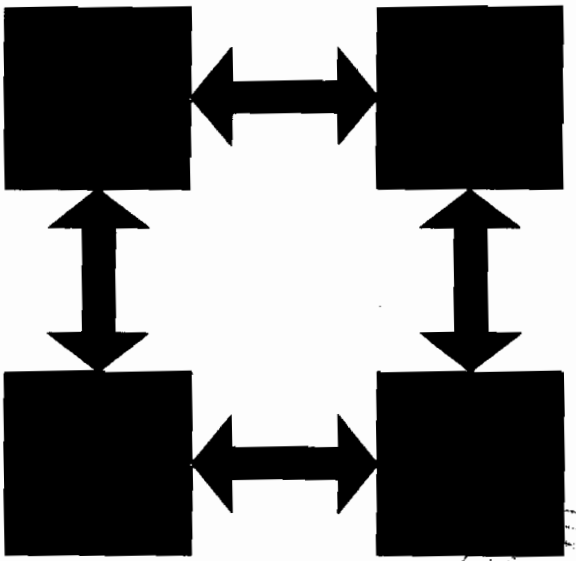

INTERNATIONAL WORKSHOP ON

COMPUTER ARCHITECTURE

sponsored by : ENSIMAG - ACM - AFCET

with the partial support of : EUROPEAN RESEARCH OFFICE

GRENOBLE - JUNE 26 - 28 th 1973

GENERAL ORGANIZATION

WORKSHOP CHAIRMEN

Prof. J. KUNTZMANN, University of Grenoble, France
S. S. HUSSON, IBM, Poughkeepsie, USA

COMMITTEES

Program Committee

Chairman : Professor L. BOLLIET, University of Grenoble, France

Dr. C.J. CONTI, IBM, Poughkeepsie
Prof. P.J. DENNING, Purdue University, USA
Dr. J.K. ILIFFE, ICL, Stevenage, UK
Dr. E.W. RIEGEL, Burroughs, USA
Prof. R. ROSIN, University of Aarhus, Denmark

Publicity Committee

Dr. O.B. MADSEN, University of Aarhus, Denmark
Dr. J. MERMET, University of Grenoble, France

Organization Committee

Prof. G. SAUCIER, University of Grenoble, France
Prof. F. ANCEAU, University of Grenoble, France

SECRETARIAT

Mme CHALAND, CII Scientific Center, Grenoble, France
Mme DUFOUR, University of Grenoble, France
Grenoble Accueil

SCIENTIFIC PROGRAM

TUESDAY JUNE 26

MORNING

8.30 REGISTRATION

9.00 PROFESSOR J. KUNTZMANN'S INTRODUCTION, OPENING REMARKS

PRESENTATION OF COMMITTE, FORMATS, ETC., BY S.S. HUSSON

9.15 PRESENTATION OF SESSION BY FIRST SESSION CHAIRMAN, PROF. L. BOLLIET

WORKSHOP INVITED PAPERS

9.30 DR. AMDHAL, ARCHITECTURE METHODOLOGY

10.15 G.G. SCARROTT, SYSTEM DESIGN OBJECTIVES FOR THE 70'S

11.00 J.B. DENNIS, THE CURRENT CHALLENGE TO COMPUTER SYSTEM ARCHITECTS

11.45 R. CASE, TRENDS IN COMPUTER ARCHITECTURE

AFTERNOON

TWO PARALLEL SESSIONS

14.30 1. TOP DOWN DESIGN
CHAIRMAN, O.B. MADSEN, UNIVERSITY OF AARHUS, DENMARK

THE LINGUISTIC DEFINITION OF COMPUTER ARCHITECTURE,
DR. H. BERNDT, SIEMENS AG, MUNICH, GERMANY

A METHODOLOGY FOR THE GLOBAL DESIGN OF INFORMATION SYSTEMS - SYSTEM SPECIFICATION AND SIMULATION LANGUAGE
PH. DARONDEAU, UNIVERSITY OF GRENOBLE, FRANCE

