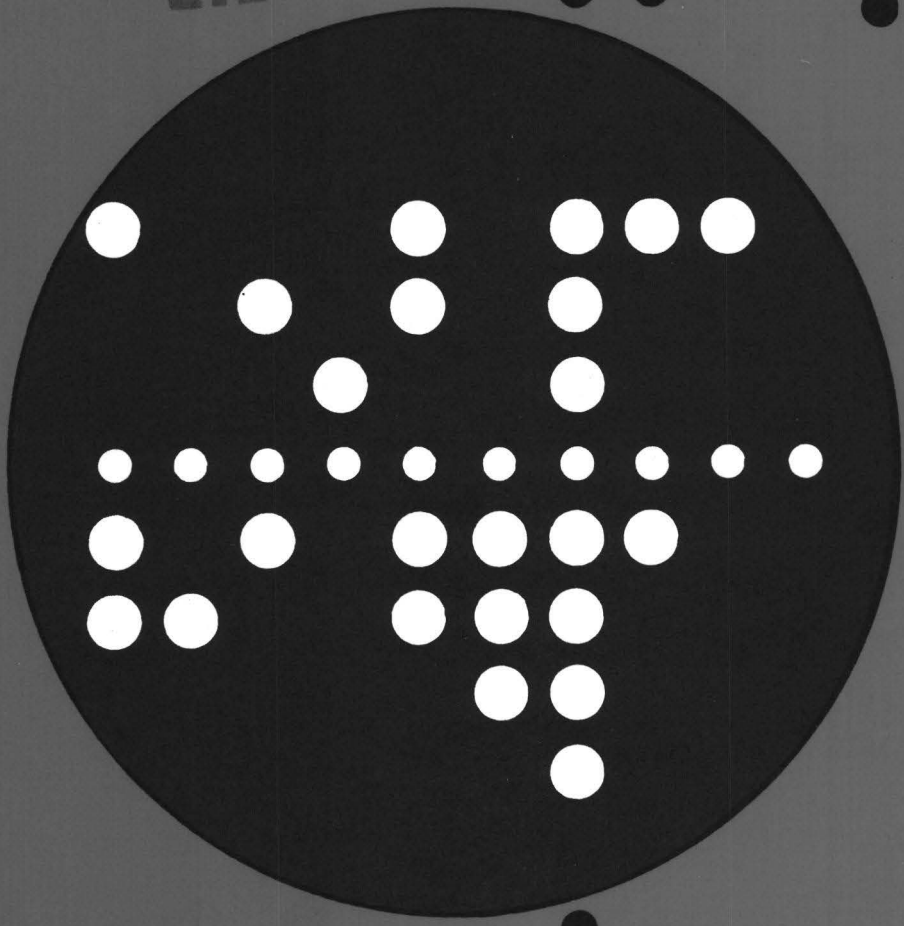
Commission of the European Communities

Joint Research Centre - Ispra

Computing Centre Newsletter



February 1978 No 18

Contents

Note of the Editor

The present Newsletter is published monthly except for August and December.

The Newsletter includes:

- Developments, changes, uses of installations
- Announcements, news and abstracts on initiatives and accomplishments.

The Editor thanks in advance those who want to contribute to the Newsletter by sending articles in English or French to one of the following persons of the Editorial Board.

Note de la Rédaction

Le présent Bulletin est publié mensuellement excepté durant les mois d'août et décembre.

Le Bulletin traite des:

- Développements, changements et emploi des des installations
- Avis, nouvelles et résumés concernant les initiatives et les réalisations.

La Rédaction remercie d'avance ceux qui veulent bien contribuer au Bulletin en envoyant des articles en anglais ou français à l'un des membres du Comité de Rédaction.

Editorial Board / Comité de Rédaction

Computing Centre References

		Room	Tel.
Manager	J. Pire	1816	732
Adjoined	G. Gaggero	1874	787
Computer Room	P. Tomba	1857	797
Adjoined	A. Binda	1857	797
Peripherals	G. Nocera	1825	767
System Group	D. Koenig	1839	742
Adjoined	P.A. Moinil	1841	704
Informatics Support	G. Gaggero	1874	787
o General Information	G. Hudry	1873	787
o Program Information Service	G. Gaggero	1874	787
Adjoined	S. Leo Menardi	1884	721
o Graphics and Support to Users	H.I. de Wolde	1890	753
Adjoined	A. Pollicini	1882	743
Application Packages	A. Inzaghi	1887	755
Programming Languages	C. van den Muyzenberg	1848	781

Editor : Jean Pire
Layout : Paul De Hoe
Graphical and Printing Workshop, JRC Ispra

Evolution de l'utilisation de l'ordinateur du Centre de Calcul 1973 - 1977

J. Pire

Il y a maintenant plus de cinq ans que l'ordinateur IBM 360-165 est en service au Centre de Calcul d'Ispra. Il peut être intéressant de suivre l'évolution de son utilisation au cours des années 1973 à 1977.

Nous avons pensé que quelques graphiques seraient plus parlants qu'un long discours.
Chacun des brefs paragraphes ci-dessous n'est qu'un commentaire de l'un des graphiques publiés en fin de cet article.

1. *Heures CPU en mode problème batch* (fig. 1.a)

   Le facteur d'accroissement exponentiel annuel moyen est de 1.24.
   L'énorme accroissement constaté en 1976 est à mettre en corrélation avec l'augmentation de la taille de la mémoire. Entre 1973 et 1975 le facteur d'accroissement exponentiel était de 1.20.

2. *Heures CPU en mode problème conversationnel* (fig. 1.b)

   L'accroissement exponentiel annuel moyen est de 1.61, c'est cependant seulement au cours de 1977 que l'utilisation a fortement augmentée (double par rapport à 1976) par suite de la mise en service de IMS. Une augmentation spectaculaire peut être attendue en 1978 avec l'introduction de T.S.O.

