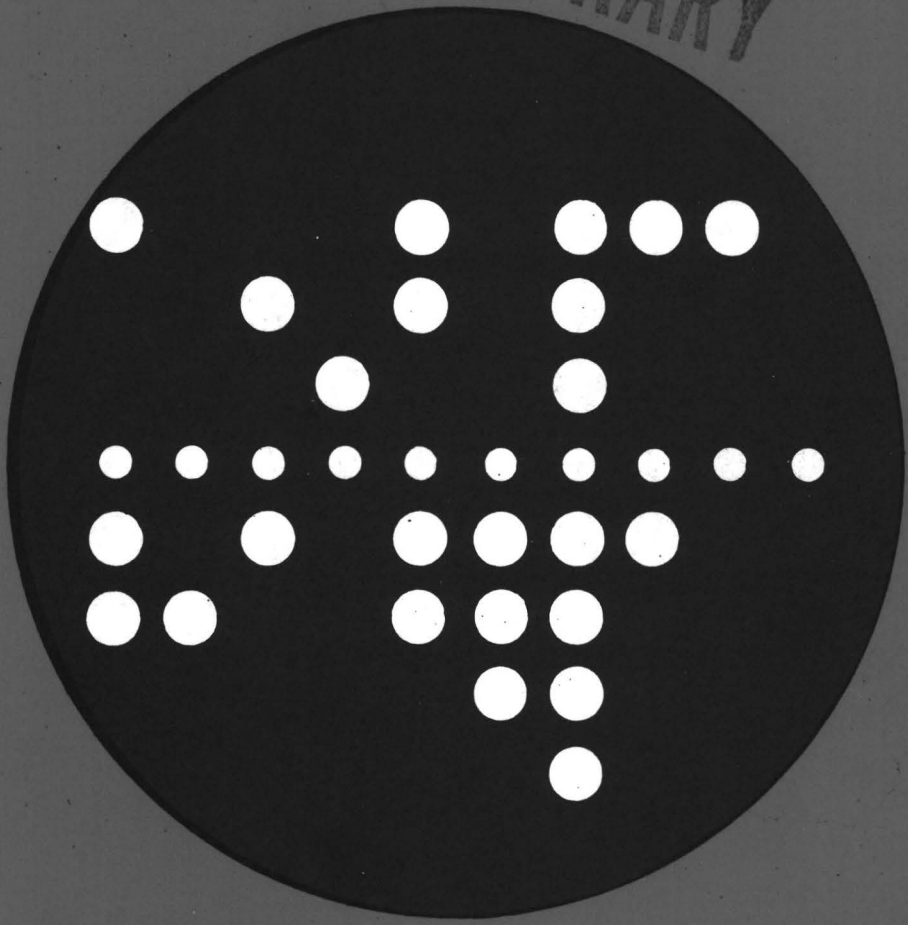
COMPUTING CENTRE NEWSLETTER

January 1979 - N°. 27



Commission of the European Communities
JOINT RESEARCH CENTRE
Ispra Establishment

CONTENTS

EDITORIAL NOTE.

The Computing Centre Newsletter is published monthly except for August and December.

It describes developments, modifications and specific topics in relation to the use of the computing installations of the Joint Research Centre, Ispra Establishment.

The aim of the Newsletter is to provide information of importance to the users of the computing installations, in a form which is both interesting and readable.

The Newsletter also includes articles which are of intellectual and educational value in order to keep the users informed of new advances in computer science topics.

Layout: Paul De Hoe (Graphical and Printing Workshop, JRC Ispra).

Administration and contact address:

Ms. A. Cambon (tel. 730)
Support to Computing
Building 36
J.R.C. Ispra Establishment
21020-ISPRA (Varese)

CHANGES IN THE NEWSLETTER.
M. Dowell.

In this, and subsequent Newsletters, you will notice a change in the format of the information and in the type-face. The Newsletter is now being produced by using the Extended Text Composer (ETC) facility which is available on the IBM 370/165. The system has already been used in the production of the three "green books" which are at present available. It is hoped that by using the ETC facility the time taken in the production of the Newsletter will be minimized. The article written by Mr. Gaggero in this issue will be the last which is produced in the style previously adopted.

At the same time modifications to the format and content of some of the information in the Newsletter are being introduced.

The list of the personnel of the Computing Centre has been modified. Information regarding the advisory service is now given as well as references to the relevant persons to contact for system and TSO registration.

The format and content of the statistics tables will be changed to give more meaningful information about computing installation utilization. It is intended to include an article which explains fully the significance of these statistics in one of this year's Newsletters. The first of the new set of statistics for 1979 appears in this issue.