A FINAL HORIZONTAL ARCHITECTURE FORM FOR INFORMATION PROCESSING
M.J. MARCUS, IBM, POUGHKEEPSIE, USA

16.30 ROUND TABLE

14.30 2. FAULT TOLERANT COMPUTERS
CHAIRMAN, MME G. SAUCIER, UNIVERSITY OF GRENOBLE, FRANCE

TRENDS IN FAULT TOLERANT COMPUTER ARCHITECTURE
F.P. MATHUR, UNIVERSITY OF MISSOURI, COLUMBIA, USA

DESIGN FEATURES OF AN AEROSPACE MULTIPROCESSOR
J.S. MILLER, INTERMETRICS, CAMBRIDGE, USA

FAUTL TOLERANT COMPUTERS, TWO EXAMPLES, SAPHIR AND MECRA
MM. MERAUD, DELAMARE AND ZUSCHMIDT, SAGEM, PARIS, FRANCE

16.30 ROUND TABLE

WEDNESDAY JUNE 27

MORNING — TWO PARALLEL SESSIONS

9.30 1. MULTIPROCESSORS
CHAIRMAN, DR. ASPINALL, UNIV. OF SWANSEA, SOUTH WALES, U.K.

A DYNAMICALLY RECONFIGURABLE MULTIPLE MICROPROCESSOR
V. LESSER, CARNEGIE MELLON UNIVERSITY, USA

COMPUTING WITH MULTIPLE MICROPROCESSORS
J.V. LEVY, DEC, MAYNARD, USA

A MODEL OF PARALLEL DIGITAL SYSTEMS COMPOSED BY MUTUALLY ASYNCHRONOUS MODULES
CASAGLIA, CONTI AND VANNESCHI, CENTRO ENI, PISA, ITALY

ASYNCHRONOUS NETWORK OF SPECIFIC MICROPROCESSORS
F. DROMARD AND G. NOGUEZ, INST. DE PROGRAMMATION, PARIS, FRANCE

11.30 ROUND TABLE

9.30 2. ARRAY PROCESSORS
CHAIRMAN, R.E. MERWIN, SAFEGUARD SYSTEMS OFFICE, ARLINGTON, USA

IMPLEMENTATION OF A HIGHER LEVEL LANGUAGE ON AN ARRAY MACHINE
M.V. VINEBERG AND A. AVIZIENIS, UNIV. OF CALIFORNIA, LOS ANGELES, USA

AN ELEMENTARY ARRAY WITH PROCESSING AND STORAGE CAPABILITIES
S.F. REDDAWAY, ICL, STEVENAGE, UK

AN ARCHITECTURAL DESCRIPTION OF A PARALLEL ELEMENT PROCESSING ENSEMBLE
C. VICK AND R.E. MERWIN, SAFEGUARD SYSTEMS OFFICE, ARLINGTON, USA

11.30 ROUND TABLE

AFTERNOON — THREE PARALLEL SESSIONS

14.30 1. SOFTWARE-ORIENTED FIRMWARE
CHAIRMAN, R.L. GRIMSDALE, PROFESSOR, UNIVERSITY OF SUSSEX, U.K.

INSTRUCTION FETCH TECHNIQUES USING PROGRAM EQUIVALENCE
D.E. EASTWOOD, UNIVERSITY OF WYOMING, LARAMIE, USA

A QUEUING MODEL OF PRIORITY MULTIPROGRAMMING
I. MITRANI, UNIVERSITY OF NEWCASTLE UPON TYNE, UK

CHARACTERISTICS OF A GENERAL PURPOSE TRANSACTION PROCESSING OPERATING SYSTEM
J.M. GROSS, ICL, BRACHENELL, UK

16.30 ROUND TABLE

CONTINUE ..

