THEME 2

Impact of New Hardware and Software Technology

Effets des nouveautés technologiques sur le plan du matériel et du logiciel

Einführung und Auswirkung neuer Hardware- und Software-Technologien

Leere Seite
Blank page
Page vide

NEMESIS: a Joint Effort of Computer Users in the Netherlands

NEMESIS: Résultats d'un effort concerté des ingénieurs aux Pays-Bas

NEMESIS: eine gemeinsame Leistung der niederländischen Computeranwender

Gerard KRUISMAN
Consult. Eng.
R + K Consulting Engineers
Rijswijk, the Netherlands

Gerard Kruisman, born 1936, obtained his civil engineering degree at the University of technology, Delft, Netherlands. He became a consulting engineer, specializing in new fields in engineering. He is an advocate of multi-disciplinary cooperation within the profession, and has helped to create many organisational structures, especially in the application of CAD techniques.

SUMMARY

As a result of close cooperation between engineering users of the GENESYS system new ideas on the use of engineering systems have been developed into a concept for a new Netherlands Modular Engineering System (NEMESIS). The paper deals with the basic experiences gained with the use of the Genesys system and the policy criteria for developing a new system based on the advanced technology available today. An inventory of user requirements is provided as well as the results of a survey on CAD/CAM interests in the Dutch industry. Organizational and technological aspects of the NEMESIS development project are presented.

RESUME

A partir des expériences réalisées par de nombreux utilisateurs du système GENESYS, les bases d'un nouveau système informatique modulaire, destiné aux applications techniques, ont été élaborées en Hollande (Netherlands Modular Engineering System). L'article présente les principaux critères qui doivent guider les développements s'appuyant sur une technologie moderne. Un inventaire des besoins des utilisateurs est établi ainsi que l'intérêt manifesté par l'industrie lors d'une enquête sur la CAO. Des suggestions sur l'organisation et les aspects techniques du développement du projet NEMESIS sont formulées en guise de conclusion.

ZUSAMMENFASSUNG

Dank enger Zusammenarbeit zwischen den Anwendern des GENESYS-Systems wurden neue Gedanken in Bezug auf die Anwendung technischer Systeme ausgearbeitet, die zum Entwurf eines neuen niederländischen technischen Modulsystems (Netherlands Modular Engineering System: NEMESIS) geführt haben. Der Text behandelt wichtige Erfahrungen mit dem Gebrauch des GENESYS-Systems und Kriterien für die Entwicklung eines neuen Systems auf der Grundlage der heute verfügbaren modernen Technologie. Eine Liste der Anforderungen der Anwender wird gegeben, sowie eine Übersicht über die Resultate einer Umfrage über die CAD/CAM-Interessen bei der niederländischen Industrie. Organisatorische und technologische Aspekte des NEMESIS-Entwicklungsprojekts werden erwähnt.

CONTENTS

- Introduction
- Experiences with the Genesys system
- Towards a new engineering system
- Engineering user environments
- User requirements
- Technology of Nemesis
- Nemesis project
- Conclusions
- Acknowledgements

INTRODUCTION

During the IABSE Colloquium on 'Interface between computing and design in structural engineering', Bergamo, Italy, 1978, the author presented a paper entitled 'The unknown triangle', in which paper relationships between a human being and his surrounding environment are described as subjective experiences. Through education, training, cooperation or other ways of exchange of experience between human beings, subjective experiences may become intersubjective experiences. Natural languages, sciences, political trends, religions etc. have at least some parts in them that bind groups of people together in such a way that the relationships between members of such group may be considered to frame an intersubjective reality.
The same applies to engineers. They have their technical background - civil engineering, mechanical engineering or otherwise. Within each discipline there are specialists like structural engineers who have built up their intersubjective reality focused on design and realization of the structures that provide strenth and stability to our built-up environment.
During recent years computer based tools have become available that enable engineers to store and process data they use for the design and construction of these structures. Data can relate to the design process, such as geometrical descriptions of the structure, structural materials, deformations and strength behaviour etc., to the production preparation, such as technical specifications, component lists, planning and routing schedules etc., and to the production itself. In the latter case one can think of NC machines, optimized material usage, material and product storage management etc.
Computer based tools have in some case become rather specialized. Examples are automated draughting systems and programmable handlers like industrial robots. Although still primitive in comparison to human abilities, fascinating examples are available of robots, equipped with vision and tactile sensors, which have a 'learning' capacity.
Designers of computer based tools for production engineering look at present production procedures, analyse and streamline working methods and information handling process, trying to create tools that fit into and integrate the production process. In this paper the author will confine himself to the design process and report on attempts that are being made in the Netherlands to develop a 'generally applicable engineering system'. The closing remark[1] provided at the end of the above mentioned paper contains a stimulus for the development of such a system but it also encompasses discouragement, disappointment and ultimate failure if the subjectivity of the experience of human individuals is not taken into account as a major fact of life.

[1] From the foregoing (the paper) it may be clear that to attempt to define a general interface between design and computing is fruitless. Each designer and program developer will define his own interface. Through continuous discussions and cooperation these individual interfaces will be unified and at least match, at least a little.

It is therefore not without reason that the new engineering system has been named NEMESIS (Netherlands Modular Engineering System) after the Greek goddess of retribution for human recklessness.

AN ENGINEERING SYSTEM

Take a micro CPU chip in one hand and a detailed drawing of some structural component in the other.
Although the micro chip in the hands of a designer can produce the drawing, there is quite a job to perform and specialized experiences to provide in order to be able to elaborate the chip into an engineering tool that enables the designer to carry out the necessary design calculations and to produce the drawing.
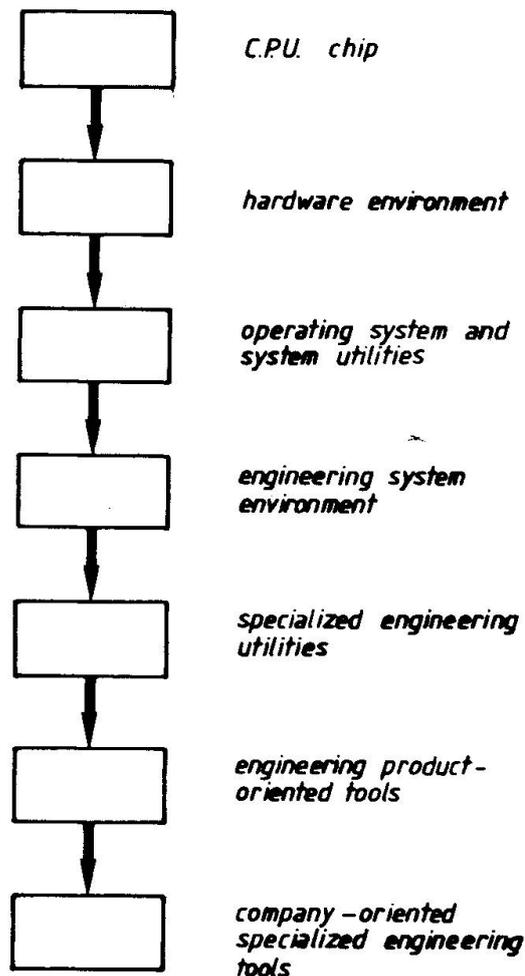

Fig.1   Chain of specialized knowledge required
          to make a micro chip useful to an engineer
          in his company-environment

The idea of bringing together a group of computer programs and extracting from them common functions like data structuring and handling functions, mathematical functions, drawing or plotting functions, I/O functions based on a so-called problem-oriented language (POL), etc. is not new.
The integrated Civil Engineering System (ICES) and the General Engineering System (GENESYS) are - amongst others - developments of the sixties and they are still in use after almost twenty years.
Once a user has learnt the POL methodology, he will be familiar with all the engineering application programs contained within the system.
Programmers who have learnt to use the engineering system functions (in the case of ICES contained in the ICETRAN language and in the case of GENESYS in the GENTRAN language) will be able to effectively produce userfriendly programs for the engineering end user.
Although these engineering systems are still in use and have enthusiastic supporters, be it within the worldwide ICES User Groups (IUG) or on a smaller scale the Association GENESYS Netherlands (VGN), the existing engineering systems cannot easily be adjusted to the new developments in hard- and system software. The systems are based on hardware concepts from the sixties. System components cannot easily be replaced and the system architecture is of a 'closed' type.
To connect such a system with an external utility such as a DBMS (Data Base Management System) or with an automated draughting system is quite a costly job. This is not a characteristic of only these engineering systems. There have become available very powerful (and costly) draughting system which fail to have possibilities to connect external systems^2 . Other disadvantages of existing engineering systems are the lack of effective programming aids. Although the dedicated programming languages like Icetran and Gentran enable the programmer to write much more user-friendly and effective programs with the aid of built in dedicated I/O functions (data-entry and report functions, command interpreters and definition functions), the writing of these programs requires more effect than direct programming in e.g. Fortran. Managers of engineering user environments cannot easily be persuaded to adopt an engineering system that costs a yearly licence fee, imposes a computer overhead, requires extra education of end users and programmers, leads to more expensive computer programs which cannot use specific features of the in-house computer systems and where the cost-effectiveness of the problem solving process cannot be proven by the system.
Notwithstanding these disadvantages it is a fact that user environments can become enthusiastic about the possibilities such systems offer to improve the quality of the problem solving process.
Quality improvement results from the high level user interface which the engineering problem solving programs (application programs) within an engineering system offer to the designer. Through this improved communication technique the designer can 'play around' with his problem. He can easily vary structural parameters and investigate in this way the sensitivity of his structure to deviations of actual values from assumed values during design, or vary the geometry of the design and prove that the geometry he supposes to be an optimal solution, really meets the requirements.
Experience results from 'trial and error',trespassing borders marking commonly accepted impossibilities.
Just watch a playing child trying to impose its imagination on the environment, leading to confrontation and thus to experience. Artists perform experiments, using all kinds of new techniques in order to find new ways to express themselves. Why not provide such facilities to the designer, so he can play around with the procreations of his mind? Isn't that the intrinsic value of Computer Aided Design? Engineering systems shall provide an effective CAD toolkit to the designer to enable him to gain experience on the products of his mind. He cannot afford a structure to collapse in the real world, but in his subjective world models should have an ability to show consequences of ultimate design.

^{2} see also FACE-report 2 'The automation of draughting work' [17]

EXPERIENCES WITH THE GENESYS SYSTEM

In 1972 the GENESYS system was presented officially to the engineering society
at Loughborough, UK, and in 1972 the Department of Waterworks in the Netherlands
(Rijkswaterstaat) signed a licence-contract with GENESYS Ltd. Within the
Association of Computer Users in Engineering (CIAD) a project group 'Integrated
systems' showed much interest in the GENESYS system and in the developments of
engineering application subsystems that were performed by the Rijkswaterstaat.
The system became used by consulting engineering firms and others, but the
expected widespread use by engineering environments did not show up. Main bottle-
necks were the relatively high licence fee for the main system, lack of
specialist subsystems based on Dutch codes of practice, thresholds to learn and
use new working methods and the fact that a governmental body is not equipped to
promote externally the use of such a system. The GENESYS system design is such
that about 95% of the Fortran source code is computer-brand independent and the
remaining 5% of the code contains the interface to a particular computer and its
operating system. The support of many different implementations appeared to be a
limiting factor.
In 1975 discussions started between Rijkswaterstaat and the project group on the
establishment of an independent GENESYS Centre in the Netherlands, resulting in
the formal charter of the Association GENESYS Netherlands in the beginning of 1977.
The Association groups together users and potential users of the GENESYS system.
Shortly afterwards the Foundation GENESYS Netherlands was created by CIAD, Rijks-
waterstaat and the newborn Association. It was the intention (and in the meantime
this has taken place) that after some years both CIAD and Rijkswaterstaat would
withdraw from the Board of the Foundation and the vacant seats within the Board
would be filled by representatives of the Association.
It is the main purpose of the Foundation to exploit the GENESYS system in a
professional way, whereas the Association is a forum for the users where they
can exchange experiences, work together on developments and express their
requirements in a structured way towards the Foundation. After about five years
this organisational structure has been settled and it is functioning quite
successfully.
The Association applies the same success formula for its functioning as CIAD,
which is not surprising because there is a strong connection to CIAD. Most of the
members of the Association are members of CIAD too.
During the five years of its existence representatives of the 15 member
organisations have worked hard on a number of subjects related to the improvement
of the effective use of the GENESYS system. A total number of about 50 reports
have been produced.
Subjects of reports are
- organisational procedures and annual reporting (11)
- evaluation of existing (British) subsystems (3)
- development reports on new (Dutch) subsystems (4)
- bottleneck investigations on the use of the GENESYS system (3)
- new GENESYS implementations (1)
- inquiry on user requirements for and development of draughting
  facilities attached to the GENESYS system (9)
- development of facilities for interfacing existing Fortran programs
  to the GENESYS system (4)
- development of quality classification standards for subsystems to be made
  generally available through the organisation (4)
- investigation on improvements of the existing GENESYS system (2)
- investigation into and preparation of the development of a new
  engineering system NEMESIS (12)

Which means that much effort has been given to the consolidation of the
Association in order to obtain a firm basis for extensive exchange of information,
quality improvement and the possibility to effectively deal with new arising needs

of the users. Especially the initiative to develop ideas on the use of computers
by engineers in near future, taking into account new technologies both in hard-
ware and in software, attracted new members to the Association, who were not
GENESYS users themselves. It is felt within the Association that the original
objectives of the developers of the GENESYS system still hold.
- uniform main system with general functions suitable for use within engineering
  application programs
- high degree of portability of main system and subsystems, with the computer-
  dependent operating system interface concentrated in the main system
- standardized and unambiguous user interface and documentation
- high degree of user friendliness,

In marketing the GENESYS system in the Netherlands the Dutch GENESYS organisation
(Association + Foundation) with a number of thresholds which present potential
user environments to use the system.

Important thresholds are
- in case of existing computer using engineering environments (non-GENESYS-users)
  . effort to be made by the user environment to become familiar with the
    specific GENESYS working method.
    This Threshold is in agreement with [15] where the development of a level 4
    I/O standard is not recommended as long as the previous levels are not
    generally accepted. The I/O user interface of the existing GENESYS system
    meets - at least to a high degree - the requirements for level 4.
  . resistance from existing users who are familiar with existing Fortran pro-
    grams. As a result of the often cumbersome I/O rules (which differ per
    program) experienced users become experts. It is their skill to use such
    programs. User friendly programs make their skill redundant.
    The general question is 'Why should we change over to a new system with all
    related problems, both financial and organisational, only to get available
    programs we already use in Fortran'.
    Only where GENESYS subsystems provide problem-solving tools which are
    required at that time experienced user environments may become
    interested to look at the subsystems and thus at the system.

- in case of existing GENESYS using engineering environments
  (thresholds to put more effort in extending the use of the system over the
  organisation, e.g. involving more departments or writing new subsystems)
  . up to recently the GENESYS system was available only on mini and main frame
    computers. Especially in the latter case the volume of required direct
    memory can be disadvantageous to multi-user environments. Users sometimes
    have to wait rather long before the 'large-GENESYS-program' can run on the `
    computer.
  . writing subsystems requires programmers trained in the use of the Gentran
    programming language. Since there is little experience available, programmers
    have to become experienced mainly through 'trial and error', which is not
    the most efficient (although effective) way.
  . although the use of user-oriented command structure almost obliges, but at
    least seduces, the programmer to design a functional model, the GENESYS
    system does not provide programming aids to define a structured process or
    a process-oriented data structure. Some programming aids are available (e.g.
    the TRACE statement), but testing is still as uneasy as in normal Fortran.
  . as the GENESYS system is almost a 'closed' system, other software systems
    or application programs can hardly be connected to the GENESYS system. The
    GENESYS system does not provide facilities for external data transfer or
    working within a network.
  . as the GENESYS system is developed in the late sixties the system architec-
    ture is based on hardware and software concepts known at that time. Modern
    concepts available cannot be used within the system and its subsystem (e.g.

menu-techniques, digitizers, text processing, draughting techniques).
as the GENESYS system is one monolith structure, it cannot easily be
tailored to the needs of individual user environments (e.g. solving a
small problem using a small subsystem requires almost the same computer
resources as solving a large problem using a large subsystem).

The general feeling within the Association is, however, that the original GENESYS-
'idea' was and still is excellent. The resulting GENESYS product - developed
about 14 years ago - is still usable for small engineering environments which
start using computers. As a micro-version of the GENESYS system is available
now, hardware costs will not be a limiting factor for these environments
anymore.
As soon as those engineering environments start to make their requirements more
advanced and want to integrate modern techniques they become trapped by the
system.

An often heard objection from opponents of the GENESYS system is
the computer-runtime overhead required by the system.
Advocates of the use of the system reply with the statement that the required
man-hour time to solve a problem is much less. However, no data were available.
One of the projects of the Association focused on this problem.
At the University of Technology at Eindhoven in the Netherlands a test was
performed using three groups of students; none of them had any knowledge of
technical application programs and most of them even did not have any
experience in the use of computers.
Each of the students had to compute stresses in a truss structure under several
loading conditions.
One group had to solve the problem with help of the GENESYS-Vlasko program,
another group with the ICES-Strudl program and the third group with the STAEX
program. The latter program being a normal Fortran program with a general
cumbersome input structure.
The following figures resulted from this test.
Required man-hours (fortran-program taken 100)

	STAEX	ICES-STRUDL	GEN.-VLASKO
acquaintance with user manual	31	27	10
input preparation	34	49	26
input correction	25	13	22
output interpretation	10	5	7
	100	94	65

From these figures it may be seen what influence an easy input preparation
- through easy understanding of the underlying model - has on the overall
required man-hours.
A similar conclusion can be drawn from the figures related to the average amount
taken 100)

	STAEX	ICES-STRUDL	GEN.-VLASKO
Average of required times of assistance	100	108	77

The figures on required computer time to solve the problem (including error
runs) were not very consistent. Leaving out data which could not be explained
(the students had to collect the data) the following figures resulted from
the test (Fortran program = 100)

	STAEX	ICES-STRUDL	GEN.-VLASKO
Average required computer time/run	100	219	363

Due to the uncertainty of the test data, these figures should not be taken too literally, but a system overhead of about a factor 2 for the GENESYS system has been found in other circumstances too.
If the costs of computing for solving a problem, using normal Fortran programs, are taken at about ¼ of the man-hour costs, which assumption does not seem to be unreasonable at this time, following figures would result from changing over to the use of an equivalent GENESYS subsystem (man-hours costs of Fortran-program = 100)

	Fortran program	GENESYS-subsysteem
man-hour costs	100	67 (factor 2/3)
computer costs	25	73 (factor 3)
Total costs	125	142

With a tendency of increasing man-hour costs and decreasing costs of computing, the cost figure will favour the use of engineering systems in future.
In some cases the costs of personal (micro) computers are accounted for in an overhead percentage on the man-hour costs. In those cases the use of an engineering system (especially an easy to use interface in conjunction with an easy to understand user manual or other means of information transfer) directly pays out. Easy use and understanding means, however, a larger environment where the rules and standards are generated and generally accepted.

TOWARDS A NEW ENGINEERING SYSTEM

At the start of the Association a survey was held among the GENESYS users on existing bottlenecks in the use of the system. Main bottlenecks have been described in the previous chapter. As a result of the survey the organisation GENESYS Netherlands contemplated its policy. Three policy lines were defined to meet the stated user requirements.

policy line 1 : Improve facilities of the GENESYS system through additional developments.

Major developments were
- interface package to Fortran programs ('Fortran-hook')
- draughting package (Calcomp-Gino-F simulation)
- micro GENESYS on 8-bit micro computer under CP/M

policy line 2 : Discuss with British contract partner renewal of the existing GENESYS system

Major developments were
- investigation of future market potentials of the GENESYS system
- investigation of the feasibility of merging the GENESYS system with a more recently written complementary system (BOS-system).

policy line 3 : Definition of a new, engineering system to be developed in the Netherlands

Basic starting points (for a new system which has been given the name NEMESIS are
- continuation of present working methods and attained standardization of computer use
- continuation of investments in existing GENESYS subsystems
- anticipation on the general use of micro computers
- system to be based on an 'open-growth' architecture, which means the system must have the ability to be tailored to the user requirements and existing hardware in a particular user environment together with the ability to grow simultaneously with growing needs of this environment
- knowledge on system software design and development to be increased and intensified in Dutch engineering software industry through system development

- continuous standardization of computer use to be based on continuing
  inventory of user requirements
- existing international contacts to be maintained and intensified.

The policy statements under this line are more of a long-term character and aim
at innovation of Dutch industry from an informatics point of view catching up
with developments in the industrialized countries.
A project group was installed with the task to define preliminary specifications
for the new system and to investigate the feasibility of development.
The project group produced in three years a total number of 9 technical reports
with the following subjects

- collection of building stones for the approach of the task to be performed
- inventory of similar, already existing systems or intended developments as
  far as known
- inventory of user requirements, based on the building stones collected
- general description of a concept for NEMESIS
- related work documents produced by the project group and other relevant
  documents
- investigation on the user friendliness of a GENESYS subsystem compared with
  other similar application programs
- the technology within NEMESIS
- inventory of on-going research projects at universities and research centres
  in the Netherlands of which results can be useful to the NEMESIS development
- NEMESIS development strategy
- NEMESIS project organisation

As the NEMESIS system is intended to become infrastructural software for CAD/CAM
applications a step-wise development is foreseen, each successive step resulting
in an improved system relative to the previous step. A start should be made with
the development of a relatively simple system, followed by several steps in order
to reach a fully grown-up system and going over in an 'innovation permanente',
where new technologies from the market are continuously incorporated in the
system.
The development of NEMESIS is foreseen as a catch-up manoeuvre of Dutch industry
with CAD/CAM developments in industry in other countries.
It is recognized by the project group that the propagation of objectives of
system development and spread of knowledge to be within the system and the
attainment of enthusiastic involvement of large numbers of potential users is of
the same order of importance as the incorporation of new and advanced technology
in the system.
Especially the abstract character of the system makes it difficult to persuade
potential users of the future applications software to become attached to the
system and to understand the necessity of such system development, and they have
difficulty in estimating their future interest. Only with help of the existing
GENESYS system and subsystems one can show on a piece of (micro) hardware what
is meant. There is an enormous gap between the NEMESIS development environment
and non-computer oriented managers of engineering design and construction
environments. 'Unspoiled' managers believe computers can be used by engineers in
the same, simple way as they can drive their cars and providing the tools to
make that attitude possible in future is exactly what the developers of NEMESIS
are aiming at.
The gap between results from the complexity of human activities and flexibility
of human behaviour in comparison with the 'stupidity' or low-level artificial
intelligence of today's computers.

At several locations in the Netherlands CAD/CAM developments in industry are on
their way. However, only large companies can afford to investigate possibilities
of implementati   CAD/CAM tools in their production processes.
Small and medium-sized companies do not have resources to follow developments

in the market 'to play' with possibilities in order to find out what is needed
and what is useful. They have no time and money to analyse their production
processes and to select and implement adequate computer based tools in those
processes. In the diagram a qualitative approach of the indicated CAD/CAM
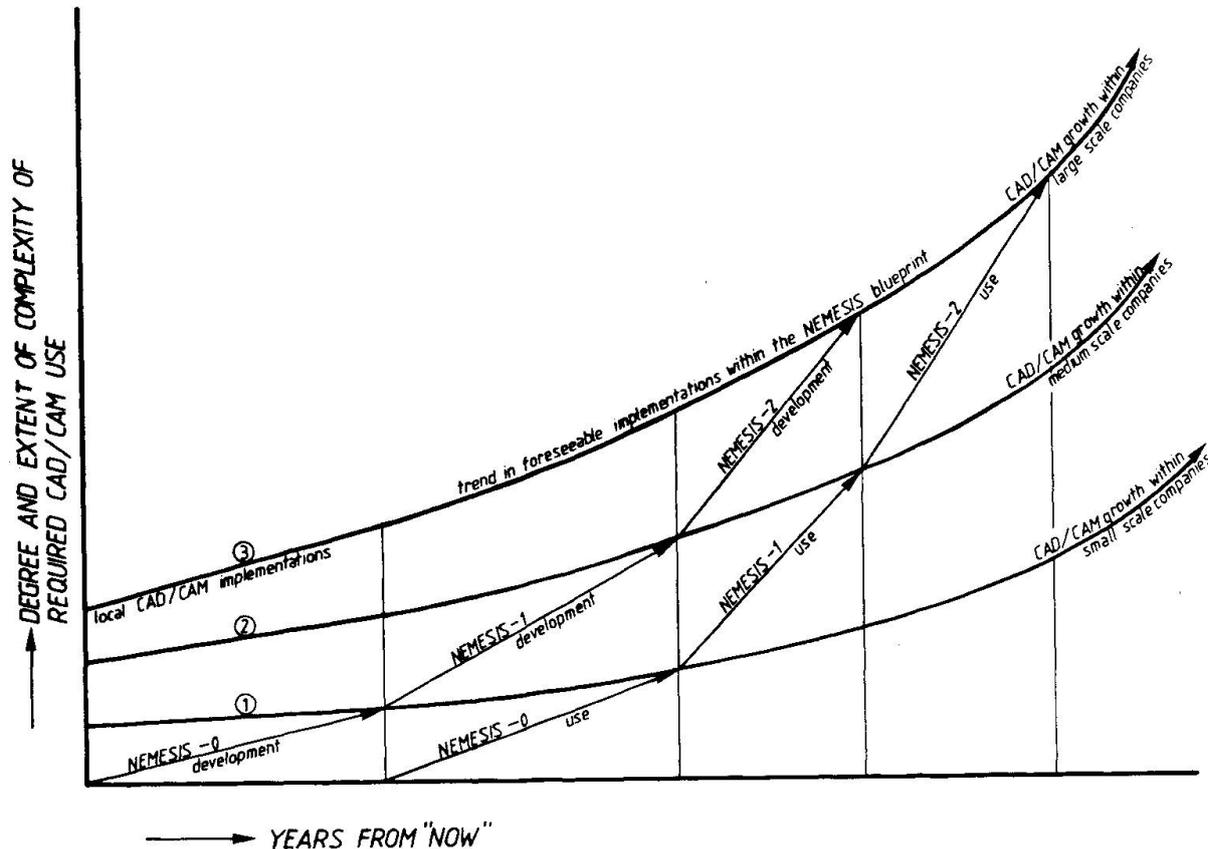problematics is shown.

Fig.2 NEMESIS as a CAD/CAM catch-up manoeuvre

(1) large scale companies
(2) medium scale companies
(3) small scale companies

From a survey on the interest in CAD/CAM among Dutch industries carried out by
a joint steering group of CIAD, Royal Institute of Engineers and the Netherlands
Association on Informatics (INCAD-project) following figures resulted. The survey
was done with a questionnaire sent out to 5309 engineering companies, representing
about 16% of the total number of engineering companies in the Netherlands
(estimated total number 31.285). The response was rather high. 15% of the
questionnaires was returned, which already points at a relatively high interest
of the companies on the subject. The interest of the respondents appeared to be
differentiated according to

- interest in receiving adequate information   22%
- interest in consultancy (both providing and receiving)   29%
- interest in cooperative efforts   16%
- interest in governmental, financial facilities   22%

Expectations on CAD/CAM becoming of current interest (cumulative)

Expectations on CAD/CAM becoming of current interest (cumulative)

	CAD	CAM
- at this time	14%	8%
- within 2 years	23%	12%
- within 5 years	37%	21%
- within 10 years	51%	31%

From these figures it may be concluded that the potential market for CAD/CAM infrastructure software can be estimated at a minimum of about 0.51 x 0.15 x 31285 = 2400 companies. If about 10% of these companies can be stimulated to use the NEMESIS system it means that about 240 companies will have to carry the financial burden of the system development and maintenance.

The magnitude of engineering companies, expressed in number of employees, and response percentages are provided in the following table.

	Percentage of total number	Response	Interest in CAD	CAM
0 - 20 employees	84.5%	12%	28.2%	15.8%
20 - 50 employees	9.3%	24%	8.6%	5.5%
50 - 100 employees	3.5%	29%	4.1%	3.8%
100 - 500 employees	2.3%	55%	7.0%	3.9%
500 employees	0.4%	100%	3.0%	2.2%
	100 %		50.9%	31.2%

Related to the above mentioned group of 240 potential user companies the distribution would become

	number	percentage
0 - 20 employees	132	55
20 - 50 employees	41	17
50 - 100 employees	19	8
100 - 500 employees	34	14
500 employees	14	6
	240	100

More than half of the companies potentially interested in the system appear to have less than 20 employees.

A preliminary estimate of development costs is $Dfl.10.10^6$, which would mean a contribution for development costs of about f 42.000,= per company together with yearly costs for maintenance and renewal estimated at 25% of the development costs resulting in about f 10.500,= per year.
It must be borne in mind that these costs only relate to the NEMESIS infrastructure software so these costs have to be increased by costs of company dependent application software. The developers are convinced that the development costs for the infrastructure software are too high for small and medium-sized firms. As a result the Dutch government has been asked to bear these costs.

ENGINEERING USER ENVIRONMENTS

Computer programs are written for users. In the case of engineering programs the users are engineers. Engineers who want to solve technical problems with the help of computers. The program is a means of communication between the engineer and his tool, the computer. Next to the skill of problem solving, the engineer is expected to be a skilled operator of the program on the computer. He has to

'translate' his technical problem into the mathematical model the program is
based on. In general the problem solving engineer using a program-tool is called
'engineering end user'. In order to do so, the program shall be presented to the
engineer in such a way that he can easily understand the program. Which means,
the program must fit into his intersubjective reality.
In some case the engineering end user is the program writer too. In that case he
has entered the world of computer specialists. He has learned some programming
language in which he can model the problem he wants to solve.
Because modelling a problem into a mathematical model, programming and testing
it, will require quite an effort, it has become common practice to make the
problem solving process more abstract, cq. the problem to be solved is
generalized in such a way that similar problems fit in as well.
The more the problem-solving process is generalized, the more effort the
engineering end user will have to put in at the moment he wants to solve his
particular problem.
There grows a stress field between the costs of programming and the costs of
problem solving. In administrative environments almost equal problems (e.g.
salary computations) are solved over and over again with a program that is very
near to the problem. In engineering environments specialist problems are not
often repeated, problems have a tendency to be quite different each time the
engineer wants to use the program. In order to spread the costs of program
development, programs are therefore designed as general as possible.
The art of programming has become a skill on its own. Problem solving has been
stated as an information process, which can be analysed, structured, systemized,
programmed etc. The problem solving process of the programmer is program
development. The result is a problem solving process for the engineering end
user. The programmer, from the own nature of his work is familiar with the use
of computers and thus searches for programming tools. For sake of systematics
he is called 'the application programming user'.
The application programming user could make the problem solving tools he needs,
but like it was with the engineering end user, new skills are required and
this new expert user of the computer system is called 'the system developing
user'.
In the context of the development of an engineering system, this user of a
computer system is the last category taken into account. Otherwise we would end
up with an integrated circuit designer, solving the design problem of an
electronic component for a new generation of computers.
Aside the technical hierarchy there is the manager of the technical problem
solving process. He is interested in the results of the process, which is not
only the quality of the answer, but also the performance of the problem solving
process. The effectiveness and efficiency of the process is (part of) his
management task. In case of providing computer services as a separate commercial
activity the performance of the problem solving process even becomes a major
item. The manager could want to monitor and measure the performance of the system
with help of the system and so he becomes a user himself, 'the system performance
user'.