There will be a stronger emphasis, in the Newsletter, on articles which relate directly to the facilities available from the Computing Centre services. This does not, however, imply a change in the policy of including articles of educational and intellectual value.

At this time, we should like to apologise for the delay in the production of this issue of the Newsletter, and also the previous issue. The delays which have occurred have been further exacerbated by the necessity to exclude articles because the information contained within them had lost its value due to the delays in publishing.

We hope that you will continue to read and enjoy the Newsletter, and that you will transmit to the members of the Editorial Board your comments on the new format and any ideas for further changes. Also, offers of articles for inclusion in the Newsletter from users of the service are still very welcome.

EDITOR'S NOTE:
This article was prepared using the Extended Text Composer (ETC).

Basic Requirements for a Modern Integrated Modular System for Engineering

G. Gaggero

Abstract

This paper is an attempt to point out the most important requirements for a modern engineering system, taking into account recent advances in hardware and software technologies.

Introduction

Computer-aided techniques for design and product realisation are becoming increasingly ambitious and involve interaction between various technological competences.

Raw computer power does not, in itself, solve the problem of computer use in engineering activities. There is the need for tools which aid the engineers in the formulation and resolution of their problems by providing a computer-aided environment in which they may carry out complex design processes.

We may call these tools an integrated engineering system.

There is no doubt that the development of an integrated system is a rather complex task, as it involves a combination of both data-processing and scientific programming tasks. The problem is further complicated when one realizes that design is not a fixed algorithm which can be programmed once and then simply used. Additions and modifications to existing analysis and design procedures can be required in order to closely fit the current design needs.

This fact imposes that the environment provided by the system is easy to use at both the implementation and user levels. It must provide complete facilities for initiating and integrating application subsystems, for defining data bases, for creating and accessing data within these data bases, and for creating and using user or problem oriented languages. In this environment, engineers must have complete control over the operation of all subsystems and the data.

Considerable efforts have been devoted during the last ten years to design and develop integrated systems for engineering. A description of the most significant implementations can be found in several reports, (1) to (9), (40).

Some of these systems are of special interest for the attention paid to the fundamental problems of modularity, programming languages, problem oriented languages, data structures, data management, and integration. Very few, however, have machine-independence as a basic design specification; this in spite of the fact that portability is a fundamental requirement for a wider use of

the systems in the engineering practice and a real impact of computers on the manufacturing and construction industry.

In addition, if one looks at the application programs developed in the frame of these systems, it must be regognized that, with a few exceptions, more emphasis was given to the analysis aspect than to the design, manufacture and administration aspects.

On the other hand, the experience provided by the existing systems is certainly of great value for any further development. In our opinion, this is true to the point that only a deep critical analysis of what has been done, together with an investigation on the rapidly changing hardware and software technology, can indicate the right way to go further.

This paper, without aiming at giving a complete answer to the problem of how should be the «modern engineering system», is an attempt to point out the most important requirements.

The Environment for the System

In this section we discuss the hardware and software aspects of the environment in which the system has to be developed and used.

In particular, we consider the machine organizations and the operating systems, as they are at present and as they can evolve, to be important issues for the design of the system.

Machine organization

When the first generation of integrated engineering programs was developed, they were primarily intended for use on batch or timeshared third generation systems.

Now technical and scientific application software is operated on many different types of installation, including

- both large and small batch systems
- timeshared multiaccess systems
- single user dedicated minicomputers
- distributed computer systems
- networks of general-purpose computers.

Each of these systems has both advantages and disadvantages and technological developments in the hardware are rapidly changing the situation.

Generally speaking, large batch systems are more cost-effective than small ones, provided they can be conveniently loaded. Remote job entry makes it possible to share one large installation among many users, but the users can be disappointed by an unsatisfactory turn-around time.

Multiaccess systems offered for some time the only cost-effective approach to interactive computing. The recent drop in the cost of memories and processors

has opened the door to single user dedicated minicomputers. They can provide a faster and more reliable response than timeshared systems which tend to become overloaded and to respond slowly at peak periods.

Large multiaccess systems are susceptible to hardware failure that block the whole system. Distributed computing, in which two or more processors are available to perform any task, offers the more satisfactory method of increasing the system reliability.

Finally large-scale computer networks can offer users a means of access to a large variety of coomputers, thus reducing the need of program transfer.

Trying to predict the hardware developments, which will play an important role in the next ten years, three major points must be mentioned:

- Semiconductor MOS storage is decreasing in price and new disk types (e.g. electronic disk) with a speed between magnetic disk and semiconductor storage will be produced. This will increase throughput by reducing the delay in paging, buffering and file handling, and the user will have a larger amount of core store available.
- Advances in integrated circuit technology will provide the minicomputers with a computing power comparable with the large computers.
- Telecommunication networks will play an increasing role.