3. *I/O disques* (fig. 2.a)

   Le nombre d'I/O disques a connu une pointe en 1976 par suite d'un usage intensif du système CORIG de la part de l'administration.
   En 1978 l'augementation de l'acivité due à T.S.O. sera probablement compensée en partie par une nouvelle diminution de l'utilisation de CORIG.

4. *I/O bandes* (fig. 2.b)

   Le nombre d'I/O bandes est en augmentation, mais semble s'être stabilisée en 1977. Le rapport des travaux demandant le montage des bandes et des travaux sans montage est en nette diminution. En 1973 il était de 0.33, il est maintenant environ 0.25.
   Vu l'adjonction de mémoires à accès direct, nous espérons que ce rapport diminuera encore.

5. *Nombre de travaux présentés* (fig. 3.a)

L'augmentation du nombre de travaux est constante, mais faible: en moyenne de 6000 travaux par an.

6. *Lignes imprimées* (fig. 3.b)

Le nombre de lignes imprimées a augmenté spectaculairement en 1974, lors de l'introduction du système CORIG et en 1976 lors de l'exploitation intensive du même système.
L'introduction de systèmes conversationnels devrait stabiliser ce paramètre sur la valeur de 1977 sinon le faire décroître dans les années à venir.

7. *Cartes lues* (fig. 3.c)

Le nombre de cartes lues est à peu près stable depuis 1974 alors qu'il était en forte croissance au cours des années précédentes. Il diminuera probablement dans l'avenir.

8. *Cartes perforées* (fig. 3.d)

Le volume des cartes perforées par l'ordinateur est en constante diminution. Nous espérons que ce phénomène continuera et même s'accentuera encore.

Caractéristiques des travaux présentés

9. *Nombre de lignes imprimées par travail* (fig. 4.a)

Après une croissance forte en 1974, cette valeur s'est stabilisée à environ 2850 lignes par travail.

10. *Nombre de cartes lues* (fig. 4.b)

Il est en constante diminution, de plus en plus les cartes sont mémorisées sur disques; il est passé de 350 à 290 cartes par travail; la sécurité des travaux en est accrue.

11. *Nombre de cartes perforées* (fig. 4.c)

Il est diminué de près de la moitié entre 1973 et 1977. La diminution a été de 33 à 18 cartes en moyenne par travail.

12. *Le temps CPU en mode problème* (fig. 5.a)

Il a presque doublé en cinq ans.

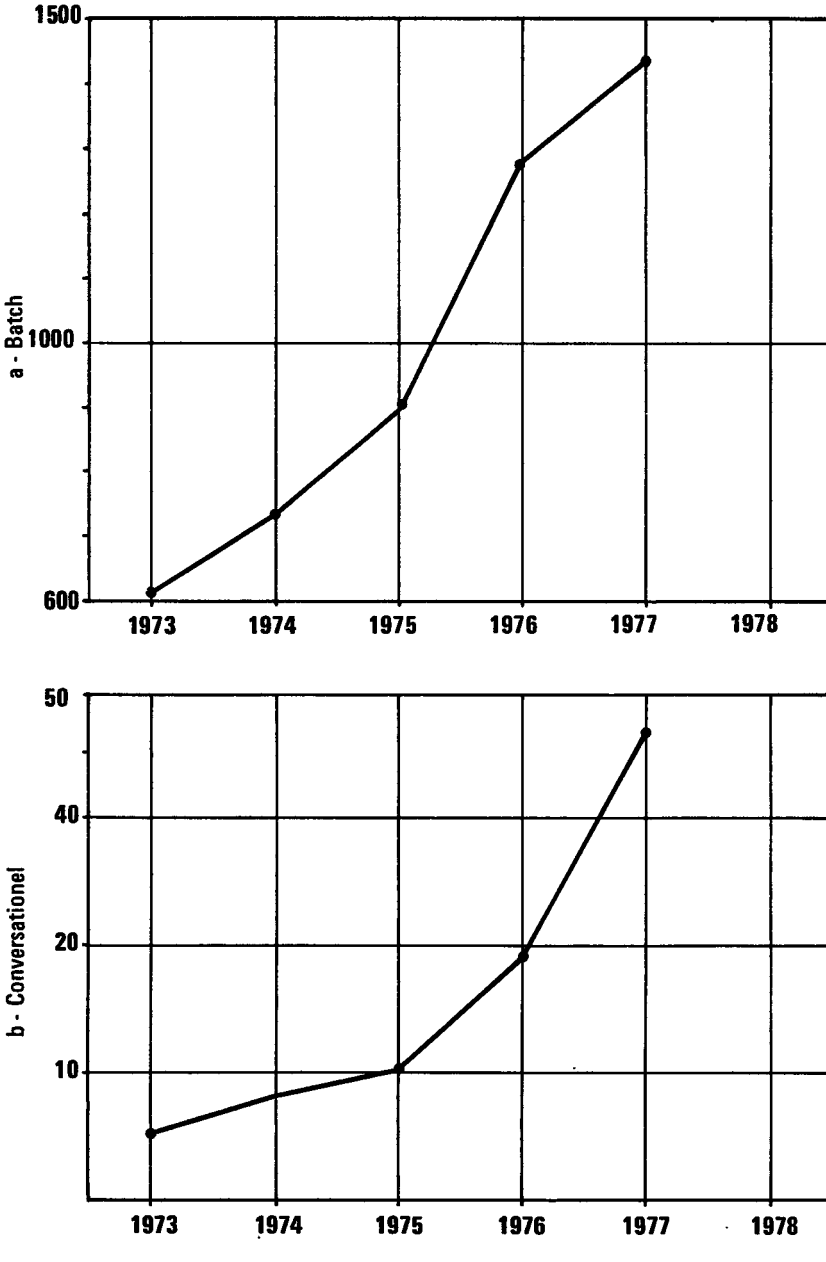


Fig. 1 — Heures C.P.U. en "problem mode"

13. Le nombre d'I/O disques (fig. 5.b)

Après une forte augmentation en 1974 (CORIG) il est en diminution, car les travaux scientifiques augmentent et les travaux administratifs diminuent.