Up till now individual user types have been dealt with. One can think of several
users of the same type forming together a user environment of that type.
However, in almost any case, there is a mixture of user types. Almost any user
environment will have users of different user types.
In the case of the NEMESIS preparations, four different user environments have
been distinguished as a more or less defined mixture of the individual user types
mentioned.
This has been done because the system as a whole has to serve the user environment
within a company. The system shall need adequate interfaces to the different user
types and have an appropriate structure to tailor the system to the need of the
user environment. The following user environments have been defined :

- engineering environment, where the engineering end use of the system predominates.(at least 2/3 of the system use by engineering end users)

- research and development environment, where the development of application programming together with engineering end use predominates (at least 2/3 of the system use by both user types, but not more than 1/2 in one of the groups)

- information processing environment, where the centre of gravity of activities lies in application programming together with system development. (at least 2/3 of the system use by both user types, but not more than 1/2 in one of the groups)

- service environment, where the performance management of the system predominates (at least 2/3 of the system use deals with performance processing. In case of a service centre the internal use is measured only).

Fig.3   Matrix of user environments and user types

Environments can be subdivided in case a more detailed differentiation is required. One can think of the engineering end user environment split up in

- engineering management
- engineering design and analysis
- engineering administration (specs. cost estimates, planning etc.)
- engineering drawing
- engineering construction
  etc.

Such a subdivision has a meaning only if the system provides special facilities
to tailor the system to each sub environment.
Of course the definitions of user environments do not cover all possibilities
but a preliminary inventory of user environments within the Association proved
that all members could be typified unambiguously by one of the defined user
environments.
In 1980 the number of members of the Association was 12,

- engineering environment   6 members
- R&D environment   4 members
- information processing environment   1 member
- service environment   1 member

USER REQUIREMENTS

An engineering system as is the case for any program shall be based on user
requirements. User requirements, however, are quite difficult to collect, because
a user cannot develop needs if he is not aware of bottlenecks in the existing
situation and possibilities to improve.
Satisfied users have no needs and thus no requirements. Satisfaction is an
integral acceptance of a situation without desires to change the situation.
Ignorance of possibilities to improve often leads to satisfaction. The reverse
approach is a general marketing strategy. Stress the benefits of new products in
comparison with existing products and users will become dissatisfied with their
present situation. The same applies to ideas. From the scala of available
possibilities users develop desires and based on these desires more dedicated
possibilities can be developed. It is an interaction between questions and
answers between users and developers.
To this and the NEMESIS project group worked out a set of preliminary
requirements and had these judged by existing user environments.
From dissatisfactions resulting from the use of GENESYS facilities in comparison
with new facilities available on new technological products, more general
requirements have been deduced :
The NEMESIS system shall constitute a system frame work that

1. is an effective aid to the various users

2. is adaptable to present and evolving needs and computer facilities of
   individual user environments

3. guarantees continuity of working methods, conventions adopted and investments
   made

4. provides a common basis for cooperation and exchange of information

From these main requirements a top down structure of more detailed requirements
has been developed.

1. Effective aids to users

(A prelimninary subdivision under this heading has been made on basis of the
 user type specification from chapter 4).

1.1   Effective aids to engineering end users

1.1.1 The system must provide an easy-to-use interface between the user and the
      engineering problem solving aids.

1.1.2 The system must provide a systematically ordered range of day-to-day
      engineering problem solving aids.

1.1.3 The system must provide advanced engineering problem solving aids.

1.1.4 The system must provide convenient aids to store, handle and represent
      engineering data.

1.2   Effective aids to application programmers

1.2.1 The system must provide an easy-to-use interface between the user and the application programming aids.

1.2.2 The system must provide a range of aids for structuring and programming application programs.

1.2.3 The system must provide libraries of well-tested and documented modules to be used in application programs.

1.2.4 The system must allow R&D environments to implement advanced facilities at a preliminary stage of development.

1.2.5 The system must provide convenient aids to become trained in and have support on the use of the system and its programming aids.

1.3   Effective aids to system programmers.

1.3.1 The system must provide system programming aids.

1.3.2 The system must provide convenient aids to store, handle and represent the system and related documentation.

1.3.3 The system must provide convenient aids to become trained in the system and the system programming aids.

1.3.4 The system must provide convenient implementation aids.

1.3.5 The system must provide means to interface the system to other systems.

1.4   Effective aids to system performance users.

1.4.1 The system must provide an easy-to-use interface between the user and the the system performance aids.

1.4.2 The system must provide a systematically ordered range of management aids to manage a cost-effective use of the system.

1.4.3 The system must provide convenient aids to store, handle and represent data related to management aids.

1.4.4 The system must provide convenient aids to become trained in the use of the system and the management aids.

2. Adaptability to present and evolving needs and computing facilities of individual user environments

(A preliminary subdivision under this heading has been made on basis of hardware, software and working methods)

2.1   Adaptability to present and evolving hardware configurations.

2.1.1 The system must be portable over a wide range of computer configurations, ranging from relatively small to very large computer configurations.

2.1.2 The system must be adaptable to new hardware developments.

2.1.3 The interface between the user and the system must be brand-independent.

2.1.4 The system must provide upward compatibility in the interface between user and system, dependent only on the magnitude of the computer configuration.

2.1.5 The system must provide convenient and adequate aids to time an application program on a particular machine.

2.2   Adaptability to present and evolving software systems

2.2.1 The system must be adaptable to a wide range of operating systems, ranging from relatively simple operating systems to very complex systems.

2.2.2 The system must be adaptable to new developments in operating systems.

2.2.3 The system must be connectable to other software systems.

2.2.4 The system must be extendable by individual user environments.

2.2.5 The system must have an architecture that allows replacement of individual facilities.

2.3.   Adaptability to present and evolving working methods.

2.3.1 The system must provide a smooth change-over from present individual working methods to system oriented practice at the individual pace of the user environment.

2.3.2 The system must provide adaptability to evolving working methods.

3.   Continuity on working methods and investments

(A preliminary subdivision under this heading has been made on basis of maintenance, education, improvement, compatibility)

3.1   Maintenance.

3.1.1 The system must provide convenient and systematic means to maintain the system.

3.1.2 The system must provide convenient and systematic means to become acquainted with the system.

3.2   Education

3.2.1 The system must provide continuity for existing and future investments in engineering education.

3.3   Improvement.

3.3.1 The system must provide convenient and systematic means to improve the system.

3.3.2 The system must provide appropriate means to administer the sequence and status of improvement.

3.3.3 The system must provide the ability to recover earlier versions.

3.4   Compatibility

3.4.1 The system must provide upward compatibility in successive improvements in order to provide continuity to investments in education, data structures, program and skill.

3.4.2 The system must provide means to use existing GENESYS 2.6 subsystems.

4.   Common basis for cooperation

(A preliminary subdivision under this heading has been made on the basis of standardization, exchange of experience and marketing).

4.1   Standardization

4.1.1 The system must incorporate present general standards and directed towards standards under development.

4.1.2 The system must correspond with existing GENESYS input-output conventions.

4.2   The system must be an adequate means to stimulate cooperation between users from different user environment.

4.2.2 The system must be an adequate means to be used by scientists and research workers in order to stimulate throughput of research know-how to practising engineers.

4.2.3 The system must be an adequate means to stimulate the exchange of subsystems, basic software modules, problem oriented modules and data structures between user environments.

4.2.4 The system must be an adequate means to be used by engineering education in order to prepare students for engineering practice.

4.3   Marketing

4.3.1 The system must be adequate means to be used by service organisations.

Based on these requirements an inquiry has been made among existing GENESYS user environments within the Association. The respondents consisted of 6 engineering environments and 2 R&D environments according the definitions provided in the previous chapter. Results are contained in [4] .
The priority sequences of the engineering environments and R&D environments are given in following schemes, where priority 1 means highest priority.

requirement	eng.env.	R&D env.
effective aids	4	3
adaptability	2	1
continuity	3	4
common basis	1	2

effective aids	eng. env.	R&D env.	adaptability	eng. env.	R&D env.
eng. end users	1	2	hardware env.	1	1
appl.prog. users	2	1	software env.	2	2
syst. prog. users	3	3	working methods	3	2
system prog. users	4	4			

continuity	eng. env.	R&D env.	common basis	eng. env.	R&D env.
maintenance	2	2	standardization	1	3
education	4	1	exp. exchange	2	1
improvement	3	2	marketing	3	2
compatibility	1	3			

From these priority schemes interesting conclusions can be drawn on the attitudes of the distinguished user environments. Engineering environments seek for unification, whereas R&D environments want to spread their knowledge. The relatively low appreciation by engineering environments for adaptability to working methods and education may result from the general unaquaintance with the process aspect of working methods and training facilities.
Next to the above mentioned requirement hierarchy a collection has been made of existing features which could be built into the system. Of course this would become a rather incoherent collection if no systematization was done.
Within the system five main functions have been distinguished and each main function is dubdivided into ten general subfunctions together forming what has been called a 'cabinet of functions'.
In each box required features can be placed, together giving a rather extended collection which is very useful in the design of specifications for an engineering

system to be newly developed. This 'checklist' has been provided in [3] .
The inquiry, mentioned before contained judgement of these features too.
The defined main functions are :

- run process management
  Through this function the user must be able to define and control the hardware
  he wants to work on and to monitor the system performance.
- training management
  Through this function the user must become aware of the capabilities of the
  system and become trained in the use of these capabilities.
- data management
  Through this function the user must be able to store, handle and represent
  data. For the engineering end user these data will apply to engineering data,
  for the application and system programmer to program data (statements,
  structures and program independent figures) and for the permance user to
  system measurement data.
- process management
  Through this function the user must be able to transform data sets into other
  data sets. (problem data in- and output) He must be able to monitor system
  provided processes of data transformation.
- documentation management
  Through this function the user must be provided by the system with up-to-date
  documentation (= explanation) on the system.
  Documentation data shall be coupled as much as possible within the software.
  Documentation in the form of paperwork, which cannot be generated by the system,
  must be kept to a minimum and be as imperturable as possible to modifications in
  the software in order to prevent incompatibility between software and documen-
  tation.

The defined subfunctions are :

- specification   (of what has to be done or stored)
- error control   (checking of specifications and relations to other
                          specifications)
- monitoring   (of what is going on or stored)
- intervention   (in what is going on)
- modification   (of what has been specified or stored)
- deletion   (of what has been specified or stored)
- measuring   (of what is going on)
- protection   (of what has been specified or stored)
- security   (on what is going on or has been stored)
- representation   (of what is going on or has been stored)

In the scheme below priorities are shown as they have been indicated by the
user environments

main function	eng. env.	R&D env.
run process man.	2	4
training man.	4	5
data man.	1	1
process man.	2	2
doc. man.	3	3

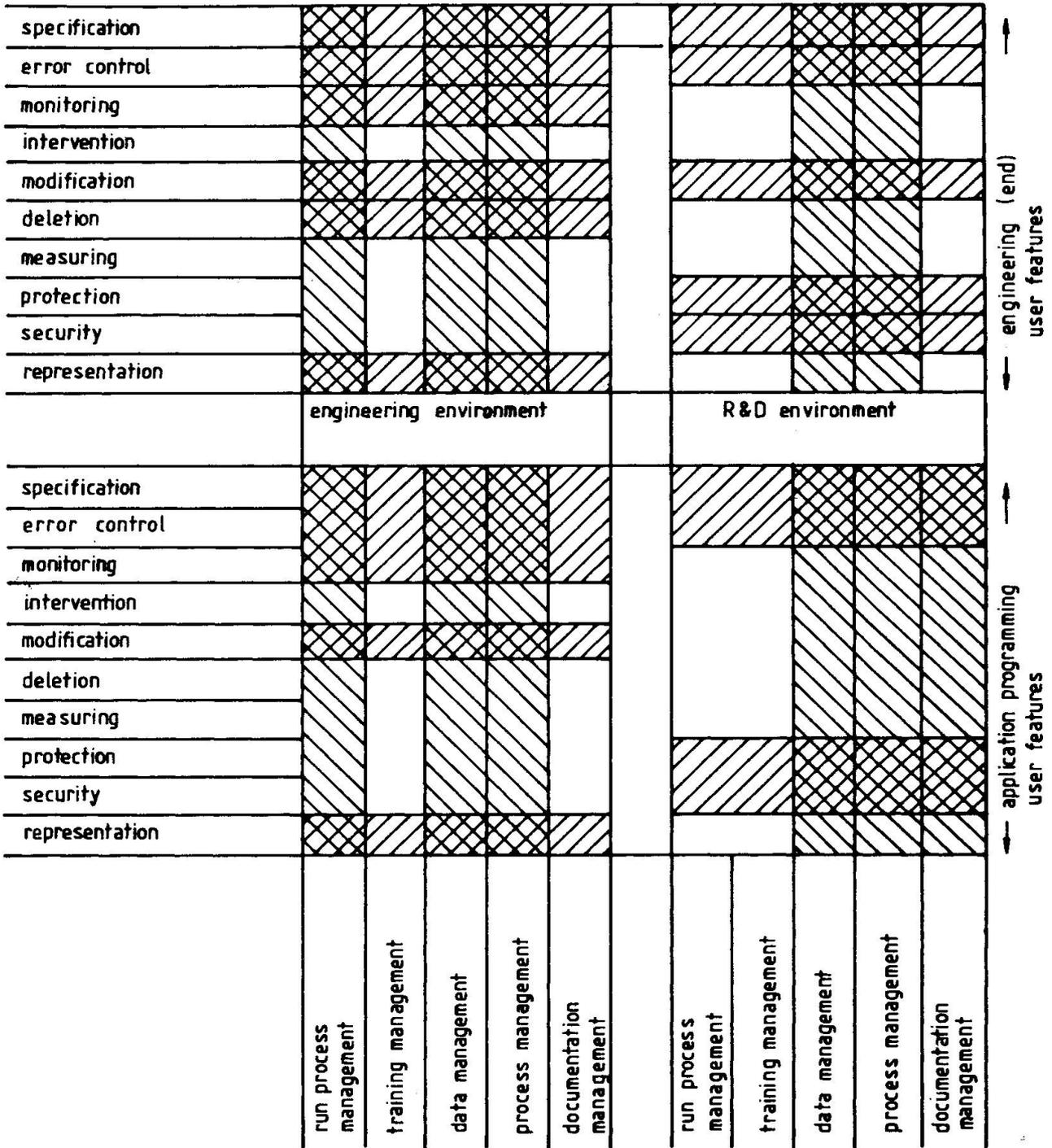In the following schemes columns and rows above average interests are shown.

Fig.4 Main and sub-function making up a 'cabinet of functions'. Above average interests of users are shown.

NEMESIS is defined as a systematic frame work of mutually tuned technical facilities. As a result of specification of the individual facilities as extendable and in their components replaceable modules the system gets an 'open-growth' character.
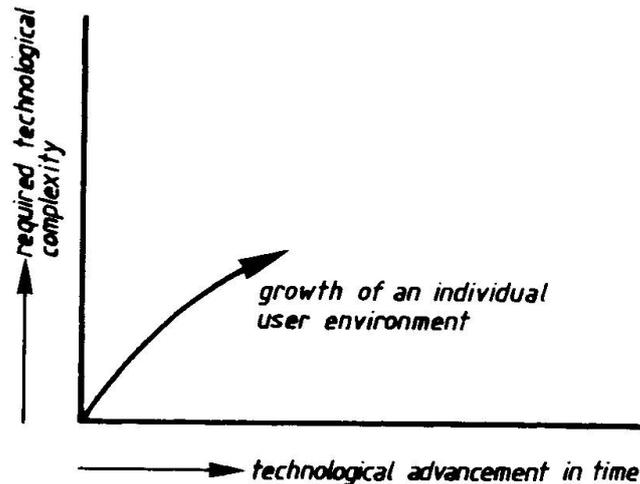

Fig.5   NEMESIS as an 'open-growth' system shall follow
        the growth of the user environment.

The NEMESIS system shall consist of

- a coherent frame work of specifications and standards for hardware components, their connection and monitoring to be used within the system

- a coherent frame work of conventions and standard for the architecture, integration and use of software components to be used within the system.

- an increasing number of hard- and software components, fitting in the above mentioned frame works and supporting the use of the standardized engineering tools within the system.

The novelty of NEMESIS will not be found in new components to be invented but in an integration of existing hard- and software components within a system of newly defined conventions and supporting software.
Main elements of the NEMESIS system will become

- micro-technology based on single chip processing components connected by standardized bus structures and providing standardized connections to support functional hardware components

- workstation concept, based on decentralization of intelligence.
  The NEMESIS workstation shall consist of all the hard- and software components required at that moment by a user for solving his problem.
  Basic hard- and software components of the workstation are

o a central monitor
o system management
o communication management
o application management
o operating system interface

o user interface
o network interface

The following scheme provides a preliminary architecture of the workstation.

- network-concept, based on integration of differentiated system use.
  Differentiation can take place internally and externally.
  Internal differentiation takes place as soon as internal functions are assigned
  to specialized system components with processing capacity independent of the
  central processor. One can think of specialized system components for data
  management, 'number crunching', graphics etc.
  External differentiation takes place as soon as external user interface
  functions are assigned to specialized workstations. One can think of work-
  stations dedicated to the user by the user types defined in the previous chapter.
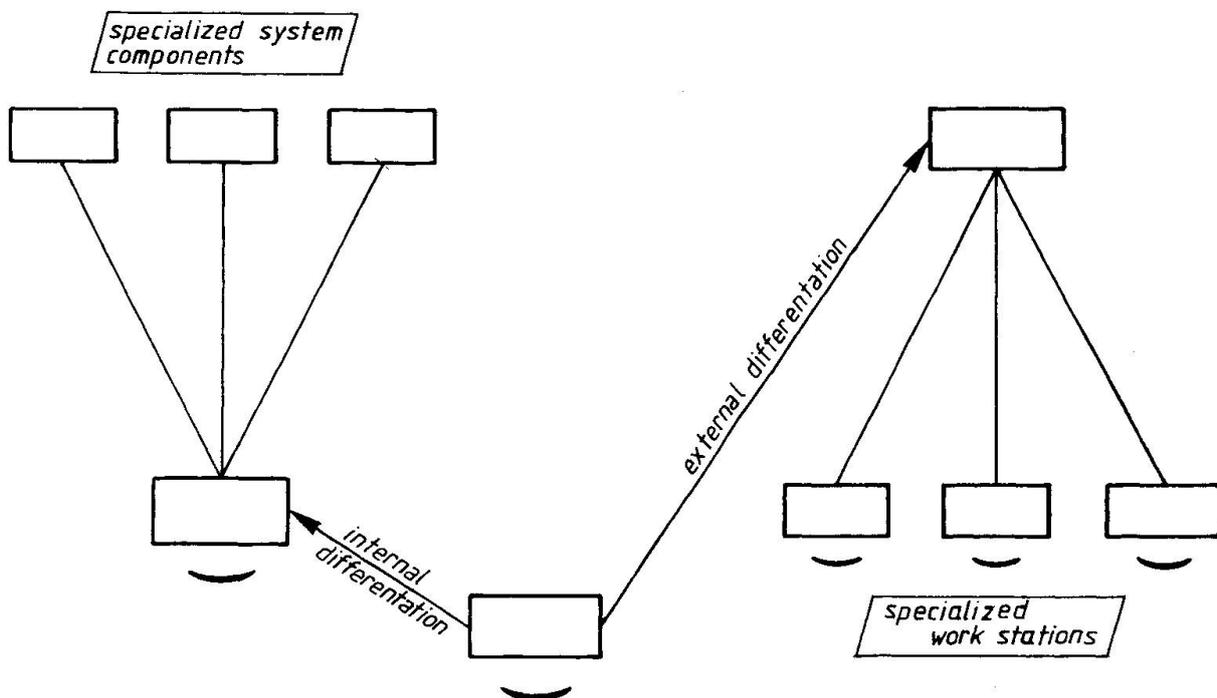
Fig.6   Internal and external differentiation of NEMESIS
        in future development steps

Networks tailored to the specific needs of individual user environments
result from the required adaptability of the system.
The network shall provided possibilities to interface other systems, such
as existing data base systems, measurement systems, industrial process monitoring
system, etc.

- portable operating system concept.
  Main frame and mini computer manufacturers in the past designed their own hard-
  ware components and wrote their own operating system. Portability of application
  software was difficult because of the many different operating systems.
  Successful attempts to standardize programming languages have been made, but
  hardware manufacturers tried very hard to avoid portability by adding powerful
  programming features in order to seduce programmers to use non-standard language
  features.

Their main incentive was to bind the client to their products and provide
continuity to their business.
The specialisation of micro hardware component industry changed the computer
manufacturing market remarkably. Many small companies assembled micro
computers choosing from available hardware components.
As the number of different micro-processors is small, system software,
manufacturers can develop standard operating systems fitting to each of the
micro processors. Well known operating sytems are CP/M on 8-bit processors
and UNIX on 16-bit processors.
In order to avoid the many interfaces to individual operating systems as is the
case with the GENESYS system and because NEMESIS is expected to be based on
micro technology one of the available de facto standard, portable operating
system will be chosen to base NEMESIS upon.

- modular architecture.
  The modular architecture of NEMESIS shall make it possible to build highly
  specialized and user environment oriented systems with a minimum of environment
  dependent software. Companies derive the right to exist and survive due to
  the fact that they are specific and differ from others. As computer systems are
  tools to be used within specific manufacturing processes, the systems themselves
  must become specific by adding special and company-oriented or application
  software blocks. A schematic view on the modularity of NEMESIS is shown below
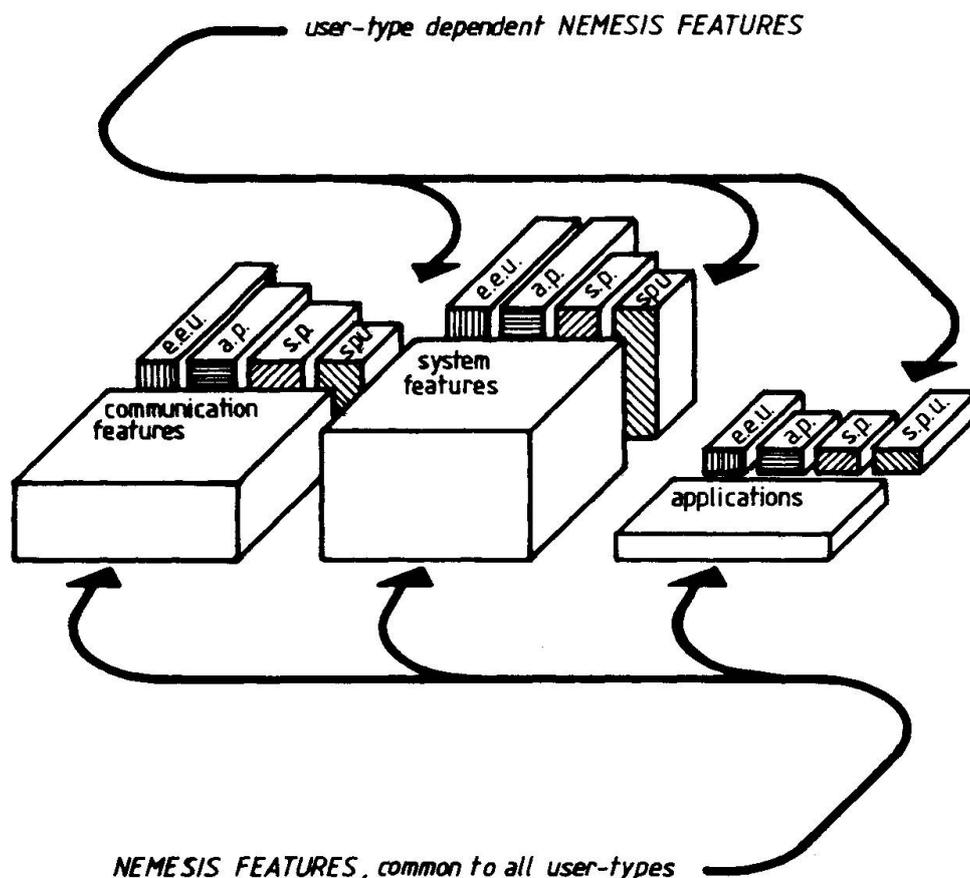 (abbreviations according user-type definitions).

Fig.7   NEMESIS as a box of modular tricks

- standardized data concept.
  As NEMESIS is intended be used as a means of communication, a standardized
  framework for exchange of data is a prerequisite. Exchange of data will take
  place at many levels.

  o between system (data base) and application programs
  o between application programs
  o between application program and user
  o between users within one technical sector
  o between technical sectors.

- standardized user interface concept.
  Many I/O devices have become available to the user to communicate with the
  computer. As the choice for a particular I/O device is often dependent on the
  problem solving process to be performed and depends on the taste of the user
  too, the user shall be as flexible as possible in this choice.
  Which means that the devices shall not have only a standard plug and a standard
  communication protocol. Each device must provide a standard data set in order
  to make them mutually interchangeable.

NEMESIS DEVELOPMENT PROJECT

As mentioned before, the NEMESIS development is planned to take place in
several successive steps, each step ending in an improved product, where user-
experience from the previously developed system is brought together with results
of research into a system that aims at a higher application level.
In the figure below the development cycle of step 1 (i = 1,2,3) has been shown.
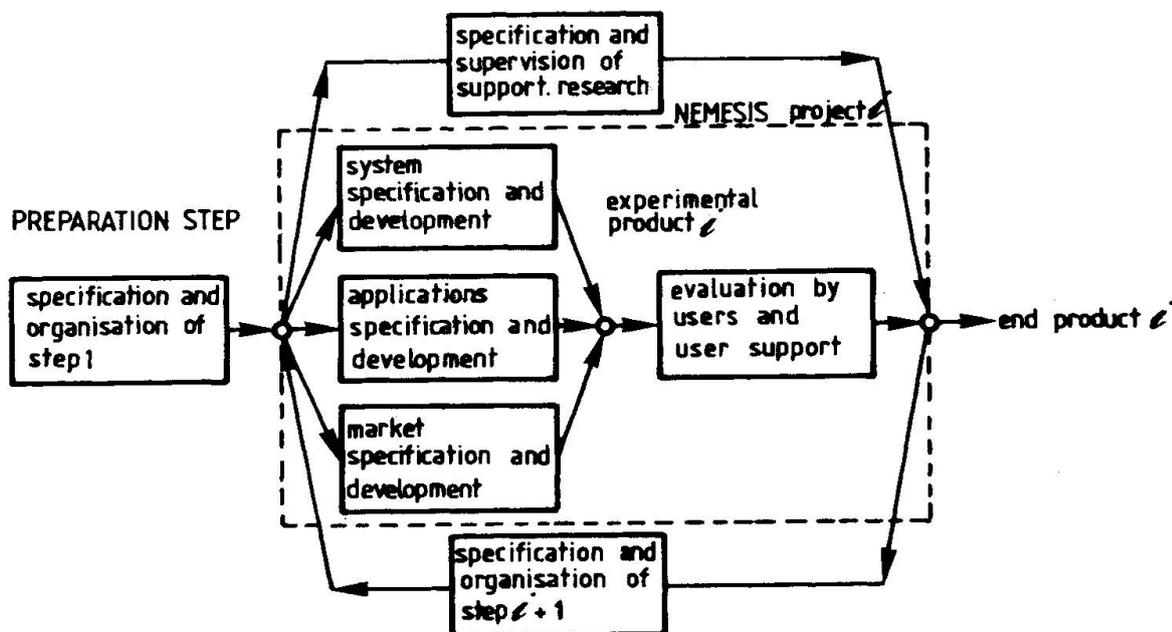For the start-up of step 1 there is a separate preparation step.

Fig.8   NEMESIS development as a project cycle with steps i (i = 1,2,3)

Gravity centres of successive NEMESIS development steps are

NEMESIS 0   : development and elaboration in workable concepts of standards
              and conventions, announcement of development objectives among
              potential users in the market, implementation of a confined
              NEMESIS system in generally available 8 or 16 bit micro hardware

NEMESIS 1   : implementation of advanced hard- and system software technology,
              extension of the NEMESIS system, internal and external
              differentiation of system components

NEMESIS 2   : implementation of communication concepts and differentiation
              of hardware and software concept, integration of differentiated
              user environments.

Each development step will be subdivided into a number of development phases,
each phase ending in a milestone where the decision is taken to go into a next
development phase

```
    milestone 0   o   decisions to execute the development step
       phase 1    ↓   concept phase
    milestone 1   o   approval of architecture and strategy
       phase 2    ↓   specification phase
    milestone 2   o   approval of component specification
       phase 3    ↓   development preparation phase
    milestone 3   o   approval of design and market strategy
       phase 4    ↓   detailed design phase
    milestone 4   o   approval detailed specification
       phase 5    ↓   development system components
    milestone 5   o   preliminary approval system components
       phase 6    ↓   development experimental system
    milestone 6   o   preliminary approval experimental system
       phase 7    ↓   evaluation and user support
    milestone 7   o   final approval on system for marketing
       phase 8    ↓   maintenance phase
    milestone 8   o   decision to take system out of the market
```

A preliminary development organisation scheme is shown. It is the intention
of the development organisation to let as many as possible development companies
(who are also potential users of the final product) take part in development
work.
This decentralized approach has been chosen in order to foster user affections
at an early stage (create an intersubjective user environment) although this
approach aggravates the tasks of projectmanagement.

SUMMARY AND CONCLUSIONS

The initiative to the NEMESIS system development is a result of close cooperation
of computer users in engineering who started and extended their cooperative
efforts on basis of the use of the GENESYS system.

The NEMESIS system development will be realized in several steps, each step
ending in a technologically improved system at a higher infrastructure level.

The NEMESIS system development is a means to catch up with CAD/CAM developments
in other industrialized countries, and shall be an incentive for many other
developments (especially in the application area).