14.30 2. COMPUTER DESIGN BASED ON HIGH LEVEL LANGUAGES
CHAIRMAN, DR. H. BERNDT, SIEMENS, MUNICH, GERMANY

LANGUAGE DIRECTED COMPUTER DESIGN
D.B. WORTMAN, UNIVERSITY OF TORONTO, CANADA

MICROPROGRAMMING FOR CAN CAI LANGUAGE
R.S. MCLEAN, INSTITUTE FOR STUDIES IN EDUCATION, TORONTO, CANADA

USAGE OF DESCRIPTORS IN A COBOL MACHINE
R. CHEVANCE, CII, LOUVECIENNES, FRANCE

16.30 ROUND TABLE

14.30 3. ANALYTICAL PAPERS
CHAIRMAN, MME A. RECOQUE, CII, LOUVECIENNES, FRANCE

AXIOMATIC TREATMENT OF THE SYNCHRONIZATION OF PROCESSORS
C. SCHOLTEN, PHILIPS, EINDHOVEN, THE NETHERLANDS

DESIGN OF INTERFACES AS STOCHASTIC CHANNELS WITH MEMORY
M. DEPEYROT, CII, LOUVECIENNES, FRANCE

A MODULAR COMPUTER SYSTEM ADAPTED FOR TEACHING COMPUTER ARCHITECTURE
J. HEBENSTREIT, E.S.E., MALAKOFF, FRANCE

16.30 ROUND TABLE

THURSDAY JUNE 28

8.30 GENERAL ARCHITECTURE
CHAIRMAN, J.K. ILIFFE, ICL, STEVENAGE, UK

INFLUENCE OF TECHNOLOGICAL EVOLUTION ON SYSTEMS ARCHITECTURE
F. MAISON, CII, LOUVECIENNES, FRANCE

A CLASSIFICATION OF CENTRAL PROCESSOR ARCHITECTURES
B.J. MOORE, LOGICA LTD, LONDON, UK

SUGGESTION FOR AN EXTENSIBLE CAPABILITY-BASED MACHINE ARCHITECTURE
B. LINDSAY, TECSI SOFTWARE, PARIS, FRANCE

ARCHITECTURE OF THE EDELWEISS SYSTEM
G. BLAIN, INSTITUT DE PROGRAMMATION, PARIS, FRANCE

11.00 GENERAL DISCUSSION
CHAIRMAN, PROFESSOR L. BOLLIET, UNIVERSITY OF GRENOBLE, FRANCE

INVITED PAPERS

METHODOLOGY OF ARCHITECTURE

Gene M. Amdahl

Architecture is essentially an exercise in economic tradeoffs, an aggregation of many compromises. It is sort of like designing a bibini, in that it's a challenge in attempting to encompass an impressive array of essentials with an immodestly small amount of material. It is more closely characterized as deciding what not to include rather than as by what could be put in.

Addressing methodology requires placing relative emphasis on different design and evaluation processes, as such it is quite dependent upon the architectural task one is addressing such as: whether it is a new design or an improvement of an older one; whether it is a single product or a line of products; whether it serves an established or extended market or a new one; or, whether it is aimed at a specialized market or a generalized one.

Differences in logical complexity of the architectural task are apparent between new design and improvements. One may design new releases of old products which would involve improved cost performance, reliability or serviceability.

In such case improvements would result mainly from minor changes in design, technology replacement and optimization of manufacturing processes. Or one may design new products in an established architecture requring work to reaching to market requirements for products of substantially higher performance, lower cost, greater reliability, different functional balance than existing products or to substantial progress in technological state-of-the-art. Or one could be designing new products with new functional architecture as a requiring work to reaction to changes in competitive environment, requirements for new functions of data processing components, a too rapidly growing volume of transactions in data bases,

needs for more reliable and secure data processing than can be offered in the old architecture, a desire to expand into new market areas, or the discovery of a recognizably more effective architecture for addressing one or more of the preceding.

