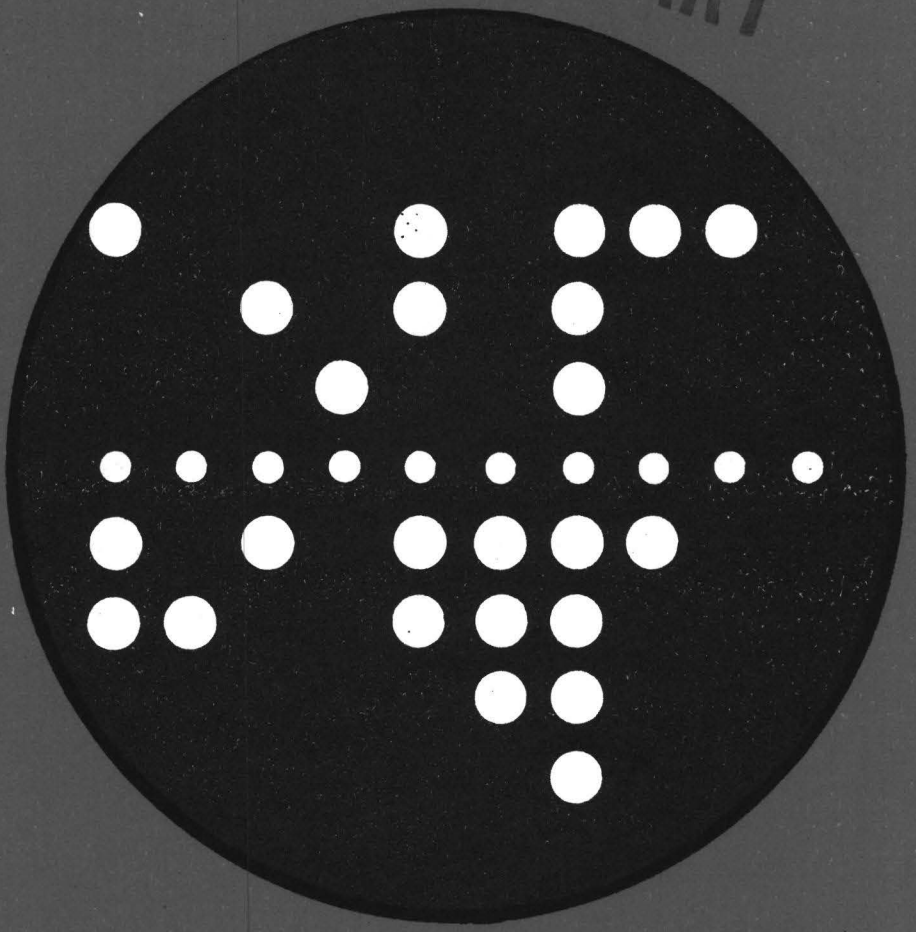
COMPUTING CENTRE NEWSLETTER

July 1978 · No 23



Commission of the European Communities
JOINT RESEARCH CENTRE
Ispra Establishment

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.

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	1883	1259
Adjoined	A. Pollicini	1886	701
Application Packages	A. Inzaghi	1887	755

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

The European Informatics Network Demonstration

A. Endrizzi

This is the third article in a series decribing the work being performed on the project and will describe the EIN demonstration that took place on April 5th.

The demonstration

The basic idea behind the demonstration was to present the «Concourse» of computers of different makes and located at various European sites.
For this purpose a series of simple applications was agreed between the partners and performed concurrently at each site.
Each partner utilized his own equipment such as local interactive terminals, printing facilities and computers to access the services offered by the remote partners.
Those participants not having enough resources, in terms of equipment and developed sofware, joined the demonstration via dial up connections to the closest subscriber host of the network.

The «distributed» audience could feel the presence of a giant multipurpose machine build up by the connection of a number of data processing installations in Europe, each of them offering its own application services.

The system

The traffic of data is handled and regulated by the various logical functions implemented on top of the telecommunication lines connecting the dispersed equipment.
The basic data transmission service is supported by the five packet switching nodes located at the primary centres sites and connected together by international lines. They are capable of transferring atoms of information (packets) from any source to any destination.

Source and sinks of data are represented by the connected computers running application programs and/or driving peripherals. The connection is usually supported by a local full duplex line linking the host computer to the closest packet switching center.