14. Le nombre d'I/O bandes par travail (fig. 5.c)

Il est en légère augmentation; la proportion des travaux exigeant des bandes étant en diminution, nous en concluons que les bandes requises contiennent des fichiers plus volumineux que ceux utilisés dans le passé.

15. Temps équivalent par travail (fig. 5.d)

Il augmente régulièrement; non seulement le nombre des travaux augmente, mais aussi leur ampleur.

16. L'importance du temps CPU par rapport au temps équivalent (fig. 6. - 8)

Il est en augmentation depuis 1974. Après l'explosion des travaux administratifs, les travaux scientifiques reprennent le dessus.

17. Occupation mémoire d'un travail moyen (fig. 6.a)

Nous voyons que la tendance à l'augmentation s'est nettement accentuée en 1976 par suite de l'agrandissement de la mémoire centrale.
Une analyse plus fine de la tendance montre qu'elle ne fait que s'accentuer.

18. Perforation manuelle de cartes (fig. 6.b)

La quantité de cartes perforées manuellement a touché son maximum (environ 800.000 cartes) en 1973, elle diminue régulièrement jusqu'en 1977 (625.000).
Depuis 1975 les variations sont faibles: environ 380.000 cartes pour l'administration et 200.000 cartes pour les travaux scientifiques.
Nous espérons que la décroissance reprendra en 1978.

19. Utilisation du Centre de Calcul par les différents objectifs

Une petitte analyse de la Table 1 montre que le travail pour l'administration en 1977 n'a plus représenté que 16% du total contre plus de 30% en 1975.
Les travaux pour les tiers quoique constituant une ressource non négligeable ne représentent que 8% des prestations totales.

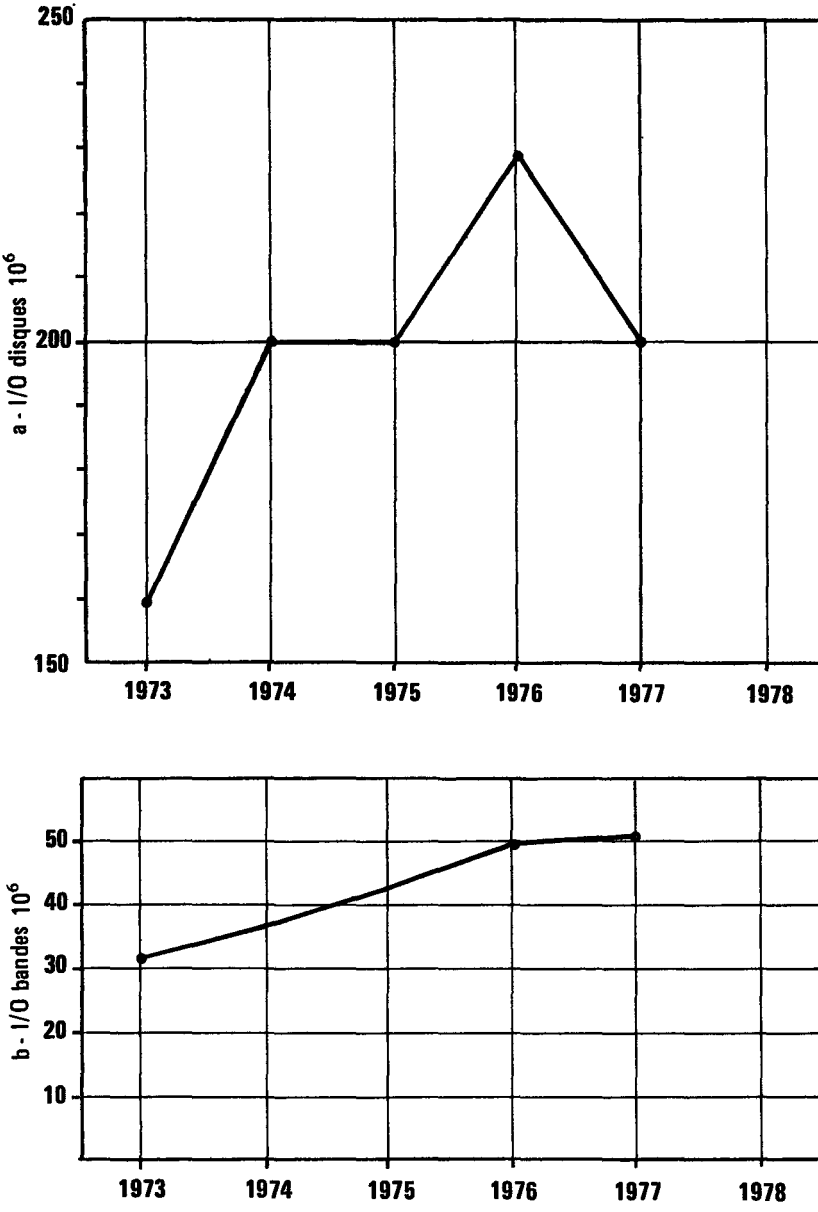


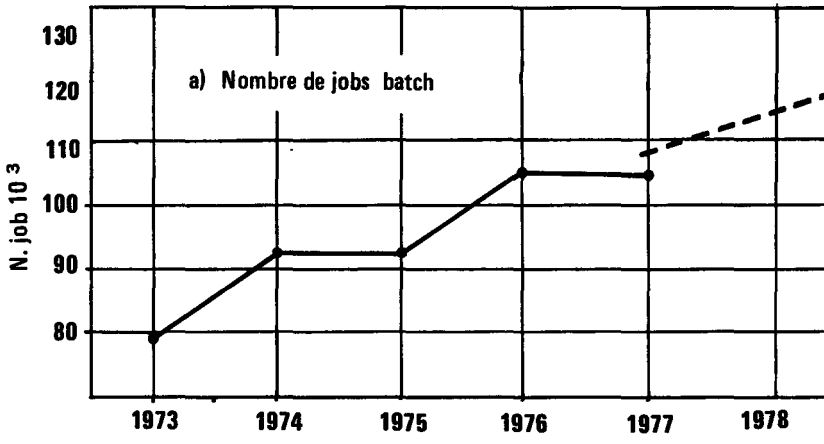
Fig. 2 — I/O en "problem mode batch"