Operating systems

The design of a system of the type we are considering can be heavily influenced by the features offered by the available operating system.

This should be avoided as the system has to be used on different installations and thus under different operating systems.

In our opinion the right approach is neither to base the design on a specific operating system (taking advantage of all the advanced and sophisticated features) nor to limit the design by using the only features that can be found as common denominator of the considered operating systems (this would result in a poor system).

One should, first, determine the requirements to be met, excluding those which cannot be supported by the present technology, and define an abstract machine with well defined functions to be used as primitives in the design phase.

The actual implementation of these primitives will be based for each operating system on the functions which are directly or indirectly available.

Of special interest is the problem of interactive use of the computer. Interaction is appropriate to a system for engineering applications for several reasons.

In first place, several applications like engineering design require an interactive process, in which the user solves his problem by repeatedly changing parameters and evaluting the results. This type of interaction requires a substantial amount of computation at each cycle and does not necessarily need to be performed under an interactive operating system. If it is, the speed of response is not a critical factor.

A second type of interactive use of the computer lies in the preparation of data prior to the evaluation process. The user makes a series of additions and modifications to the data; each step involves only a negligible amount of computation, but the speed of response is particularly important.

This type of interaction requires an interactive operating system.

Small machine operating systems generally support only a limited number of terminals, whereas large computers have multiaccess operating systems that allow for a larger number of concurrent users, but have a slow response time. Large multiaccess systems are required when a large main memory and a high computational speed are essential, or if there is the need of sharing files with many users.

A good solution could be offered by single user minicomputers linked via a network to large multiaccess computers and served by special multiaccess file systems. This could allow for an effective use of both local and remote resources.

From the user point of view it is important that the command language for controlling the local and the remote operations is the same.

Computer Networks

Computer networks constitute one of the most interesting evolutions in the use of computers. The possibility of building networks is going to be considered by a number of installations in order to add flexibility and access facilities to the existing systems. Some manufacturers and service companies do already provide network systems. PTT's start to offer new types of data transmission services at national and international scale.

At European level the COST Group 11 had prepared, mid 1971, a proposal for a cooperative venture in the field of computer networks.

The purpose of the project, (known as EIN, European Informatics Network), is to permit the sharing of resources, to allow the coordination of research, to compare ideas for national networks and promote the agreement of standards, being a model for future networks.

EIN is a heterogeneous type of network made up of a set of nodal centres. Some of them are conventional multiaccess computing systems with mainframes serving terminals while others are complete private networks of distributed computers and terminals. Hardware architecture and operating systems (CDC, CII, IBM, CTL, UNIVAC) varies significantly from one site to another. Homogeneous end-to-end protocols enable the communication, and standard interfaces allow the development of higher levels of service like terminal support, file transfer, remote job entry, distributed data bases.

Largely based on the technology developed by the EIN project, the European network for public service, EURONET, is expected to become operational in 1979. In a near future the user will have access, by means of networks. to a wide

variety of facilities provided that the software he has to use allows him to move without difficulties across machines and operating systems.

This requires the availability of portable general purpose systems which present the user a standard interface independent from the computer accessed through the network.

Programming Techniques and Languages

The first twenty years of programming efforts have been dominated by the need to rewrite or reworking software to transfer it to each new generation of computer or to computers of different makes. In addition, software was generally expensive, unreliable, difficult to understand, debug, test, and maintain.

The following facts have become increasingly clear:

- software which is reliable, portable, understandable and maintainable must be built,
- software has become more sophisticated and complex so that techniques for producing and ensuring quality in it have to be integrated into the software production process.

One can now speak of unproved programming technologies such as:

- Structured Design (modular design, composite design, etc)
- Structured Programming
- Top-Down Development (hierarchical structuring, abstract resources, etc).

These techniques proved to produce reliable systems, because they introduce a discipline which simplifies the design and the implementation phase, (10) to (25).

A modern engineering system should profit of the new techniques at two levels:

1. the design and implementation of the Basic System
2. the programming facilities offered to the developer of application subsystems.

Basic System Level

For the first level it is like to have software specialists with a good knowledge of methodologies like Structured Design (26), and of modern criteria for modular decomposition and information distribution within a system, (27) to (32), doing the design and the implementation, (37).

Coding, debugging, modification, and maintenance considerations suggests to use a well structured high level language instead of assembler for the development of the system. The efficiency can always be improved, when necessary, by rewriting few functional parts in the assembler language of the run-time computer.

Compilers for high level languages are like to be available on the most important

large scale computers and cross-compilers for minis become more and more available.