Each host computer normally runs a number of application programs and terminal handlers.
This means that packets in arrival have to be classified and routed to the proper destination inside the host itself.Packets to be transmitted need also to be «marked» in such a way that they can be addressed to the proper destination program.

This multiplexing and demultiplexing function inside the hosts is performed by the so called Transport Station. The TS is a distributed process running in

all hosts and charged with the naming conventions that are necessary to set up and operate the conversations between remote processes.
The TS is also provided with the appropriate intelligence to recover from the sporadic malfunctioning of the packet switching network due to line break down or congestion.
The TS acts much the same way as telephone and mail services do. It provides liaisons (telephone calls) which carry the full duplex conversation between correspondants in the form of streams of messages. It also provides facilities to send lettergrams (letters) from one network address to another.

The contents of the messages or lettergrams is completely transparent to the TS.
The semantic meaning of the exchanged information has to be agreed by the interlocutors.
Point to point agreements on the language to be used for the communication are to be avoided. In fact the software which is capable of interpreting the exchanged information should be valid for any conversation of the same type.

The standardization of the language to be used by the communicating processes guarantees the possibility to bypass the incompatibilities between hererogeneous systems and services. The common language is obviously different from the one used within a particular system, so that each network host has to adapt itself to the agreed standards.

On top of the TS a Virtual Terminal layer was defined which corresponds to the notion of a standard interactive terminal. All network hosts had to interface via software translation modules, their terminal handlers to the VT.
This approach implies that local terminals are to appear as standard terminals and local applications are to be modified or interfaced in order to be capable of driving standard terminals.
In this way any terminal connected to the network system is able to logon any conversational service which accepts network users.

Similar criteria will be followed in the future to provide the EIN network with the file transfer and remote job entry services.

The demonstrated facilities

The demonstration was intended to show the operation of the various layers mentioned above.

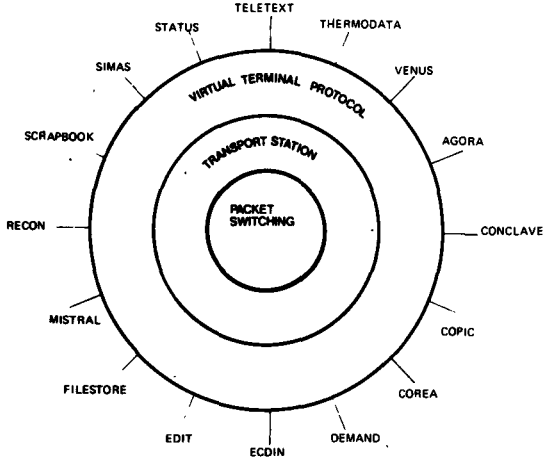
The packet switching service was illustrated first. This was done by sending data and commands packets to the switching nodes. Those packets were addressed to the testing and monitoring facilities that run in the nodes. Those facilities, once activated, provide the demanding end with packets carrying status information such as line behaviour, traffic, routing paths.
Packets were also sent to the CYCLADE, DCN and EPSS networks which happen to be connected to EIN.

Via DCN and EPSS, ARPANET was supposed to be reachable. The experiment to link us to the Pacific Coast was also scheduled but for some reasons we were unable to perform it. We now know that other EIN partners succeded in that.