The NEMESIS system is intended to be a means of integration of many already
available hardware and software components based on standards and conventions to
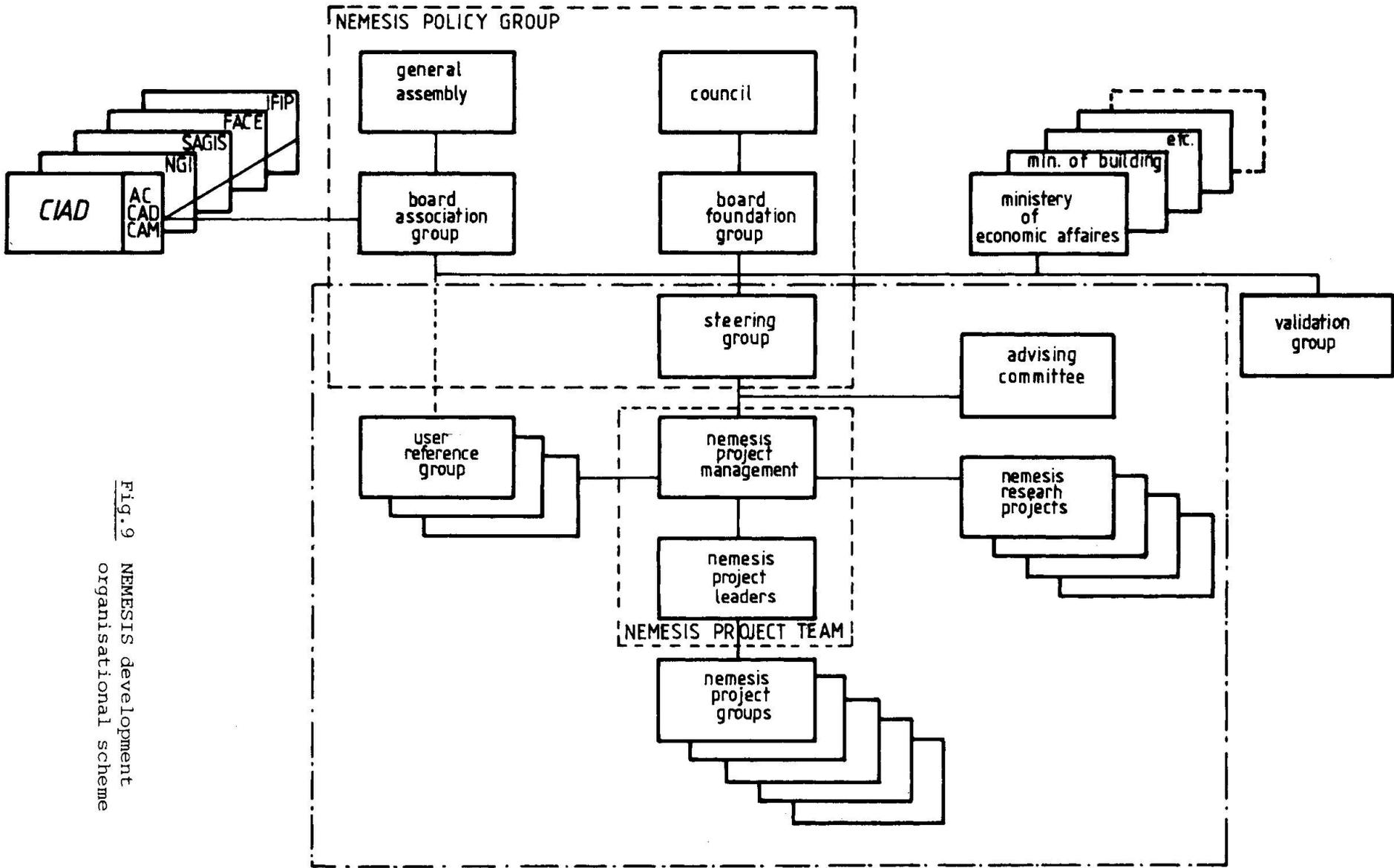
NEMESIS POLICY GROUP

general assembly

council

board association group

board foundation group

ministry of economic affaires

min. of building

etc.

CIAD

AC CAD CAM

NGI

SAGIS

FACE

IFIP

steering group

advising committee

validation group

user reference group

nemesis project management

nemesis researh projects

nemesis project leaders

NEMESIS PROJECT TEAM

nemesis project groups

Fig.9   NEMESIS development
        organisational scheme

be newly developed by larger user groups. As such the NEMESIS system may act as a concentrator for users who are confronted with a dazzling stream of new hardware and software products on the market, which they cannot evaluate each on its own.

From a survey on CAD/CAM interests among engineering companies in the Netherlands, it appeared that more than 50% of the companies that returned the questionnaire (which was 16% of the random sample) expects CAD/CAM techniques to be implemented in their organisation within 10 years.
From these results the conclusion may be drawn that there is enough potential interest for the NEMESIS system as a CAD/CAM infrastructure system to justify its development.

ACKNOWLEDGEMENT

The author wants to stress the fact that the NEMESIS ideas and further elaboration contained in this paper are the results of close cooperation within the project group NEMESIS of the Association and many others. The members of the project group are Messrs. J.H.A.E. Amkreutz, S.I.E. Blok, N.T. van Harpen, W.Nijenhuis and F.P. Tolman, and the author.
The Boards of the Association and Foundation, the staff of CIAD, the CAD-steeringgroup and representatives of the Ministries of Economic Affairs, Housing and Public Works and Waterworks have contributed to the project too. Each from their own point of view. The author expresses his gratitude to all participants in the project for the opportunity to present the NEMESIS ideas at an international forum.

REFERENCES

1. Reports VGN2 and 2.1   : Inventory of bottlenecks in the use of the GENESYS system, 1978 (Dutch)
2. Report VGN-4.3   : Preliminary quality classification criteria for GENESYS, 1981 (Dutch)
3. Report VGN-10   : Qualitative approach towards a new, user environment oriented engineering system, 1979 (English)
4. Report VGN-10.1   : Inquiry on user requirements for a new, user environment oriented engineering system (NEMESIS) 1981, (English)
5. Report VGN-10.2 and 10.3 : Concept of NEMESIS, a new engineering system, volume 1 and 2 , 1981 (English)
6. Report VGN-10.4   : Investigation of the user-friendliness of a GENESYS subsystem in relation to other similar programs, 1981 (English)
7. Report VGN-10.5   : Inventory of research projects and development groups which could support the NEMESIS system development, 1982, (Dutch)
8. Report VGN-10.6   : The technology of NEMESIS, 1982 (Dutch)
9. Report VGN-10.7   : NEMESIS development strategy, 1982 (Dutch)
10. Report VGN-10.8   : NEMESIS project organisation, 1982 (Dutch)
11. NEMESIS nota 1   : Policy report 1 "NEEM NOU NEMESIS..." from the oards of VGN and SGN, 1980 (Dutch)
12. NEMESIS nota 2   : Policy report 2 "NU NAAR NEMESIS..." from the Boards of VGN and SGN, 1982 (Dutch)
13. Samenspel, an information brochure for Dutch industry on CAD, 1981, published by the CAD steering group (Dutch)
14. Results of an inventory on CAD/CAM interests in Dutch industry, 1981, CAD-steering group, unpublished. (Dutch)
15. Feasibility study of common I/O conventions for the building industry, EEG predevelopment study, RIB, Germany 1982 (English)
16. The performance specification of a workstation for the building industry EEG predevelopment study, CICA, Great Britian, 1982 (English)
17. The automation of draughting work, FACE report nr. 2 1982 (English)

Integrated Program Systems for Mini- and Micro-Computers

Systèmes de programmes intégrés pour mini- et micro-ordinateurs

Integrierte Programmsysteme für Mini- ind Micro-Computer

Heinz PIRCHER
Civil Engineer
TDV
Graz, Austria

Heinz Pircher, born 1940, obtained his civil engineering degree at the TU Graz, Austria. After five years as an assistant at the «Institut für Stahlbau, Holzbau und Flächentragwerke» he founded TDV in 1970. He is active in the development of civil engineering software and its application in his own computing centre.

Wernfried HAAS
Physicist
TDV
Graz, Austria

Wernfried Haas, born in 1947, obtained his degree in technical physics in 1972 at the TU Graz. From 1972 to 1976 he worked there as an assistant in computer controlled measurements. He joined TDV in 1977 and has been working in the development of several FEM-codes and as a system analyst.

SUMMARY

The authors' opinion of the present state and the further development of civil engineering software is put into discussion. Of particular interest are micro-computers and their influence on future development. Examples demonstrate their universal applicability.

RESUME

L'opinion des auteurs concernant l'état actuel et les développements futurs de programmes pour ingénieurs est mis en discussion. L'apparition des micro-ordinateurs sur le marché et leur influence sur les développements futurs sont particulièrement intéressantes. Quelques exemples montrent l'application universelle de ce type d'ordinateurs.

ZUSAMMENFASSUNG

Die Meinung der Autoren zum momentanen Stand und zur weiteren Entwicklung von Bauingenieur-Software wird zur Diskussion gestellt. Von speziellem Interesse sind Micro-Computer und ihr Einfluss auf die zukünftige Entwicklung. Beispiele demonstrieren ihre universelle Anwendbarkeit.

1. INTRODUCTION

In European civil engineering, the majority of the planning work is done by "Civil Engineering bureaus". One can say that the typical size of such a bureau lies between 10 and 20 people. Only a few bureaus employ more than 50 people. The construction offices of the larger building companies are also of similar size.

All questions which concern the application and development of new working methods are therefore made with respect to the special situation concerning firms of this size. Only steel construction is an exception, because here a large part of the planning and construction work is done in large steel construction firms, which have different criteria for the introduction of new working methods.

One can be certain that the difference in average size between the two is partly responsible for the different path taken by steel construction firms in applying computer-methods to the one taken by the average civil engineering firm.

Many technical, economic and organizational reasons have moulded the present computer applications and the possibilities for further development.

2. TECHNICAL ASPECTS OF ENGINEERING SOFTWARE

Looking at the current situation of civil engineering software, we realize that structural analysis has been developed and that several software systems are already in practical use. The computer is also used for various dimensioning problems.

The latest and perhaps the last step in development is to also use the computer to produce all the drawings needed to build the structure. The slogan "CAD = Computer Aided Design" was first created by mechanical engineers and now even civil engineers think about a complete solution integrating structural analysis, dimensioning tasks and graphic modules. If one acknowledges "CAD" to be the aim of actual development then the term "CAD" should not be taken too simply, namely merely "graphic data processing". Everything such as using a computer to define the shape of a construction, to define its dimensions and to produce plans necessary to build it should be included.

Looking at the current situation one realises that for many applications software solutions are already in practical use:

2.1. Structural Analysis

Here, the use of computers is a matter of course.There are programs for all types of processors and the possibility to use these programs in a computing centre or on one's own computer. A further development will be the improvement of present programs. Basically all important problems have been solved.

An important point to mention is that today all complicated structural analysis problems, that are solved by computers, are supported by graphics. This is especially the case when using "Finite Element Method" programs. Today the checking of input values for complex structures is unthinkable without graphic support. The same applies for the representation of results.
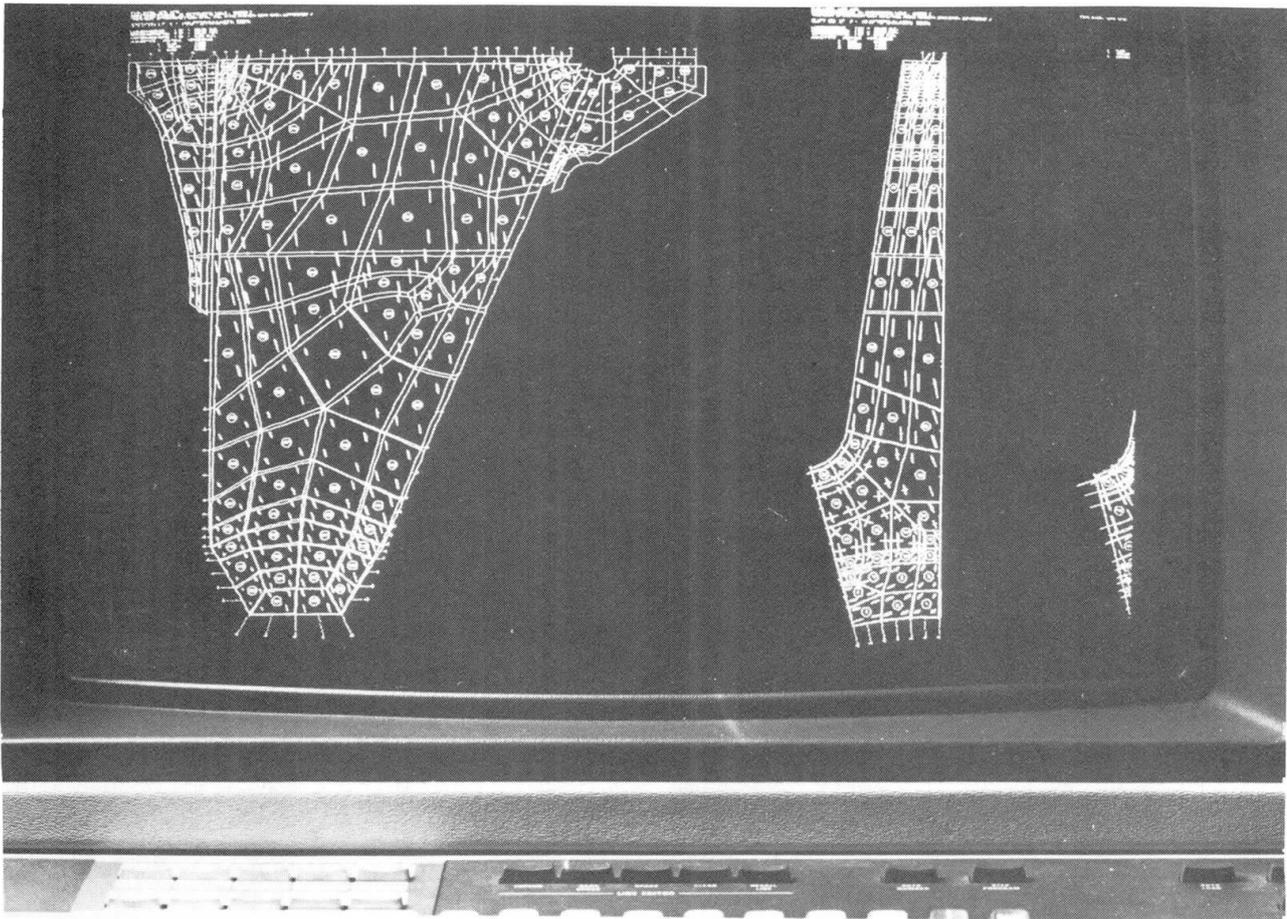
<u>Fig. 1</u>   Example of representation of results of a F.E.M.
            calculation

One can say with hindsight that the solution to this problem paves
the way for a much broader use of these methods. The use of graphics
for F.E.M. calculations was a major starting point for the CAD
development.

It is noteworthy that today's F.E.M. programs with extensive inter-
active graphics for input control, element net generation and re-
presentation of results are also available in BASIC for desk top
computers.

2.2. Reinforced Concrete

The dimensioning of the reinforcement and the production of the
corresponding plans is the main work of many engineering bureaus.
Programs to calculate the required reinforcement belong to the basic

tools of every computer used in this field.

The "dimensioning programs" are often coupled to structural pro-
grams. For various constructional elements (beams, supports, slabs,
etc), programs exist to draw the reinforcement plans, also coupled
to simple dimensioning routines.

Furthermore it should be noted that this sort of solution is al-
ready available for very cheap computer systems.

What is missing is a general, comprehensive solution including
statics, dimensioning and graphics in one integrated system.

The structural programs have been concepted in such a general way
that the data from any structural system, or from any type of con-
struction can be captured onto the computer system.

Even the dimensioning has been similarly comprehensively solved
for all relevant construction elements. The next logical step would
be towards the automatic drawing of a similar general solution and
to connect this to an available structural analysis program. This
is one of the most up-to-date-problems in software development in
civil engineering.

These efforts are only successful, if a certain level of standardi-
sation is achieved.

The traditional ideas of "how a plan should look like", differ and
a rationalisation is only possible if all those active in this
field agree. The programmer needs a final and precise algorithm !

A further problem is that drawing reinforcement plans is partly
routine work for a computer and partly creative engineering; they
are not separable. The many constructional details and definitions,
which first become apparent during drawing, cannot be decided upon
by the computer.

It is obviously very important to develope user-friendly dialogue-
programs. The engineer makes his decisions and needs easy-to-use
talking here about a model for computer supported work - as opposed
to a completely automatic solution, which would only be possible in
a few special cases.

2.3. Pre-stressed-concrete

In this field, there are highly development solutions available,
especially for pre-stressed and prefabricated elements. The problems
here are not very complex and programs already exist for cheap
desk-top computers, which take care of all calculation and have
either full or partial graphic support.

Program systems for the design of pre-stressed concrete bridges have
been developed, which apart from calculationg all important values
even include the ability to draw plans for the laying of the cables:

Definition and calculation of the stressing cable geometry and pre-
stress-forces using all values for the stressing protocol, load case
creep + shrinkage and all verifications required by design code,
representation of the pre-stressing tendons in plan view and vertical
section, and the production of a finished plan by bringing such views
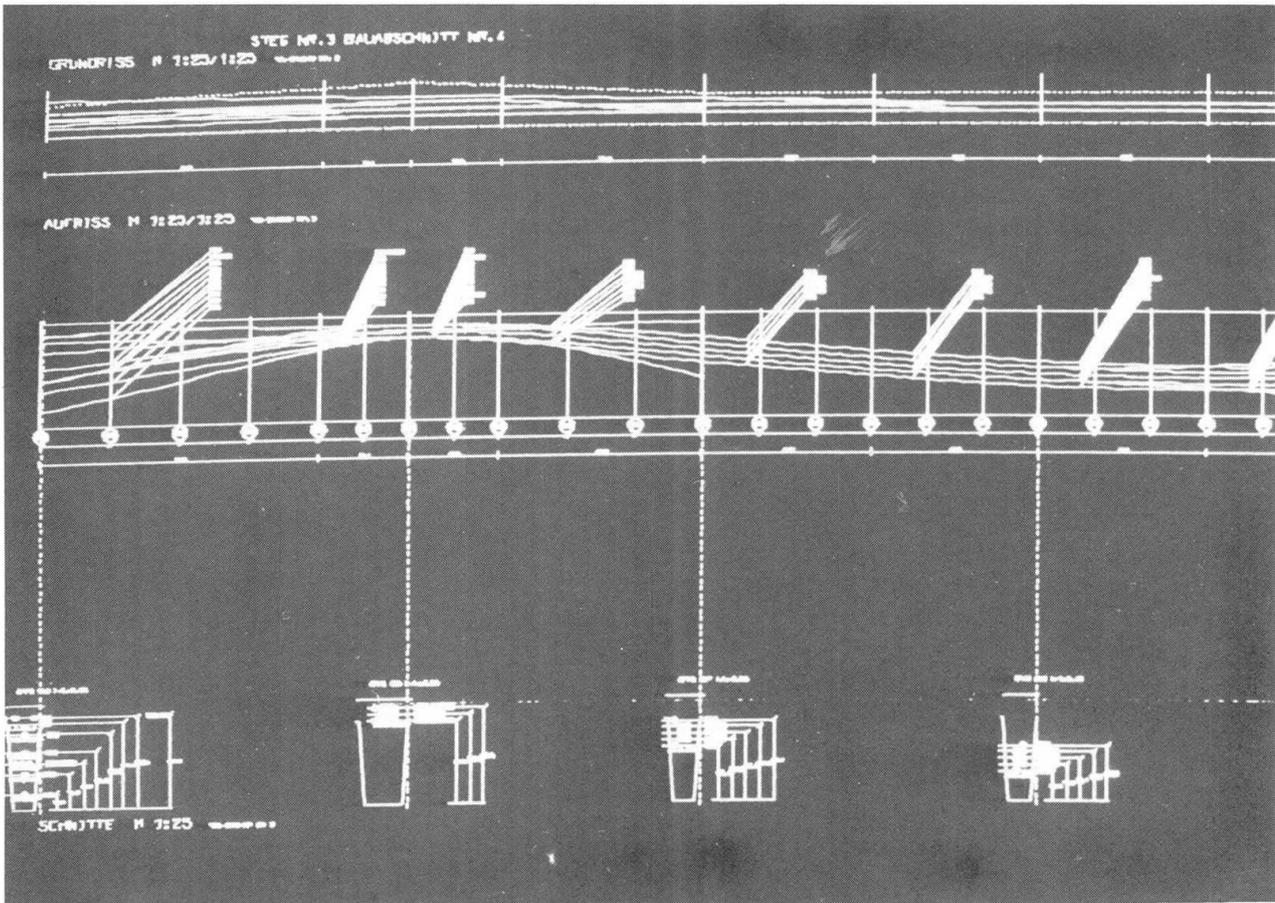and cross sections together (see Fig. 2).

Fig. 2   Example of pre-stressing cable plan

The program package "Pre-stressed-concrete" is integrated into a program for structural analysis and the whole system is a good example of a complete solution for calculation, dimensioning and drawing.

Even in this case it is important when using the computer sensibly that the pre-programmed course supports the engineer but does not replace him. Therefore, great emphasis has been laid upon dialogue input and great care has been taken in programming the graphic representations.

The program is used by a great number of Austrian engineering bureaus and civil engineering firms. It is even used in computing centres as well as on private machines. It has also been sold to nameworthy engineering bureaus abroad (approx. 7Ø installations, approx. 5Ø in Austria).

There are two versions:

BASIC for desk-top computers (the performance is obviously limited by BASIC) and FORTRAN for mini-computers (this version also runs on all mainframes).

2.4. Steel construction

In steel construction, the main stress of actual development lies in the direction of computer aided design.

A problem seems to be that available program packages concentrate mainly on the drawing of plans. The coupling of data to structural analysis is problematic. Much work is still required to include into the overall program package constructive details (cross sectioning, profile choice, determination of connections) with all required calculations to achieve a general solution for calculations, construction and drawing of plans.

The dependency on standards plays an important role in the development of technical programs.

Since the design codes and "unwritten" working conventions differ so much from country to country, international cooperation is made difficult. This is awkward for programmers.

Program users find it difficult, if not impossible to use solutions developed in other countries and the programmer does not have access to foreign markets if he conforms only to local conventions.

3. ECONOMIC ASPECTS OF ENGINEERING SOFTWARE

At the beginning it was pointed out that the majority of potential and actual users have a certain size, ie. 10 to 20 people. The cost of computer applications must therefore be within reach of such firms.

In spite of the small size of firms there is a need for a complete solution including structural analysis, computer supported dimensioning and drawings like a C.A.D.-system.

When purchasing a computer system the costs have to be regarded: the initial cost of the system and the running costs (mainly maintenance and personnel). In order to estimate the initial costs, one must consider that integrated solutions are mainly a matter of proper data organisation. Only if results and constructional details are properly stored in a data base, can all the necessary data for structural analysis and drawings be efficiently extracted.

The computer must therefore be equipped with sufficient external memory. To guarantee fast access to the database, a powerful processor is required. In this way, costs quickly escalate up to 100.000,-- or 150.000,-- $. Yet, one still needs a powerful plotter (with sufficient paper size and quality) and a graphic terminal with good resolution.

The costs for graphic hardware lie between another 30.000,-- $ and 50.000,-- $.

Then, there is of course, the software. The initial cost of such software systems lies between 100.000,-- and 200.000,-- $. If it is not possible to reduce these costs, then this system will have to earn its user 5.000,-- $ a month, which for the majority of potential users is highly improbable.

From the programmers point of view, these development projects will last for many "man-years". To do this privately in one's own average sized engineering bureau would therefore only be possible in exceptional cases. A professional software house would need to sell approx. 50 program systems in 2 years to be able to keep the soft-

ware price down to such a level, that the total cost (hardware + software) does not escalate too far.

These statements make it clear that with todays costs it is difficult to strike a balance between cost and usefulness if one wants to buy the complete C.A.D.-solution. This may be the explanation for the fact that a large part of structural analysis and CAD development is supported by the state.

4. FUTURE OUTLOOK

There are two ways out of this dilemma:

That the expected continuing fall in hardware costs in the coming years increases the circle of potential users to such an extent that software development costs can be better divided. Then the profitability of obtaining such a system will be easier proven. Please note, that the actual development of micro computers is a very important fact in this context and will therefore be dealt with in chapter 5.

Restriction of problems in order to take out only specific topics that are of real interest and concentration on practical points can reduce the hardware and software specifications so much that the profitability for user and producer is guaranteed.

The author has been tackling these problems for many years and has succeeded in implementing engineering software on mini- and even microcomputers. The special considerations necessary for such implementations will be explained in the next chapter.

5. MINI-MICROCOMPUTERS

5.1. What is a mini ?

The question: "What is a minicomputer ?" cannot be answered easily or fully. Some years ago, the answer would have been easier. At that time, a minicomputer was a digital computer with a word-length of 16 bits, preferrably used for process-controlling purposes in real-time applications.

The 16-bit word length restricted the addressable memory space to 64 KBytes. This meant that no program on a mini could exceed that magic limit. Some of the minicomputer soft- and hardware characteristics are given in the following table:

MINI
16 Bit word length
64 KB Addressable memory space
fast floating point operations
all peripheral devices
electronics one generation younger
than on mainframe
low price compared to mainframe
broad range of system-software available
(from compilers to communications network software)

SUPERMINI
32 bit word length
some MB addressable memory space
price less than mainframe

MICRO   8 - 32 bit word length
                         more advanced electronics CPU = 1 chip
                         slower than mini
                         sometimes no floating point hardware
                         not so many peripherals
                         very cheap

To continue the history - very soon after minicomputers had
appeared on the market their ability of solving technical problems
economically was discovered.

Upto this time all the available FEM codes had been developed -
mostly at the universities - on big mainframe computers. Of course,
they were too expensive to be used in everyday engineering practice.

Since then, many people have started to use the FEM-method on mini-
computers, but not only because they are cheaper.

There are also other good reasons for preferring a minicomputer to
a mainframe:

- minicomputer are usually easier to use
- they have a more modern operating system which is better designed
  for
- interactive method of working
- easy file (data) handling
- it can be located where the engineer works,
- so he has direct access to all peripherals like printers,
  plotters, tapes, disks

Therefore, it seems that all these advantages of a mini-micro-
computer outweigh its disadvantage of being slower than a mainframe.
Very often, the high speed of the mainframe is lost anyway, when
plots or printouts have to be sent to the user.

Of course, many tasks remain which have to be done preferrably on
really big machines. But in my example I would like to show to what
extend minicomputers can be used. A connection between intelligent
terminals, decentral minicomputers and mainframes connected in a
hierarchical tree structure seems to be a widely accepted concept.

5.2. Programming concepts

Due to the characteristics of a mini-computer, especially its
address space restriction to 64 Kbytes, special programming
strategies had to be devised for the implementation of FE-codes on
them.

First of all, instead of one big software-system containing all
possible element types and all thinkable algorithms a family of
program modules has to be constructed. There, the number of
available elements is reduced to those which offer good performance
for a wide range of applications.

These different modules communicate with each other by means of
disk files.

Secondly, an overlay structure as simple as possible has to be de-
fined to use the available memory space as efficiently as possible.

As a result, the programs are disk i/o oriented. This could lead
to machine dependent programming (Disk i/o is not standardized !).

But by special programming techniques where all i/o is put into special subroutines this problem can be solved. It is then very easy to include system subroutines in these subroutines, which make disk i/o very fast, compared to standard i/o speeds.

Consequently the algorithms have to be chosen, not only considering their incore efficiency, but sometimes mainly with regard to the number of i/o operations performed.

Of course, the concept has to be made so flexible that big memory. can be used, when it is available. Think of the new generation of micros with a 16 bit CPU and an addressing capability of about 8 MBytes.

6. EXAMPLES

To illustrate our concept of technical programming, some examples will be presented in this section. It has to be pointed out that all these examples have been solved on very small computers.

In Fig. 3, results of a plate analysis are shown. The program used for this purpose runs on a basic programmable desktop-computer with 64 KBytes main memory and a floppy-disk capacity of approximately 512 KBytes. The cost for such a configuration is currently less than 316.000,-- $.

Fig. 4 shows the FE mesh for the analysis of a wheel of a generator. Cyclic symmetry has been taken into account for this system, comprising of about 350 shell elements.

In Fig. 5, the mesh for a 3-D analysis of an arch dam, approximately 200 m high, is shown. This problem too, has been solved on a mini-computer with 256 Kbyte of main memory, 20 Megabytes of disk space and of a price of about 45.000,-- $.

Finally, I would like to stress that all the software mentioned in this paper is already implemented, and functioning on modern 8-bit and 16-bit-micro-computer. We have excellent experience in using them. Naturally, there is a limit to what extent the small machine can solve the problem in acceptable time. Within this limitation the solution using micro computer is unbeatably cheap. Please note, that the arch dam (fig. 5) can also be analyzed using a micro computer during a single night.

All these experiences strengthen the opinion that the new generation micro computer can be a base for the development of complete CAD-Systems including everything needed for structural analysis, dimensioning and drawing of plans.

7. SUMMARY

In order to supply the engineers with cost-effective and easy usable computer support, it is proposed to concentrate future work on the following:

Specialized software solutions for special applications, as opposed to general, expensive and big software-packages.

Usage of cheap and efficient mini- and micro computers instead of centralized mainframes.