The Basic System is the logical interface between the application subsystems and the hosting system (hardware and operating system), it should present to the developer of application subsystems an abstract machine with standard functions.

Its structure must be modular so that logically independent functions are implemented in functionally and physically independent modules.

Application Subsystems Level

Coming to the second level, i.e. the programming facilities, we can consider it as the logical interface between the developer of application subsystems and the Basic System (or the abstract machine).

It will essentially consist of programming tools which allow the programmer to implement engineering procedures and to build problem oriented languages for using them.

It is out of question that the programming language (or languages) to be used at this level is a high level procedure oriented language, but which language to adopt ? This is a central issue because the language must allow the programmer access to all the facilities offered by the System, including possibly dynamic resource allocation, modules and data management, data structures and files handling, graphic representation; it must allow the programmer to express algorithms and procedures in a natural way leaving the structure and logic of the program clearly visible to the reader; it must be portable.

Looking at the languages which are presently available, we can find a number of modern very powerful languages, such as PL1, ALGOL 68, LISP, APL, PASCAL, which incorporate many of the advances made in the field of programming languages.

However, none of them satisfies all the requirements stated above; in particular, they are still familiar to a limited number of programmers and compilers exist on a limited number of computers.

One cannot avoid to take into consideration the «old» FORTRAN, which leaves much to be desired as a language, but is the most widely available scientific programming language. On many small and medium-sized machines it is the only high level language provided by the manufacturer; it has been adopted as an ISO/ANSI/ECMA Standard and it exists a large viable environment of programs written in, and programmers using FORTRAN.

Of course FORTRAN is far from providing the programmer with all the facilities and possessing all the desirable qualities we pointed out, even with the improvements introduced into the new ANSI standard (39).

Nevertheless, the approach adopted in such systems as ICES and GENESYS, (i.e. building a language which is a «superset» of FORTRAN in the sense that missing facilities are implemented by adding new statements) can, in our

opinion, provide a satisfactory solution. There is, of course, the cost of writing a pre-compiler and the overhead of using it, but there are also some important advantages, namely:

- more freedom in designing the language and complete control over it;
- possibility of coping with changes or variants in FORTRAN compilers (e.g. for different computers) without changing the programs;
- possibility of checking the prgram without compilation, possibly issuing more clear error messages and producing useful documentation.

Data Structures and Data Management

In the first stage of development of integrated systems, the algorithmic or procedural aspect received the major attention, and the supply of data was subordinated to it. Systems consisted of a set of problem-solving programs, complemented by simple data management routines.
Also the problem of data structures was essentially seen from the point of view of internal storage structure.

Now it has been fully realized that information structures should be considered at two levels. One is the logical level, as typically seen by the application-program user; the other is the internal or physical level, where one is concerned with details of how the structure is implemented.

At the logical level the user should be provided with functions to perform operations at the internal level, each of these functions operating on the physical storage structure.

Data structures are a relevant issue for:

- run-time data
- data files for passing information between communicating programs
- long-life data base.

Run-time data are likely to be handled in the main memory, but they have typically a dynamic character. It is important that the programming language provides logical structures and means for implementing internal structures. Declarative methods for defining fields as well as facilities for manipulating such structures as arrays, tables and lists are essential. A virtual storage mechanism has also proved to be very useful for applications involving large data structures.

Reference to run-time data should be as much efficient as possible, as these data are normally subject to a large amount of computation.

Data files for passing information between communicating programs do not need to have a complex organization, but it must reflect the logical and not the physical organization of data. Data structures should be addressable by symbolic names and files should be self-descriptive to make the communicating programs as much independent as possible.

Long-life data need a special consideration. Only a system which provides facilities for storing, updating and retrieving long-life data can be considered a real information system.

Now there is no doubt that a system to be used in the engineering field should not only provide means for writing analysis/design procedures but also facilities for handling large collections of inter-related data characterized by a long life and different procedures (and users) accessing them

The experience made in the field of administrative or commercial applications suggest that a data base oriented approach is adopted for engineering applications too.

As it has been pointed out in (35), a well structured and designed data base is not only an integral component of an application but can be the basis for the design and implementation of a wide variety of different programs grouped around it and, by this fact, integrated.

Data Base Management Systems provide languages for data definition and data manipulation at the logical level and functions to implement the physical data structures. Existent DBMS's are quite powerful and well developed, so that the most convenient approach seems to be that of providing the engineering system with a suitable interface to them. One should however try to design this interface independent from the actual DBMS used.