Differences in constraints on performance of hardware realizations to be demonstrated are apparent between a single product and a line of compatible products. For a single product, one may have a clearly specified cost/performance objective in order to address a particular market segment, or one could have freedom to select that cost/performance which capitalized most effectively upon the characteristics of an available technology. For a line of products the freedoms are somewhat reduced, for the members of the line must essentially satisfy at least two requirements: that the spacing of performance ratios be rather regular and that the cost/performance be essentially constant. If this line relates to an older product line, another pair of requirements probably are also applied: that the spacing of new performance ratios are about the same as before and that the new line be spaced almost but not quite one spacing ratio higher than the old line, at least when executing the old applications. These spacing constraints are economic considerations relating to uniform coverage of the market for the first pair and relating to revenue upgrading for the new line for the second pair of constraints.

Differences in efforts needed to characterize the market appears between established and new markets. An established market has its two major aspects already defined: the functional requirements and the economic tradeoffs. The development of a new market requires extensive study, modeling and trial tests to ascertain the operational and economic viability of any proposed solution. The new market area may fall in either of two categories: activities or processes presently being performed in an alternate, probably manual, manner; and activities or processes which are postulated as providing beneficial returns if adopted. The first of these categories implies that the application resisted economic or operational solution with the previous generation of products and, hence, is probably only marginally soluble this time. The second category probably requires repeated demonstration under a variety of circumstances

to develop adequate acceptance to establish a market.

Differences in the nature of the architectural solutions and of the evaluation testing are apparent between a product specified for a specialized market as opposed to a generalized one. A particular example might be the market for solution of partial differential equations. In such a market one might be able to postulate several varieties of parallel architectures, each of which possessed particular advantage for particular situations, and each of which may possess superior advantage over a sequential arechitecture for the majority of situations. In such a market, one must pay particularly careful attention as to whether or not this is a "real" market; that is, is there also a need to perform processing that is other than parallel. If so, how much? Can the users afford both a parallel and another computer? These questions are important for the efficacy of the processor yet may end up being determined by the irregular undiciplined processing.

One must also determine the applicability of compiler techniques to the automatic utilization of the parallel componentry of the system, for the potential capability will almost certainly not be fully realized if hand coding is required.

The architecture of computing systems is by its very nature an inter-disciplinary activity. The major burden of planning, coordinating and performing architectural tasks will rest with a core architectural group. To be able to cope with problems of varying scope and importance and to function efficiently and productively within a dynamic corporation, an intensive cooperation of key professionals from major corporate departments is an ultimate requirement. On the one hand they provide the architectural group with the knowledge and expertise of specialists, and on the other hand their participation in the architectural activity gives them an intimate feeling of participation and of the overall goals of product strategy and architecture so that they are motivated and informed to carry it through in their departments.

The necessary complement to such a distributed architectural activity is the centralized core architectural group, composed of professionals with

interdisciplinary backgrounds and interests. They synthesize the expertise and state-of-the-art of different areas into structures which relate to the needs of the data processing market.

I like to think of architectural activities as divided into four categories: 1) functional architecture, which is the synthesis role; 2) evaluation of architecture and products; 3) technology evaluation and selection; and 4) design architecture which validates the realizability of the planned products.

The functional architecture, or synthesis, requires as input: the product strategy to be followed; the market research data base with analysis and projections of users' EDP functional needs; system activity measurement data with distribution and correlation of activities; programming concepts of advanced operating systems, structure of high level languages and their translators and data base and data communication structures and accessing methods; trends in computer architecture; structure, algorithms and performance of storage hierarchies; super-reliable computer architecture with field engineering and reliability statistics; technology studies and evaluations; and concurrent experimenting with trial machine organizations for each product for each major synthesis attempt.