This concept has been followed in our company for more than ten years. It seems that nowadays - especially due to the advantage of the new generation micro computers - this concept is accepted by a wide group of scientists and engineers.
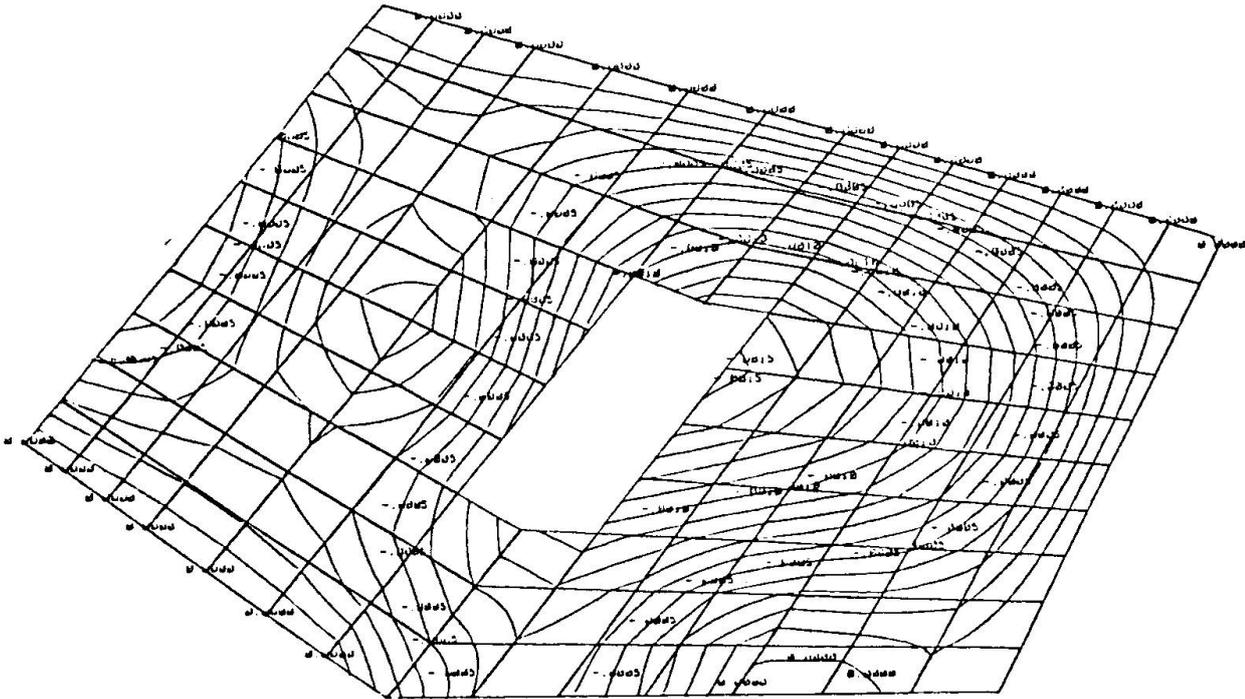
CONTOURS - DEFLECTION W
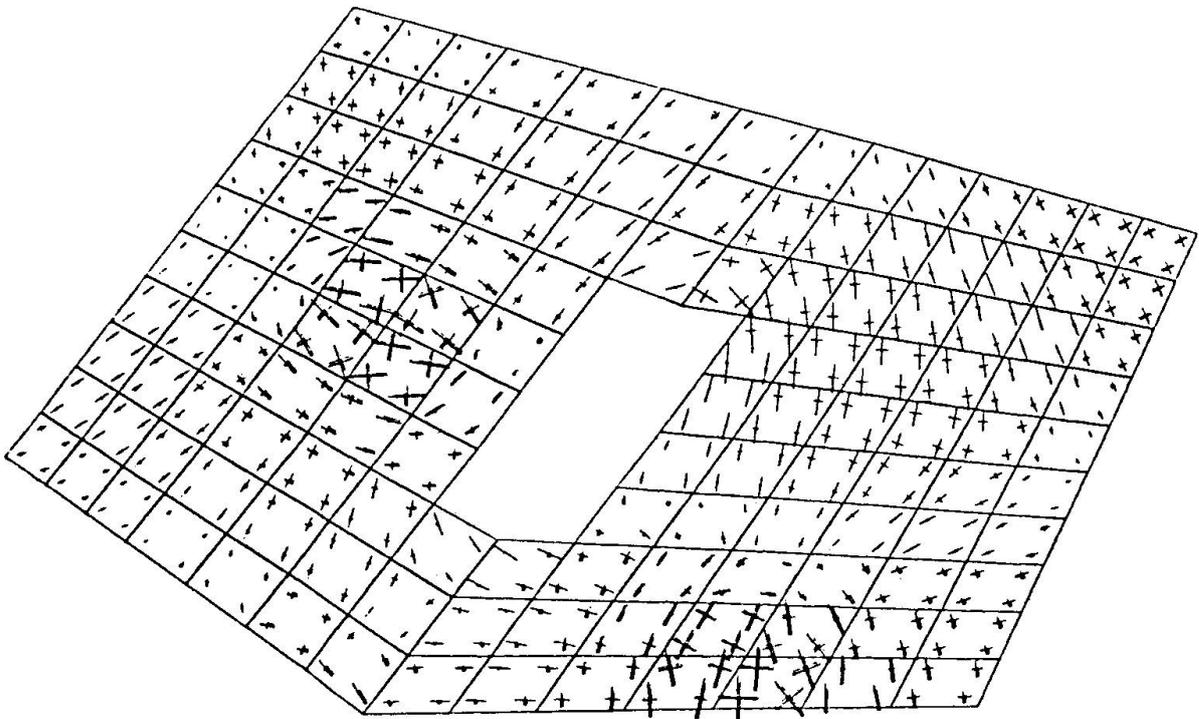
PRINCIPAL MOMENTS

Fig. 3

AXONOMETRIE

LAENGENMASSTAB: 1 CM = 100.00 R.E.
AXONOMETRIE   RICHTUNG   VERKUERZUNG
          X   15.00   1.000
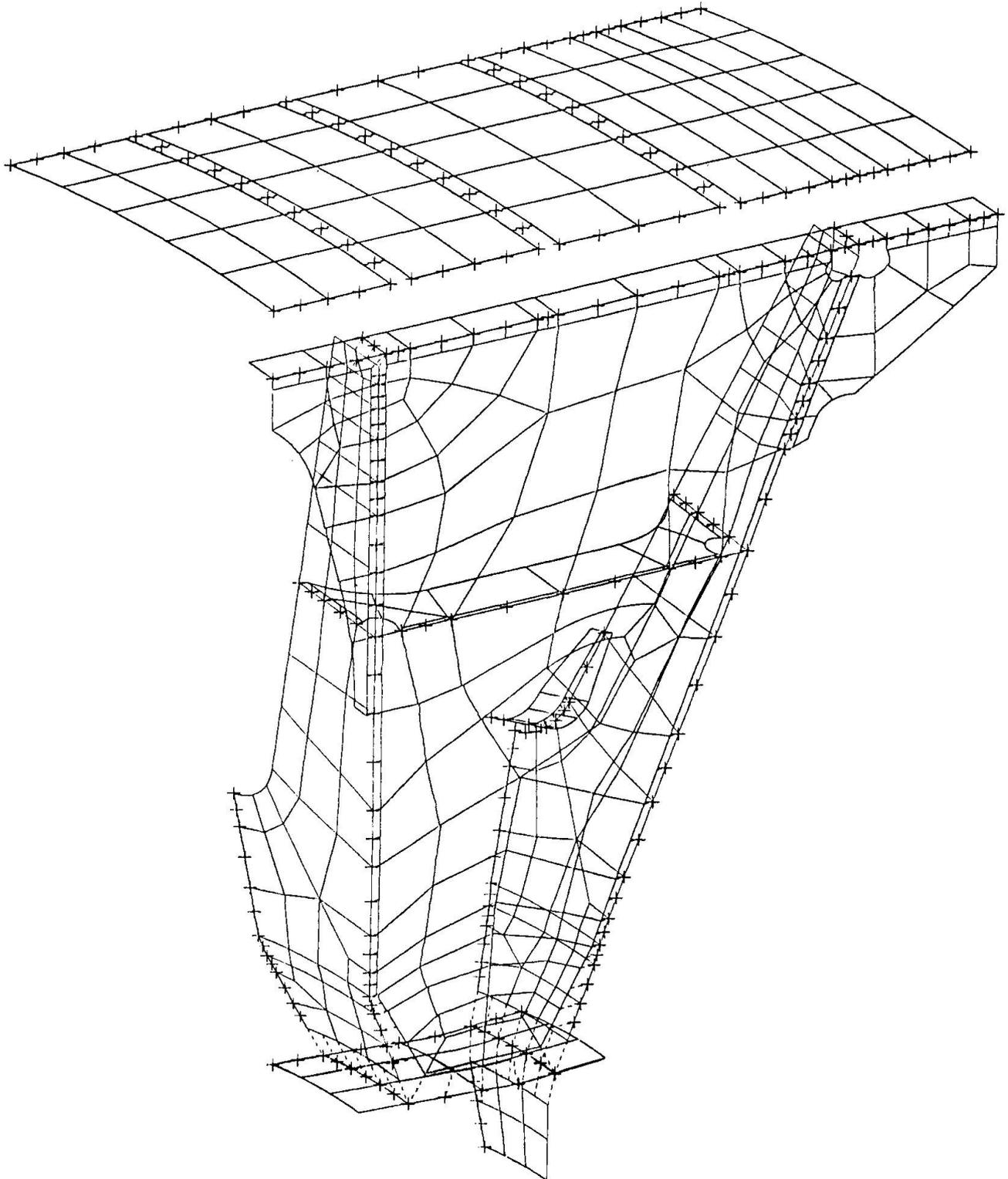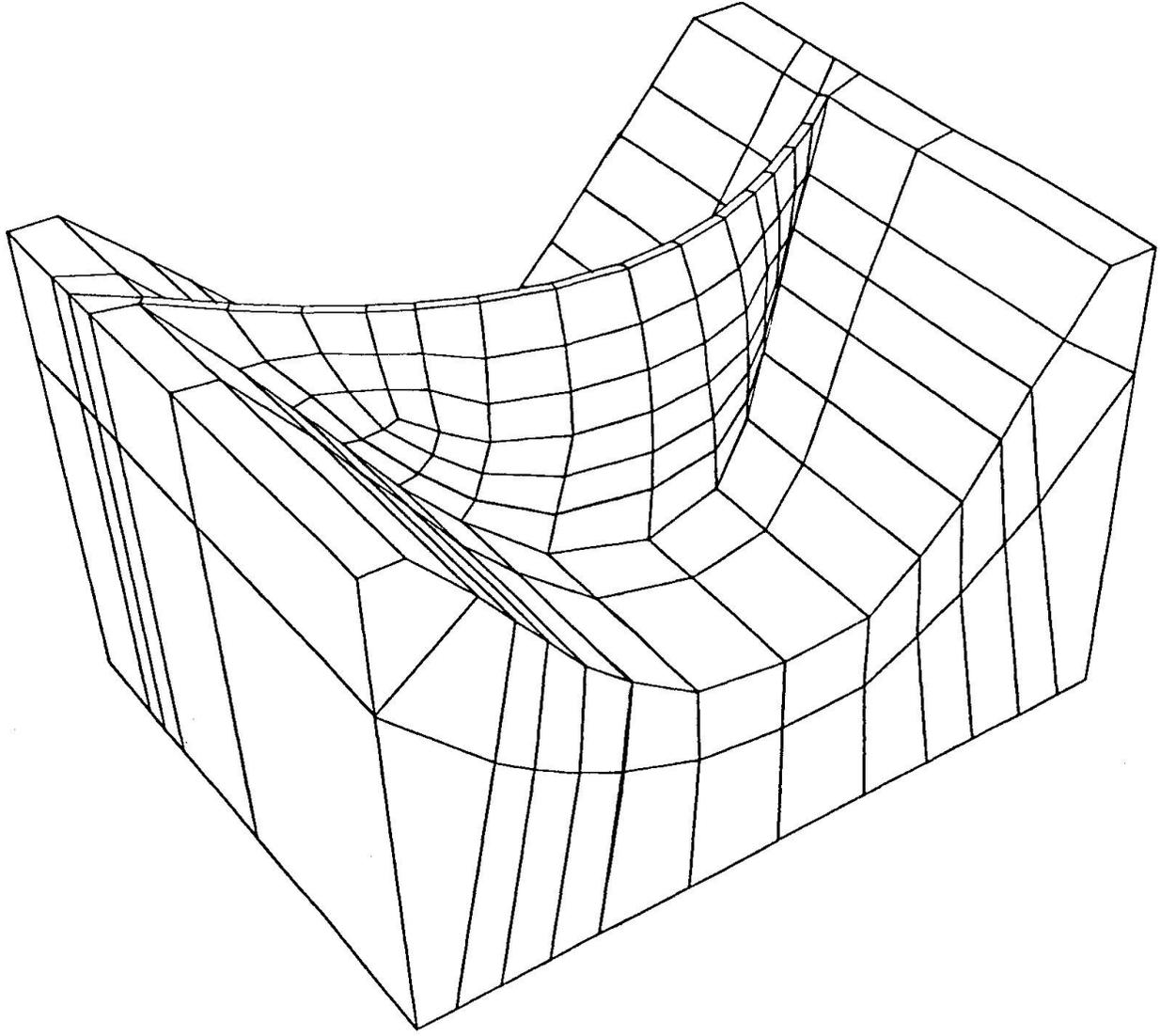   Y   100.00   1.000
          Z   160.00   1.000

Fig. 4

M 3 V I E W : Zentralperspektive
ARCH DAM --- UNSYMMETRIC SYSTEM
===
TECHNISCHE DATENVERARBEITUNG, DIPL.-ING. HEINZ PIRCHER, 8010 GRAZ, LUTHERGASSE 4

Fig. 5

Standardized Input-Output Conventions – State of the Art and Trends

Standardisation des entrées/sorties – Etat actuel et tendances

Standardisierte E/A-Konventionen – Stand der Technik und Trends

Mun AHN
Civil Engineer
RIB
Stuttgart, Fed. Rep. of Germany

Mun Ahn, born in 1936, obtained his civil engineering degree at the Seoul National University and at the University of Stuttgart. For twelve years he was engaged in the field of development and application of EDP programs for civil engineering and CAD problems. Mun Ahn is a senior consulting engineer with RIB.

SUMMARY

This paper reviews the findings of the European Community predevelopment study – I/O conventions – for the building industry within the Member countries completed in April 1982. The aim of the study was to assess the feasibility of European Standards for input and output (I/O) conventions for computer programs used by designers and builders in the construction industry.

RESUME

Ce rapport décrit les résultats d'une étude préliminaire portant sur les conventions d'entrées – sorties (E/S) pour l'industrie de la construction dans les pays de la Communauté Européenne. Le but de cette étude est d'évaluer la possibilité d'établir un standard européen pour les entrées/ sorties des programmes utilisés par des ingénieurs dans l'industrie de la construction.

ZUSAMMENFASSUNG

Dieser Bericht beschreibt das Ergebnis einer EG-Studie – E/A-Konventionen – für die Bauindustrie innerhalb der EG-Mitgliedstaaten. Das Hauptziel der im April 1982 fertig gewordenen Studie war die Untersuchung der Durchführbarkeit des europäischen Standards für E/A-Konventionen für die Computerprogramme, die von Entwerfenden und Bauunternehmern der Bauindustrie benutzt werden.

1. INTRODUCTION

In 1977 a report was published by CIAD Consortium: "The effective use of computers within the building industries of the European Community", EEC Study Projekt T/1/77. In this report several recommendations were made, one of which was to establish some input and output (I/O) conventions for computer programs used by the building industries of member countries. Accordingly the Commission appointed Rechen- und Entwicklungsinstitut für EDV im Bauwesen - RIB e.V., Stuttgart to discover whether international agreement about I/O conventions was feasible, and, if so, to describe a framework in which such conventions could be defined and implemented.

This predevelopment study on I/O conventions was controlled by a Technical Committee appointed by the Commission; the committee representing every member country. An international study team assembled by RIB made a preliminary survey of computing in the building industries of all member countries and presented its findings as an interim report.

The findings of the interim report were then analysed; the analysis being presented as the final report which led to firm conclusions - and set out recommendations - as summarized below.

2. FINDINGS

There are different I/O conventions among member countries. There are different conventions among firms in each member country. There are also internationally used systems such as ICES and GENESYS which employ still other conventions. Yet this report concludes that universally acceptable I/O conventions are still feasible. The reason for this fortunate conclusion is that computer-aided design (CAD) systems are currently in a state of development from the old fashioned "batch" to the more modern "interactive" approach to presenting input data. Such I/O conventions as are well established - and in which there is conflict between originators - are mostly conventions concerned with batch usage.

That is not to say that batch usage is dead. On the contrary when immediate response is not essential designers may prefer to present original data in the conventional way, using an interactive approach only for modifying and refining the original design. For every aspect of the proposed framework in this report, each interactive facility has its corresponding batch facility. This philosophy is tenable because at the fundamental level of input there is seldom much difference between the syntactical requirements of an entity (say an integer) regardless of country, company, or proprietory system.

Small but irritating differences between I/O conventions in programs originating from different sources have hitherto made interchange of programs - even the marketing of well established programs - difficult if not impossible. Definition of a very few standard conventions of syntax (integer, real, keyword etc.) would make marketing and interchange much easier. In a small office that gains initial experience using, say, a cheap turn-key CAD submitting data would have to be relearned when that office graduated to a bigger system.

Computing is now an essential element in the design and construction of
buildings. Therefore some conventions - at very least the definition of syntax
of fundamental items of data - are essential to the building industries of
member countries. The obvious advantages (in terms of costs avoided and time
saved) are:

- there would be less learning time for users in mastering each new system

- the documentation explaining the use of each new system would be shorter, and
  simpler both to read and to write

- by using standard I/O modules there would be less programming statements to
  write when developing each new system

- because of standard I/O procedures the use of each new system would be easier
  to explain, hence the system easier to market

A framework for such conventions is proposed.

3. CONVENTIONS

The report describes four "levels" of I/O convention. Levels are hierarchical
in the sense that conventions at each level encompass these at lower levels.

As well as describing levels at which I/O conventions should be defined the
report proposes input and output modules be written to handle data prepared
by the user according to these conventions.

The four levels are briefly exemplified below, where AP stands for Applications
program and IOM stands for Standard Input/Output Module:

Level 1. At this level a call to the IOM causes a single value to be input.
         The programmer of the AP may compose a prompt for the IOM to issue
         when the AP is being used interactively:

         SECTION WIDTH (mm)?

Level 2. The user issues a "command" (one of a set of words defined by the
         applications programmer) making the IOM cause entry to a correspon-
         ding section of the AP. Necessary data are then input by calls to the
         IOM precisely as at level 1: ⸰

         READY?

When the program is being used interactively the command may be selected from a menu displayed by the IOM:

TYPE 1 FOR INPUT
     2 FOR ANALYSIS
     3 FOR OUTPUT

READY?

Level 3. The user may issue commands modified by parameters. It is not necessary for the AP to call the IOM for each item as at levels 1 and 2

BEAM 115,,225 (default assumed for 2nd parameter)
DELETE BEAM1 THROUGH BEAM5

Level 4. At this level the applications programmer has a command-definition language. Examples of level 4 are seen in GENESYS and ICES where the command-definition language of GENESYS is the more sophisticated. Examples from the user's point of view are limitless in scope, but below is a command as defined in a manual describing an AP of GENESYS. Below that is a possible input of that command by a user

Command definition

ALLOW SETTLEMENT OF value,,MM AT SUPPORT integer
AND

Command input

ALLOW SETTLEMENT   OF 1.5,,MM AT...
           SUPPORT 2 AND OF 2,,MM AT...
           SUPPORT 5

4. RECOMMENDATIONS

The report makes no recommendations about conventions for graphical I/O because an international standard on this subject was being prepared at the same time as this study. However, the report concludes that I/O conventions for alphanumeric data are feasible and should be defined. The four levels described are enough to cover the design and construction processes of the building industry (Fig. 1), and there is adequate expertise in member countries to define these conventions and produce corresponding IOMs of technical excellence.
The recommendations are:

- I/O conventions and corresponding IOMs should be implemented to level 3 starting as soon as possible

- During implementation of the first two levels the conventions at level 3 should be defined so as to ensure upward compatability

- When level 2 conventions are established - and feedback obtained from use in the field - level 3 conventions should be revised (if necessary) and implemented

- All conventions and corresponding IOMs should be made as independent of proprietory machinery and software as possible

- Conventions should aim at compatability with the work station specified in a parallel EC Predevelopment study (CICA, April 1982)

- As for as possible the project should be funded by the EEC. Benefits are expected to exceed costs very soon

- The IOMs should be distributed to those who require them against a low fee to cover costs of maintenance and distribution

- The IOMs should be written in three programming languages: Fortran, BASIC and Pascal

- Assuming success of level 3 in the field work should continue on a unified approach to I/O conventions (including those for gaphic I/O) in the building industry by definition and implementation of level 4.

Area of Computer Application	Level 1	Level 2	Level 3	Level 4	1 Initial brief	2 Preliminary design	3 Costumer's approval of preliminary design	4 Scheme design	5 Approval of licensing authorities	6 Detailed design	7 Preparation of tender documents	8 Preparation of specialist constructors' tenders	9 Preparation of constructors' tenders	10 Analysis of tenders	11 Awarding of contract	12 Plan construction	13 Monitor project costing,planning,cash flow	14 Schedule/order resources	15 Site co-ordination and management	16 Monitor project performance - constructor	17 Preparation of interim valuations	18 Modifications to project	19 Completion of project	20 Final settlement of accounts	21 Documentation of completed project	22 Historical analysis of project	23 Monitor customer satisfaction
Design	●	●	●	●	X		X		X													X				X	
Structural analysis	●	●	●	●			X		X													X				X	
Drawing production and registr.	●	●	●	●	X		X		X													X			X		
Building codes and regulations	●	●	●	●	X		X	X	X	X	X	X										X					
Specifications and bills of quantities	●	●									X											X					
Estimating	●	●					X		X		X	X	X	X	X												
Planning and monitoring of activities		●	●				X		X		X	X				X	X		X	X		X			X		
Material and resource management	●	●	●															X		X							
Commodity information	●	●	●				X		X		X	X	X			X	X					X			X		
Financial																	X	X	X	X	X	X	X	X	X	X	X

Figure 1   Areas of Computer Application v. Stages in the Building Process with I/O levels

Leere Seite

Blank page

Page vide

Role of Database Management Systems in Structural Engineering

Systèmes de gestion de base de données dans le domaine des structures

Datenbank-Management-Systeme im Bauingenieurwesen

Steven J. FENVES
Professor
Carnegie-Mellon University
Pittsburgh, PA, USA

Steven J. Fenves, born in 1931, received his degrees in civil engineering from the University of Illinois, where he taught until 1972. His teaching and research activities deal with computer-aided engineering, with emphasis on representation of standards, databases and expert systems.

William J. RASDORF
Assistant Professor
North Carolina State Univ.
Raleigh, NC, USA
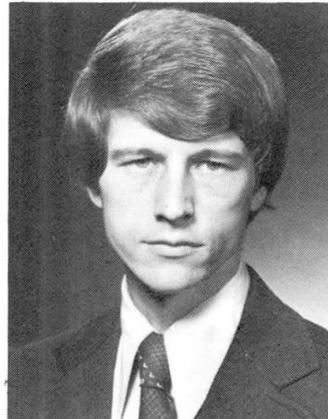
William J. Rasdorf, born in 1951, received Bachelor's and Master's degrees in Architectural Engineering from the Pennsylvania State University. He worked as an Architectural Engineer for 3 years. He received his Ph.D. from Carnegie-Mellon University where he became interested in computer representation of engineering design data.

SUMMARY

The future integration of structural engineering application programs will depend critically on integrated databases which provide access to information in essentially arbitrary sequences, and which automatically perform a large portion of integrity checking on the data. One source of such design databases are the database management systems (DBMS) evolving from management applications. The paper surveys such systems and presents some extensions needed.

RESUME

L'intégration future des programmes d'application dans le domaine des structures dépend de façon critique de la mise en place de bases de données intégrées. Celles-ci doivent permettre l'accès aux informations selon des séquences essentiellement arbitraires et assurer automatiquement une large part des contrôles d'intégrité des données. Les systèmes de gestion de bases de données (SGBD) issus des applications de gestion constituent un point de référence. L'article examine ces systèmes et propose certaines extensions nécessaires à leurs applications dans le domaine technique.

ZUSAMMENFASSUNG

Die zukünftige Vereinheitlichung von strukturellen Applikationsprogrammen wird kritisch von integrierten Datenbanken abhängen, die Zugang zu Information in wichtigen, willkürlichen Abläufen verschafft und die automatisch eine breite Integritätsüberprüfung ausführen. Eine Quelle solcher Entwurfsdatenbanken sind die Datenbank-Management-Systeme, die von Management-Anwendungen stammen. Dieser Bericht betrachtet solche Systeme und behandelt einige erforderliche Zusätze.

1. INTRODUCTION

The decade of the 80's will see a major trend towards the integration of stand-alone structural engineering application programs into comprehensive design systems. The bunifying element of such systems will be a central data base, containing up-to-date information about the evolving design. The organization and operation of such a database has to be drastically different from the file or secondary storage management systems now in common use. Integrated databases must allow users and application programs to access information in essentially arbitrary sequences without regard to the internal organization of data. Furthermore, such databases must automatically perform a large portion of the consistency and integrity checking on the data, both within the structural design process itself and in integrating structural data with other disciplines participating in the design.

Database management systems (DBMS's) incorporating physical and logical data independence and consistency enforcement have evolved for management applications, and many DBMS's are commercially available. A major challenge is to evaluate such systems for their applicability to structural design and to implement necessary additions or extensions.

The paper is organized into three parts. First, the the present use of data are critically evaluated. Second, the relevant concepts of DBMS's and the major existing database models are briefly introduced, with primary emphasis on the relational model. Third, a class of necessary extensions to the relational model is presented, dealing with the representation and processing of constraints that arise in structural design. Illustrations are drawn from a prototype structural design database system recently developed.

2. DATABASE IMPLICATIONS OF STRUCTURAL ENGINEERING COMPUTER USE

2.1 Trends in Computer Use

Computer usage in structural engineering design has had an explosive growth in the past 15 years. Around 1970, the only common structural engineering applications were analysis (usually restricted to linear elastic models), and detailing of certain repetitive structural components. Except for some simple file structures for storing input data and results or for providing restart capabilities on long runs, essentially all data had a lifespan restricted to the duration of a single computer run.

Since the early 1970's, computer usage has grown in four distinct directions, all of which have a major impact on the lifespan of the data and the manner in which they are used:

- **Increase in depth.** Computer programs are now commonly used at all stages of design of a structure (conceptual, preliminary and detailed); and for fabrication and construction scheduling and control. It has therefore become important to "capture data at the source," so that the output of one stage can serve directly as input to the next one.

- **Increase in breadth.** The range of programs used at any one stage has grown similarly. For example, it is now common practice to perform a linear, static analysis for one set of loads, and a dynamic and/or nonlinear analysis for another set of environmental conditions. While some large analysis packages support all these analysis options, it is not uncommon that different programs must be used for the different analyses. It is therefore necessary to have the common input data available in a *program-independent* fashion.

- **Increase in integration.** As the other design disciplines (e.g., architecture, mechanical and electrical engineering) have increased their computer usage, there has been an increasing need for two-way interchange of information between the structural engineer and the other disciplines; it becomes less and less desirable to re-enter the shared data manually or even to re-format them with ad-hoc programs.

- **Increased need for flexibility.** The combination of the three factors described produces a fourth one, namely, the need to accommodate much more flexible design sequences than was the practice in the past. Thus, programs must accommodate a much larger range of "inputs," in terms of the decisions previously made.

2.2 Present Usage of Data

The present methods of data storage and access in structural engineering fall short of the needs. These methods generally belong to one of the following three categories:

- **Temporary files.** Most large structural analysis programs store and access information in temporary files (usually unformatted or binary) for segmentation purposes, backup/restart, or postprocessing).

- **Explicit interface programs.** When the output of one program serves as input to another, interface programs are written for performing the necessary conversions and reformatting.

- **Text files.** Many organizations save both input and output data in alphanumeric formatted text files. Input files can be edited for changes and resubmitted for reanalysis, and output files can be scanned for relevant information retrieved.

The first two methods exhibit strong physical and logical data dependence, i.e., the content and organization of the data is totally dependent on the needs of the programs using them. The last method, by contrast, exhibits *data ignorance*; the file management system only knows the name of the file, and knows nothing about its contents.

These are a number of notable exceptions to the limited range data storage and access methods discussed above. Only a few of these can be reviewed here.

- Both GENESYS [9] and ICES [18] contain a common system for data storage and communication among subsystems. ICES contains a subsystem, TABLE, specifically for storage and access of data. POLO [10] goes a step further and contains a database management system, FILES [11].

- A number of application-independent centralized databases have been developed. In systems such as IPAD [3, 15] or COMRADE [1, 17], all data are stored in a single common pool and are accessible to all users.

- General finite element pre- and post-processors such as UNISTRUC [22] and FASTDRAW [13, 14] represent the model in a "neutral file"; when model generation and manipulation is completed, a "source file" for a specified analysis program is generated. Similarly, results from any of the supported analysis program can be reformatted into a "neutral file" and post-processed.

- GLIDE (Graphical Language for Interactive Design in Engineering) [5, 6] is a

prototype system using a flexible database supporting high level data abstractions and geometric modelling capabilities.

A recent survey of database applications in engineering practice [2] shows that 53 out of 153 firms surveyed use some form of database applications, but that the majority of the use is in accounting, personnel records and project management. Only 23 firms use databases for analysis or design.

2.3 Shortcomings of Present Usage

The present methods of data storage and access have three major shortcomings. First, these methods are not suited for supporting flexible design sequences, where the programs may interact in a variety of sequences, where multiple iterations have to be performed, and where multiple alternative designs may have to be generated and results compared. Such flexibility cannot be provided if the internal representation (content and organization) of the data is integrally dependent on the specific program(s) using the data.

Second, these methods provide no general mechanisms for either querying or updating the database. All updates or queries must be specifically programmed, using the specific internal representation of the data.

Third, and possibly most importantly, there are no general ways of insuring or monitoring the integrity and consistency of the stored data. One cannot impose constraints such as "beam depth ≤ clearance allowed" unless one knows exactly where and in what format "beam depth" and "clearance allowed" have been stored.

3. DATABASE MANAGEMENT SYSTEMS

3.1 The System

In administrative data processing applications there has been a major trend toward the use of DBMS systems. DBMS's are designed to store data in a manner independent from programs, to allow programs to access and retrieve data, and to provide a means of inserting, deleting, and modifying data. A DBMS can be represented as a series of layers:

- Core. The actual data as it is stored on a physical device.

- Interface. The layer immediately surrounding the central data core is a collection of software that enables the core to be used. It is the communications link between the stored data and users; it acts as a transfer mechanism that converts the "raw" data from its stored physical form to a logically structured format.

- Users and applications. The outermost layer consists of database users and application programs accessing and performing operations on the data.

Each level of the DBMS contains a different *view* or description of the data. A view of the core shows the physical layout of the data as it resides on a storage device. This is the view of systems designers and programmers concerned with system performance, data indexing, and data location [4]. The organization of the data at this level is referred to as the physical structure of the data.

Software associated with the second layer of the DBMS converts the physical structure of the core data to a logical structure or *schema*. The schema is an overall *representation*

of the data describing it in logical rather than physical terms. The overall schema is further converted by the software associated with the outermost layer into smaller views referred to as *subschemas*, portions of the overall logical data representation used by application programs. The schema provides an overall view of the logical representation of all of the data, while a subschema provides a limited view structured for a particular application.

One of the key objectives of a DBMS is to allow different applications to use the same data in a program—independent fashion. To do so requires a data structure where all data is pooled together at the physical level and selected views of the data are appropriately available.

The DBMS must also provide the capability for the database to grow and change as needs dictate. Over the life of the database it must be possible to dynamically change its logical structure as new types of data are added and new programs developed.

Flexibility in a database is possible only if *physical* and *logical data independence* is maintained between the layers of the DBMS. The physical data structures of the core must be entirely separate from the data structures of the logical outer layers perceived by users and programs. The connection between the layers is maintained by the DBMS software. Changes to either the data or the programs can be made without requiring changes to the other. Only the layer of DBMS software that interfaces with the change needs to be modified. In this manner the database need not be restructured when new programs are written and existing programs need not be rewritten when changes are made to the data structure.

There are three major capabilities provided by a DBMS:

• Data definition. Data definition defines the schema and builds the *framework* into which attribute values are placed. Data definition is performed by the database administrator using a data definition language (DDL).

• Data modification. Data modification includes insertion, modification, and deletion of data values and is performed by the database user using a data manipulation language (DML).

• Data retrieval. Data retrieval consists of obtaining desired information from the database and includes the ability to search, manipulate, and query without the necessity of writing application programs. Data retrieval is performed by the user using a *query language*.

The DBMS communicates with application programs written in a standard programming language through a *host language interface*. The interface consists of a set of call statements to DML procedures that initiate the desired operations on the database.