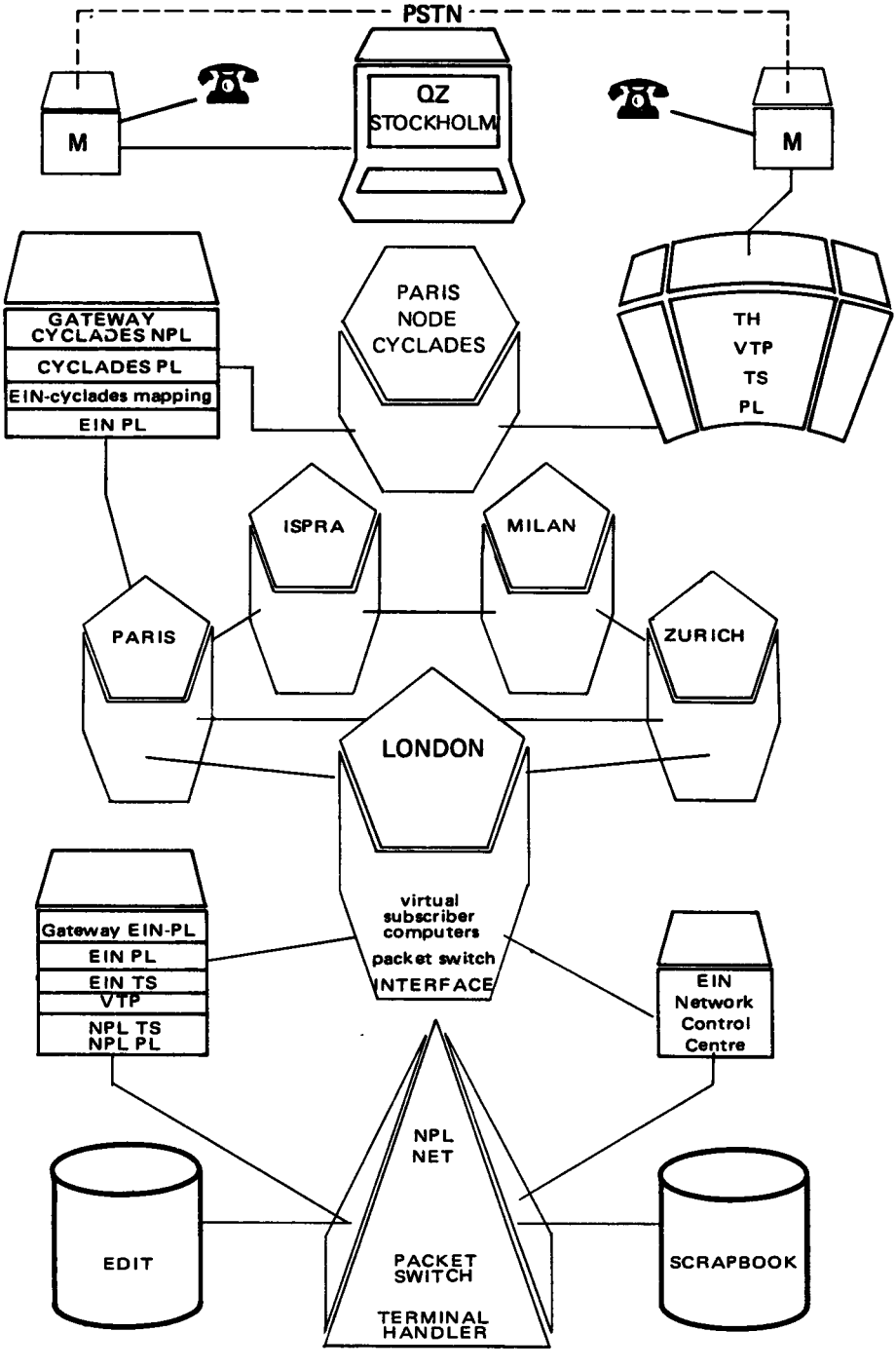
The Transport service was demonstrated by exchanging messages with various location.