REFERENCES

1) Ross, D. (ed.) «ICES System General Description», MIT, Dept. Civil Eng., R67-49 (1967)
«ICES - An Integrated Computer Based System for Engineering Problem Solving», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972

2) «GENESYS Reference Manual», GENESYS Ltd, 2nd Ed. Oct. 1974
Allwood, R.J., Maxwell, T.O.N. «GENESYS - A Machine Independent System», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972
Shearing, B.H., Alcock, D.G. «GENTRAN», idem
Shearing, B.H. «The Implementation of GENESYS», idem

3) Deprez, G. «SYSFAP Description Générale», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972
Baar, E. «SYSFAP Organisation Interne», idem

4) «Informationssystem für Technik IST», Handbuch der Kommandosprache, Best. Nr. D14/40312, Programmierhandbuch, Best. Nr. D14/40313, Siemens AG, München
Pahl, P.J. «ISB - Das Informationssystem für das Bauwesen», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972

5) Schlechtendahl, E.G. «Grundzüge des integrierten CAD-Systems REGENT», Angewandte Informatik 11/76 pp. 490-496 (1976)

6) Hatfield, F.J., Fenves, S.J. «An Information Organizer for Coordinating Modular Programs», Univ. of Illinois, CESL Rep. 5 (1970)

7) Kayser, O. «MIRIAM - A Matrix Based System for Solving Engineering Problems», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972

8) Becque, A.J. «LPR - An Integrated Program and Data Management System for Engineering Applications», Doctoral proefshrift, Amsterdam, 23.3.1977

9) Lopez, L.A. «POLO - A Supervisor for Integrated Systems Development», Colloque International sur les Systèmes Intègres en Génie Civil, Liège, 1972

10) Naur, P., Randell, B. (eds.) «Software Engineering», Report on the Conference sponsored by NATO S.C., Garmisch, Germany, Oct. 1968

11) Liskov, B.H. «Guidelines for the Design and Implementation of Reliable Software Systems», The MITRE Corporation, ESD-TR-72-164, Febr. 1973

12) Miller, E.F., Jr., Lindamood, G.E. «Structured Programming: Top-down Approach», Datamation, Dec. 1973, pp. 55-57

13) Buxton, J.N., Randell, B. (eds.) «Software Engineering Techniques», Report on the Conference sponsored by NATO S.C., Rome, April 1970

14) Tou, J.T. (ed.) «Software Engineering», Proceedings of the Third Symposium on Computer and Information Sciences, Miami Beach, Florida, Dec. 1969, Academic Press, New York, 1970

15) Böhm, C., Jacopini, G. «Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules», Communications ACM, May 1966, pp. 366-371

16) Dijkstra, E.W. «GOTO Statement Considered Harmful», Communications ACM, Vol. 11, No. 3, March 1968, pp. 147-148

17) Dijkstra, E.W. «Notes on Structured Programming». T.H.E. Report No. EDW-248, 70-WSK-03, 2nd Edition, April 1970

18) Miller, E.F., Jr. «Extensions to FORTRAN and Structured Programming - An Experiment», General Research Corp., RM-1608, March 1972

19) Mills, H. «Mathematical Foundations for Structured Programming», IBM Corp., Federal Systems Div., Report No. RSC-71-5108, 1971

20) Mills, H. «Top Down Programming in Large Systems» in Debugging Techniques in Large Systems, R. Rustin ed., Prentice Hall, 1971

21) Wirth, N. «Program Development by Stepwise Refinement», Communications ACM, April 1971, Vol. 14, No. 4

22) Judd, R. «Practical Modular Programming», the Computer Bulletin, Jan. 1970, pp. 4-7

23) Wirth, N. «Systematic Programming», Prentice Hall, Englewood Cliffs, 1973

24) Wirth, N. «The programming Language PASCAL», Acta Informatica, 1, 1971, pp. 35-63

25) Naur, P. «Programming by Action Cluster», BIT 9, 1969, pp. 250-258

26) Stevens, W.P., Myers, G.J., Constantine, L.L. «Structured Design», IBM

System Journal, Vol. 13, No. 2, 1974, pp. 115-139

27) Dennis, J.B. «Modularity», Advanced Course on Software Engineering Bauer F.L. (ed.), Springer-Verlag, 1973

28) Parnas, D.L. «On the Criteria to be Used in Decomposing Systems into Modules», Communication ACM, 15, 12, Dec. 1972, pp. 1053-1058

29) Zilles, S.M. «Procedural Encapsulation: A Linguistic Protection Technique», SIGPLAN Notices, 8, 9, Sept. 1973, pp. 142-146

30) Parnas, D.L. «Information Distribution Aspects in Design Methodology», IFIP 1971

31) Frost, D. «Designing for Generality», Datamation, Dec. 1974, pp. 59-61

32) Myers, G.J. «Composite Design: The Design of Modular Programs», Technical Report TR00.2406, IBM, Poughkeepsie, N.Y., Jan. 1973