3.2 Common Data Models

A *data model* defines the overall logical structure of a database [12]. It provides the structural framework into which the data are placed. Three database models have come into common usage: the hierarchical, network, and relational models.

Hierarchical Data Model. The hierarchical model is *tree structured*. It is composed of *nodes* connected together by *links* as shown in Figure 1. The nodes may be grouped into horizontal layers called *levels*. As Figure 1 shows, a hierarchy is a *multilevel* data model. The tree structure of the hierarchical model implies that each node may be linked to more

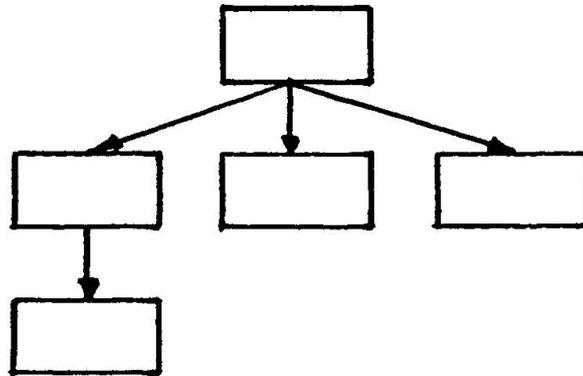than one node below but to only one node above itself [12].

Figure 1: Hierarchical Data Structure

A node represents a *type* of entity about which information is stored. An entity may be an object such as a bolt, a column, or an entire frame. Each entity has certain descriptive information associated with it. This information determines the entity *type* and is referred to as the *attributes* of the entity. Links represent relationships between the entity types. The direction of the link indicates a relationship of one to many from the tail of the arrow to its head.

Network Data Model. A *network* is a directed graph. The network model is a multilevel data model in which each node may be linked to more than one other node in *both* upward and downward directions [4]. This is the distinguishing difference between the hierarchical and network models; it allows relationships to be established horizontally within levels between different entity types as well as vertically between levels.

Figure 2 illustrates an example of a network data structure. It has basically the same structure as the hierarchy of Figure 1 with the addition of multiple network links. It can readily be seen that the hierarchical model is a special case of the network model.
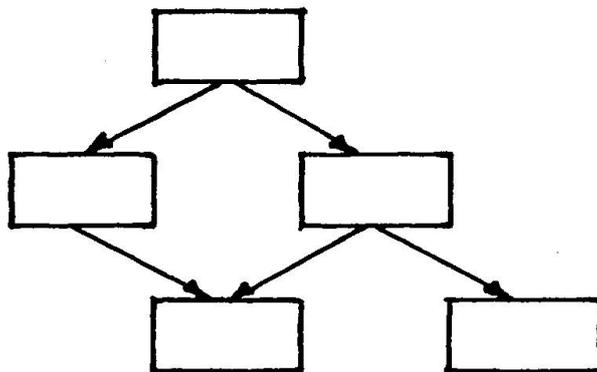
Figure 2: Network Data Structure

Relational Data Model.   A *relational model* is a *single level* model consisting of a collection of *relations* represented in two-dimensional tabular form [19]. Associated with the relations is a set of operators that allow for the insertion, deletion, modification, and retrieval of data. A figure for the relational model similar to Figures 1 and 2 would simply contain a collection of nodes without any links between them. There are no predefined hierarchies or networks in the relational model. Links needed between nodes are automatically created by the relational DBMS upon demand and an access path is established to any node.

Figure 3 illustrates the structure of a relation. The rows of a relation called *tuples* and its columns are called *attributes*. All attribute values are drawn from the same *domain* i.e., they are of the same data type. Each tuple represents an entity and contains a value for each attribute. All tuples are distinct; duplicates are not permitted [16]. Tuples and domains have no order; they may be arbitrarily interchanged without changing the data content and meaning of the relation. Tuples are accessed by means of a *key*, a single attribute or a combination of attributes that uniquely identifies a tuple.

A standard shorthand notation to represent relations is as follows:

RELATIONname (ATTRIBUTE1name, ATTRIBUTE2name, . . .)

with the BEAMS relation of Figure 3 being represented as:

BEAMS (Designation, A, d, b$_f$, t$_f$, t$_w$).

The name of the relation is listed first followed in parentheses by the names of all of its attributes. The underlined domain of a relation is the key.

BEAMS

Designation	A	d	bf	tf	tw	
W12x50	14.70	12.19	8.077	0.641	0.371	
W12x58	17.10	12.19	10.014	0.641	0.359	←---*TUPLE*
W14x26	7.67	13.89	5.025	0.418	0.255	
W14x38	11.20	14.12	6.776	0.513	0.313	

ATTRIBUTES ←--- (points to header row)

←----- RELATION

--DOMAIN

Figure 3: The Structure of Relations

3.3 Model Comparison

The similarities between the multilevel hierarchical and network models are evident. The network model is more flexible, allowing non-hierarchical or, multi-hierarchical relations to be defined. This added flexibility results in greater representational power, although it still does not afford the representational capabilities of the relational model.

Relationships among entities and constraints among attributes impose critical requirements on a database. All user queries and updates generally cannot be anticipated prior to the establishment of the structure of the database. To satisfy diverse access needs, links may be required between any of the components of the database. Both the hierarchical and network models are composed of precisely defined links between the nodes. As a result, the structure of the database is fixed and cannot easily be changed.

Both the hierarchical and network models present difficulties in representing many to many relationships. An additional disadvantage is that *loops* are not permitted, i.e., relationships cannot be established between a record type and itself. This is particularly disadvantageous in a structural engineering database where such relationships are common: beams are connected to adjacent beams; columns to adjacent columns; etc..

Neither of these disadvantages occurs with the relational model. Its use requires knowledge of only one data construct and its underlying access mechanisms are hidden from the user. The user needs to be concerned only with the content of individual relations. The hierarchical and network models do, however, allow for efficient implementations. Because hierarchy and network links are implemented as pointers, node traversal is direct and fast. These efficiencies contrast with the relational model whose primary disadvantage at the present is its lower efficiency of accessing.

An additional advantage of the relational model is its ability to avoid common anomalies through *normalization*, the process of removing dependencies from among the attributes of relations. The concept of normalized relations is an integral part of the relational model and it promotes the achievement of well structured data while providing a degree of automatic integrity and consistency checking.

Most existing commercial database systems are based on the hierarchical and network models. The relational database model is the newest model and only recently have commercial relational database systems such as RIMS [7] and SQL [20, 21] become available. However, as efficiency problems are reduced, relational implementations will see far greater use. The versatility and flexibility of the relational model make it ideal for managing structural engineering data.

4. THE FUNCTION OF DATABASE MANAGEMENT SYSTEMS IN STRUCTURAL ENGINEERING

4.1 Potential Role

DBMS systems offer many advantages for structural engineering use. Program-data independence is one of these. A DBMS stores data in such a way that many different applications, both existing and planned, can use it. The data is structured for efficient use but it is not tailored to a particular application program.

DBMS systems provide a structure for storing a wide variety of data. A collection of data far more global than that needed by a single application program or a single design

discipline can be achieved, linking together many different applications around commonly shared data. This integration offers promise of significant improvement in overall information storage and handling.

DBMS systems provide the conceptual framework for organizing design data so that all database needs can be efficiently and uniformly supported. Two aspects of DBMS's are of particular significance. One is the intellectual discipline required to formulate the overall framework or schema of the entire database, an absolutely essential step but one likely to be ignored when a system of programs is allowed to grow in an uncoordinated fashion. It is worth emphasizing that the schema can be extended and the database reorganized as needed without affecting existing applications. The second significant aspect is the availability of query languages, so that many accesses to the database can be made directly, without the need to first develop an application program.

It is to be expected that structural engineering design systems will be developed based on both the network and relational data models (hierarchical models have already been essentially supplanted by network models). In view of the conceptual advantages of the relational model and the vast amount of research and development for making it more general and efficient, it is likely that eventually it will become predominant. The comments that follow are based on this development.

4.2 Extensions Needed

To assume a major role in structural engineering, extensions to the present relational DBMS capabilities are needed. These are particularly necessary in the areas of run-time storage management, extension of data types, and integrity management.

Run-Time Storage Management. The efficient operation of data dependent systems and the integrating capabilities of data independent systems are both desirable characteristics. The logical extension to existing DBMS systems to achieve both simultaneously is a facility for run time storage management. Such a facility would temporarily restructure those portions of the database needed by an application program during its operation. Application programs interfacing using a DBMS with a run time storage management system would be treated as modules that utilize the DBMS only for input and output. For program input the DBMS would *export* the data to the storage management system. Output data from the application program could be processed directly using DBMS facilities.

Extended Data Types. From a structural engineering perspective one of the significant drawbacks of existing relational DBMS's is their limited number of available data types: integers, reals, and character strings. These types are sufficient for business applications but engineering applications require the use of a wider range of data types including vectors, arrays, matrices, etc..

Integrity Management Enhancements. Extensions are also needed in the area of integrity management. The built-in integrity controls of present relational DBMS systems are extremely limited. In the majority of cases only constraints on single attributes of single tuples can be enforced. These are not sufficient to satisfy engineering design integrity management needs. Functional dependencies between multiple attributes of tuples must be established. Where functional dependencies result in defined constraints, those constraints must be automatically enforced. Mechanisms to achieve this integrity capability are introduced below.

5. INTEGRITY AND CONSISTENCY ISSUES

5.1 Functional Integrity

Functional integrity and consistency with respect to governing laws must be enforced by application programs. These constraints cannot be built into the database because they must invoke high-level complex sequences of computations.

For example, a structural analysis application program must output a set of member forces and member properties in accordance with structural laws of compatibility and equilibrium. As a result the interface between the database and the program is very limited. The interaction between them is simply an input/output transformation. It is taken for granted by the database that the output it receives from the application program is consistent with the constraints built into the program.

5.2 Constraint Needs

Although the global constraints mentioned above cannot be built into the database, there are many constraints and design functions that can be. Design criteria, interaction constraints, consistency constraints on redundant data, and iteration control are examples of constraints that must be enforced in a structural engineering database.

Design Criteria. Design criteria specify limitations on attribute values and may be defined by codes, standards, and specifications or by a database user or designer. Codes, standards, and specifications comprise a class of constraints designed to assure the functionality and usability of the entity which they govern. They consist of a set of requirements that govern the design and behavior of the components of the entity [8]. Examples of such design criteria are:

$$M_{provided} \geq M_{required} \text{ for a member, and}$$

$$b / t \leq K * \lceil F_y \text{ for a plate element of a steel member.}$$

User-defined constraints arise from each user's personal design style. They may be issued at any point during the design process and incorporated into the database. An example of a user-defined design criterion would be:

$$F_y \leq 50ksi$$

limiting yield stress to a maximum value of 50 ksi.

Interaction. Each design discipline deals with a set of attributes commonly used by designers of the discipline. Often individual attributes of one discipline are related to attributes of other disciplines. Such attributes are referred to as boundary attributes. Interaction constraints are those which define relationships between boundary attributes. One example of an interaction constraint is:

$$d \leq clearance.$$

This constraint relates the beam depth, d, under control of the structural designer, to the available clearance, dealt with by the architectural subsystem.

Consistency. Consistency is a special case of integrity and deals either with an attribute redundantly stored in more than one database location or with dependencies among attributes where dependent values are calculated on the basis of independent data and constraints among the attributes. In the first case each occurrence of the attribute is

equivalent and must have the same value. In the second case the value of the dependent data may be computed when it is needed.

The constraint among beam section properties:

$$S = 2 * I / d$$

can be used to illustrate redundancy and consistency. If the values of S, I, and d are all stored, their integrity must be maintained with regard to the constraint, i.e., if the value of any one of these attributes is changed, the value of one of the others must also be changed; otherwise the consistency of the relationship and the integrity of the database are violated. The constraint is thus used in a passive *checking* mode. Alternatively, it is possible to store any two of the three attribute values, calculating the third as needed. In this manner the constraint is used in an active *assignment* mode and integrity is automatically maintained.

In structural engineering databases redundantly stored attribute values and consistency constraints among attributes are necessary. Unfortunately, the only tool available in a relational DBMS to prevent redundancy and consistency problems is normalization. But structural engineering data storage needs cannot be satisfied by normalization because they require that a variety of interrelated data be stored together, as shown above, so that all potentially needed data is immediately available for use.

Iteration. One important capability that must be supported by an engineering database is the design of an entity through multiple iterations and the control of the process of iteration. The database must provide the ability to distinguish between iterations and to orderly assemble subsequent generations of data.

The control of iteration can be achieved using constraints. To determine if the result of a structural analysis–sizing iteration has converged, for example, one must measure the difference between the moment of inertia, I, before and after sizing using the constraint

$$| I_{new} - I_{old} | / I_{new} \leq epsilon.$$

If the difference is less than the acceptable tolerance, sizing is complete and another design stage may be initiated. Otherwise, additional iterations are necessary.

5.3 A Model

Introduced above were a number of requirements a database must satisfy to be an effective structural engineering tool. Reference [16] describes a new relational data structuring scheme that eliminates problems of redundancy, update anomalies, and other irregularities caused by intrarelation dependencies among attributes. The new model does so by introducing into relations new attributes that record the *status* of all constraints defined on the relation. The integrity of intrarelation dependencies is thus always monitored. At the same time, the model retains immediate local access to all of a relation's attribute values.

As an example consider the relation

MEMBER (ID, S, I, d)

containing the attributes of the section properties constraint introduced above. The alternative to normalizing this relation is to introduce into the relation the new attribute **sectionOK**

MEMBER (ID, S, I, d, sectionOK)

that monitors whether or not the constraint is satisfied. The attribute sectionOK is a boolean domain whose value is determined by the constraint. The constraint itself is stored in the database as a function. Whenever one of S, I, or d is changed, or when a new tuple is added to the relation, the constraint function is invoked and its value (true or false) is computed and stored in the tuple.

This model can be generalized to handle multiple constraints. All of the constraints introduced earlier can be converted to assignment form as follows:

$$\text{strengthOK} := M_{provided} \geq M_{required}$$

$$\text{sectionOK} := S = 2 * I / d$$

$$\text{clearanceOK} := d \leq \text{clearance}$$

$$\text{iterationOK} := |I_{new} - I_{old}| / I_{new} \leq \text{epsilon}.$$

These assignment statements would be inserted into functions and stored in the database. The modified relation would then contain the following attributes:

$$\text{MEMBER (ID, S, } I_{new}, I_{old}, \text{ d, } M_{provided}, M_{required}, \text{ strengthOK,}$$

$$\text{sectionOK, clearanceOK, iterationOK).}$$

In this manner the status of multiple constraints among intrarelation attributes can be monitored and recorded.

An additional capability provided by the model is the assignment of attribute values to dependent data items in such a way that the governing constraint(s) is automatically satisfied. To do so requires the constraint checking function to be converted to an assignment procedure in which the left hand side of the assignment statement is the dependent attribute. Converting the sectionOK constraint redundant in this manner results in the new assignment procedure:

```
PROCEDURE S (I, d : real;  VAR S : real;  VAR sectionOK : boolean);
    BEGIN
        S := 2 * I / d;
        sectionOK := TRUE
    END;
```

The designer thus gains a new tool with which he can assign results known to be consistent with other attribute values in the database.

The primary advantage of the model is that the database user has direct access to the values of all of the attributes in the relation while insuring a measure of intrarelation integrity and consistency. Even though functional interdependencies are retained, the integrity of the tuples can at all times be determined by invoking the stored checking functions and recording the values returned.

6. CONCLUSIONS

The clear trend in structural engineering computer usage is towards higher levels of integration of individual programs, both horizontally (among different structural applications)

and vertically (between structural applications and the applications of other participating disciplines, from planning to construction management and facility operations). At the same time, there is a similar trend towards more flexible use of programs in response to the wide *range of design and analysis sequences needed in diverse projects.   The data exchanged and shared among the applications is the key to integration.

There are three paths that organizations may take in their approach to data usage. One is to continue extending the present mode of high program-data dependence; this approach is highly self-limiting, and there is a clear indication that this mode cannot be fruitfully continued. Second, we can wait for the development of an "ideal" system, incorporating all structural engineering needs. The third approach, advocated in this paper, is to build on the highly developed – and rapidly developing – area of DBMS's, adding those specific extensions necessary for structural design.

A DBMS-based structural design database will provide much more than a passive repository of data, and will serve as an active agent in the design process. Many of the design control functions will be part of the database management activity.   Consistency management with respect to a wide range of design constraints can be made an integral part of the DBMS. Other consistency management functions, such as monitoring spatial conflicts, can also be included in the DBMS. The need for a centralized database incorporating a high level of consistency management will become even more acute as the trend towards more decentralized computing, including personal computing, accelerates.

ACKNOWLEDGEMENTS

REFERENCES

[1]   Bandurski, A. E., and Wallace, M. A.
        COMRADE Data Management System Storage and Retrieval Techniques.
        In *Proceedings 1973 National Computer Conference and Exposition*, pages 353–
            357. AFIPS, Montvale, NJ, 1973.

[2]   Bland, R.
        Private communication.
        1982.

[3]   Burner, B., Ives, F., Lixvar, J., and Shovlin, D.
        The Design, Evaluation, and Implementation of the IPAD Distributed Computing
            System.
        In *Proceedings of the First Conference on Computing in Civil Engineering*, pages
            126–144. American Society of Civil Engineers, New York, NY, June, 1978.

[4]   Date, C., J.
        An Introduction to Database Systems.
        Addison Wesley, Reading, MA, 1977.

[5]   Eastman, C., and Thornton, R.
        A Report on the GLIDE2 Language Definition.
        Technical Report, Computer-Aided Design Group, Institute of Physical Planning,
            Carnegie-Mellon University, Pittsburgh, PA, March, 1979.

[6]   Eastman, C.
        An Introduction to GLIDE: Graphical Language for Interactive Design.
        Technical Report, Computer-Aided Design Group, Institute of Building Sciences and
            Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 1980.

[7]   Erickson, W. J., Gray, F. P., Limbach, G.
         Relational Information Management System
         Version 5.0 edition, Boeing Commercial Airplane Company, Seattle, WA, 1981.

[8]   Fenves, S.J. and Wright, R.N.
         The Representation and Use of Design Specifications.
         In W.J. Hall (editor), *Structural and Geotechnical Mechanics*, pages 277-304.
             Prentice-Hall, Englewood Cliffs, NJ, 1977.

[9]   Genesys Limited.
         GENESYS.
         Technical Report, Genesys Limited, Loughborough, England, 1976.

[10]   Lopez, L. A.
         POLO: Problem Oriented Language Organizer.
         Journal of Computers and Structures 2(4):555-572, 1972.

[11]   Lopez, L. A.
         FILES: Automated Engineering Data Management System.
         In *Sixth Conference on Electronic Computation*, pages 47-71.   American Society
             of Civil Engineers, New York, NY, 1974.

[12]   Martin, James.
         Principles of Data-Base Management.
         Prentice-Hall, Inc., Englewood Cliffs, NJ, 1976.

[13]   *FASTDRAW/3 Reference Manual*
         MCAUTO — Graphics Products, McDonnell Douglas Automation Company, St. Louis,
             Missouri, 1978.

[14]   *FASTDRAW/3 Interactive Postprocessing Reference Manual*
         MCAUTO — Graphics Products, McDonnell Douglas Automation Company, St. Louis,
             Missouri, 1980.

[15]   Miller, R. E. et al.
         Feasibility Study of an Integrated Program for Aerospace Vehicle Design (IPAD).
         Technical Report, Boeing Commercial Airplane Company, Seattle, WA, 1973.

[16]   Rasdorf, W. J.
         Structure and Integrity of a Structural Engineering Design Database.
         Technical Report DRC-02-14-82, Design Research Center, Carnegie-Mellon
             University, Pittsburgh, PA, April, 1982.

[17]   Rhodes, T. H.
         The Computer-Aided Design Environment Project (COMRADE).
         In *National Computer Conference*, pages 319-324.   AFIPS Press, 1973.

[18]   Roos, D.
         ICES System Design
         Second Edition edition, The MIT Press, Cambridge, MA, 1967.

[19]   Sandberg, G.
         A Primer on Relational Database Concepts.
         IBM Systems Journal 20(1):23-40, 1981.

[20]   *SQL/ Data System General Information*
         International Business Machines (IBM), White Plains, NY, 1981.
         Report GH24-5012-0.

[21]   *SQL/ Data System Concepts and Facilities*
         International Business Machines (IBM), White Plains, NY, 1981.
         Report GH24-5013-0.

[22]   *UNISTRUC 2 Reference Manual*
         Control Data Corporation, Minneapolis, MS, 1979.

Engineering Databases

Bases de données pour ingénieurs

Datenbanken im Ingenieurwesen

Klaas van der WERFF
Dr. Ir.
Univ. of Technology
Delft, the Netherlands

Klaas van der Werff, born 1942, obtained his mechanical engineering degree at the University of Technology, Delft, the Netherlands. As a senior researcher at the Delft TU he was involved in the finite element formulation of the kinematics and dynamics of mechanisms. Present work is directed to CAD.

SUMMARY

To investigate the applicability of Database Management System (DBMS's) for engineering purposes, some prototype-implementations have been performed. These implementations were based on the use of a Codasyl-type DBMS and a relational type DBMS. The applications involved include storage and manipulation of a finite element network. One of the major conclusions from the study is, that due to the fact engineering applications have to keep record of part-subpart relationships, geometric relationships as well as computational relationships, DBMS's are of limited use in an engineering environment. In particular, control structures in the application software tend to be too complex.

RESUME

Afin d'examiner les possibilités d'application pour l'ingénieur des systèmes de gestion de bases de données (SGBD) un certain nombre d'expériences pilotes ont été réalisées. Elles ont mis en œuvre des systèmes répondant aux normes CODASYL et de SGBD relationnelles pour la manipulation de réseaux d'éléments finis. Les conclusions de l'étude conduisent à considérer que ces systèmes sont d'un intérêt limité dans l'environnement technique de l'ingénieur, dû au fait notamment que les relations à maintenir entre les structures et les sous-structures, sur le plan de la géométrie et du calcul, deviennent trop complexes par ce type de méthodes.

ZUSAMMENFASSUNG

Um die Anwendbarkeit eines Datenbank-Management-Systems für Ingenieurzwecke zu untersuchen, wurden einige Prototypstudien durchgeführt. Diese Implementationen wurden auf einer «codasylen» und einer «relationalen» DBMS aufgebaut. Die Anwendungen schlossen die Speicherung und Manipulation eines finiten Element-Netzwerkes mit ein. Eine der hauptsächlichsten Schlussfolgerungen dieser Studie ist, dass DBMS in der Ingenieurumgebung beschränkt anwendbar sind, weil ingenieurmässige Anwendungen Teil-Unterteil-Abhängigkeiten, geometrische Abhängigkeiten und berechnete Abhängigkeiten aufzeichnen müssen. Im speziellen neigen Kontrollstrukturen in der Anwendungssoftware zu grosser Komplexität.

1. INTRODUCTION

Computer Aided Design is rapidly advancing. The application of databases is an important component in the development, particularly because the use of databases is a kind of integrating factor in order to come to computer aided engineering.
In order to get insight in the essentials of engineering data base management systems, in 1980 a project group has been formed in which 9 companies and institutions cooperated. This project group is one of the activities of CIAD, a dutch institute for computer applications in engineering practice. The results will be made available in the form of a final report.
The study is directed to the following topics:
  - formulation of a set of demands which has to be met by an engineering application of DBMS's,
  - investigation whether or not the DBMS's selected provide adequate facilities to meet the demands,
  - providing a checklist which covers all relevant aspects in selecting a proper DBMS.
In addition the applicability of DBMS's for engineering purposes has been investigated. To this purpose some prototype implementations have been performed. This last aspect gets much attention in the present publication.

2. WORKING METHOD

The three major topics were treated by separate working groups. Another subgroup was asked to study a specific CODASYL-type DBMS and a relational type DBMS. Furthermore two characteristic engineering problems should be implemented. The experience was distributed among the whole project group.
The selection of the characteristic problems had to be made first, because this influenced the work of other subgroups also. The members of the project group were asked to formulate problems representative for their field of interest. The outcome of this enquiry made clear that a wide variety of engineering activities had to be covered. The DBMS should not only deal with a constant part for storage of standardized parts, design procedures and material properties, but also with structures, machines and installations composed from these elements. Furthermore the DBMS should be used for storage and retrieval of measured and calculated data in such a way that the origin of these data would be maintained. The DBMS should support typical engineering activities such as analysis and modification of complex structures. Various ways of presentation of the database contents must be possible, including graphical presentation of objects and measurement data.

3. REQUIREMENTS FOR AN EDBMS

For the definition of the programme of requirements for an engineering database management system it was decided that the starting point had to be the engineers view on the problem. First of all the requirements and design criteria were studied from a number of existing systems. This led to a first formulation of the requirements. Next the participants of the working group were asked to describe their engineering problems for which an EDBMS was thought to be a solution. In consecution these representative problems were confrontated with the first draft of the programme of requirements mentioned before. After evaluation the programme of requirements was brought in a final form.
In general an EDBMS should meet at least the requirements for an administrative DBMS. Specific properties directed to engineering processes must then be added. These specific properties are determined by the character of the engineering process, including the whole cycle of problem definition, design, production and testing. Within the four phases indicated a number of characteristic

activities can be distinguished:

- conception   - decision   - production
- analysis   - administration   - checking
- calculation   - documentation   - testing

It must be noticed however that the engineering process contains many loops in which activities with a routine character such as analysis, are followed by activities with a creative character such as selection. The routine work should be done by the computer where the creative part is a human activity. The EDBMS should support this process.
The requirements to be imposed to a EDBMS depend very much on the character of the engineering process. These requirements can be grouped as follows.

-1. general requirements

A number of general requirements have to be satisfied, such as security, concurrent use possibilities and maintenance.

-2. user friendliness

Both the engineer end user as well as the application programmer have their own ideas about database systems.
- They are not interested in the internal structure of the system.
- They expect the computer to be fast in jobs they can do fast.
- They want to see something happening after a minimum of input.
- They don't worry about the availability of computer facilities.
- They must be protected against themselves.
- They want to sophisticate their system use as they grow familiar with the system.
- They want to be warned for possible catastrophes as a result of their own action.
- They are allergic for manuals and slow in understanding the meaning of error reports.
- They want to use the system casually with a minimum of preparation.

-3. data

Which data is to be stored in depends on the function of the engineering processes for which the database is used. In almost every case the ultimate function is a description of an installation consisting of a large set of elements, groups of elements and their mutual relations. Elements and groups of elements have the meaning of the all necessary information varying from a single parameter, via groups of function values to reference to an external file. The relations mentioned have the meaning of defining the functional relationship between the elements. They are essential for the changing configuration during the design process. Without further comments a number of requirements originating from the data is given below.
- elements, groups of elements, mutual relations must be handled
- these data are often fuzzy
- storage and retrieval of criteria and design rules
- the data structure has in principle a dynamic character
- stepwise refinement of structure definition and variables must be possible
- the origin of the data must be administrated
- two types of data: dynamic project data (product database) and static data (constant database) with library structure
- topological and geometrical description of a spatial structure of installation
- separate storage of graphics structures
- coupling between graphics and functional structure.

-4. aspects of use

The EDBMS should allow multiple ways of use such as:
- multiple use of data
- user friendly interface with analysis programmes
- very general graphical-alpha numerical report facilities
- interactive use possible (low response times)
- hardware independant.

4. PRACTICAL EXPERIENCE IN APPLYING GENERAL DBMS's

The primary objective of research was answering the question: "what are the possibilities and problems when using administrative DBMS's for engineering purposes?". In order to answer this question the following procedure has been applied.
- inventory of that specific engineering problems in which a DBMS approach might be helpful within the reach of the members of the projectgroup.
- definition of two characteristic problems based on these engineering problems.
- design of a general conceptional scheme covering these two problems.
- implementation of this scheme on a network DBMS, a relational DBMS and also on an 'Integrated Engineering System'.
- evaluation.

The inventory of specific engineering problems has already been described in chapter 2.

4.1. Selection of two characteristic problems

Two problems were selected for being characteristic problems:

a) channel plate problem
b) finite element structure

These problems were chosen for the following reasons.
The channel plate problem, which will be discussed in detail later on, is a typical example of an assembly of predefined parts. The parts have different connections with various physical and geometrical properties. Apart from the geometrical position in the assembly, also the logical connections between the parts are essential. The database should contain all information necessary for graphical display and for engineering calculations and storage. The finite element structure was chosen for two reasons. First of all a finite element model has the possibility of processing a large amount of data with a simple structure. Second we have the possibility to work with different levels in the database e.g. a ship structure divided into different sections.

4.2. Design of the general conceptual database

As a first demand it was stated that both characteristic problems should be stored in the same database. For that reason a general conceptual scheme had to be developed to meet this demand. It is to be considered that the scheme cannot result in an optimal result for the individual problems.
Much attention was given to the problem of modeling the hierarchial structure present in all engineering problems. It is common use to structure a design according to the scheme in Fig. 1.
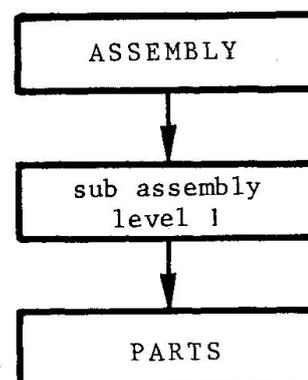
Fig. 1. Hierarchial structure.

The disadvantage of such a model is that
the number of hierarchical levels is
undefined beforehand. Furthermore a sub-
assembly can appear at different levels
in assemblies. For this reason an approach
was chosen according Fig. 2, which is
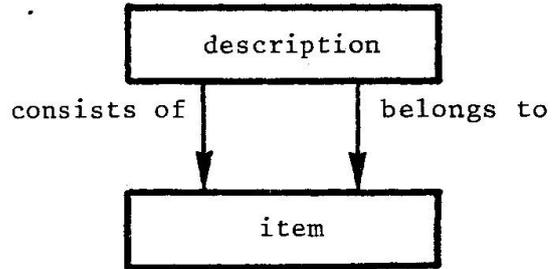much like commonly used 'Bill of materials
model'.

Fig. 2. Alternative model allowing
hierarchical structure.