TABLE I : Utilisation des installations de calcul

Projets/Comptes d'Affectation Contrats	IBM 370/165 Janv. - Déc. 1977
Sureté des réacteurs	1.200,3933
Comb. au Pu & Act.	38,9332
Déchets nucléaires	29,9069
Energie solaire	9,4351
Hydrogène	1,4658
Etudes fusion	25,0923
Environnement & ressources	222,2175
M.E.T.R.E.	39,5052
Informatique	532,2901
Gestion mat. fiss.	12,7417
Serv. gén. admin.	461,5748
Serv. gén. techn.	15,5119
L.M.A.	0,2127
ESSOR	106,6061
Support Commission	79,3654
Contrats avec tiers	227,7090
Total	**3.002,9610**

Conclusions

Les demandes de travaux présentées au Centre de Calcul au cours des cinq dernières années sont en constante augmentation tant en nombre qu'en durée d'exécution et en taille de mémoire requise.
Les opérations périphériques requises ont atteint un plafond ou sont en diminution.
L'introduction des facilités conversationnelles, dont l'utilisation s'intensifie de façon très satisfaisante, accentuera encore très probablement ces deux tendances.
C'est à l'usage à but scientifique et technique de l'ordinateur que ces tendances sont dues, l'utilisation à des fins administratifs est par contre en heureuse diminution.


a) Nombre de jobs batch
N. job 10 3
130
120
110
100
90
80
1973
1974
1975
1976
1977
1978

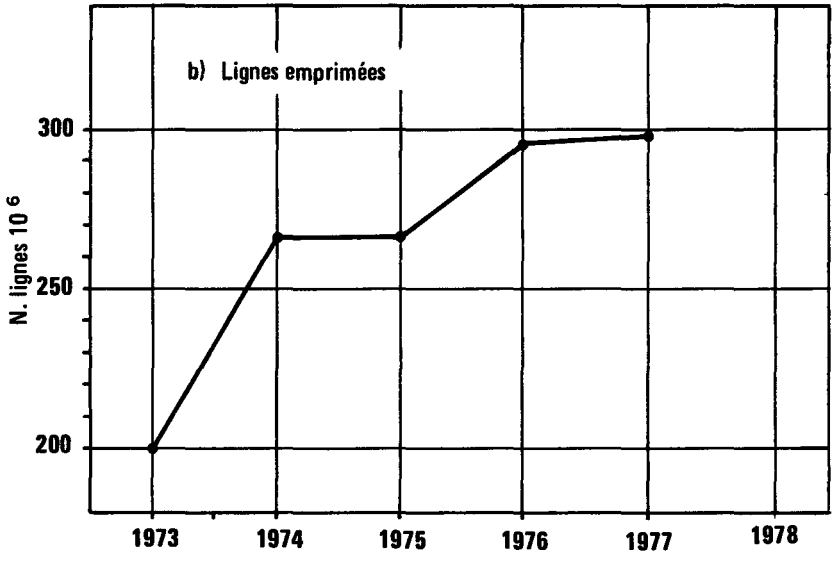
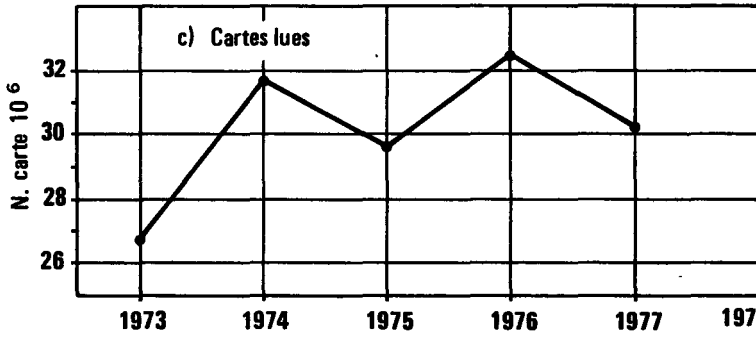

b) Lignes emprimées
N. lignes 10 6
300
250
200
1973
1974
1975
1976
1977
1978

Fig. 3


c) Cartes lues
N. carte 10 6
32
30
28
26
1973
1974
1975
1976
1977
1978

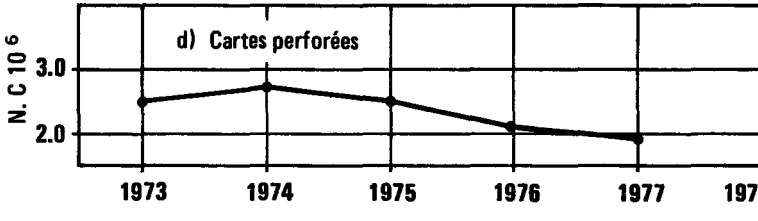
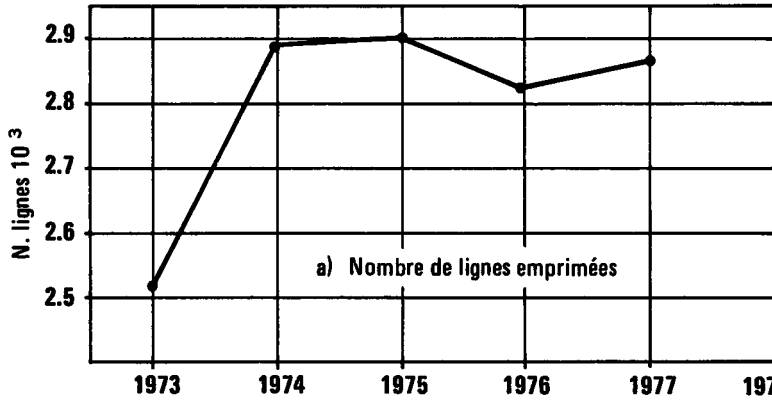
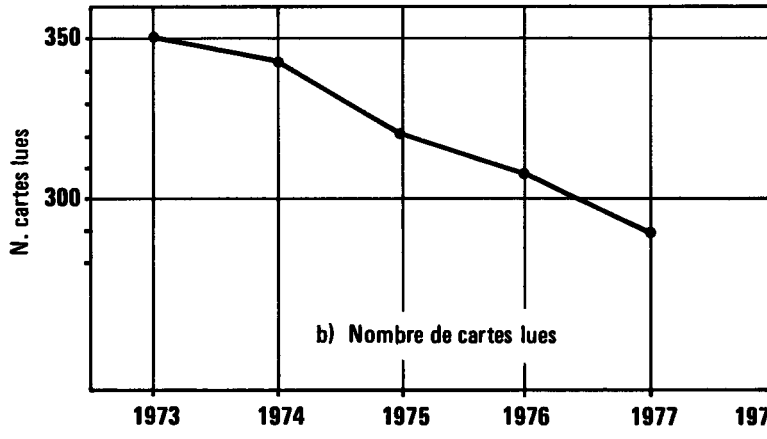

