many of these methodologies one of the main logical design tasks is to divide the transformation into processing *modules*, for example by drawing data flow diagrams. During the past eight years the author has observed that different student designers tend to partition the same transformation in different ways. Their modules differ substantially (cf [14]); not merely by one or two instructions, but by entire functions. This means that modules represent the outcomes of alternative decisions regarding *the grouping of functions*. For example, one designer might construct a module which includes functions A-B-C-D whereas another might include functions B-C-D-E. The existence of such alternatives indicates that the designer should *optimise* the grouping of functions into logical segments (cf [5]).

Those methodologies that include partitioning in the logical design phase typically call for further partitioning in the physical design phase. In physical design, logical modules are *packaged* to form *load units* such as programs, overlays, subroutines, etc. These physical segments can be constructed in many alternative ways, and so the designer is obliged to search for the best grouping of functions into physical segments [10]. This establishes a basic premise: *the partitioning decision seeks optimal groupings of a system's functions.*

Next, it is shown that partitioning rules should express ranking relationships between alternative groupings. If a design decision ignores the inherent characteristics of an information system's components, the system will not work as well as it might. Therefore design decisions ought to be based on laws – statements about the attributes of system components [8]. The Philosophy of Science suggests that useful laws express *relationships* between the attributes [17]. According to the basic premise, the components involved in the partitioning decision are *functions*, and their attributes are *grouping* and *success*. Ideally, the relationship between those attributes would be expressed as an equation with a numerical measure of success on the left and a formula involving grouping variables on the right. However, concept formation in the systems field has not yet reached the necessary level of precision. So the designer has to be content with the next-best, namely a *ranking* relationship of the form A > B [15].

Finally, Precedents from the natural sciences, the social sciences and Artificial Intelligence suggest that any rule is usually restricted in the scope of its applicability [6, 7, 8, 18]. So one would expect that a partitioning rule will also be restricted in scope. This implies that it should mention some condition, C, which characterises the *situation* in which it applies. Therefore we get the *normal form* of a segmentation rule:

A > B if C.

Classes of Rules

It will now be shown that the normal form involves three distinct dimensions of variability: rules can aim at different kinds of success, involve different kinds of groupings, and pertain to different situations. Therefore the three dimensions *success*, *grouping* and *situation* can serve as a basis for classifying needed partitioning rules.

First, we consider the *success* dimension. A previous paper [9] identified three independent criteria of information system success. Proceeding from a simple I-P-O model, it showed that a system can be designed to produce alternative combinations of outputs from alternative combinations of inputs using alternative combinations of processes. Users attach different values to different output combinations; so a success criterion *effectiveness* is defined as the ratio N of actual output value to maximum possible output value. Similarly suppliers of inputs attach different costs to different input combinations: so another success criterion *economic efficiency* is defined as the ratio E of actual input cost to minimal input cost. Also, some process combinations waste more resources than others: so *technical efficiency* is the ratio T of actual to minimal resource usage. A system can be effective without being economically efficient, or technically efficient without being effective, and so on. Therefore the three criteria are independent. That means the designer needs three different classes of rules:

effectiveness rules .... N(A) > N(B) if C
economic efficiency .... E(A) > E(B) if C
technical efficiency .... T(A) > T(B) if C

Next, we consider the *grouping* dimension. Yourdon and Constantine [18] identified two alternative ways of grouping any pair of functions X and Y: they may either be *associated* in the same segment (X+Y) or *dissociated* in different segments (X/Y). That means each of the three classes identified above may contain sub-classes of rules that call for:

- association of X and Y in preference to dissociation, i.e. N(X+Y) > N(X/Y), E(X+Y) > E(X/Y) and T(X+Y) > T(X/Y)
- dissociation of X and Y in preference to association, i.e. N(X/Y) > N(X+Y), E(X/Y) > E(X+Y) and T(X/Y) > T(X+Y).

Furthermore, Yourdon and Constantine identified a need for *priority* rules. Given three functions X, Y and Z, it may be better to associate X+Y rather than Y+Z; similarly it may be better to dissociate X/Y instead of Y/Z. That calls for another two subclasses:

- association of X and Y in preference to Y and Z, i.e. N(X+Y) > N(Y+Z), E(X+Y) > E(Y+Z) and T(X+Y) > T(Y+Z)
- dissociation of X and Y in preference to Y and Z, i.e. N(X/Y) > N(Y/Z), E(X/Y) > E(Y/Z) and T(X/Y) > T(Y/Z).