In this synthesis phase of architecture one is most strongly tempted to succumb to the Lorelei call of the latest most innovative architectural concepts. But I would warn you most loudly to be suspicious of new ideas. I warn you, not because I dislike them, but because I become intrigued with them. Their power of attraction lies in the things that they do well. However, the bitter lesson of experience is that the true capability of a computer is far more often determined by the things it does <u>not</u> do so well than by those it does do well. A second rule to observe is to avoid adding to the computer structure additional capabilities to supplement its performance in specific applications. These additions add little or nothing to the computer's capability in general application and the pursuit of these capability additions will soon affect the machine clock cycle to the point where the net effect is an overall negative one.

The last rule I suggest is to avoid capitalizing upon a functional property unique to a single technology. It is all right to do so in a particular

product realization, but if it is done in the architecture the subsequent technology improvements may severely limit its effective life.

The evaluation of architecture and products requires the development as well as the application of a methodology, tools, models and data base for evaluating functions, reliability and cost/performance of products, product alternatives, product design alternatives and of systems using those products. The evaluation methodology must be carefully formulated to give rather complete and reasonably precise results in order to provide adequate distinction between competing solutions. System activity measurement tools must be developed which allow acquisition of data from external actual operating sources as well as from simulated equivalents. From these measurements the generation, maintenance and interpretation of a measurement data base must accrue. Proper interpretation is difficult, but distribution and correlations exhibited can yield exceedingly useful analytical tools for preliminary selection of alternatives and may also yield model transformations which permit more rapid system modeling and evaluation. In testing applications be certain to give adequate weighting to the existing developed applications. These, particularly the least exotic ones, are the economic foundation of the computer industry. If these oldies are not handled efficiently in a new architecture do not presuppose that a modest rewrite effort will properly accommodate them to a new architecture, for the costs would probably aggregate to a prohibitive level. Be careful to identify as much of the overhead and exception handling situations as you can, for people are prone to fail to include these tasks and almost never unnecessarily include them. Finally, I'll give a caution to be sure that a workload is characterized by operations to be performed rather than by division of time. This is a simple trap people so often get caught in. One hears, "We do this kind of processing half the time, so by doing it ten times as fast the system will perform overall more than five times as fast". Instead, the statement should have been, "We do this kind of processing for half the operations, so by doing it ten times as fast the system will perform overall less than twice as fast". Quite a difference!

The technology evaluation requires detailed studies in the packaging, logic design and memory areas particularly, with emphasis on manufacturing

processes, heat dissipation, maintenance, reliability and costs. Studies in wireability, loading characteristics and achievable integration levels of the logic are important. The set of technologies to be employed for the set of products in the line should be chosen, with alternates if possible, so that product verification through cost and performance can be determined.

The design architecture or architectural and product verification process requires the architects to provide at least one internal organizaticnal structure, using the selected technology, for each product in the planned line. These structures must specify critical design parameters within the proposed technologies and with these be able to meet the functional, performance, reliability and cost specifications of the products. I would visualize these organizational structures to be worked out in cooperation with the product development engineering groups involved and to be utilized by these groups as their initial design plan.

I have tried to give a methodology which turns out to have many cookbook attributes in lieu of method at times. The subject is too broad and too pervasive to permit a truly methodical approach, but continued attempts to be analytical, to apply consistent evaluation determinations, even if somewhat crude, will be valuable in the long run.

INTERNATIONAL COMPUTERS LIMITED
RESEARCH AND ADVANCED DEVELOPMENT

SYSTEM DESIGN
OBJECTIVES FOR THE 70's

by

G.G. Scarrott

June 1973

CONTENTS

1. Introduction

We can tackle the problem of specifying objectives for computer system design from two main points of view - that of a typical individual company engaged in making its living from computer business, or alternatively by considering the requirements of society as a whole. In my opinion, these two apparently contrasting views, if analysed in sufficient depth lead to similar conclusions since no large company can prosper unless its products exploit up-to-date technology to meet real social requirements. Accordingly, this paper gives a personal view of the objectives, derived by such arguments, which designers should try to achieve in systems introduced in the next few years.