33) Whitten, D.E., de Mainde, P.A.D. «A Machine and Configuration Independent Fortran: Portable Fortran (PFORTRAN)», IEEE Transactions of Software Engineering, Vol. SE-1, No. 1, March 1975

34) «Computer Languages in Buildings», Proceedings of a Symposium by Correspondance, CIB (International Council for Building Research Studies and Documentation), W52, Draft Report ETI-76.1258

35) Smits, H.P. «An ICES/IMS Interface», ICES Journal, March 1977, pp. 150-155

36) Fujii, Y. «Engineering Data Base Package Based on IMS», ICES Journal, March 1977, pp. 228-256

37) Rutten, H.S. «The Seeming Contrariety Between Theory and Practice: Viz. The Meagre Usefulness of Existing Application Software», First European ICES Users Group Conference, Delft, 1977

38) Schlechtendahl, E.G. «Potential Design Characteristics of a 'Future ICES'», First European ICES Users Group Conference, Delft, 1977

39) American National Standards Institute «ANS Programming Language FORTRAN x3.9-1978»

40) Gaggero, G. «An Introduction to Modular Systems», JRC Computing Centre Newsletter, No. 23, July 1978

UTILISATION DE L'ORDINATEUR DU CENTRE DE CALCUL EN 1978.
J. Pire.

Dans le numéro 18 de février 1978 nous avions fait une brève analyse de l'évolution de l'utilisation de l'ordinateur IBM 370-165 du Centre de Calcul au cours de la periode 1973-1977. Dans un article du numéro de novembre 1978 nous avons parlé du développement qu'a connu l'utilisation de T.S.O. La question qui peut se poser est d'analyser la répercution que ce développement peut avoir eu sur l'utilisation 'BATCH'.
Disons tout de suite que les travaux BATCH demandés au cours de 1978 ont constitué une charge plus lourde qu'au cours des années précédentes, néammoins certains 'details' ont varié sensiblement. La table I ci-dessous fourni pour 1978 les paramètres qui avaient fait l'objet d'un graphique couvrant 5 années dans le numéro 18.

1)	Heures CPU en mode problème batch	:	1520
2)	I/O disques batch	:	221.5×10^6
3)	I/O bandes batch	:	47.0×10^6
4)	Nombre de travaux présentés	:	92.7×10^3
5)	Lignes imprimées	:	233.1×10^6
6)	Cartes lues	:	21.5×10^6
7)	Cartes perforées	:	1.4×10^6

Par job en moyenne

a)	Nombre de lignes imprimées	:	3.05×10^3
b)	Nombre de cartes lues	:	230
c)	Nombre de cartes perforées	:	15
d)	Temps C.P.U. (Heures)	:	16.4×10^{-3}
e)	Nombre d'I/O disques	:	2.38×10^3
f)	Nombre d'I/O bandes	:	507

Ces valeurs peuvent être reportées sur les graphiques du n.18.

Les constatations les plus évidentes sont les suivantes:
1) Le temps C.P.U. utilisé a continué à croître (5% par rapport a 1977)
2) Le nombre d'I/O disques a repris à augmenter
3) Le nombre d'I/O bandes commence à diminuer
4) Le nombre de JOB BATCH a diminué de 10%
5) Le nombre de lignes imprimees a commencé á diminuer
6) Le nombre de cartes lues a baissé de l'ordre de 30%
7) Le nombre de cartes perforées est diminué de 30%

Au sujet des cartes lues nous devons encore ajouter que 5% environ des cartes lues proviennent des SUBMIT de T.S.O. Et ne passent par conséquent pas par le lecteur de cartes.
Nous sommes donc en bonne voie d'élimination de la carte perforée. Voici en outre quelques données concernant les distributions statistiques de quelques caractéristiques des travaux présentés.

Travaux produisant moins de N lignes imprimées

N =	1000	2000	4000	8000	16000	32000	64000	128000
% =	53.89	13.77	12.40	10.02	6.02	2.56	1.16	0.18
Cum =	53.89	67.66	80.06	90.08	96.10	98.66	99.79	99.97

Travaux comportant moins de N cartes à lire

N =	100	200	400	800	1600	3200	6400
% =	57.80	13.28	11.87	9.03	5.28	2.28	0.23
Cum =	57.80	71.08	82.95	91.98	97.26	99.54	99.77

Travaux ayant eu un 'turn-around time' inférieur a T min.