Therefore twelve categories of partitioning rules are identifiable:

	association		dissociation	
	X+Y > X/Y	X+Y > Y+Z	X/Y > X+Y	X/Y > Y/Z
effectiveness N	class 1.1	class 1.2	class 1.3	class 1.4
economic efficiency E	class 2.1	class 2.2	class 2.3	class 2.4
technical efficiency T	class 3.1	class 3.2	class 3.3	class 3.4

Each of these categories consists of several rules which express the same relationship, e.g. N(X+Y) > N(X/Y), but under different conditions. These categories may be further subdivisible by *condition type*, but no useful typology seems to be available at present.

Existing Rules

A literature search was carried out to find instances of the twelve categories. 32 rules were found. They were translated into the normal form [11]; then classified by the type of relationship expressed, and finally factorised by the relationship-type to yield the following list (? indicates that no rules were found).

1.1 N(X+Y) > N(X/Y) if:
- a) real time X and Y are time-critical [4]

1.2 N(X+Y) > N(Y+Z) if ....?

1.3 N(X/Y) > N(X+Y) if:
- a) environmental changes affect X but not Y [3]

1.4 N(X/Y) > N(Y/Z) if ....?

2.1 E(X+Y) > E(X/Y) if:
- a) X and Y depend on the same design decision [12]
- b) X and Y access common data [3 & 12]
- c) X and Y process data in a sequence which is subject to change [12]

2.2 E(X+Y) > E(Y+Z) if
- a) Y is more cohesive with X than Z [16]

2.3 E(X/Y) > E(X+Y) if:
- a) X and Y do not interact [3]
- b) environmental changes affect X but not Y [3]
- c) X+Y is incomprehensibly complex [3]
- d) X+Y has a complex external interface [3]
- e) X in Y is subject to change [2]
- f) X iterates Y [2]
- g) several X perform Y [2]

2.4 E(X/Y) > E(Y/Z) if:
- a) Y is less strongly coupled to X than Z [16]

3.1 T(X+Y) > T(X/Y) if:
- a) X and Y process the same data [2]
- b) X transmits a high volume of data to Y [2 & 4]
- c) X iterates Y [18]
- d) X accesses Y with high volume [18]
- e) X accesses Y with high frequency [18]
- f) X and Y are computationally intensive [4]
- g) X is executed shortly before Y [4 & 18]
- h) real time X and Y are constrained by I/O speed [4]

3.2 T(X+Y) > T(Y+Z) if:
- a) Z iterates Y which iterates X [18]
- b) Y accesses X with higher volume than Z [18]
- c) Y accesses X with higher frequency than Z [18]
- d) the time interval X-Y is shorter than Y-Z [18]

3.3 T(X/Y) > T(X+Y) if:
- a) X is optional [2 & 18]
- b) X is only used once [18]
- c) X transmits data to a sort Y [18]
- d) X receives data from a sort Y [18]
- e) X is computationally intensive but Y is not [4]
- f) X is executed at regular intervals and Y is not [4]

3.4 T(X/Y) > T(Y/Z) if ....?

Conclusions

Three major implications emerge from the foregoing. First, one observes that categories 1.2, 1.4 and 3.4 are empty, and that 1.1, 1.3, 2.2 and 2.4 are sparsely populated. These gaps suggest that many information system partitioning rules still remain to be discovered: they *represent problem areas* demanding further research. For example, the sparsity of effectiveness rules prompted the author into a research project aimed at finding some new members of class 1.3. Three were found [10]; namely N(X/Y) > N(X+Y) if:

- X and Y extract temporally independent user data types
- X collects source data which predates the user data extracted by Y
- X and Y collect source data at different frequencies, and one of the source data types is available less frequently than the user data type derived from it.

Secondly, the normal form resolves *apparent contradictions* between rules proposed by different authors. For example, Emery's rule

> "Operations which must be performed repetitively, ... in an iterative loop, should normally be defined as separate modules" [2]

seems to contradict the Yourdon-Constantine rule

> "Include in the same load unit modules connected by iterated reference" [18].

However when the two rules are cast into the normal forms 2.3f and 3.1c one can see that they address different success criteria. So the disparity is simply a conflict of objectives.