Let us consider first the point of view of an individual company. Computer business is highly competitive. Day to day, or year to year decisions in running any company are properly based upon the overriding necessity to survive and make a profit. Ideally, even the choice by a computer compnay of system design objectives should be based on these some criteria. However, the time required to bring a computer system project to fruition considered against a background of rapid evolution in computer technology and users requirements is so long that the profitability criterion, although a proper meosure of success after the event, is of little value as a guide to planning. Although it is impossible to produce credible finoncial figures as a foundation for long range planning, it is quite

possible to plan rotionally by tackling the problem closer to its roots in human affairs.

Every computer user must believe when he mokes his purchose decision thot the computer will do a useful job for him.

The decisive competition between computer componies therefore occurs when systems are sold. Success in selling must be bosed on a combination of good marketing methods and a product relevont to the users needs when the sole takes ploce. System design has on oll pervoding influence on the odaptability of an information system to continuously changing market requirements. The choice of objectives for system design is therefore in effect a choice of bottle fields for the subsequent competition in the morket place.

Ill chosen objectives, even if totally achieved, may be irrelevant by the time the competition takes place; well chosen objectives, even if only partially achieved, may confer decisive advantages. Several yeors must elapse between choosing objectives and facing the consequential moment of truth, by trying to sell systems derived from thos: objectives, so that their choice should properly be based on a forecast of social requirements and technological passibilities.

This stotement of the problem although originally derived from the point of view of a typical individual company expresses the selection criteria for design objectives from the point of view of society as a whole. The choice of system design objectives must therefore be based on an analysis of the social purpose of computers ond the technologicol opportunities.

2. How Should One Specify System Design Objectives in 1973?

Since the computer industry is now some 25 yeors old, one might suppose thot the objectives for system design would be olreody well understood, even if they are not achieved; this is not so however, since the design and opplication of computers has alwoys been an open-ended exploratory process. At the present time, there is o fortunate coincidence of three separote lines of argument which together can leod to a better specification of objectives thon has been possible for previous generations of computers. The main components of these new lines of argument ore the fruition of semiconductor technology, experience of the inadequocies of present computer systems and an improved understanding of the humon purpose for which information systems are required.

2.1. Semiconductor Technology

Up to the present the only effect of semiconductor technology on computers has been that they now work well enough to do a useful job, whereas in the days of electron tube technology computers were so unreliable thot only very dedicated users were prepared to put up with them.

The moin outlines of computer design are still the same as they were before semiconductor technology was introduced. We still have o clear distinction between the programmed automaton ond the storage devices; a design decision which occurately reflected the ovailable techniques some 25 yeors ago. We moke extensive use of compiling

in the implementation of high level languages - a strategy which was introduced at least 15 years ago when the crude processing speed was very much less than can easily be achieved now.

The tactical manoeuvreability conferred by the availability of LSI has recently been used to introduce "mini-computers" Following this trend, it is now possible to change our mental image of a computer system from "a set of storage devices and the central processor" to a "team of storage and processing devices well adapted to the specific work load".

The ratio of internal processing speed to external communication speed has consequently increased so greatly that it is now appropriate to re-introduce some of the interpretive techniques which were tried but abandoned when high level languages were first introduced. Neither the concept of the "well adapted team" nor the extension of interpretive techniques has yet been adopted in commercial systems.

2.2. How Do Present Systems Perform?

Many present computing systems are certainly successful whereas others are a continual headache to their users. The main factor which determines into which of these classes each installation falls, is the rate of change of requirements. Where the users requirements change slowly, it is quite possible to evolve a total system which

is well matched to them. Where, however, the requirements change rapidly this adaptation process cannot be made to take place. The main problems in present designs can therefore be listed as follows:

2.2.1. To a very great extent it is the human users who have to adapt to their computer rather than the other way about.

2.2.2. Present systems incorporate many logically unnecessary links between sub-components. Each deliberate modification therefore necessitates so many consequential changes due to the parasitic cross links between sub-elements that it is difficult to evolve the system to match changing requirements. Even the software could now be better described as "brittleware".