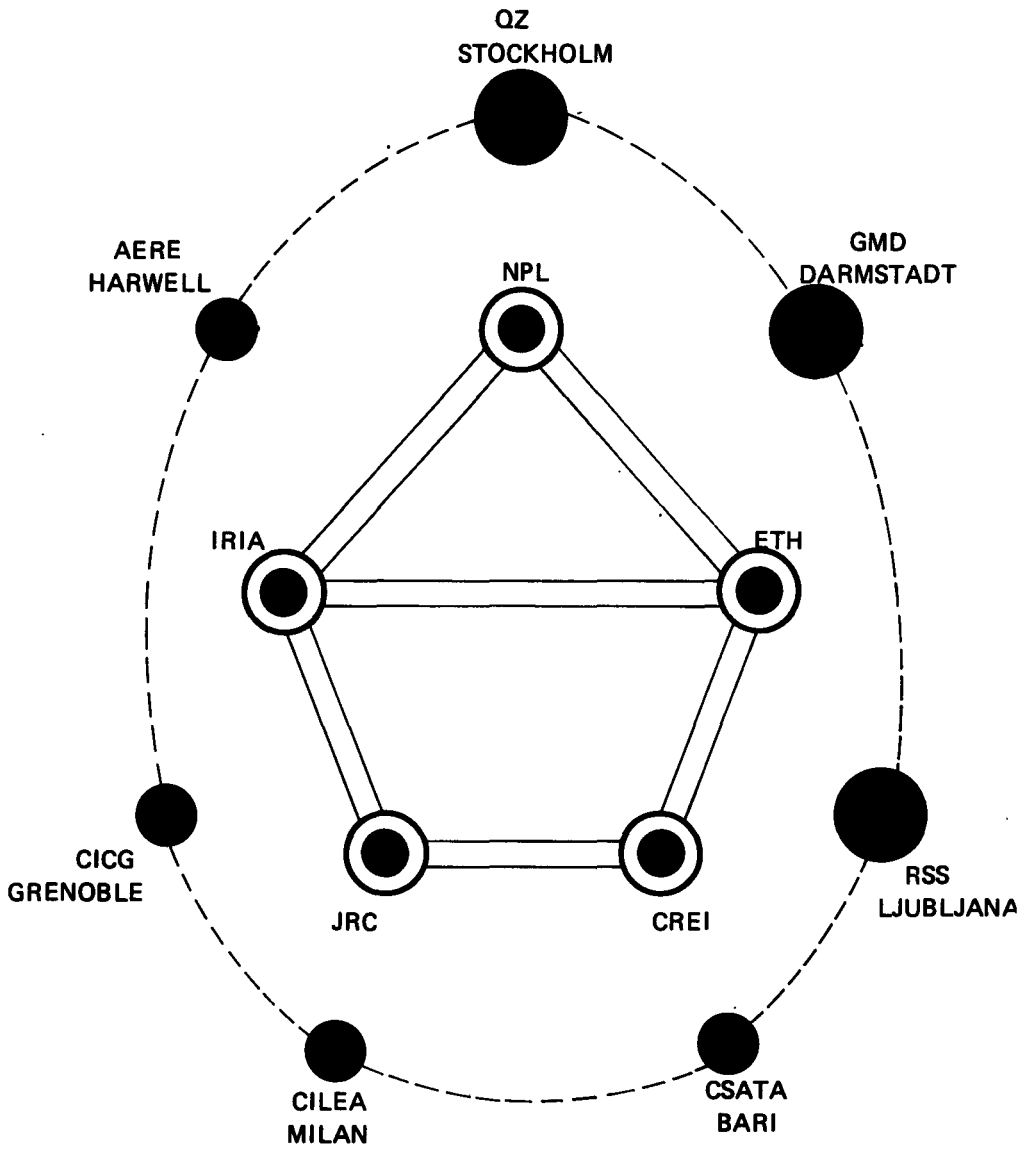
A distributed application n called «Conclave» was run successfully. It consists of a simplified teleconferencing service which allows a number of participants to build and retrieve a common file containing the contributions of the «speakers».
The virtual Terminal Protocol allowed those partners having implemented it, to access to a number of remote timesharing services and information retrieval systems.
From our terminals we ware able to log on to the Univac timesharing service in Milan and to the CDC Venus service in Zuerich.

Remote users utilized our corea and ECDIN facilities.
During the demonstration we experienced some malfunctions which introduced some delays into the scheduled agenda or prevented us to show some other implemented features. After a preliminary analysis jointly performed by the EIN implementors, we can now say that those faults originated from hardware malfunctioning and from software unreliability. As a justification it should be mentioned that EIN is a research project and as such hardware and software maintenance services are not supported properly. Nevertheless it is believed that despite some temporary faults, the network system behaved nicely and the cooperation of a great number of heterogeneous systems performing activities was demonstrated.



SOME EIN SERVICES


PSTN
M
QZ
STOCKHOLM
M
GATEWAY
CYCLADES NPL
CYCLADES PL
EIN-cyclades mapping
EIN PL
PARIS
NODE
CYCLADES
TH
VTP
TS
PL
ISPRA
MILAN
PARIS
ZURICH
LONDON
Gateway EIN-PL
EIN PL
EIN TS
VTP
NPL TS
NPL PL
virtual
subscriber
computers
packet switch
INTERFACE
EIN
Network
Control
Centre
NPL
NET
EDIT
PACKET
SWITCH
TERMINAL
HANDLER
SCRAPBOOK



THE CONCOURSE OF COMPUTERS

ACCOUNTED WORK UNITS 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	43	55	50	49	74						
accumulation	51	94	149	199	248	322						

ACCOUNTED WORK UNITS 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	213	283	232	202	317						
accumulation	211	424	707	939	1141	1.458						