d) Cartes perforées
N. C 10 6
3.0
2.0
1973
1974
1975
1976
1977
1978

Fig. 3


N. lignes 10 3
2.9
2.8
2.7
2.6
2.5
a) Nombre de lignes emprimées
1973
1974
1975
1976
1977
1978


N. cartes lues
350
300
b) Nombre de cartes lues
1973
1974
1975
1976
1977
1978

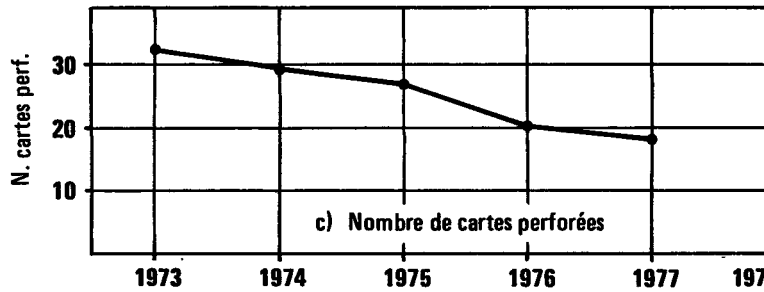

N. cartes perf.
30
20
10
c) Nombre de cartes perforées
1973
1974
1975
1976
1977
1978

Fig. 4 — Par Job

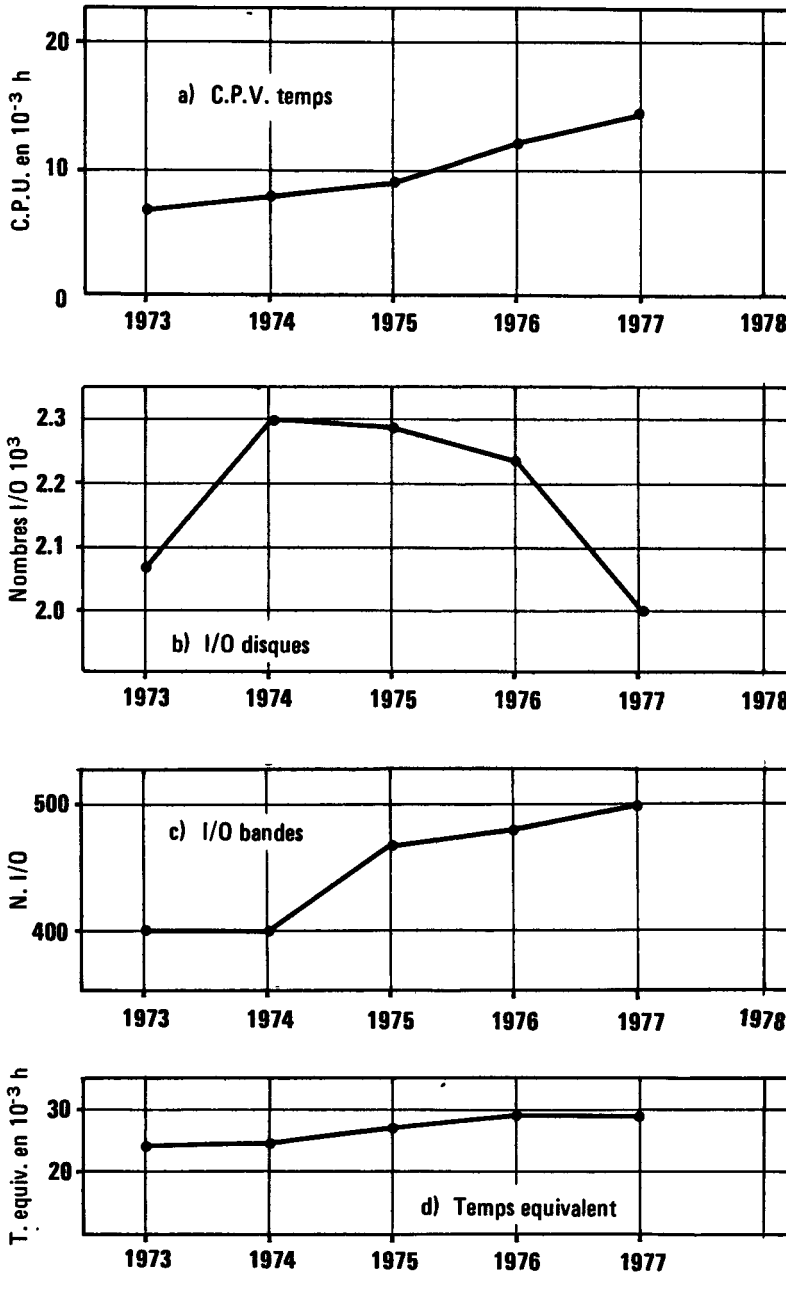
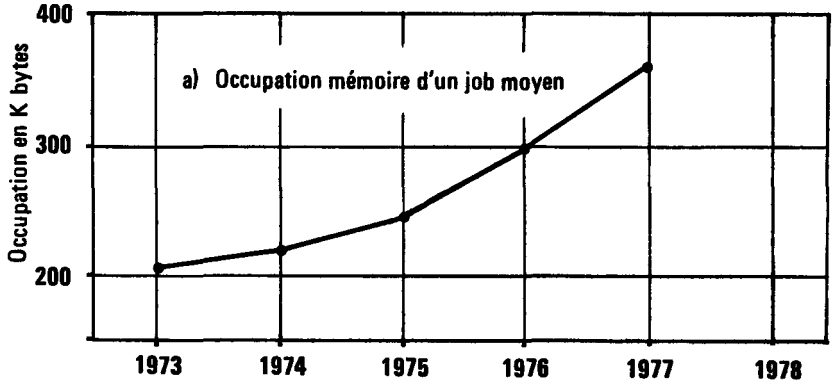