It is shown in the occurance diagram Fig. 3 how a hierarchical structure can be
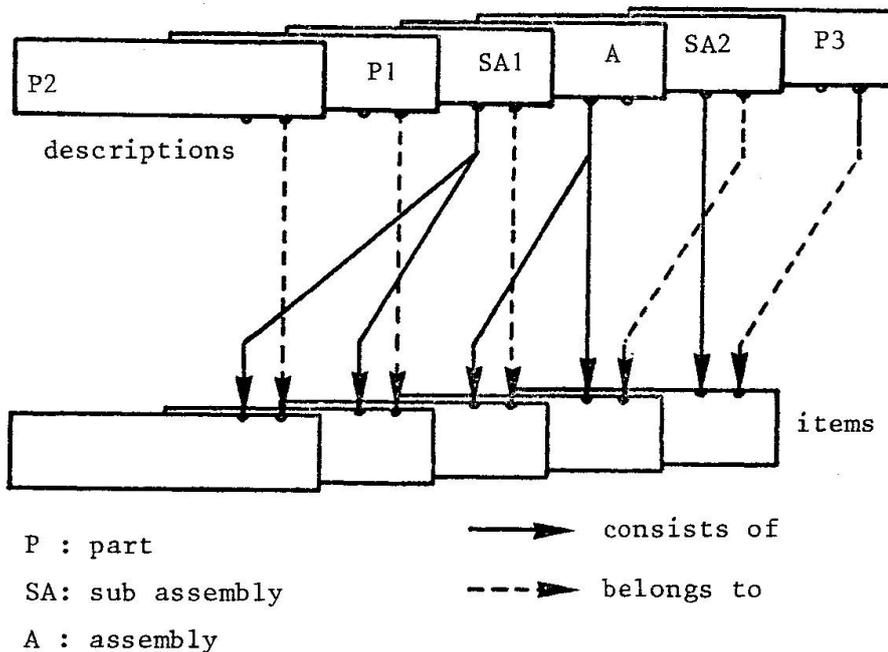modelled according to the bill of materials structure of Fig. 2.

P : part
SA: sub assembly
A : assembly

────▶ consists of
── ─▶ belongs to

Fig. 3. Occurance diagram of assembly.

In the proposed scheme we have two types of records: the description records
and the item records. The description record contains a formal description of
an assembly, sub-assembly or part. The item record represents an instance of a
sub-assembly or part according to the description it belongs to. The item record
contains information about the geometrical position of the item as a part of the
(sub) assembly to which it belongs. A sub assembly can thus be regarded as a
part of the main assembly. Item records always belong to a single assembly.

The method will be illustrated by the problem of a bicycle. We consider the
following assembly of a bicycle, Fig. 4.
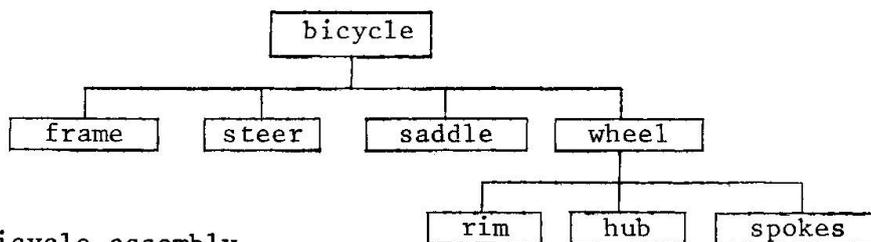
Fig. 4. Bicycle assembly.

In Fig. 5 the model of the bicycle is shown at occurance level.
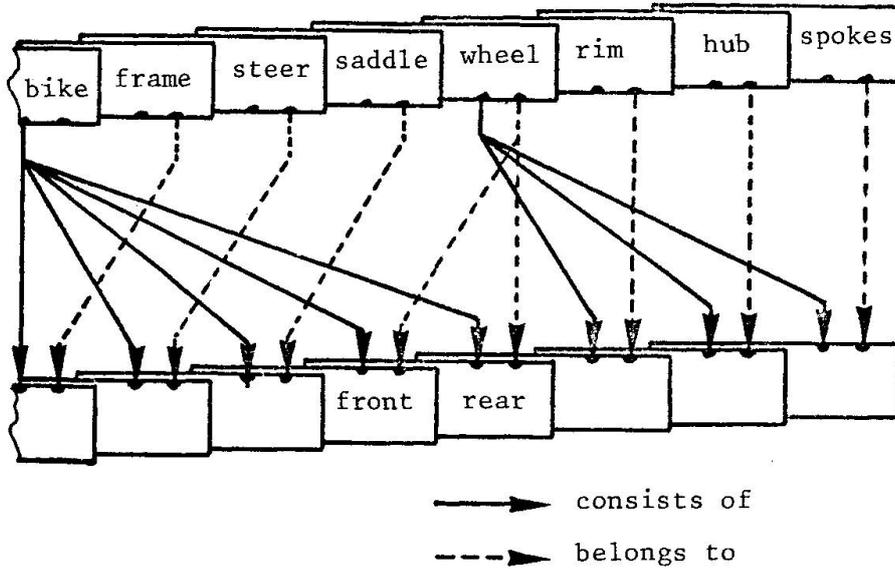
Fig. 5. Occurance diagram of bicycle assembly.

In order to be able to make design calculations the model of a structure or
component must contain the necessary physical connections between the parts.
This means that for each part connections points have to be defined. To set
up the physical connections we need link records between the connection points
(Fig. 6).
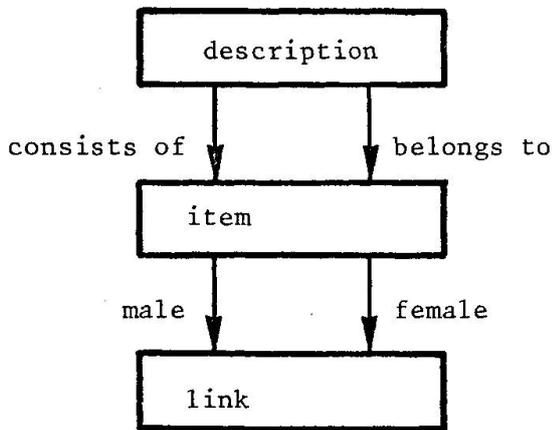
Fig. 6. Extended scheme showing link
record.

The link record is related with the two connected items by means of two sets
named male and female. This implies that linking of two items has a direction
aspect. We did not try to overcome this inconsistancy which is typically due
to the demands of the database system.
The connection points of each item are assigned an indentification.
The link record contains the identification of the linked connection points.

In Fig. 7 this process is shown for the example of the bicycle.
As an example a link record is shown which describes that connection point ≠ 3
of the frame (item No. 1) is connected with connection point ≠ 1 of the wheel
(item No. 4).
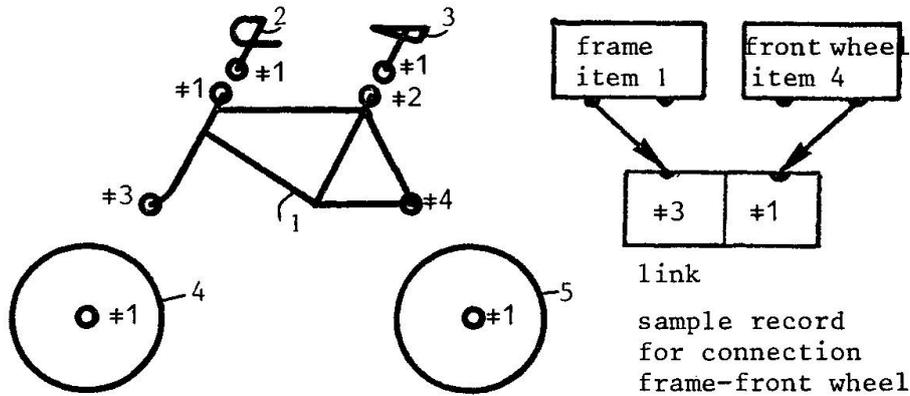
Fig. 7. Physical connection by link record (example).

An extra facility was designed in order to be able to distinguish connections
on different hierarchical levels. However this facility was not used in
practice.

In order to bring some structure in the heap of description records an index
function was used. The index allowed us grouping of selected records. This was
used for building a catalogue of standard components and a catalogue of projects
like the ship structure and the channel plate designs.
The complete scheme as it was used is shown in Fig. 8.
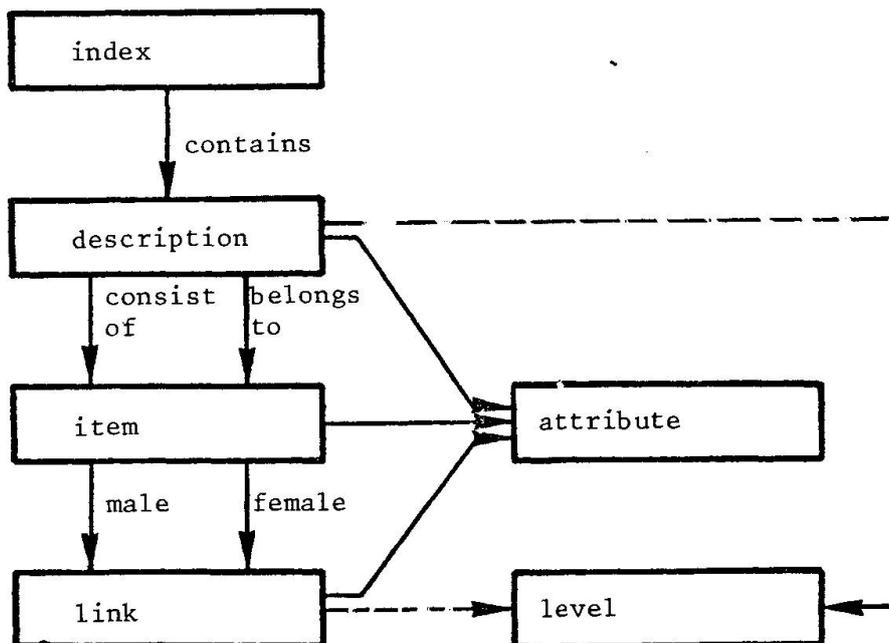
4.3. The channel plate problem

4.3.1. Channelplates

In designing logical circuits with pneumatic logical components, one solution is to use so called channel plates. The application of channel plates is similar to the use of printed circuits in electronics.
The channel plate integrates the functions of mounting components to a base plate and interconnection of the components.
When covered with the base plate, the milled grooves in the channel plate form the connections between the connection pins of the logic components. When necessary multiple layers can be used (Fig. 9).
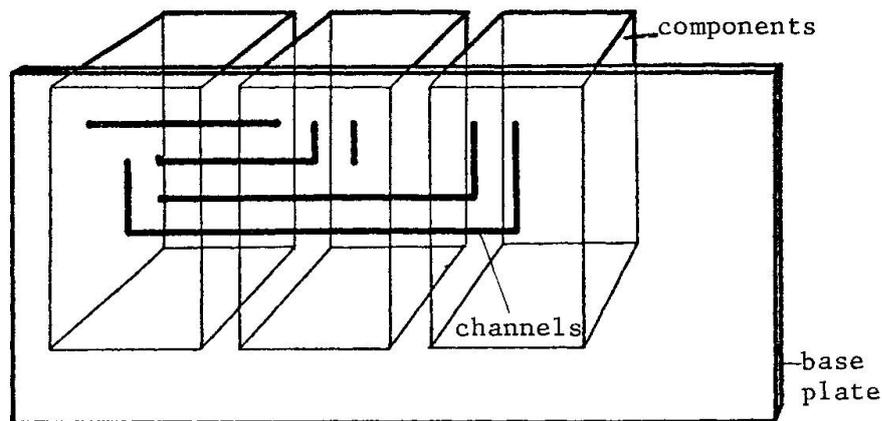
Fig. 9. Channel plate.

Computer Aided Design of the channel layout and of the manufacturing with NC tools has been a subject of study at the Delft University of Technology. The starting point of the design is an existing scheme showing the components used and their interconnections.

An important aspect of the computerized design of channel plates is the method of storage of the data describing the problem. In this report we consider the possibility of storing the data in a data base using the conceptual scheme according to Fig. 8.

The following data have to be considered:
- a catalogue of logic components containing the identification and dimensions of the component and its connection pins,

- a parts list containing the component names and part numbers used in a specific circuit,

- a connection table containing the complete specification of the pins which are to be connected,

- the placing of the parts on the channel plate containing the positions of each component.

4.3.2. The logic component catalogue

For the purpose of channel plate layout design the information shown in Fig. 10 is essential.

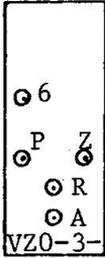| component identification	VZ0-3-PK-3	
dimensions	26	68
penname and position	A	13.	10.
	R	13.	18.
	P	5.	26.
	Z	21.	26.
	6	5.	42.

Fig. 10. Component description.

A specific logic element description will be accessed either by its component identification or by the fact that a component is a part of a specific circuit. The catalogue is realized using the indicated part of the conceptual scheme (Fig. 11).
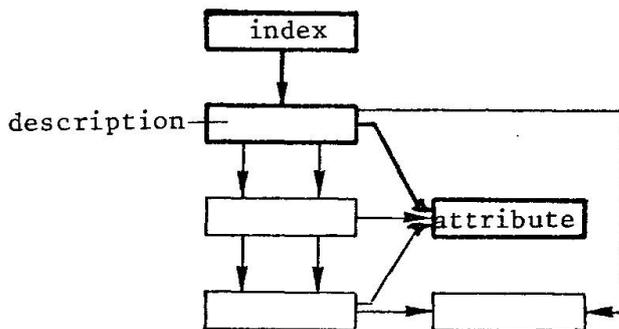
Fig. 11. Catalogue function.

An example of a catalogue containing two logic components is shown in Fig. 12.
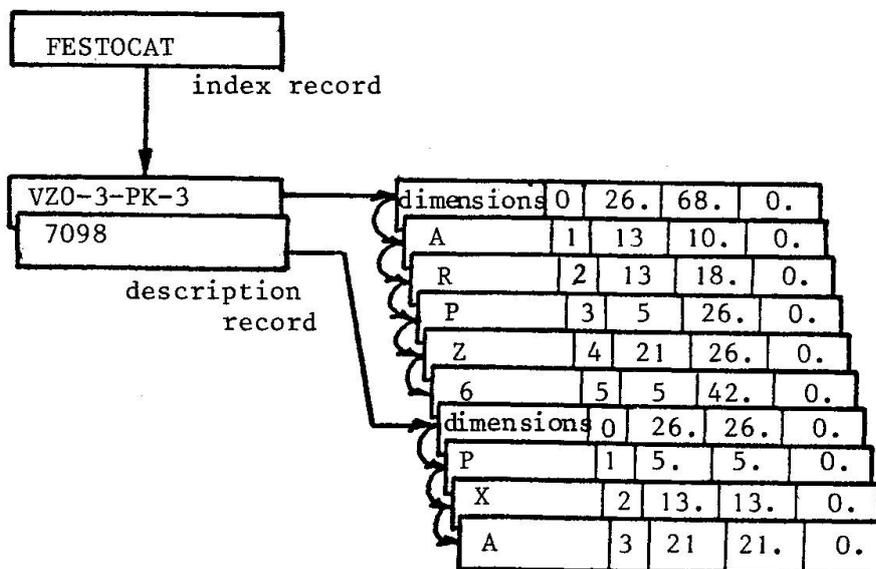
Fig. 12. Logic component data (example).

4.3.3. The parts list

It was decided that connections should be treated similar to connecting pipes in a chemical plant.
In the parts list however connections and components should be treated separately. Therefore the part with the description 'channel plate' is composed of two 'parts', the 'component list' and the 'connection list' as shown in Fig. 13.
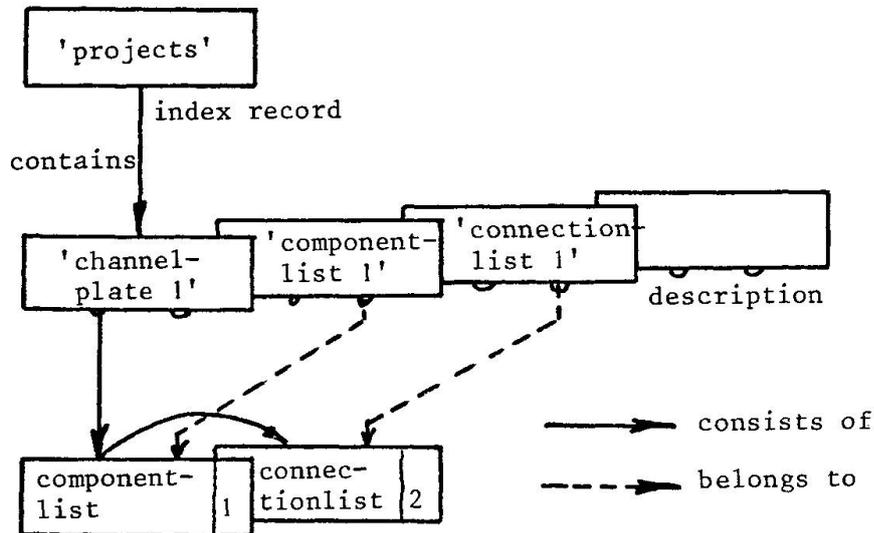

Fig. 13. Channel plate divided into component list and connection list.

Of course the description 'channel plate' is related to the index. The channel plate can be accessed by means of a direct entry on its descriptive name or by looking it up in the index under an entry called 'projects'. It has to be considered that the description records such as componentlist 1 and connection list 1 are strongly related to 'channelplate 1'.
Normally these descriptions will not be used in other channelplates. These records should allways be accessed via the record "channelplate'.

The logic components used in a specific channel plate are to be selected from the catalogue. An item record is created for them in which the part numbers are stored. The item records are connected with the description record 'component-list 1' and with the selected description record. This process is shown in Fig. 14.
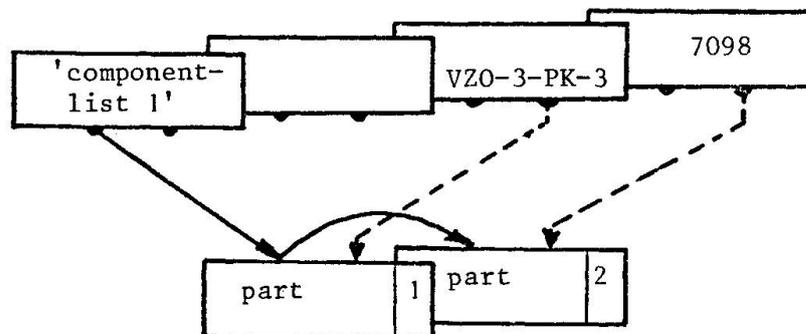

Fig. 14. Creating the componentslist.

During the generation of this part of the database it has to be checked whether
a component name given by the designer is really present in the component
catalogue.
The placing of the components on the channelplate is reached by the attachment
of an attribute record to each part containing the xy-coordinates of that
particular part.
For each channel connecting a number of pins a list is given of the pins that
are to be connected by that channel, hence a set of duplets (componentname,
pinname). Before a channel can be incorporated into the database a check is
made wether the connections are possible or not. The componentnames should be
present and they must have the specified pin. Then for each channel an item
record is stored. Next for each connected pin a connection record is stored and
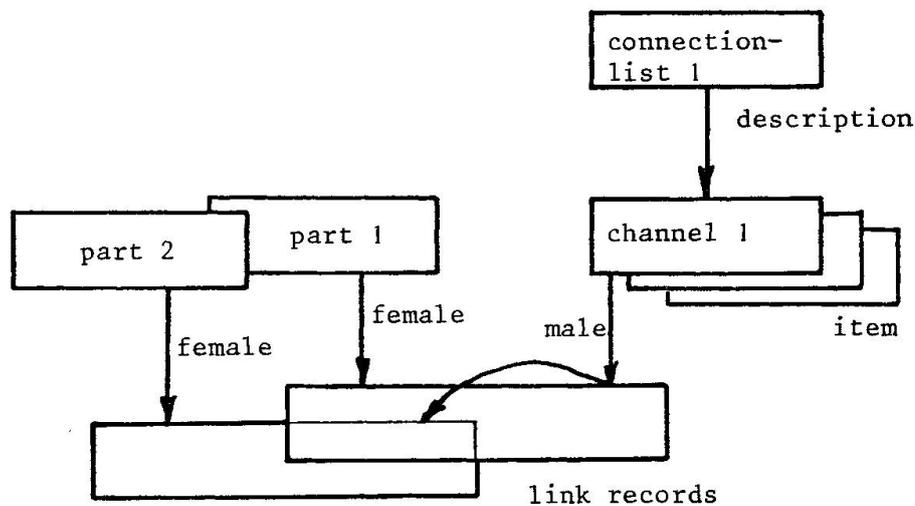linked to the indicated component (Fig. 15).

Fig. 15. Storage of connections.

In the connection record the following information is stored:

- a unique number
- the pen name
- the connection number.

Geometrical properties of the channel are stored in an attribute record which
is linked to the channel record.

This description does not include some very specific problems related to
channelplate problems.
These problems could be solved within the proposed conceptual scheme without
many difficulties.

4.4. The finite element problem

4.4.1. Finite element structure

The finite element method is a method to describe the mechanical behaviour of
engineering structures.
A structure is divided into a number of finite elements of a simple geometric
form. The mechanical properties of the individual elements are known.
Computer programs are available to calculate the response of the structure due
to the applied forces. In this context the finite element structure is defined
as the complete topological and geometrical description of the structure.

The topology describes the way the structure is divided into elements. This
results in a list of the used elements and their relation to a list of nodal
points. The geometrical description contains the actual values of the coordi-
nates of the nodal points.
The topology and the geometry are input for various calculations in the design
process. The storage of the finite element structure in a database has many
advantages. For example the database could be used for modification and control
of the configuration.

For large structures it could be necessary to divide the structure into a number
of different parts. A ship structure can be split up into different sections and
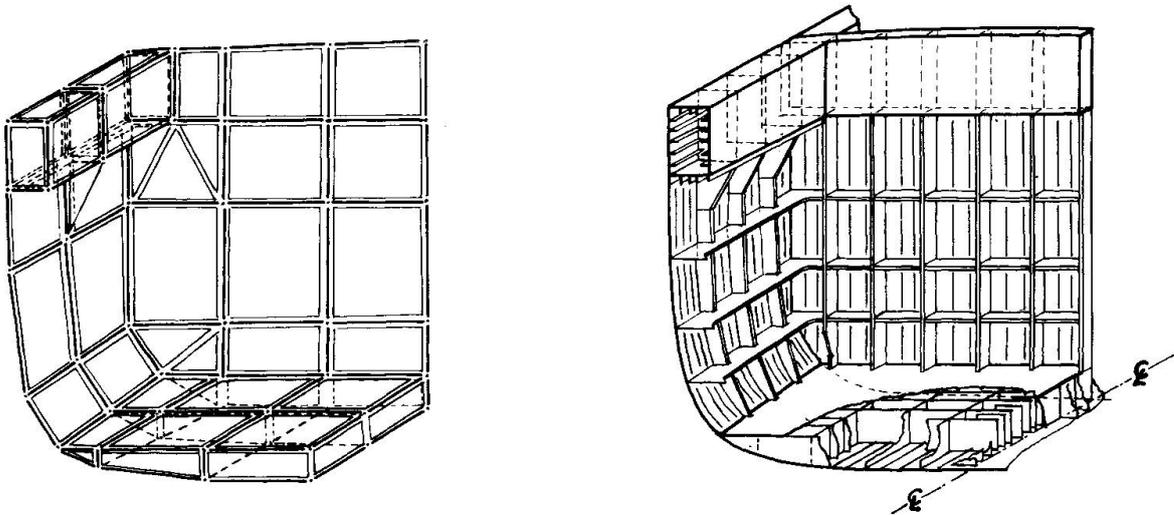each section is divided into elements. See Fig. 16.

Fig. 16. Ship structure.

In the following paragraphs it will be explained how the topological and
geometrical data can be stored in a database according the conceptual scheme.

4.4.2. Implementation of a ship structure into the database

First a description record with the ship's name is stored. Being a project this
record is linked to the index record 'projects'. The ship consists of two parts,
each being a seperate section of the ship. See Fig. 17.
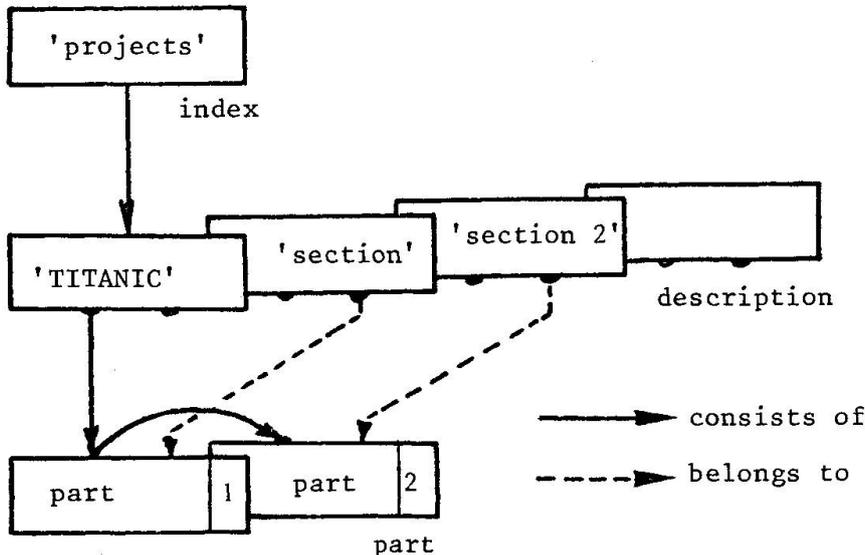

Fig. 17. Ship divided
into sections.

Next each section will be treated. To each section the elements and coordinates
are assigned in a way similar to the channel plate problem. This results in a
list of nodal points and an list of elements for each section. The coordinate
values are connected to the nodes as attribute records.
Each element record, containing the element number, is connected to the specific
nodes which are part of the element.
The connection is accomplished by means of connection records.
The assembling of two sections has not been implemented. But the conceptual
scheme does allow such an assembling.
In both problems it was needed to structure the parts belonging to a description.
The way it was implemented, by means of special lists, see Fig. 18, was really
cumbersome. Other solutions, requiring a redefinition of the conceptual scheme,
are also possible.
The connection of similar parts should be mutual, i.e. the connection from part A
to part B should be equivalant to the connection from part B to part A. But a
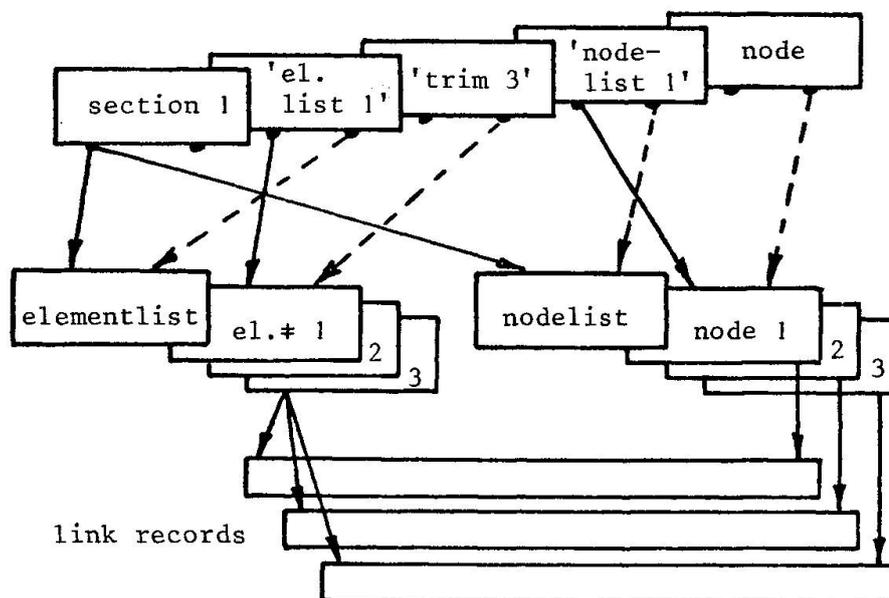good solution for this problem has not yet been found.

Fig. 18. Finite element division.

4.5. Implementation with DMS 1100

The channel plate problem and the ship finite element model were implemented
with the DMS 1100 DBMS installed at a Univac 1100 series computer installed at
the Dienst Informatieverwerking Rijkswaterstaat. DMS 1100 is a network type DBMS
supporting several host languages and a query language.
First the conceptual scheme was discussed and its final form described in detail.
As the conceptual scheme is the basis of the whole project, much attention was
paid to this part. With the help of a DMS 1100 specialist the physical implemen-
tation of the conceptual scheme was relatively simple. The implementation group
existed of four members including a DMS 1100 specialist.
The facilities like automatic rollback, back up etc. were found to be very
useful.
The performance of the system is difficult to measure. With a size of approximate
200 interconnected elements for each problem, a typical DMS 1100 retrieval of
all elements took a time of 5-10 seconds.
The more simple FORTRAN solution with a sequential and a direct access file took
about twice the amount of time.

It is felt that in many engineering applications this performance will be acceptable.

The whole implementation including the channel plate problem and the finite element model into DMS 1100 took about 200 hours.

Our experience with the implementation led to the following remarks:
- If the conceptual scheme is well chosen, there is no need for intermediate changes of the conceptual scheme.
- It is important to choose a conceptual scheme that anticipates to future complexity, because most network DBMS'es do not allow a dynamic change of the implemented scheme.
- The programmes for storage and retrieval are quite complicated. In a production environment it is recommended to develop standard programs.

To illustrate the last omission the programme past for the input of the channels will be discussed. It will be assumed that the individual elements have been checked and stored. It is of great importance that the database is always consistant and that it does not contain erroneous or incomplete information. First the database is coupled to the user programme. When the channelplate in question is found via the INDEX, an ITEM record 'channellist' is stored and automatically linked.

An unique description record 'chanlis 1' is stored as the head of the channels-list and a connection is made with the 'channellist', as follows:

```
FETCH     SYSTEM
FETCH     INDEX         'projects'
FETCH     DESCRIPTION   'channelplate'
STORE     ITEM          'channellist'
STORE     DESCRIPTION   'chanlis 1'
FETCH     ITEM          CURRENT, RETAIN
CONNECT   ITEM          SET = USES
```

Attention is asked for first the aspect of "navigating" through the database, and second the currency problem. For each record or link type there is always one specific record or link current in use. Most activities applied to the database change this currency. The CONNECT command above establishes a link between the current DESCRIPTION and ITEM records. Before the CONNECT command is issued the intended ITEM record is fetched, while retaining the current DESCRIPTION record. A single channel is then read as a set {(itemnumber, pinname)}. Before the channel is stored in the database it is checked whether the itemnumber appears in the partslist of the channelplate and whether that component has indeed a pinname as required. Furthermore it is to be checked that this pin has not been used before. Such a check does require some navigation through de database.

```
FETCH     DESCRIPTION   'elementslist'
FETCH     ITEM          itemnumber, if present then
FETCH     OWNER OF SET = USES  (establishes DESCRIPTION record)
FETCH     ATTRIBUTE     pinname, via set = omscattr; if pin exists then
FETCH     CONNECTION    pinname, via set = female. Not found means unused pin.
```

When all these conditions are satisfied, the data can be stored.

4.6. Implementation with ORACLE

ORACLE is a typical relational DBMS. At the time that this paper was written the ORACLE implementation was not finished yet. A lot of thinking work that had been done for the DMS-1100 implementation was also useful for the relational approach. The general scheme could easily be translated to a set of relations as required for ORACLE. This resulted in the following relations.