Thirdly, rules that were proposed for use in *physical design* can also be applied in *logical design*. For example, physical rules 1.1a, 2.2a, 2.4a, 3.1a-f, 3.2a-d and 3.3a-e are independent of physical considerations such as equipment, access methods, data structures etc., and can be used in drawing data flow diagrams. (In fact, the author has personally applied several of these rules in the logical design of a purchasing system). This poses serious questions about the relative roles of logical vs physical design in the system development cycle. For example, what decisions should be made in the logical design phase, and what decisions should be made in the physical phase? Which criteria should one address in logical design, and which ones in physical design? Are the rules separable into logical and physical subclasses: if so, which rules properly belong in which

category? Finally, the classification scheme might also prove useful in preparing teaching material on system design, in devising improved system development methodologies, and in developing expert systems to automate the design process.

References

[1] D.R. Chand and S.B. Yadav, [1980], Logical Construction of Software, *Comm. ACM*, **23**, 546-555.

[2] J.C. Emery, Modular Data Processing Systems Written in COBOL, *Comm. ACM*, **5**, 263-268.

[3] K. Ewusi-Mensah, [1984], Identifying Subsystems in Information Systems Analysis, *Inform. Systems*, **9**, 181-190.

[4] H. Gomaa, [1984], A Software Design Method for Real-Time Systems, *Comm. ACM*, **27**, 938-949.

[5] D.H. Hutchens and V.R. Basili, [1985], System Structure Analysis: Clustering with Data Bindings, *IEEE Trans. Software Engineering*, **11**, 749-757.

[6] P. Jackson, [1986], *Introduction to Expert Systems*, Addison Wesley, Reading/Massachusetts, p78.

[7] S. Labovitz & R. Hagedorn, [1976], *Introduction to Social Research*, 2nd ed. McGraw-Hill, New York, p6.

[8] J. Mende, [1988], A Structural Model of Information Systems Theory, *Quæstiones Informaticæ*, **6**(1), 28-32.

[9] J. Mende, [1987], Three Objectives of Information System Design, SACLA Conference, Pretoria.

[10] J. Mende, [1987], Three Packaging Rules for Information System Design, Proceedings of IV S.A. Computer Symposium, p363-370.

[11] J. Mende, [1988], Normalisation of Information System Partitioning Rules, Internal Paper, University of the Witwatersrand.

[12] D.L. Parnas, [1972], On the Criteria To Be Used in Decomposing Systems into Modules, *Comm. ACM*, **15**, 1053-1058.

[13] B. Randell, [1986], System Design and Structuring, *The Computer Journal*, **29**, 300-306.

[14] G.R. Rogers, [1983], A simple architecture for consistent application program design, *IBM Systems Journal*, **22**, 199-213.

[15] S. Siegel, [1956], *Nonparametric Statistics*, McGraw-Hill Kogakusha, London, p21-26.

[16] W.P. Stevens, G.J Stevens and L.L. Constantine, [1974], Structured design, *IBM Systems Journal*, **13**, 115-139.

[17] S. Toulmin, [1953], *The Philosophy of Science*, Hutchinson, London, p57-104.

[18] E. Yourdon and L.L. Constantine, [1979], *Structured Design*, Prentice-Hall, Englewood Cliffs/New Jersey, pp43-46, chapters 6, 7, 10, 14.

NOTES FOR CONTRIBUTORS

The purpose of the journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles and 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:

Professor J M Bishop
Department of Computer Science
University of the Witwatersrand
Johannesburg
Wits
2050

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins.

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

Manuscripts may be provided on disc using any Apple Macintosh package or in ASCII format.

For authors wishing to provide camera-ready copy, a page specification is freely available on request from the Editor.

Tables and figures

Tables and figures should not be included in the text, although tables and figures should be referred to in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Figures should also be supplied on separate sheets, and each should be clearly identified on the back in pencil and the author's name and figure number. Original line drawings (not photocopies) should be submitted and should include all the relevant details. Photographs used as illustrations should be avoided if possible. If this cannot be avoided, glossy bromide prints are required.

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 I, the number one and prime; between K and kappa.

References

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

[1] E. Ashcroft and Z. Manna, [1972], The Translation of 'GOTO' Programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

[2] C. Bohm and G. Jacopini, [1966], Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM*, **9**, 366-371.

[3] S. Ginsburg, [1966], *Mathematical Theory of Context-free Languages* McGraw Hill, NewYork.

Proofs

Proofs will be sent to the author to ensure that the papers have been correctly typeset and *not* for the addition of new material or major amendment to the texts. Excessive alterations may be disallowed. Corrected proofs must be returned to the production manager within three days to minimise the risk of the author's contribution having to be held over to a later issue.

Only orginal 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