T =	15	30	60	120	240	480	480
% =	30.61	16.50	17.17	14.83	12.34	6.27	2.28
Cum =	30.61	47.11	64.28	79.11	91.45	97.72	100.00

La plupart des travaux ayant eu un turn-around time supérieur à 480 min (8h) sont des travaux devant être expédiés en R.J.E. et pour lesquels la station était fermée au moment où le travail était prêt à être transmis.

Travaux nécessitant moins de T min. d'elapsed time

T =	1	2	4	8	16	32	64	128
% =	29.92	19.87	18.76	14.99	8.99	5.55	2.22	0.59
Cum =	29.92	49.79	68.55	82.54	91.53	97.08	99.30	99.89

Travaux nécessitant moins de N kilobytes (pourcents du nombre de travaux)

N =	100	200	300	400	600	800	1000	1200
% =	24.40	14.39	22.51	13.57	3.29	0.62	0.94	0.31
Cum =	24.40	58.79	81.30	94.87	98.16	98.78	99.62	99.93

Travaux nécessitant moins de N kilobytes (pourcents de l'elapsed time total)

N =	100	200	300	400	600	800	1000	1200
% =	7.64	22.07	27.29	26.06	8.70	2.39	3.89	1.53
Cum =	7.64	29.71	57.00	83.06	91.76	94.15	98.04	99.57

Enfin à titre de considération générale concernant la fiabilité du matériel, nous avons connu quelques arrêts graves au début de l'année, mais la situation s'est fortement améliorée et aucun arrêt important n'a été enregistré au cours des derniers six mois.
Espérons que ceci continuera.

NOTE TO ALL TSO USERS.
D. Koenig.

Because of a serious misuse of the TSO system by persons executing programs with high CPU time on TSO in the foreground region, it has become necessary to introduce the following restriction.

From 26th. March 1979 every individual TSO session will have 3 minutes of CPU time allocated. This should be sufficient for normal TSO usage (editing, compiling of small programs, test runs of small problems) during one session, even if the session is hours long. If the CPU time is exceeded within the duration of one session, the session will be automatically cancelled by the system.

The user will receive the message -

IEF450I userid ..... ABEND S322 TIME=

IKJ56470I userid LOGGED OFF TSO AT .....

Unfortunately it is not possible to give a warning to the users before this time limit is exceeded. However, users may monitor their CPU usage during a session by typing the TIME command when in TSO command mode (see TSO Command Language Reference Manual [GC28-6732-2] page 253).

Statistics of computing installation utilisation.
Report of computing installation exploitation
for the month of January 1979.

	YEAR 1978	YEAR 1979
General		
Number of working days	22 d	22 d
Work hours from 8.00 to 24.00 for	16.00h	16.00h
Duration of scheduled maintenance	24.08h	25.83h
Duration of unexpected maintenance	18.43h	9.41h
Total maintenance time	42.51h	35.24h
Total exploitation time	313.49h	316.76h
CPU time in problem mode	132.07h	166.98h
Batch Processing		
Number of jobs	8952	7440
Number of cards input	2240000	1407000
Number of lines printed	26800000	23545000
Number of cards punched	167000	183000
CPU time	130.15h	157.81h
Number of I/O (Disk)	19022000	21818000
Number of I/O (Magnetic tape)	3749000	3194000
T.S.O		
Number of LOGON's	685	1865
Number of messages sent by terminals	23500	101000
Number of messages received by terminals	98900	515000
CPU time	1.92h	8.14h
Number of I/O (Disk)	337968	1759000
Connect time	250.98h	1012.00h
IMS		
Total time service is available	-	257.00h
CPU time	-	1.03h
Number of I/O (Disk)	-	280000

Utilisation of computer centre by objectives and appropriation accounts for the month of January 1979.

		IBM 370/165 equivalent time in hours
1.20.2	General Services - Administration - Ispra	35.97
1.20.3	General Services - Technical - Ispra	1.22
1.30.3	Central Workshop	3.90
1.30.4	L.M.A.	0.01
1.90.0	ESSOR	17.24
1.92.0	Support to the Commission	1.35
2.10.1	Reactor Safety	181.83
2.10.2	Plutonium Fuel and Actinide Research	52.35
2.10.3	Nuclear Materials	8.38
2.20.1	Solar Energy	0.06
2.20.2	Hydrogen	0.43
2.20.4	Design Studies on Thermonuclear Fusion	5.28
2.30.0	Environment and Resources	14.80
2.40.0	METRE	0.24
2.50.1	Informatics	27.07
2.50.2	Training	-
2.50.3	Safeguards	7.85
	TOTAL	357.98
1.94.0	Services to External Users	10.88
	TOTAL	368.86