INDEXENTRIES { <u>ENTRY</u> },

DESCRIPTION { <u>DESCRIPTIONNAME</u> },

INDEX   { <u>ENTRY, DESCRIPTIONNAME</u> },

ITEM   { <u>DESCRIPTIONNAME 1</u>, DESCRIPTIONNAME 2, <u>ITEMNO</u> },

NODE   { <u>DESCRIPTIONNAME, NODENO</u>, NODETYPE, x, y, z },

CONNECTION   { <u>DESCRIPTIONNAME, NODENO 1, ITEMNO, NODENO 2</u> },

DESCRIPTION ATTRIBUTE   { <u>DESCRIPTIONNAME, ATTRNM, ATTRNO</u>, NVALUE, CVALUE }

ITEM ATTRIBUTE   { DESCRIPTIONNAME, ITEMNO, ATTRNO, NVALUE, CVALUE }.

The cursive attributes form a key for the tuples. It is noticed that the system
of defining connections between items has been changed with respect to the
original approach. For an assembly a list of nodes is defined to which nodes,
internal to the items, are attached by means of the connection relation. For this
reason the connection relation has two attributes NODENO, the first node
indicating the global node number, the second the local node number of the item
to be connected.
As a result of our first experience the preliminary conclusion can be drawn that
unless special precautions are made it is difficult to maintain the integrety of
the database.

4.7. Implementation with the ELPRO system

ELPRO is an acronym for <u>E</u>ngineering- and <u>L</u>ist <u>PRO</u>cessing system. ELPRO was
developed by the Dutch HBG Company based on Becqué's thesis [*]. ELPRO is
supported by P & P Engineering Consultants B.V., Rotterdam.
Characteristics:

- System is written in FORTRAN with user callable subroutines.

- System and user data are stored in tables with a row and column structure.
  Information is stored and retrieved directly from the tables by user
  written programmes.

- ELPRO supports its own input and definition language for automatic
  generation of data.

- ELPRO is an open system in the sense that the user can define his own input
  and command language.

- ELPRO has a possibility to define relations between data and use these for
  storage and retrieval (Relational approach).

- The command language allows the use of user defined procedures including
  possibilities for jumps and loops.

- ELPRO has its own report writer, a query language for system maintenance
  and a logging facility.

- ELPRO is multi user and can be implemented on small computers.

Implementation of the finite element structure in ELPRO

In ELPRO each data item has to be collected and stored in twodimensional tables.
To avoid fixed pointers every table often has to contain extra data. For the
ship structure, a language was defined to read directly the datacards produced
by the user. This resulted in the tables as shown in fig. 19.

---

[*] Becqué: LPR, an Integrated program and Data Management System for
        Engineering Applications (Thesis Delft, 1977).

ELEMENTS					
el.#	node 1	node 2	node 3	section #	shipname

NODES			
node #	rel.x	rel.y	rel.z

SECTIONS			
section #	abs.x	abs.y	abs.z

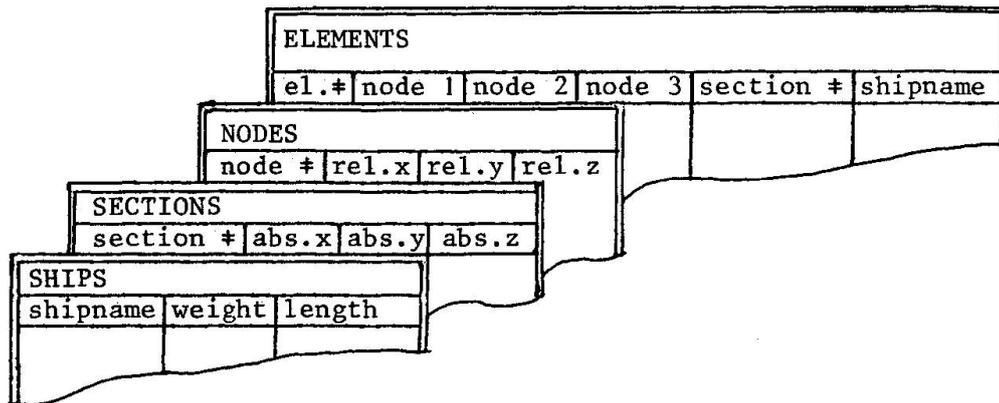| SHIPS		
shipname	weight	length

Fig. 19. Tables and relations for ship structure.

The implementation takes four steps.

- Table definition

Every item is given a name, sequence number, type, default value and a range. Tables are defined using the ELPRO definition language. No user programmes are necessary at this stage. The user uses his own problem oriented language. Data generation tools are available.

- Data entry

The tables are filled using the ELPRO input-programme. Each input record will be checked according to the table definition. At this stage the reportwriter is used to print the contents of the tables for checking purposes. Some changes in the definition or in the contents of the tables can be made.

- User programmes

With user written FORTRAN programmes communication with the database is possible via ELPRO Library subroutines.

- Running the test programme

The user can activate his programme by calling the module-command in the command table. The results are stored in work tables for examination or they are printed directly. Before running modifications can be made.

Implementation of the channelplate problem in ELPRO

In the same way as it was done with the ship, tables were defined and filled. Some minor changes of the data were necessary before they could be read, checked and   stored in the tables.

ELEMENT CATALOGUE

element			pins			
el.name	b	h	pname	p ≠	x	y
–	–	–	–	–	–	–
				–	–	–
–	–	–				

PLACEMENTS

DIMENSIONS			ELEMENTS			
L	B	n	el ≠	x	y	φ
–	–	–	–	–	–	–
			–	–	–	–
–						
–						
–	–		–			
			–			
			–	–	–	–

CONNECTIONS

dimensions			element/pins		
B	D	MA	el ≠	pname	p≠
–	–	–	–	–	–
			–	–	–
–	–	–	–	–	–
			–	–	–

ELEMENTLIST

el ≠	el.name
–	–

Fig. 20. Tables and relations for channel plate.

As shown in the figure 3 tables had to be divided into two parts. In the first part some general information is stored (overall dimensions) while in the second part more detailed information is given. The two parts have different definition rules (subtables). These division was necessary to read the user delivered information rows in an easy way. Other solutions are also possible. The implementation of the applications in ELPRO is much more simple compared with a CODASYL-type database. ELPRO was developed for an engineering environment. Therefore a number of facilities such as crash protection and integrety are less sophisticated. Execution times of ELPRO are better than with the CODASYL-type database.

5. CONCLUSION

Due to the fact that the project has not been finished yet it was not possible to give an integral result. A final report is expected to be available at the end of 1982 from the CIAD institute.

6. ACKNOWLEDGEMENTS

Leere Seite

Blank page

Page vide

Computer Aided Structural Design in Hungary

Conception des structures assistée par ordinateur en Hongrie

Gegenwärtige Lage der rechnerunterstützten Projektierung in Ungarn

Peter LENGYEL
Head of Department
Videoton Fejlesztési Intézet
Budapest, Hungary

Peter Lengyel, born 1953, obtained his structural engineering degree at the Technical University Budapest and his applied mathematics degree and doctorate at Loránd Eötvös University of Sciences, Budapest. Managing now a departement for CAD programming he has been involved in research in numerical methods in FEM analysis.

SUMMARY

The paper introduces the basic features of Computer Aided Structural Design (CASD) from the computer manufacturer's point of view, and characterizes the state-of-the-art in Hungary. The basic tendencies of CASD are summed up on the basis of statistical comparisons of computer applications in Hungary, and reflect the state of international trends towards fully automated structural design.

RESUME

Cet article présente les caractéristiques principales de la conception des structures assistée par ordinateur (CSAO) du point de vue de constructeurs d'ordinateurs et décrit la situation actuelle dans ce domaine en Hongrie. Les tendances fondamentales de l'introduction de la CSAO sont exprimées sur la base de comparaisons statistiques des dépenses effectuées pour l'informatique en Hongrie. Parallèlement, une évaluation des effort entrepris sur le plan international en vue d'une automatisation totale de la conception des structures est réalisée.

ZUSAMMENFASSUNG

Diese Arbeit präsentiert die wichtigsten Eigenschaften der rechnerunterstützten Projektierung vom Standpunkt des Rechner-Herstellers und beschreibt die gegenwärtige Lage dieses Fachgebiets in Ungarn. Die grundlegenden Tendenzen des CAD werden mit Hilfe von statistischen Untersuchungen über den EDV-Aufwand in Ungarn zusammengefasst, die wohl auch die internationalen Bestrebungen in Richtung von voll automatisierter Projektierung charakterisieren.

1. INTRODUCTION

The VIDEOTON computer factory is the largest manufacturer in Hungary, and is specialized in producing minicomputers. Since this category is generally accepted as the most suitable computer for CASD - as far as capacity and costs are concerned - VIDEOTON has been laying great emphasis on its technical-scientific software development. A structural engineering program package has been developed, which is widely used by the Hungarian and foreign users. The program library is under steady development and extension. Each of the programs has been worked out on basis of real user's wish, and thus these are frequently used by design offices. This is the way, VIDEOTON has wide ranging connections with structural engineering research and design institutions in Hungary. The aim of the present paper is to summarize the experiences obtained through this cooperation.

2. THE DESIGN OFFICES AND THEIR CONNECTION TO CASD

First of all some words about the design offices involved in structural engineering. There are structural design offices in Hungary being specialized in different fields, such as industrial structural design, design of buildings' structures, structural design of bridges, or that of water engineering, etc. The largest offices are to be found in Budapest, however the other major cities have outstanding design offices too. In Hungary the small buildings for housing families /family houses/ are mostly built on basis of typified designs offering a large choice, and being rather cheap. Therefore the general structural design problem belongs to one major individual task given to the design office in question /e.g.: design of a hospital, design of a hotel, etc./. There are of course typified designs here too, but these are - from the structural automation's point of view - out of interest. Thus the design office has always a certain given problem to solve, and is keen on having a computer for its solution. All these explain, why some of the structural design offices have a computer centre with a large staff for the solution of their momentary tasks. There are others, who prefer to buy and rent programs or have them made by university departments or software houses. The choice depends on how often occurs the given problem in the practice of the design office.

These are shortly the demands to be solved on one hand, and let us now see the other side, i.e. the offer of computer manufacturers. The offer is today extremely wide, and went through the process of a long development. This development can be best characterized by some data published by the Hungarian Central Statistical Office, /1./.

Firstly the following diagram shows the increase of the number of employees in the field of computer application and development in Hungary. (Fig.1)
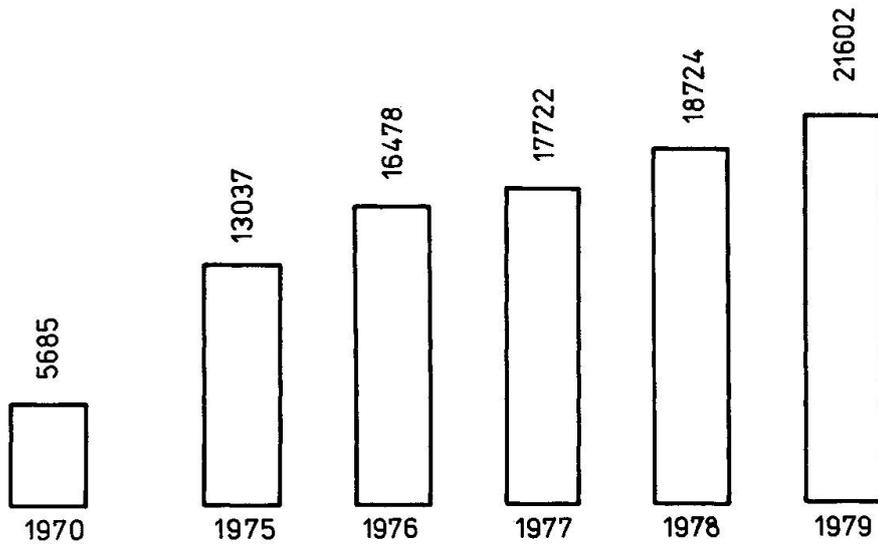
Fig.1 Number of employees in the field of computer appli-
      cation in Hungary

The next figure shows the distribution of computer application
in Hungary. The percentage ratio of technical scientific and
technical structural calculations have decreasing tendencies,
however the change of absolute values is about six times for tech-
nical scientific and more, than two times for technical struc-
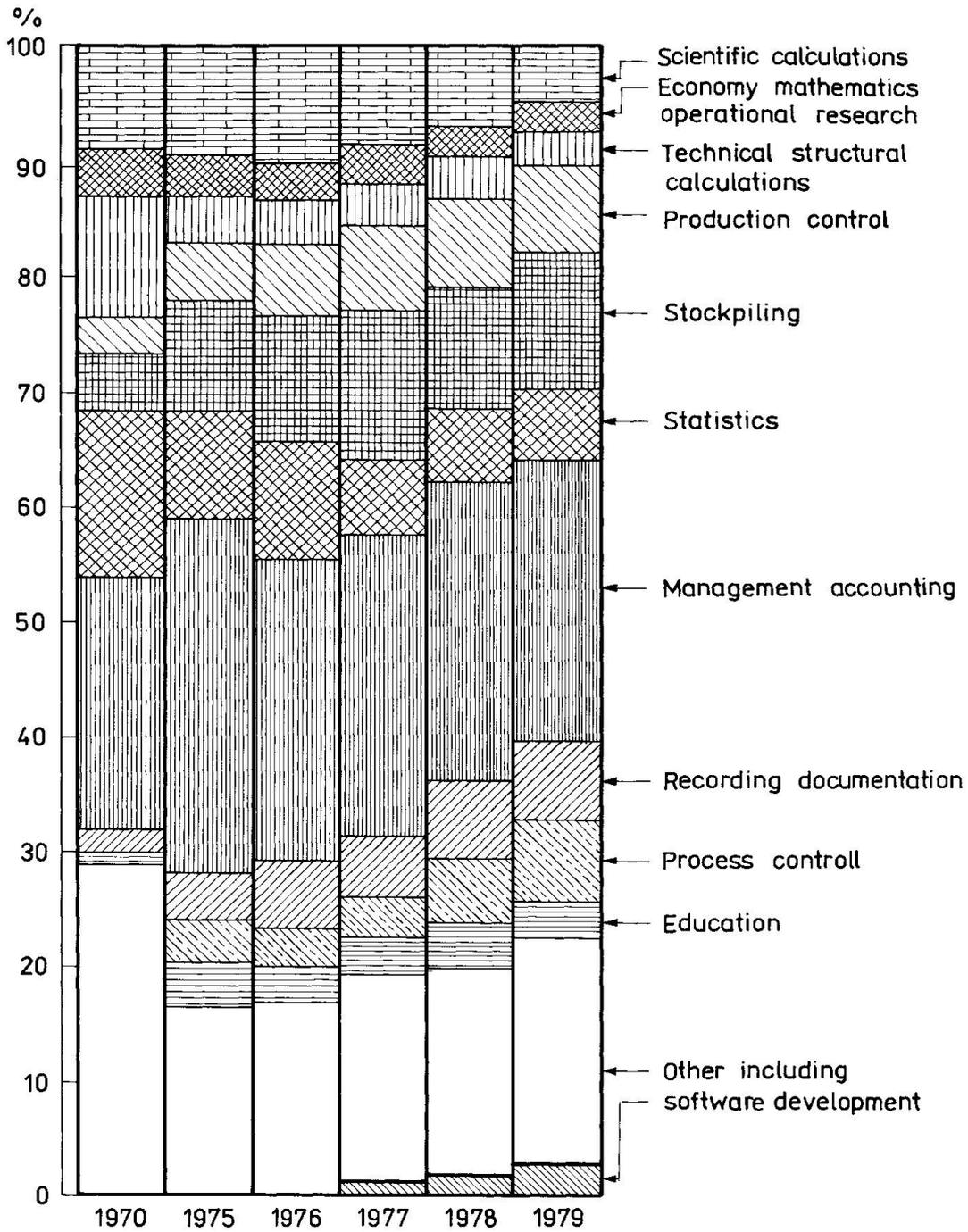tural calculations. (Fig.2)

Fig.2 Distribution of computer hours in different fields of application

Let us now consider the change of the total number of used computer hours between 1970 - 1979: (Fig.3)
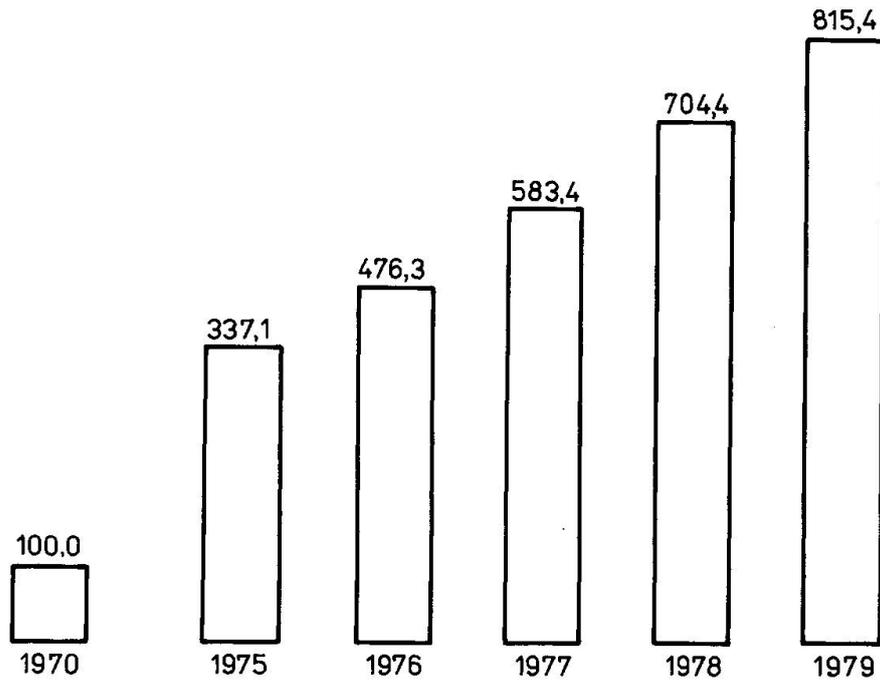
Fig.3 Total number of used computer hours

If we have a look at the utilization of computer hours the following diagram can be obtained. (Fig.4)
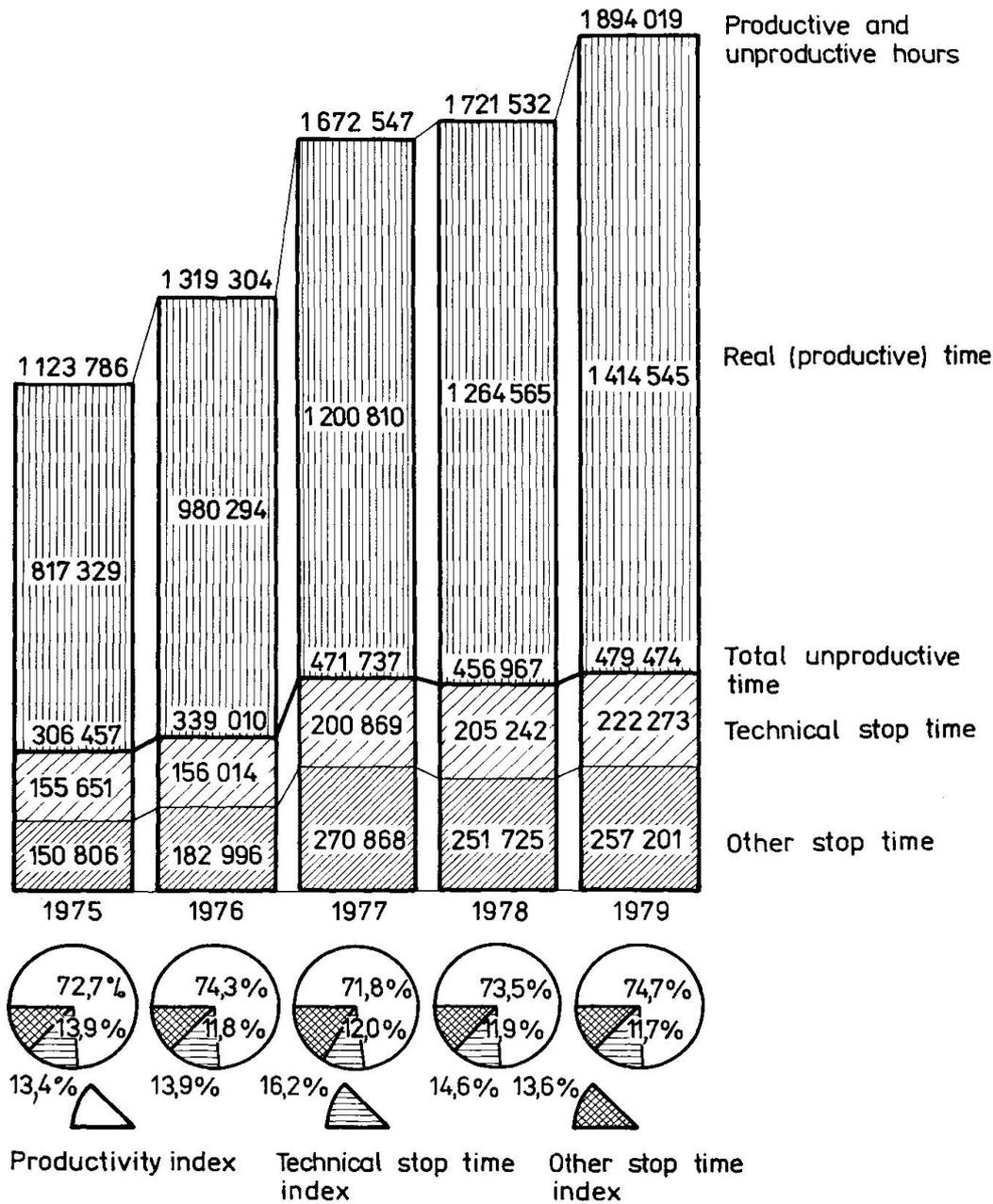
Fig.4 Utilization of computer hours

3. TENDENCIES IN CASD

The rapid change in computer application can be seen if the research and design activity in building industry is considered, which includes CASD too. Here in 1979 alone there were 94617 computer hours utilized in Hungary. Thus the increasing use of computers underlines the importance of the question, which is the proper approach to the economic utilization of these? And this question belongs to the often heard statement: the greatest obstacle against the overall use of CASD has been the lack of proper hardware in the 60's, the lack of proper software in the 70's and the lack of knowledge for proper application in the 80's.

Let us first summarize the practice of application of the Hunga-

rian design offices. In the mid 60's there were only a few companies having their own mini or mainframe computer. Others joined the trend of CASD only some years later, and started from the other end of the line, i.e. introducing pocket and later desktop calculators. The third alternative has been the case, when problems requiring larger central core capacity were solved by computer centres, however the routine tasks of smaller demands were run on desk tops located at the practising engineer.

The picture is even today very colourful. However, efforts have been done for the general introduction of multi-user systems, where the engineer has a terminal being suitable for standalone useage, and for real terminal function to a mega-mini or main-frame computer too. The definite lines having been dividing the above trends some years ago, are to disappear as the development of smaller - as far as sizes are concerned, however bigger as far as capacity is concerned -, and cheaper CPU-s and background store units is internationally accelerating. It is worth mentioning, that the background store techniques have had for a long period /roughly between 1967 - 1979/ a relatively steady level compared to the development of CPU-s. This situation has changed mostly due to the introduction of Winchester disks, and this will have stimulating effects on the new trends of CASD.

It can be stated, that the today's office of structural engineering has still more alternatives to choose from, and the size of the office determines which hardware choice is the optimal one. However, tomorrow even the big offices will have many display units, functioning as local desktops or forming networks. These technical possibilities are available at present, however they have not yet been generally introduced into structural engineering practice.

Another important aspect of CASD is the graphical input and output. The situation is here also similar to the previous one, i.e. although the technical opportunities are given, today we cannot speak about a general useage of graphical displays, that is about a total graphical interaction in the field of CASD. However, as far as graphical output is concerned the application of plotters is a generally existing commonplace of CASD.

This is true for CASD in Hungary, too. Here there is a great shortage of auxiliary man-power in the structural design work, and so of draftsmen. Therefore every CASD user has some sort of plotter, being almost steadily used.

The lack of graphical displays in the design offices can be best explained by the level of technical development and by the cost. This is proven by the fact that IBM presented its first graphical display /IBM 2250/ in 1965, but waited 12 years until the next type of graphical displays has been manufactured, because of high costs and lack of proper software. Now, as the manufacturers try to cover the field of CAD with hardware and software products as well /let us only think of the UNIS-CAD System of Sperry Univac having been first presented at the Hanover Fair in April 1982/ we can count with a much easier and cheaper availability of graphical displays. These devices are very important, because without them we will always be able to speak only about computer aided design, but not of a design automated by computer. It can be stated, that if a total automation in CASD - even including the unavoidable human interaction, however computer aided and not manual interaction - is achieved, there will be a much greater

and general enthusiasm towards CASD, than today. This total
automation is very difficult to get at, mainly because of the
points of view of software and application. /One might think for
exampleof the problems having been experienced during the useage
of large program systems./

4. CONCLUSIONS

The present paper has mentioned some international effort, trends
and their link with CASD in Hungary. As there is a large amount
of man-power concentrated in computer application in Hungary -as
shown in the previous figures -, and as all of the design offices
have some sort of experiences with CASD, the general trend, i.e.
the strengthening of CASD can be felt in Hungary, too. This means
that the number of multi-user systems is increasing, and the pop-
ularity of micros is also ever greater. On the other hand there
is a very strong software development capacity, which provides
structural programs for general aims, but for the solution of
individual tasks too. All these form the basis of the general
useage of graphical devices, which mean the first step toward the
development of

                CASD ———— CASD

i.e. Computer Automated Structural Design out of Computer Aided
Structural Design. The speed of this development is very hard to
predict in advance, however the speed of another development
might already been demonstrated in the field of computer appli-
cation. The number of used punch-cards, and printer paper sheets
in Hungary has shown the following change during the period of
1960-1979. Figure 5 gives the impression, that concerning the
duration of tendencies in the field of computer application one
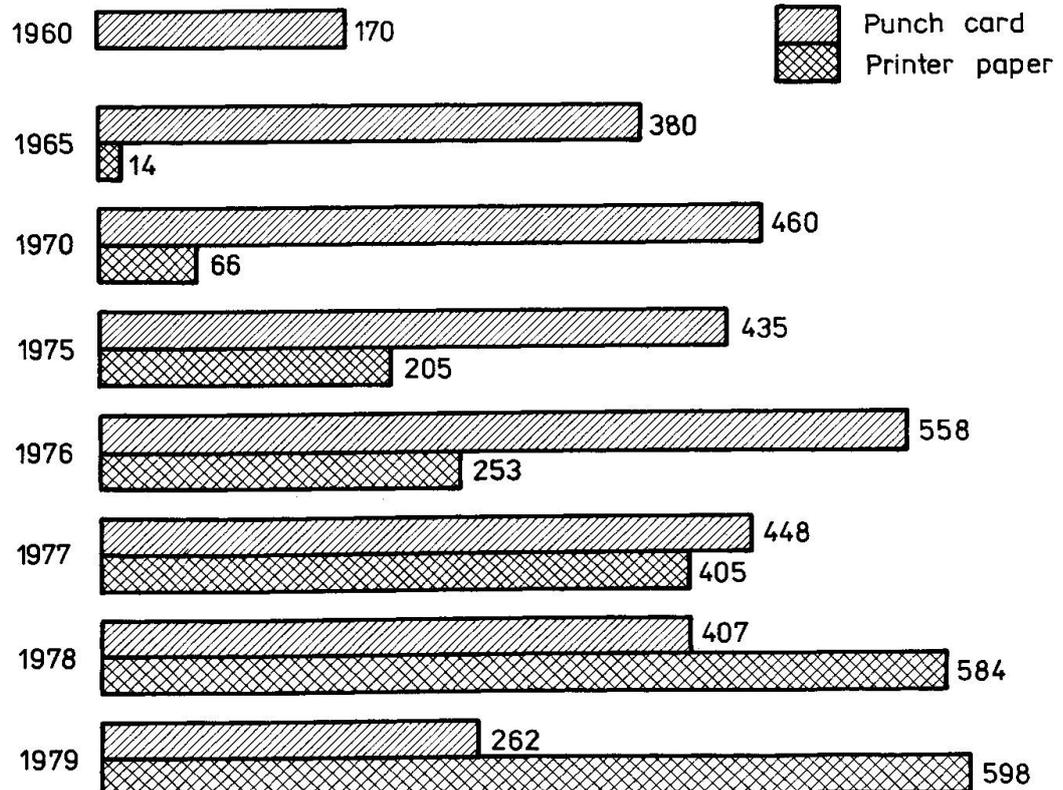can never go for sure. Pieces in millions

Fig.5 Number of used punch-cards and printer papers

REFERENCE

1. Statistical Yearbook of Computer Application 1980. Statis-
   tical Publishing Company, Budapest, 1981. /In Hungarian/

Leere Seite

Blank page

Page vide

Adaptive Techniques of Finite Element Computations

Techniques adaptatives appliquées au calcul par éléments finis

Einwirkung anpassfähiger Verfahren zu Finite-Elemente-Berechnungen

Alberto PEANO
Dr. Sc. Civ. Eng.
ISMES
Bergamo, Italy

Alberto Peano, born in 1946, received his doctoral degree at Washington University, St. Louis, Missouri. He has worked in the industry as a stress analyst as well as in the Polytechnic University of Milano as an assistant in the Structural Engineering Dept. Alberto Peano is head of the Mathematical Modelling Division of ISMES.

SUMMARY

Adaptivity is the capability to increase, either automatically, or with minimal user interaction, the number of degrees of freedom in regions where an error tolerance has been exceeded. The availability of adaptive computer programs will drastically simplify preprocessing of finite element analyses and is particulary useful for three-dimensional application. A benchmark problem is also presented.

RESUME

La notion d'adaptatif s'applique à la possibilité d'ajuster, automatiquement ou sous l'effet d'une intervention minimale de l'utilisateur, le nombre de degrés de libertés à la précision souhaitée dans les zones où les tolérances ne sont pas observées. La disponibilité de tels programmes apportera une simplification considérable au niveau des pré- et post-processeurs, notamment pour les applications tridimensionnelles. Un exemple est exposé à titre de comparaison.

ZUSAMMENFASSUNG

Mit Anpassbarkeit meint man die Fähigkeit, automatisch oder mit geringster Benutzereinmischung die Zahl der Freiheitsgrade dort zu erhöhen, wo eine vorgeschriebene Fehlertoleranz überschritten wird. Die Verfügbarkeit anpassfähiger Computerprogramme wird die Vor- und Nachlaufarbeit von Finite-Elemente-Analysen sehr erleichtern und kann mit Vorteil bei dreidimensionalen Berechnungen angewandt werden. Ein Benchmark-Test wird dargestellt.