C.P.U. en 10^{-3} h
20
10
0
a) C.P.V. temps
1973
1974
1975
1976
1977
1978
Nombres I/O 10^3
2.3
2.2
2.1
2.0
b) I/O disques
1973
1974
1975
1976
1977
1978
N. I/O
500
400
c) I/O bandes
1973
1974
1975
1976
1977
1978
T. equiv. en 10^{-3} h
30
20
d) Temps equivalent
1973
1974
1975
1976
1977

Fig. 5 – Par Job


400
300
200
Occupation en K bytes
a) Occupation mémoire d'un job moyen
1973
1974
1975
1976
1977
1978

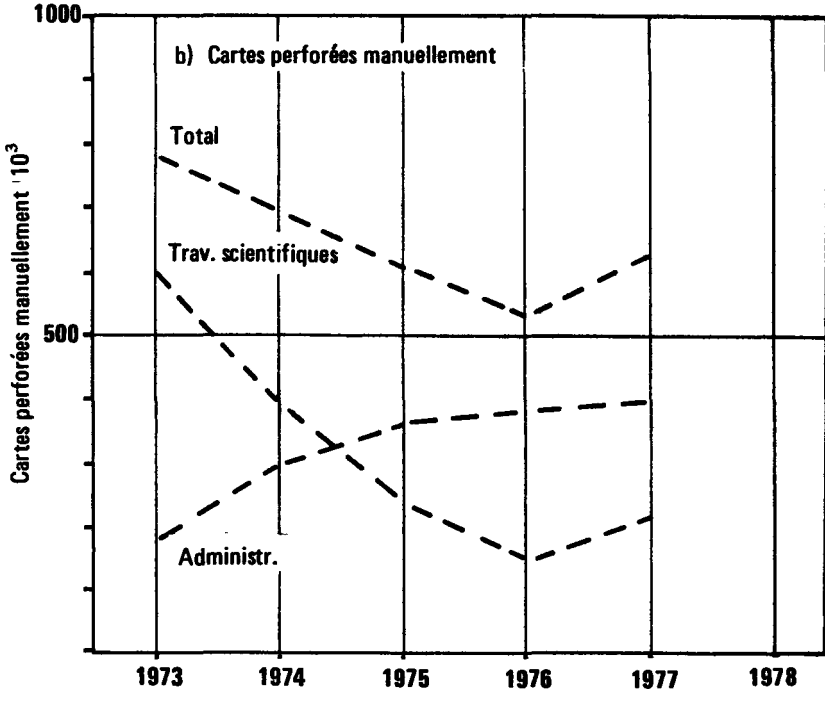

1000
500
Cartes perforées manuellement $\cdot 10^3$
b) Cartes perforées manuellement
Total
Trav. scientifiques
Administr.
1973
1974
1975
1976
1977
1978

Fig. 6

Statistics of computing installation utlization

Report of computing installation exploitation for the month of January

	YEAR 1978	YEAR 1977
Number of working days	22.50 d	20 d
Work hours from 8.00 to 24.00 for	352.00 h	16.00 h
Duration of scheduled maintenance	24.08 h	27.00 h
Duration of unexpected maintenance	18.43 h	86.10 h
Total maintenance time	42.51 h	113.10 h
Total exploitation time	337.57 h	206.90 h
CPU time in problem mode	132.07 h	71.36 h
Conversational Systems:		
CPU time	2.20 h	1.10 h
I/O number	427,000	136,000
Equivalent time	5.10 h	2.00 h
Elapsed time	293.00 h	75.00 h
Batch processing:		
Number of jobs	8,952	6,246
Number of cards read	2,240,000	1,827,000
Number of cards punched	167,000	96,000
Number of lines printed	26,817,000	16,478,000
Number of pages printed	602,000	364,000

BATCH PROCESSING DISTRIBUTION BY REQUESTED CORE MEMORY SIZE

	100	200	300	400	600	800	1000	1400	total
Number of jobs	2323	2975	1767	1235	215	78	63	1	8657
Elapsed time (hrs)	51	143	183	204	46	19	27	0.3	673
CPU time (hrs)	2.5	19	35	44	14	5	11	0.1	131
Equivalent time (hrs)	16	49	72	84	21	10	16	0.2	268
Turn around time (hrs)	0.4	0.7	1.2	1.6	2.3	3.6	3.0	1.9	0.9

PERCENTAGE OF JOBS FINISHED IN LESS THAN

TIME	15'	30'	1^h	2^h	4^h	8^h	1^D	2^D	3^D	6^D
% year 1977	33	49	65	77	87	94	97	98	99	100
% year 1978	41	59	77	90	96	99	99	99	100	

Utilisation of computer center by the objectives and appropriation accounts for the month of January