BATCH PROCESSING DISTRIBUTED BY REQUESTED CORE MEMORY SIZE

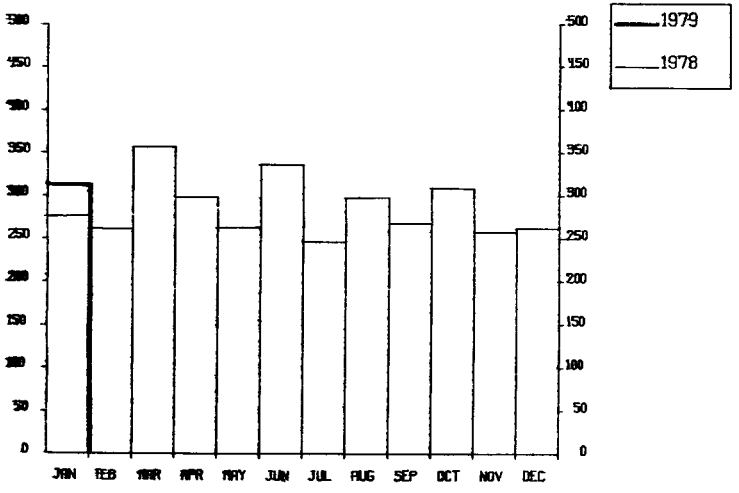
	100	200	300	400	600	800	1000	1200	1400	>1400
No. of jobs	1813	2210	1508	1012	251	144	83	24	-	13
Elapsed time	45	168	216	182	60	67	28	13	-	0.3
CPU time	2.5	20.5	56.0	33.9	16.6	15.4	7.0	3.6	-	0.0
"Equiv" time	15.1	53.7	89.1	72.2	29.1	31.7	13.8	7.3	-	0.3
"Turn" time	0.5	2.0	2.7	2.7	4.2	6.5	3.7	3.0	-	-
I/O (disk)	1345	4351	4585	5157	1769	2319	981	525	-	39
I/O (tape)	1059	933	345	742	53	21	1	3	-	-

NOTE.
All times are in hours.
"Equiv" means equivalent.
"Turn" means turn around.
All I/O transfers are measured in 1000's.

PERCENTAGE OF JOBS FINISHED IN LESS THAN

TIME	15mn	30mn	1hr	2hrs	4hrs	8hrs	1day	2day	3day	6day
%year 1978	41	59	77	90	96	99	99	99	100	100
%year 1979	32.0	43.6	56.6	69.5	83.9	94.2	99.1	99.9	100	100

HISTOGRAM OF TOTAL EQUIVALENT TIME(HRS)



Projected Total for 1979 = 3748 hours.
Total for 1978 was = 3424 hours.

REFERENCES TO THE PERSONNEL/FUNCTIONS OF THE COMPUTING CENTRE.

			Room	Tele.
Manager of The Computing Centre		J.Pire		
Responsible for User Registration	Ms.	G.Rambs		
Operations Sector				
Responsible for the Computer Room		P.Tomba		
Substituted in case of abscence by:		A.Binda		
Responsible for Peripherals		G.Nocera		
Systems Group				
Responsible for the group		D.Koeniq		
Substituted in case of abscence by:		P.A.Moinil		
Responsible for TSO Registration		C.Daolio		
Informatics Support Sector				
Responsible for the Sector		G.Gaqqero	1874	787
Secretary	Mrs.	G.Hudry	1873	787
Responsible for User Support		H.de Wolde	1883	1259
General Inf./Support Library	Mrs.	A.Cambon (See Note 2)	1871	730
Advisory Service/List of Consultants(See Note 1)			1870	730

A.Inzaghi A.A.Pollicini
H.I. de Wolde
R.Meelhuysen M.Dowell

NOTE 1. The advisory service is available in the same room as the Computing Support Library(room 1870). Exact details of the advisory service times for a specific week can be found at the head of any output listing(for that week).
Any informatics problem may be raised. However, the service is not designed to help users with problems which are their sole responsibility. For example, debugging of the logic of programs and requests for information which can easily be retrieved from available documentation.
If necessary, other competent personnel from the informatics division may be contacted by the consultant but not directly by the users.
The users should only contact the person who is the consultant for that specific day and only during the specified hours.
Outside the specified hours general information may be requested from Mrs. A. Cambon(see note 2) in the Computing Support Library.

NOTE 2. Mrs. Cambon is at present replaced by Mrs. C La Cognata.

HOW TO BECOME A REGULAR READER OF THE NEWSLETTER.

Persons interested in receiving regularly the "Computing Centre Newsletter" are requested to fill in the following form and send it to :-

Ms. A. Cambon
Support To Computing
Building 36
Tel. 730.

NAME .....

ADDRESS .....

.....

.....

TELEPHONE .....