ACCOUNTED WORK UNITS 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	10	11	46	23	11						
accumulation	12	22	33	79	102	113						

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	261	356	298	262	335						
accumulation	276	537	893	1191	1453	1.788						

Statistics of computing installation utlization

Report of computing installation exploitation for the month of June 1978

	YEAR 1978	YEAR 1977
Number of working days	22 d	22 d
Work hours from 8.00 to 24.00 for	16.00 h	16.00 h
Duration of scheduled maintenance	21.00 h	17.84 h
Duration of unexpected maintenance	15.83 h	14,61 h
Total maintenance time	36.83 h	32.45 h
Total exploitation time	334.988h	319.55h
CPU time in problem mode	183.178h	143.71 h
Conversational Systems:		
CPU time	3.60 h	3.93 h
I/O number	501.000	845.000
Equivalent time	7.10 h	9.85 h
Elapsed time	415.00 h	351 h
Batch processing:		
Number of jobs	8.542	10.392
Number of cards read	2.019.000	2.895.000
Number of cards punched	116.000	166.000
Number of lines printed	27.663.000	28.265.000
Number of pages printed	614.730	632.000

BATCH PROCESSING DISTRIBUTION BY REQUESTED CORE MEMORY SIZE

	100	200	300	400	600	800	1000	1400	total
Number of jobs	1961	2680	1881	1027	296	26	92	34	7997
Elapsed time (hrs)	57	181	224	188	65	4	51	12	782
CPU time (hrs)	3	25	41	33	18	0,7	15	7	143
Equivalent time (hrs)	17	54	73	71	30	1	30	8	284
Turn around time (hrs)	0.4	1.3	1.5	2.4	3.5	3.0	4.6	8.0	1.4

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	46	64	80	90	97	99	99	99	100	
% year 1978	27	43	61	78	93	99	100	/	/	/

Utilisation of computer centre by the objectives and appropriation accounts for the month of June 1978

		IBM 370/165 equivalent time in hours
1.20.2	General Services - Administration - Ispra	73.55
1.20.3	General Services - Technical - Ispra	0.64
1.30.4	L.M.A.	–
1.90.0	ESSOR	36.11
1.92.0	Support to the Commission	1.30
2.10.1	Reactor Safety	149.55
2.10.2	Plutonium Fuel and Actinide Research	1.95
2.10.3	Nuclear Materials	0.60
2.20.1	Solar Energy	–
2.20.2	Hydrogen	–
2.20.4	Design Studies on Thermonuclear Fusion	4.38
2.30.0	Environment and Resources	16.46
2.40.0	METRE	1.11
2.50.1	Informatics	25.48
2.50.3	Safeguards	3.60
309	Programming Support	2.16
	TOTAL	316.89
1.94.0	Service to External Users	11.43
	TOTAL	328.32

An Introduction to Modular Systems

G. Gaggero

Abstract

An overview of the various approaches adopted in the development of modular systems is presented. Three modular systems which have a high degree of generality , (CARONTE, ICES, and GENESYS), are described.

Introduction

With the advent of third generation computers an increasing interest in the development of modular systems was observed. Several different systems have been implemented; some of them are specific to particular fields of application of computers, other ones are quite general in purpose.

There are various motivations at the base of modular systems development.

In first place, the rapid growth of computer power stimulated the increase of the number and complexity of the tasks performed with the aid of computers. Stand-alone programs written to solve a specific problem turned out to perform only a subtask of a more complex task. It appeared evident the need to integrate stand-alone programs into systems of programs. But integration requires standardization and the logical complexity of large systems requires a clear structure of programs; so that economy and efficiency suggested a second step to be made, the design of modular programming systems.

Another important motivation is the need of a rationalization of the production, dissemination and use of software. Modular systems are expected to reduce the present wastage of man-power and money caused mainly by software duplication, by promoting a more effective sharing of programs.

The aim of this paper is to outline the most important aspects of the modular system approach, and to describe three systems, (CARONTE, ICES, and GENESYS), which have a high degree of generality, and are presently in use at the JRC Computing Center.

An Outline of Modular Systems

Taking into account the basic philosophies and technical approaches, three types of modular systems can be identified.

One can classify into a first category those modular systems which have been developed to link together several pre-existing stand-alone programs with a minimum of changes. They essentially automate the job of taking the output of a program and preparing the input for the next one.