		IBM 370/165 equivalent time in hours
1.20.2	General Services - Administration - Ispra	50.83
1.20.3	General Services - Technical - Ispra	1.02
1.30.4	L.M.A.	–
1.90.0	ESSOR	15.07
1.92.0	Support to the Commission	1.52
2.10.1	Reactor Safety	150.35
2.10.2	Plutonium Fuel and Actinide Research	1.15
2.10.3	Nuclear Materials	1.94
2.20.1	Solar Energy	1.29
2.20.2	Hydrogen	–
2.20.4	Design Studies on Thermonuclear Fusion	2.16
2.30.0	Environment and Resources	8.86
2.40.0	METRE	2.21
2.50.1	Data Processing	24.00
2.50.3	Safeguards	3.00
	TOTAL	263.40
1.94.0	Services to External Users	12.61
	TOTAL	**276.01**

EQUIVALENT TIME TABLE FOR ALL JOBS OF THE GENERAL SERVICES - Monthly and Cumulative Statistics

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1977	44	74	78	32	26	36	27	25	27	31	40	34
accumulation	44	118	196	228	254	290	317	342	369	400	440	474
Year 1978	51											
accumulation	51											

EQUIVALENT TIME TABLE FOR THE JOBS OF ALL THE OBJECTIVES AND GENERAL SERVICES - Monthly and Cumulative Statistics

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1977	135	218	312	193	180	269	244	196	277	275	284	179
accumulation	135	353	665	858	1038	1307	1551	1747	2024	2300	2584	2763
Year 1978	211											
accumulation	211											

EQUIVALENT TIME TABLE FOR THE JOBS OF THE EXTERNAL USERS - Monthly and Cumulative Statistics

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1977	13	14	18	16	13	22	19	18	27	25	21	20
accumulation	13	27	45	61	74	96	115	133	160	185	206	226
Year 1978	12											
accumulation	12											

EQUIVALENT TIME TABLE FOR ALL JOBS OF ALL USERS - Monthly and Cumulative Statistics

	January	February	March	April	May	June	July	August	September	October	November	December
Year 1977	158	241	314	242	202	294	266	217	299	299	318	235
accumulation	158	399	713	955	1157	1451	1717	1934	2233	2532	2850	3085
Year 1978	276											
accumulation	276											

Programming Foundations

W. Boettcher, R. Jaarsma, A.A. Pollicini

The following article has been written as a summary of the course "Programming Foundations" as was organized by CREST, IRIA and the University of Toulouse, December 1977.

We considered the contents of importance for all programmers and decided to publish the article here. Other summaries of conferences, if there are of general interest, are also welcome.

The Editors

The title "Programming Foundations" suggests the presentation of basic and commonly agreed principles of programming. However the course was not so well structured and there were no connections between the lectures. It became clear that there is a great distance between the world of academic computer scientists and the world of daily computer users which look for solutions of real problems and work mainly with the software tools supplied by the computer companies.

Nevertheless many practical problems were discussed and also methods which can be well applied in practice were taught.

We think that the following headings cover the greatest part of that was said.

1. Software Development Management
2. Better Languages
3. Program Verification and Correctness Proofs.

The preliminary documents are available at the Computing Support Library (Mrs. A. Cambon).

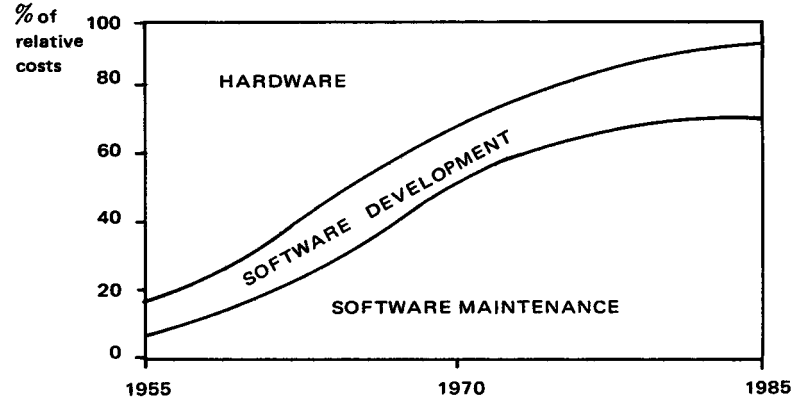
It is planned by the IRIA to publish a book, based on this course, next autumn.

It was amazing to discover that there are now so many students at universities who study informatics as an adult, independent discipline.

We try below to summarize the courses about the above listed subjects and we may go more into detail if desired.

Software Development Management

The actual need of Software involves a big amount of investments. Furthermore the ratio between Software and Hardware costs is continuously increasing, as shown in pictorial form by the Boehm's curve (1975).



These economical considerations lead to look at Software development as an engineering activity to be organized, let us say as the production of a car.

A series of lectures were dedicated to the global Software cycle. This topic was mainly illustrated by M. Jastrabsky, who represented the industry.

Software development may benefit by planning methodologies like PERT, already applied to control complex projects of other nature. The first need in doing that is to split the project into well defined subtasks, clearly ordered in time and interfaced by disciplined use of the resources.

Secondly, proper tools must be provided to carry out the project. Four main phases are identified in each Software project:

- Analysis of the requirements and problem definition,
- Design of the solving process,
- Coding the solution in a computer acceptable form,
- Integration and testing the computer program.

Each phase requires:

- Separate estimates in time and resources,
- Periodical reviews and progress reports,
- Final evaluation and documentation.

As for any other application product, Software must be preserved from failures, therefore error checking must be applied at any stage of the project.

The most expensive errors are the ones made in the design phase, especially when they are only detected in the production phase. Therefore the design should be the crucial point in the project, in order to build a well formed final product, containing as less as possible conceptual errors.
This means that programmers become engineers and their job must firstly and mainly consist in thinking and finding solutions, while writing statements becomes a less important task of their profession.

In the lectures of M. Galinier the emphasis was put on the design phase. The design starts with a global view of the problem which is developed step by step in more detail.
Each step may identify a level of abstraction of the problem. Design methodologies can be based on this concept and the one developed at the Toulouse University was described.