2.2.3. The computing productivity of present systems is ill-defined. It is quite common for the user to discover to his surprise that the computing power provided by his installation differs greatly from that which he expected.

2.2.4. Although one might suppose that computer installations are installed primarily to help people, the communication means provided between man and machine is ill-adapted to the real requirements.

2.2.5. Attempts to devise and adopt a universal high level language have failed and can now be seen to be of doubtful value.

2.3. The Human Requirements

The earliest computers, before exploration had beenmade of their potential fields of application, were conceptually simple. Since those innocent days, the range of known computer applications has expanded enormously so that as computers became useful they also became complex and incomprehensible because it was difficult to formulate general simplifying theorems concerning computer design ond operation for the very wide range of applications.

In 1973 we are at last beginning to see our way through this jungle. We can now recognise that there is a common factor behind the diverse range of computer applications; it is people.

We may define the term "social information" to refer to the information which men use to effect their co-operative behaviour in organised groups. There are recognisable habits of human thought which mould the organisational structure of such social information largely independently af the detailed activity. We therefore now have the opportunity to design our computer systems to be well adapted to the natural behaviour of such social information.

When eventually this adaptation process is taken to its logical conclusion we should perhaps regard a computer, not as an information system in its own right but as an artificial subsystem which must be plugged into the naturally occurring human information system in order to justify its existence.

3. What Features of the Behaviour of Social Information can be Recognised?

It is useful at this point to sketch in the outline of a typical main stream computer applicotion. A computer user makes his living from some primary activity, e.g. running an insurance company or a retail store. His primary activity will be partly internal to his organisation and partly external, e.g. in the sense that he has commercial relations with suppliers and customers.

The user with advice from the computer manufacturer devises a data base which provides a dynamic description adequate for his purpose of the current state of his primary activity. The organisation of the data base, the specification of its access mechanisms and manipulation facilities is collectively termed a "Data Base Management System" and its design is a problem of great subtlety. If the user's primary activity is complex and unpredictable it is more difficult to design a DBMS for it than if the primary activity is comparatively simple. It is therefore useful to introduce the concept of "intrinsic disorder" to refer to a recognisable feature of a primary activity which largely determines the difficulty of designing and implementing a DBMS for it.

We can already recognise that a primary activity with high intrinsic disorder leads to greater emphasis on transaction rather than batch processing and the introduction of concepts such as virtual files. Up to the present, however, we have only just begun to

tackle the problem of designing a DBMS for primary activities with high disorder. Indeed, there is a regrettable tendency among computer system analysts to regard high disorder in the primary activity as evidence of incompetence rather than humanity. Fortunately, there is also a great deal of order in most primary activities. It is often possible to structure the data base so that many of the accesses to it, to record events in the primary activity, can be made by a comparatively simple address calculation. Indeed, the majority of computer applications at the present time are of this nature.

The main features of the natural behaviour of social information can therefore be summarised as follows:

3.1. The behaviour pattern is typically a combination of order and disorder. The proportion of the two, however, varies from one primary activity to another.

3.2. The ordered component of the behaviour pattern is largely represented by the widespread use of set structures in the data base organisation.

3.3. The most common set structure is the tree possibly due to the ubiquity of the hierarchical social organisation.

3.4. The disordered component arises from the fact that there is no unique tree for describing the association patterns of a typical mass of information and the trees themselves are "living" things which are frequently changed to suit varying circumstances.

3.5. Other structures such as rings, chains or networks can be regarded as combinations of superposed trees and, indeed, one may speculate that they arise in that way as a by-product of multiple trees rather than as naturally occurring structures in their own right.