A second category includes those systems whose modules and interfaces have been planned from the start and have been developed following standard rules.

To the third category belong problem orientated language modular systems. In these systems modularity is a consequence of both the need to integrate

the operation of several separate but related subtasks and the problem orientated language communication approach.

In the first two types of systems, communication of data between modules is generally accomplished through a peripheral or backing store. The layout of data within the interfaces is under the control of the programmer and he must provide each module with suitable read/write instructions or calls to retrieve or store data.
In the systems of the third type the interface concept is replaced by the virtual storage concept, which relieves the programmer of knowing whether or not the data is in core or stored peripherally. A language (generally an extension of FORTRAN) is provided to make data access transparent to the programmer.

Another interesting aspect of modular systems is the way in which the control of the computational sequence through the various modules is provided.

From a logical point of view different types of path control can be identified:

- those in which the path is pre-set when the system is programmed,
- those in which the path is pre-set but self-developing during execution on the basis of discriminations which are identified in advance,
- those in which calculational paths are controlled interactively by the user during execution.

The first type of control can be usually obtained without a supervisor (or executor or driver program) by simply arranging modules as subsequent steps of a job and using theJob Control Language (JCL) provided by the Operating System.

On the contrary, a driver program is required in order to implement the second type of path control. Arithmetical and logical capabilities of an algorithmic language are required to programme the complex logic of a self-developing path. In addition, the driver program must provide an efficient modules management to allow looping paths.

Concerning the third type of control, one easily recognizes the need for a conversational driver program with a powerful communication language, and, hopefully, graphical display facilities.

A further important aspect of modular systems is the way in which they are implemented. As the efforts and costs associated to the development and maintenence of a comprehensive library of modules are normally quite high, machine independence is a basic requisite.

Various approaches to this problem have been dopted:

- writing modules and driver program in ANSI FORTRAN or in a low level subset of FORTRAN;

- confining all data management operations in the driver program and accessing external data through standard calls to subprograms;
- writing modules in a «private» language which can be easily translated into the various dialects of a high-level language (e.g. FORTRAN) of general use.

A high degree of portability of modules is generally obtained by these means. Of course the problem is more difficult with the executor which must rely for modules and data management on the facilities provided by the various operating systems. No general solution has been found to this problem, except that of providing different executors for different computers.

CARONTE System

CARONTE system was started in 1967 at the CETIS Division of the Ispra Establishment of the Joint Research Centre. The first version became operative on the IBM 360/65 computer in 1969. Use experience and portability considerations suggested a series of improvements which resulted in the development of two further versions in 1971 and 1972. The most up-to-date version of the system, called CARONTE Fortran, is 90% written in Fortran IV language and is running on IBM 360 and 370 series computers.

CARONTE has been working since many years at various computer installations, (Ispra; CNEN, Bologna; BELGONUCLEAIRE, Bruxelles; CAMEN, Pisa).

The Philosophy of CARONTE

CARONTE was developed to provide an automatic control of the execution of a sequence of modules involving transmission of data from module to module.

A basic design specification of the system was that pre-existing programs could be handled as modules with a minimum of modifications.

Input and output data of a program to be inserted into the library of CARONTE do not have to be standardized. Groups of data that could be received from or passed to another module have to be organized into numbered sets, called «INTERFACES». Each interface has its own lay-out of data, but interfaces identified by the same number must have the same lay-out.

When a module produces a set of data to be used by another module, the set is stored, under the control of CARONTE, into a data file, called «DATA-POOL», and receives a label consisting of:

«module name»«module occurence»«interface number»«control flags»

Conversely, when a module requires a set of data produced by another module, the set is transferred, under the control of CARONTE, from the DATA-POOL to the module.

However, since input and output INTERFACES are not necessarily standardi-

zed, it may happen that a rearrangement of data (e.g. convertion of data from an interface to another one; merging of two or more interfaces) become necessary to provide a correct transfer of data.

This task is performed by «LINKING MODULES», which must be written to provide INTERFACES manipulation, (they operate on one or more interfaces to produce another interface).

It must be noted that intervention of LINKING MODULES does not require any user action, being controlled interely by CARONTE.

Of course, if CARONTE is used to control a library of modules having standardized interfaces instead of a library of pre-existing programs linked together a posteriori, the need for LINKING MODULES is strongly reduced.