- The most external entity is called Project and is composed by a hierarchy of Abstract Machines (AM).
- Each AM covers a well defined action, assigning it some resources and operations on them.
- Software resources are classed in internal ones, which are defined in the AM itself (using a PASCAL-like data type mechanism), and external ones which are defined in AM of higher levels.
- Operations are grouped in units called Modules, which may be internal if used only within the AM defining it, or external if activated by Modules belonging to AM of higher levels.
- Software resources needed by a Module must be declared as parameters classed in input only, output only and input plus output, so that the entire data flow is explicitly evident.

A Language for Abstract Machines Design Approach (LAMDA) was defined and a processor implemented to obtain a documented support of the design.

Starting from the codified design produced by LAMDA, by means of an interactive processor for semiautomatic program generation, a PL/I program is developed.

Another series of lectures was devoted by P. Henderson to the use and management of libraries to support the development of large Software Systems.
If the components of the libraries (JCL, source texts, object texts and case data) become highly inter-related, it may be useful to store them in a relational data base.

The principles of such a data base were discussed and some data models described:

- A developmental model which represents the relationships among different components concurring in a program development process.
  When a component has to be modified, this model points out directly what other components will become invalid as well as the needed processes to generate a valid copy of them.
- A structural model dealing with components of separate versions of a system.
  When a specified configuration of the system is requested, this model picks up all the components to build it and even the specific case data to test it may be selected from the library.

Better Languages

A programming language can be seen as a tool to describe algorithms. In general, solutions of problems are expressed in algorithms and algorithms are translated into programs, which are executed by a computer. Most of the programming languages are strongly computer directed and programming is mainly: "Adapting an algorithm to constraints and facilities of a computer". If however accuracy and design efficiency become more important than computer efficiency, then new concepts of programming languages are possible.

The lectures of J. Arsac, R.M. Burstall, R. Dewar and M. Griffiths were (partly) dedicated to the subject of better languages. Principles as modularity and structuring are evident and also valid in these higher level languages.

The Newsletter is available at:	*Des exemplaires du Bulletin sont disponibles chez:*
Mrs. A. Cambon Support to Computing Bldg. 36 - Tel. 730	**Mme A. Cambon** Support to Computing Bât. 36 - Tel. 730

The basic working methods may become:

- Use a very high level language to write a correct program, which is perhaps not an efficient program. The language should permit easy and rapid programming (compare APL). The program is close to the problem (and is remote from the machine level).
- Prove correctness and modify in this version.
- Transform the program correctly into a more efficient program executable by a computer. The correctness-preserving transformation may be done in different ways.
  In the ideal case there is a compiler or preprocessor to do the job.
  However transformation may be done (stepwise) by hand applying correctly transformation rules until the program is in a form which can be coded in a known language.

J. Arsac proposed an assignment free language. Correctness proofs can then be constructed by applying the mathematical concept of substitution. Such a language is in general not implemented on a computer.
A program must therefore be transformed preserving its meaning.
The method and rules to do so are an integral part of the language.

R.M. Burstall explained a system of rules for transforming programs, with the programs in the form of recursion equations. He proposed a simple functional programming language and listed explicitly those features of a language for writing correct and flexible programs. He advocated e.g. maximum use of user defined types. Thus type "data" or type "pressure" rather than type "real". Also no assignment, thus a real specification language.

A refined system of type declarations was also backed by M. Griffiths, who proposed a lot of practical improvements in languages and compilers. E.g. a compile time checks on ranges of variables and tests on completeness of possibilities should be executed.

R. Dewar spoke about the language SETL. SETL allows algorithms to be written in a maximally concise manner, without concern for selecting the data structure to be used. By using maps and sets the data is structured in such a way that it is easy to apply operations on them. Expressions of set theory are valid expressions in SETL. A reasonably efficient execution can be obtained from this SETL program. Completely separated from the SETL program itself the programmer can gain computer efficiency by detailed specification of data structures. He creates the declarations of the representation sublanguage as a separate file, which the compiler merges with the original program, by matching up module and procedure names, without any danger of introducing errors by subtle departures form the semantics of the original program.

Program Verification and Correctness Proof

In spite of a big amount of papers devoted to this subject in the last 20 years, the ideas of program proving seems to have no important impact on current programming practice. One of the reasons for this may be that a truly practical program verification technology has been slow to develop, and that the extra work to carry out a formal program verification using existing techniques is still too much.

On this subject there were the 2 presentations from J.T. Schwartz and O.J. Dahl. Both are based on the inductive assertion technique of FLOYD/HOARE. In this approach, one attaches 2 predicates to the given program text: an input assumption (pre-condition) and an output assertion (post-condition).

A pragmatic approach of J.T. Schwarts.

A program should not so much have proved correct as developed in such a way as to make its correctness evident. Therefore, rather than starting with large program texts and attempting to prove their correctness, we should prefer to work systematically with generally small units of program text. Such small units express some fundamental and essentially indecomposable element of algorithmic technique and are known *to be correct.* The rules for the manipulation and combination of correct program text units lead then to what Schwartz calls *"the correct program technology".* This technique should open the way to an interactive, correct program manipulation system, in which a group of algorithm developers could enter "ROOT" programs to be used and combined into larger programs by the application programmers.

The more theoretic approach of O.J. Dahl.

In this presentation, emphasis was put on the design and proof of abstraction mechanisms. By this it is possible to represent the semantics of an external program description by means of an *abstract function.* On the other hand, the given program text leads to a *concrete* functional description.

A (semi)-automatic proof system should carry out that the 2 functional representations are equivalent, using a set of axioms and deduction rules.

The persons interested in receiving regularly the "Computing Centre Newsletter" are requested to fill out the following form and to send it to:

Mrs. A. Cambon
Support to Computing
Building 36, Tel. 730

Nom .....

Address .....

.....

Tel. .....

EIN PRESENTATION

There will be a presentation and demonstration of the EIN Network on Wednesday 5th April, in the Amphitheatre of Building 36.
This demonstration will be a milestone in the project and will take place simultaneously in 11 centres throughout Europe.
The presentation is open to all those interested.
A detailed programme will soon be available.