Path Control and Data Flow Specification

The system CARONTE allows the execution of a single module as well as of a sequence of modules. Are also permitted loops involving two or more modules. The number of times a loop is executed can be fixed by the user or depend on the verification of a specified condition.

To the purpose of defining the sequence of modules to be executed, the user is provided with a simple control language. For instance, the sequence of modules ALFA, BETA, twice (GAMMA, LAMBDA), BETA is specified as follows: ALFA-1, BETA-1, 2*(GAMMA-1, LAMBDA-1), BETA-2

The number following the module name is the «MODULE OCCURENCE NUMBER» in the sequence.

The flow of data from module to module can be specified by the user by means of the control language.

For instance, in order to inform the system that INTERFACE 2 required by module BETA (in its first occurence) must come from INTERFACE 3 and INTERFACE 5 provided by program ALFA, the user can write either:

BETA-1(2)FROM ALFA-1(3),ALFA-1(5) or BETA-1 interface 2 name FROM ALFA-1

provided that a name has been assigned to INTERFACE2, using a special feature of CARONTE.

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

System Organization

The system consists of a NUCLEUS, which is always resident in fast memory and of a PROCESSOR.

The operations performed by the system are the following:

a) it analyses the sequence of modules specified by the user,
b) it examines what interfaces have to be transferred from module to module and determines what linking-modules are required,
c) it executes the whole sequence consisting of problem-solving modules and linking-modules.

ICES System

The Integrated Civil Engineering System (ICES) was developed at the M.I.T., starting from 1964, as a cooperative venture of US government, industry, and university groups interested in the development of a large-scale, computer-based system for Civil Engineering.

It was originally developed to run on IBM 360 series computers, but, at present, there exist several versions of ICES running on UNIVAC, CDC, SIEMENS 4004, and PHILIPS P400 computers.

The MIT version of ICES (for IBM 360, 370 computers) is available from the ICES Users Group (IUG) Distribution Agency.

The Philosophy of ICES

The ICES system is aimed at providing the engineer (or more generally the user) with a complete environment for developing and using large integrated programs. The major attention was devoted in designing ICES to the man-machine communication problem.

The structure of ICES is hierarchical in that it is composed of a set of subsystems operating under the control of the ICES Basic System.

The role of the Basic System will become clear if we proceed by considering the characteristics of a subsystem.

A subsystem is a collection of modules performing various independent but related subtasks which combined in different ways provide the solution to a particular class of problems.

The user of an ICES subsystem is meant to be skilled in some discipline but not necessarily to be an expert in the use of computers: so that each subsystem must have a problem oriented language (POL) associated with it. These POLs have the nature of commands which are composed of phrases familiar to the user.

From the user point of view, the execution of an ICES subsystem is the processing of a series of commands. When a command is put into ICES, the ICES Basic System interprets the command, i.e. determines what module(s) the use of this command implies and stores any data input with the command, it ensures the execution of the invoked module(s) and then passes to the next command.

Although command interpretation is a function of the ICES System, it is the responsibility of the subsystem programmer to specify through a special purpose language, called the Command Definition Language (CDL), the form of each command and the modules to be invoked.

Concerning the problem-solving modules, they are written in a language called ICETRAN, which is a procedure-oriented language based on FORTRAN. After an ICETRAN module is written, it is precompiled by the ICETRAN precompiler into FORTRAN. The FORTRAN compiler then produces object code from the FORTRAN input. The linkage-editor is finally used to produce a load module which is included in the Subsystem library.

An ICETRAN module can make reference to the subsystem data base, which is formed from data input through the POL, and data generated during computation by other modules.

No input/output statements have to be written by the ICETRAN programmer to transfer data from module to module. ICES provides a dynamic arrays allocation facility which is implemented via a virtual store algorithm.

The basic functions of this algorithm are to create, distroy, and move data elements (arrays or subarrays) from the primary memory to disk memory and viceversa.

ICES System Components

The system consists of the following basic components:

1) ICETRAN language precompiler,
2) CDL language compiler,
3) ICES executor, which provides modules and data management
4) ICES utility programs.

GENESYS System

In 1969 the UK Government decided to sponsor a system under which a library of integrated programs could be made readily available on a wide range of computers.

The GENESYS Centre was established to coordinate the development of the system and support the use of the software.

The GENESYS (GENeral Engineering SYStem) system consists of a Master

Program and of a library of problem solving subsystems. Eighteen subsystems are now available or being developed, for use in the Civil Engineering field.

The Philisophy of GENESYS

The GENESYS System provides complete facilities for developing and running a library of integrated subsystems.

The programmer developing a subsystem can define, through a special language, a set of commands (POL) to allow users to activate individual program modules, and a set of tables to be used for entering the bulk of data.

He can write modules of a subsystem in a language, called GENTRAN, which is based on ANSI FORTRAN, but offers more powerful input and output facilities, dynamic array definition and features for manipulating data structures (tree structure).

For the user GENESYS provides two basic facilities. The user can enter the data by using the standard tables defined by the programmer. The format is very free, but because it is a standard, errors are considerably reduced. Second, he can control the way his problem is solved by using the POL commands.

In addition, GENESYS provides data filing and editing facilities, allows the user to specify the units for input or output data, accepts certain FORTRAN statements (e.g. IF, DO, GOTO, and assignement statements) amongst POL commands.

A unique facility within GENESYS is the possibility to replace any numerical item in a table by a variable (or an arithmetic expression), which will be defined later in the commands.

From the above description it appears that GENESYS is similar in a number of ways to ICES, but it differs in several important aspects.

However, the most important difference is that GENESYS was designed to be as machine independent as possible.

Subsystem modules, written in GENTRAN, are fully machine independent. In fact, a precompiler, which is part of the GENESYS Master, translates them into FORTRAN tailored to fit the particular configuration of the run-time computer.

The Master program itself is 95% machine-independent, and the GENESYS Centre provides a version of it for each different computer. At present, the GENESYS System is running on more than 20 combinations of computer/operating systems.

It must be noted that GENESYS, like ICES, can run in batch as well as in conversational mode.

Concluding Remarks

Several other modular systems have been developed during the last decade and some of them would merit our interest. However, the three systems described in this paper, have been chosen because they serve quite well to illustrate the basic aspects of the historical development of modular systems and because they are all available for use on the computing installations of the JRC.

In a second paper, which will be shortly published, we will try to outline the basic requirements for a «modern» integrated modular system for engineering.

REFERENCES

1) G. Buccari, G. Fattori, C. Mongini-Tamagnini — «CARONTE - The Euratom System for Automatic Control of Linked Calculations» — Proceedings of the Conference on «The Effective Use of Computers in the Nuclear Industry», Knoxville, Tenn., April 1969

2) C. Mongini-Tamagnini — «CARONTE - The Euratom Modular Calculational System» — Paper presented at the Symposium «Modular Coding Systems for Reactor Calculations», Ispra 1-3 Dec. 1970

3) G. Buccari, G. Fattori, C. Mongini-Tamagnini, F. Astigiano — «CARONTE - The Euratom Modular Calculations System» — Report EUR 4779e (1972)

4) G. Fattori, A.A. Pollicini — «CARONTE - FORTRAN», EUROCOPI Program Description Series, Report No.6, June 1977

5) D. Roos — «ICES System Design» The MIT Press, Cambridge, Massachusetts, 1966

6) B. Schumaker — «An Introduction to ICES», Dept. of Civil Engineering, Research Report R 67-47, MIT, Sept. 1967

7) J.M. Sussman — «Primary Memory Management in ICES: An Engineering Oriented Computer System», Dept. of Civil Engineering, Research Report R 67-8, MIT, Nov. 1967

8) W.A. Dillon — «ICES - Programmers Reference Manual», Dept. of Civil Engineering, Research Report R 71-33, MIT, August 1971

9) D.G. Alcock, B.H. Sharing — «GENESYS - An Attempt to Rationalise the Use of Computers in Structural Engineering» in: The Structural Engineer, Vol.48, No.4, April 1970, pp. 143-152

10) «The GENESYS Reference Manual», The Genesys Centre, Univ. of Technology, Loughborough, UK

11) R.J. Allwood — «GENESYS - Machine Independent Software Sharing» in: Structural Mechanics Computer Programs, ed. W. Pilkey et al., University Press of Virginia, Charlottesville, 1974

Les personnes intéressées et désireuses de recevoir régulièrement "Computing Centre Newsletter" sont priées de remplir le bulletin suivant et de l'envoyer à:

Mme A. Cambon
Support to Computing
Bât. 36, Tel. 730

Nom .....

Adresse .....

.....

Tel. .....

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

Name .....

Address .....

.....

Tel. .....