1. INTRODUCTION

Analysis of solid continua by current general purpose programs is a very
time consuming and expensive task. Unless geometry is favourable to automated
mesh generation, preparation of the model may turn out to be a long and
tedious chore. Of course the use of advanced graphic capabilities is very
helpful. Since estimates of time and cost required for the preparation
of a solid model are sometimes excessively optimistic, perspective users
may wonder whether results will be available in due time and may select
alternative simplified models, if applicable (2D continuum or thick shell).

Moreover the results of expensive 3D analyses are sometimes of little use
because of the difficulty of interpreting and gaining confidence in the
computed stress values. Two apposite extreme situations may occur. On one
hand the user may be disappointed by severe stress discontinuities occurring
at element interfaces of coarse meshes. On the other hand the user may be
overwhelmed by the huge amount of output data.

Since the cost of data processing has decreased considerably in recent years,
it is possible to envisage a time when solid finite element models will
be as usual as 2D models have been in the seventies. The previous discussion
indicates that before this time comes, two conditions have to be met: model
preparation must be fast and simple, result interpretation must be as much
automated as possible.

It is well recognized that sophisticated graphic capabilities are essential
for an efficient solution of both problems. Graphics alone, however, cannot
achieve the ultimate goal of making 3D analysis as simple as 2D analysis
is today. Refined 3D meshes are difficult to scrutinize, if at all, on a
graphic terminal. A major revolution of the basic finite element techniques
is needed and it is effectively under way. Two aspects are particularly
relevant to this discussion, namely p-convergence and adaptivity.

P-convergence is a new efficient convergence process that has been investigated
in theoretical studies, tested in numerical experimentations and applied
to the solution of large application problems. The distinguishing feature
of this new convergence process is that the number and distribution of finite
elements are fixed, and the number of interpolating (basis) functions, which
are complete polynomials of order p, is progressively increased over each
element. On the other hand, in the conventional convergence process, called
h-convergence, the number and type of basis functions are fixed, and the
finite element mesh is refined in such a way that the maximum diameter of
the elements, h, approaches zero.

It has been found that in general the rate of p-convergence is faster than
the rate of h-convergence (unless optimally graded sequences of mesh refine-
ments are used). The fast convergence rates provided by the p-version of
the finite elment method may obviously lead to considerable savings of computer
time. A different point is stressed here, however: the p-version may
lead to considerable savings of manpower because very coarse meshes may
be used. Coarse meshes are important both in data preparation and during
assessment of the accuracy of computed results. The main reason is that
computer graphics is simplified tremedously by the use of coarse meshes.

For istance displaying results on a plane arbitrarily cutting through the solid has a cost (and a response time) which increases very rapidly with the number of elements of the mesh. Moreover coarse meshes lead naturally to the "hidden variables" concept, which, as discussed later in the paper, is essential in order to keep the computer print-out to a manageable size.

The second fundamental aspect, mentioned earlier, is <u>adaptivity</u>. A finite element software system is said to be adaptive when it possesses some local a-posteriori error estimation capability and a capability to increase, either automatically, or with minimal user interaction, the number of degrees of freedom in regions where an error tolerance has been exceeded. Adaptiveness aims to minimizing the role of the so called "engineering intuition" during post processing of finite element solutions.

Adaptiveness based on h-convergence has been examined by Babuska and Rheinboldt (1). Adaptiveness base on p-convergence has been studied both by B. Szabo (2) and by the author (3, 4). Completely automated adaptive finite element analysis of 2D continua using p-convergence has been tested at ISMES since 1977 (5). The development of a similar capability for 3D analysis has been delayed by the need of performing extensive research and of implementing a strong 3D software system based on the p-version. This software has now been released for commercial use, under the trademark FIESTA.

The theory used to develop FIESTA has been presented earlier (6). In this paper the capabilities and some software design aspects of FIESTA are discussed. The paper also shows results of a benchmark test which was solved both using FIESTA and NASTRAN.
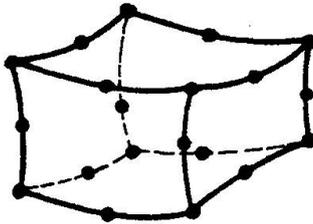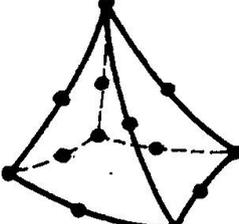
2.   MODELING CAPABILITIES OF FIESTA

The distinguishing feature of FIESTA is the capability of grading the degree of polynomial interpolation over the mesh. User's are allowed to selectively increase the order of approximation over one or more elements without any modification of the stored input data. The maximum degree of interpolation is a complete quartic polynomial.

TABLE I

Solid finite elements available in COMET-3D

| TETRAHEDRON | PENTAHEDRON |
| HEXAHEDRON | PYRAMID |

The first release of FIESTA may be applied to the static stress analysis of three-dimensional elastic continua. Isotropic, transversally isotropic and generally anisotropic material properties are accounted for. The element library is comprised of hexahedra, pentahedra, tetrahedra and pyramids (Table I).
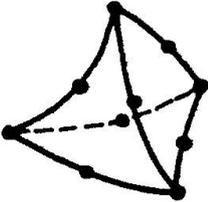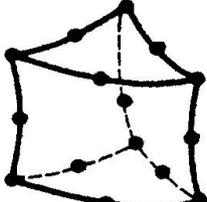
The large variety of element shapes available is very useful for modeling complex details and for mesh grading. All elements have a variable number of nodes.

A large variety of load types may be handled by the program: forces, pressures (e.g. hydrostatic load), general traction load, gravity loading, thermal strains, initial stresses. Any number of load cases and load combinations may be considered.

FIESTA incorporates strong graphic capabilities as well. The mesh and the deflected mesh may be displayed (hidden line option, available). Contours of various displacement or stress components may be plotted on any user' specified surface or segment.

3.   THE "HIDDEN VARIABLES" SOFTWARE DESIGN CONCEPT

A characteristic feature of conventional finite element techniques is the simple physical meaning of the unknowns (nodal variables) used for parametric modeling of the structural behaviour. For instance the unknowns used for stress analysis of continua are generally displacements of the same nodal points used for description of the initial geometry. Hence it is quite natural for program developers to provide printed output at all the nodal points used in the model. This feature is very practical and user oriented until the number of degrees of freedom is limited. For refined meshes and large scale analysis problems the situation is completely different:

a)   Printed information is redundant because very close displacement or stress values are printed at adjacent nodes.

b)   Printed information is excessive. The few meaningful values are not easily accessible in a print-out full of unnecessary information.

c)   The conventional choice of the problem unknowns is a well known cause of ill-conditioning in the presence of very refined meshes.

The development of adaptive programs based on h-convergence would actually worsen the problem, as many new undesired nodes are added at locations which are not meaningful for the analyst. In practical applications a compact print-out is always very useful for getting an overall impression of the solution. This is particularly important for solid continua because graphics will exhibit results only on surfaces or sections, which are best chosen after a quick look to the overall solution. For this purpose the analyst would like to have printed output only at a limited number of nodes.

This task is naturally achieved by the p-version of the finite element method. Here very coarse discretizations may be used. In FIESTA the geometric description of the solid is achieved by using standard isoparametric elements, such as hexahedra or pentahedra with straight or parabolic edges. The vertex and midside nodes of this mesh are limited in number and are distributed all over the structure including any region meaningful to the stress analyst.

The natural choice is to provide printed output at these nodes only, irrespective of the selected order of polynomial approximation. The actual degrees of freedom of FIESTA are the vertex node displacements and the amplitudes

of hierarchic higher order deformation modes added in order to reach the user specified degree of polynomial approximation. Typical shape functions used by FIESTA are shown in Figure 1. All these additional degrees of freedom are hidden to the user in order to contain the amount of printed information and to simplify the comparison of results computed for different levels of approximation.

Of course results may be printed by the program, upon user request, at any point arbitrarily located in the structure. Moreover FIESTA has the capability of intersecting the solid with an arbitrarily fine 2D grid, to be used for graphic output (contours of any displacement, strain or stress component). Of course the numerical values at the nodes of the auxiliary grid may be printed out.

In conclusion the amount of numerical data printed by FIESTA may be as limited or as large as desired and it is completely independent from the number of elements and of degrees of freedom used to model the problem.

P = 1   P = 2

P = 3   P = 4

P = 5   P = 6

Fig. 1 - Typical shape functions used by COMET for various levels of interpolation.

4.   QUALITY ASSURANCE AND INTERACTIVE MODE PROCESSING

FIESTA has been designed as a system of independent processors which operate on a common data structure. Each processor has a specific function, easily recognized by the engineer-analyst. A list of the most important processor is given in Table II. Additional processors and utilities are available for checking the data and for plotting graphs, contours and the deflected grid.

The processor structure is very important for reasons of maintenability and verificability of software. Moreover considerable advantages are accrued in case of interactive and distributed processing. Infact the various processors may be executed during different jobs. This allows several individuals to work independently during data preparation or output of graphic results in order to meet pressing deadlines. Moreover the most CPU consuming tasks may be processed by the most powerful computers of the net (in case of distributed processing) or may be scheduled during night shifts (particularly when working on large minicomputers such as VAX-11/780). An additional advantage is that printed output is generated in steps according to the user needs. For istance nodal stresses may be output after checking displacements or even after plotting stress contours on appropriate sections.
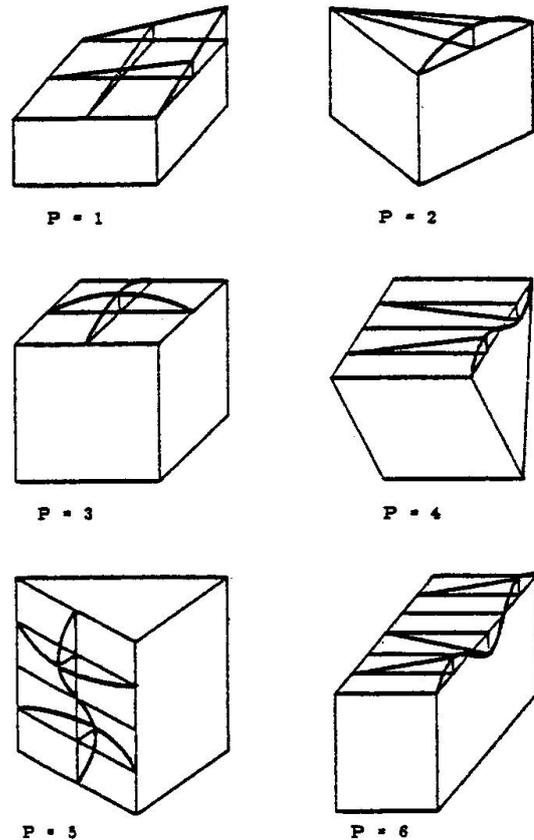
TABLE II

Typical processor functions in FIESTA

PROCESSOR	FUNCTION
TOP	Initialization and input of the grid
PROP	Input of a library of material properties
CONST	Input of Constraints
LOADS	Input of Loading data
PLEVEL	Input of element order of interpolation
LOVE	Computation of the Load Vectors
LCOMB	Input of load combinations
ARRAY	Computation of arrays needed for element matrices
STIFF	Computation of element stiffness matrices
STATIC	Matrix factorization using the frontal technique
SOLVE	Solution of the system of linear equations
DISP	Output of nodal displacements and reactions
STRESS	Output of nodal stresses
FDATA	Output on segments intersecting the grid
CMESH	Generates auxiliary 2D meshes for contour plotting
CDATA	Output at nodes of auxiliary 2D mesh

The order of execution of processors is very flexible provided a meaningful sequence is followed. In case of reanalysis, processor functions, which are not influenced by the modification of the input data, don't have to be executed again. For instance the solution of a new load case does not require triangularization of the stiffness matrix, or a change in the constraints does not imply that the element stiffness matrices have to be computed again.

A reason of concern is that the exceptional flexibility of FIESTA may ease misuse. Infact in case of subsequent modifications (e.g. new material properties) or new load cases the data sets generated by the project may become not only numerous but ambiguous as well. For instance identical files containing different version of the triagulated stiffness matrix may be available in the application data base. In order to eliminate any reason of concern, which could be very serious for users working in a quality assurance environement, an innovative book-keeping system has been built into the program. All runs of a specific project are assigned their ordinal number which identifies the computer print-out as well. The same number as well as additional information (passwords, problems titles, etc.) is used to label the data sets. In this way the program may keep a record of previous execution and issue a Project Status Bulletin at the beginning and at the end of each Computer Report. Therefore the history of any FIESTA output data is fully traceable.

5.   A BENCHMARK PROBLEM

The problem was a gear case that was cracking in an area near the gear case mounting bolts. The purpose of finite element analysis   of existing

design was to obtain a model that predicted the existing failure zones, then this model could be modified to evaluate proposed design modifications prior to manufacturing them. This study was conducted at MCAUTO (McDonnell Douglas Automation) using standard general purpose programs. Later part of it was duplicated using FIESTA for comparison purposes. A plate finite element model was made of the gear case, gear-bearing support structure and support structure. The plate elements did not model the complex gear case geometry in the area of failure. This area contained reinforcing bosses and gussets. Therefore, a local refined model using solid finite elements was required to obtain the local stresses. The loading for this refined model would be the boundary displacements obtained from the plate finite element model analysis.

5.1 The NASTRAN solution

The 8 node CHEXA solid finite elements and the 6 node CPENTA solid wedge elments from MSC/NASTRAN were used for the refined model.
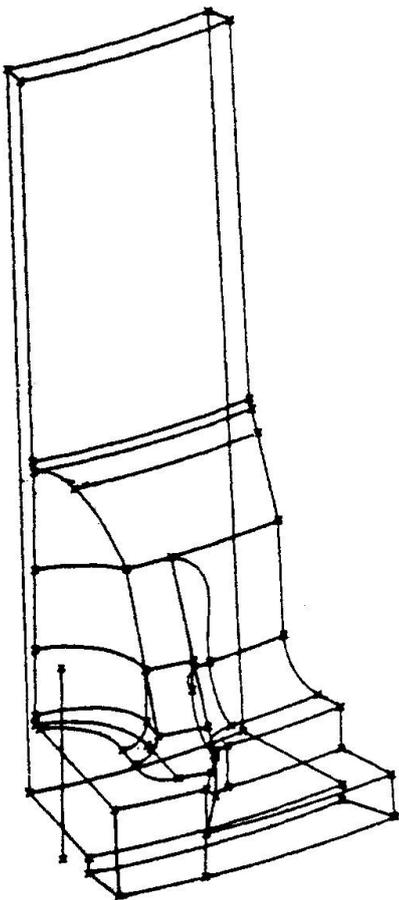

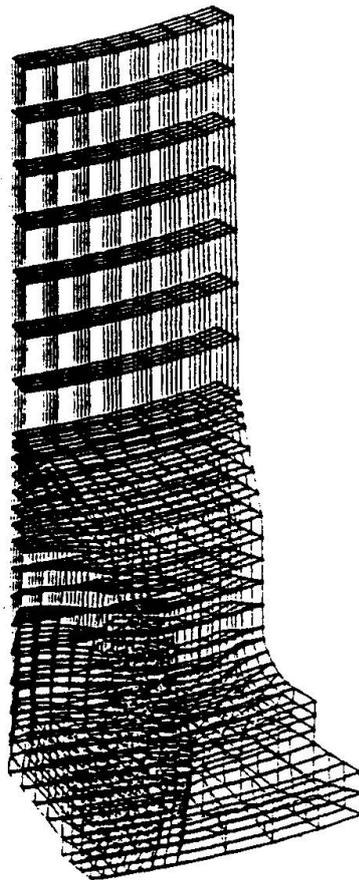
Fig. 2a - CADD model of the gear-case detail.
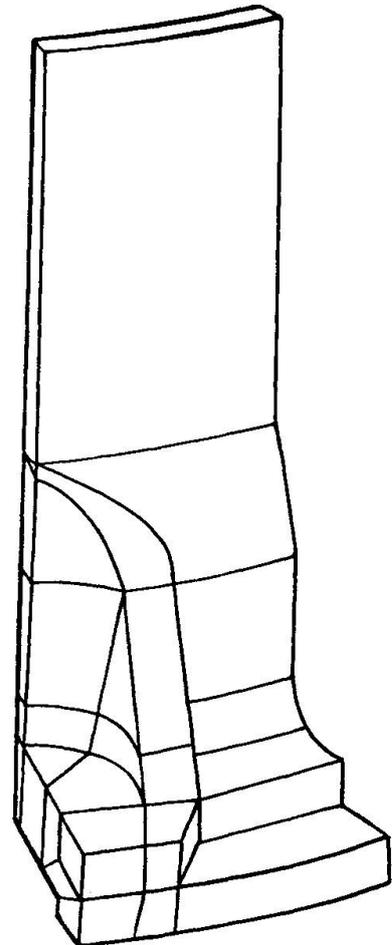
Fig. 2b - NASTRAN model.

Fig. 2c - FIESTA model.

The geometry of the gear case is rather complex. The outer surfaces of the refined model which consist of a portion of an 18° segment of the gear case was modeled using MCAUTO's Computer Aided Design Drafting System (CADD) and is shown in Figure 2a. The gusset and bosses make this a very complex shape. These geometric data was then passed to MCAUTO's FASTDRAW/3system which is an interactive graphycs finite element modeling system. FASTDRAW was used to divide this geometric model into ten volumes. The volume mesh generation was used to create the finite element model shown in Figure 2b. This model consisted of 1,788 nodes, 1,219 8-node and 35 6-node elements. The number of equation to be solved was 5,188, with an RMS wavefront of 195 degrees of freedom.

Stress results were post processed using MSGSTRESS and plots of the invariant stresses were made. Fig.3, the major principal stress, shows the highest tensile stress in the exact areas that were cracking.

5.2 The FIESTA solution

The FIESTA grid shown in Figure 2c is comprised of 28 elements and 222 nodes. The problem was solved using three different levels of interpolation: quadratic, cubic and quartic. The corrisponding number of degrees of freedom is reported in table III.

A detail of the contours of the first principal stress is shown in Figure 4. It may be noted that:

- The regions of stress concentration computed by FIESTA are in agreement with both NASTRAN results and experimental areas of failure.

- The discrepancy in the peak stress values is due to the mathematical singularity occurring at the reentrant edge. Infact the theoretical elastic solution exhibits infinite stresses along the edge and higher and higher stresses
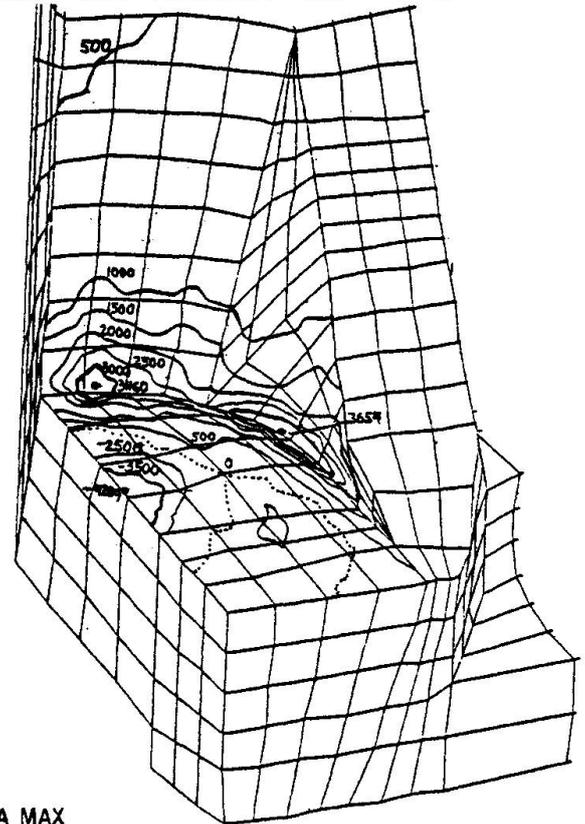
SIGMA MAX

Fig. 3 - Contours of first principal stress in the area of cracking.

TABLE III

	NASTRAN	FIESTA		
NUMBER OF NODES	1788	222		
NUMBER OF ELEMENTS	1254	28		
NUMBER OF DEGREES OF FREEDOM	5188	pl=2	pl=4	pl=6
		666	1473	2625
MAXIMUM WAVEFRONT	195	81	149	236

may be computed by providing
additional degrees of freedom
near the irregularity. Realistic
values of stresses, if needed,
should be computed by elastic
plastic analysis and require
to model the real radius of curva-
ture of the irregularity (compare
fig. 2a and fig. 2c).

— The regions of probable failure
are clearly pointed out by the
quadratic solution already.
The cubic solution is needed
just to confirm the results.
The combined computer cost of
a quadratic and a cubic solution
is obviously less than the cost
of the NASTRAN solution. Note
that quadratic elements are availa-
ble in NASTRAN as well. Coarse
meshes are difficult to use,
however, because the accuracy
of the results is not easily
assess in NASTRAN.

— The stress oscillation between
FIESTA solution are due to the
singularities at the reentrant
edge and in the area where the
bolt load is applied. This is
a very large concentrated force
which causes a singularity of
the Boussinesq type. FIESTA
provides the best energy-approxi-
mation, although it is unable
to interpolate the singular solu-
tion by standard polynomials.

Fig. 4 – Contours of first principal
stress for a) quadratic, b) cubic and
c) quartic polynomial interpolation.

This is not a reason of concern, as the mathematical irregularities
have no engineering significance. NASTRAN provides smoother but equal-
ly unrealistic stress results in the singular regions.

6. CONCLUSIONS

A new software system for the modeling of solid continua has been released
for commercial use. The new system is based on recent theoretical advances
and is sensibly different from state-of-the-art finite element programs.
An example of successful application of FIESTA has been reported. It is hoped
that the new system will get a favourable acceptance by the finite element
community. Extensive use of FIESTA is necessary to assess the merits of the
new approach.

REFERENCES

1. BABUSKA I., RHEINBOLDT W.C., A-Posteriori Error Estimates for the Finite Element Method. Tech. Rep. TR - 581, University of Maryland, Sept. 1977.
2. SZABO B., BASU P.K., ROSSOW M.P., Adaptive Finite Element Analysis based on P-convergence. NASA Conference Publication 2059, pp. 43-50, Nov. 1978.
3. PEANO A., FANELLI M., RICCIONI R., SARDELLA L., Self-Adaptive Convergence at the Crack Tip of a Dam Buttress. Proc. Int. Conf. on Num. Meth. in Fracture Mechanics, Swansea U.K./9-13 Jan. 1978.
4. PEANO A., RICCIONI R., Automated Discretization Error Control in Finite Element Analysis. 2nd World Congress on Finite Element Method, Bournemouth, U.K., 23-27 Oct. 1978.
5. PEANO A., PASINI A., RICCIONI R., SARDELLA L., Self-Adaptive Finite Element Analysis. Proc. VI Int. Finite Element Congress, Baden Baden, Nov. 1977.
6. PEANO A., PASINI A., RICCIONI R., SARDELLA L., Adaptive Approximations in Finite Element Structural Analysis. Computers and Structures, Vol. X, pp. 333-342, 1979.
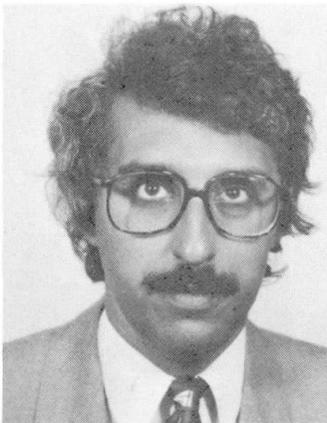
STEELSTRAC: Steel Structures Automated Coding

STEELSTRAC: système de codification automatique des structures métalliques

STEELSTRAC: Automatisches Entwerfen von Stahlstrukturen

J.P. RAMMANT
Dr. Ir.
Scientific Application Group SV
Meensel-Kiezegem, Belgium

E. BACKX
Dr. Ir.
Catholic University of Leuven
Leuven, Belgium

Jean-Pierre Rammant, born 1950 is a structural engineer and doctor of applied science from the Catholic University of Leuven. A Former assistant at this university and research associate at M.I.T., he is now directing a software-house specialized for the construction and consulting industry.
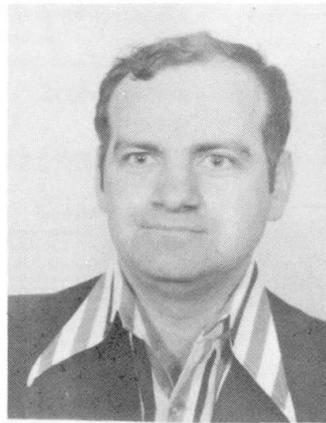
Edgard Backx, born 1948, obtained his mechanical engineering degree and his doctor of applied science degree at the Catholic University of Leuven. He spent 4 years at the Von Karman Institute making measurements in hypersonic turbulent boundary layers. Since 1975 he has been at the Catholic University of Leuven.

SUMMARY

STEELSTRAC is an intelligent software-package consisting of modular programs which can be used separately or can be built together to form a CAD package. It is developed for Wang mini-computers and programmed in BASIC. A specially developed engineering data-base system makes it possible to use a unique data-base for the structure to be designed. This data-base will be used for generating drawings, for making the necessary calculations, for detailing and for planning. STEELSTRAC allows constructors to start effectively with a moderate investment and to expand gradually the hardware and software configurations.

RESUME

STEELSTRAC est un logiciel composé de modules utilisables séparément ou qui peuvent être regroupés pour former un système de CAO. Il a été développé pour les mini-ordinateurs «Wang» et les programmes sont écrits en BASIC. Une base de données techniques développée à cet effet permet la description complète sous une forme unique de la structure étudiée. Elle fournit toutes les indications nécessaires au calcul, au dessin, à l'étude des détails constructifs ainsi qu'à l'organisation des travaux de construction. STEELSTRAC permet aux constructeurs de démarrer avec un investissement modéré et d'étendre progressivement leur installation.

ZUSAMMENFASSUNG

STEELSTRAC ist ein intelligentes Software-Paket, das aus modularen Programmteilen besteht, die entweder einzeln oder zusammen benützt werden können. Es ist auf einem «Wang» Mini-Computer entwickelt und in Basic programmiert worden. Ein speziell entwickeltes Datenbank-system macht es möglich, eine einzige Datenbank für eine zu entwerfende Struktur zu verwenden. Diese Datenbank wird gebraucht um Zeichnungen zu generieren, um die nötigen Berechnungen zu machen, für Planung und Ausarbeitung der Details. STEELSTRAC erlaubt es einem Konstrukteur effizient und mit mässigen Investitionen zu beginnen und die Hardware- und Software-Konfiguration stufenweise auszubauen.

1. THE INFORMATION FLOW IN A STEEL CONSTRUCTION FIRM

Table 1 outlines the exchange of information in a firm starting from a bid till final delivery of a structure.
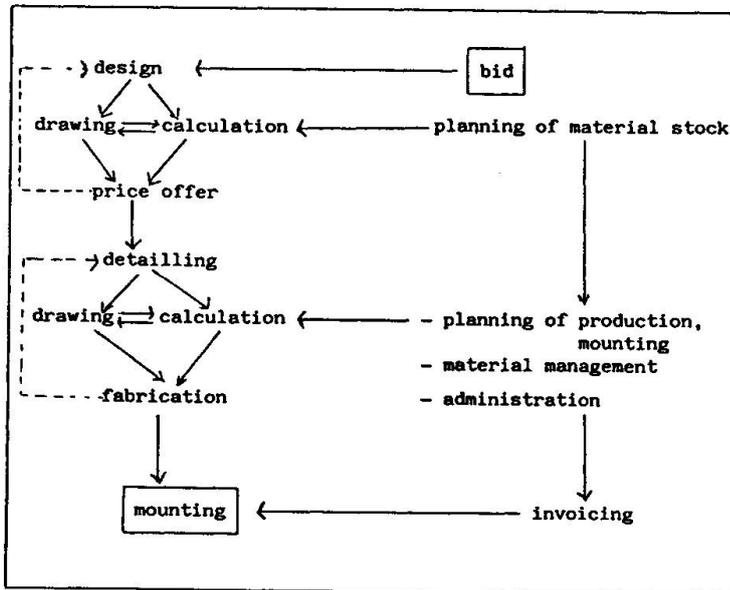
Table 1

The good progress of all these interdependent actions requires technical as well as administrative skills and the knowledge of firm dependent customs.

Because several people from different departments are involved, an exchange of internal reports and memo's is necessary and its volume increases with the size of the firm and the size of the project.

The computer makes it possible, provided that the software is available, to decrease drastically the amount of reports and memo's, thereby eliminating a large source of errors and misunderstandings.

The goal is : - to make more refined calculations
              - to automate drawing to a large extent
              - to use data-banks which contain information on selected profiles, stocks, prices
              - to have access to a library containing previous projects which could be used as a starting point for new projects
              - to assist the managing of projects and to controle costs
              - to produce punched tapes, floppies or programs for NC machines and robots.

To meet all those objectives, a large amount of software is required and is to a certain extent, firm dependent. That is the reason why a modular approach has been chosen, such that each module can be developed, tested and used on its own. These modules can be divided into 6 categories (Table 2).

	Computations	Drawings	Administration
Total Structure	(linear & nonlinear) - static - dynamic - optimization - stability - verification	- general overview - sections - projections - implantation, ...	- data for price offer
Details	- steel profiles - connections - foundation - stock-management	- monoplans - connections - anchorage - welding details - mounting plans	- parts list - punched tapes - work sheets - production plan- ning - cost controle

Table 2

Superimposed on this scheme, there is a need for a global management system which controls the follow-up of the project, handles the stock control and the financial administration.
Each category (Table 2) consists of more than one module and will briefly be described in section 2.
Section 2 gives an overview of characteristics which are common to all modules.

2. THE STEELSTRAC SOFTWARE. GENERAL CHARACTERISTICS.

It is not that difficult to develop a program on a desktop computer that gives a good answer for simple testcases. The step from writing such a program to a commercial product is not a small one.
A commercial program has to be correct, reliable, robust, easy to modify and extend, user-friendly and must be maintained by its authors.
To meet all these requirements for a series of programs which use the same database requires even more discipline and organisation from the program developers. Because the STEELSTRAC software is designed for a small computer (WANG), one has to be aware of the specific limitations of these type of computers :
1. small core
2. processor speed limitation
3. accuracy of calculation.
To take account of the first limitation, a dynamic memory allocation and management system has been designed. (1) This system has the same aim as a virtual memory system in mini-computers. This means that regardless the size of the problem, no action of the user is needed because the software will check if a variable needed is in memory or not. If it is not in memory, the required page will be loaded. This has for effect that a large problem can run as well on a 32K machine as on a 64K machine provided sufficient external memory is available. However, the problem will be solved much faster on a 64K machine because fewer paging will occur.

A special problem is posed in F.E. calculations where a large system of equations has to be solved in a small core. The first algorithm implemented used a blocked Gauss elimination.(2) This algorithm was implemented to take into account that the matrix is symmetric and banded.

The new release uses the same algorithm but the required primary and secondary core is further reduced because only non-zero blocks of coefficients are taken into account. This is especially favourable for problems with a variable band-width.

To account for the second limitation especially severe when using a BASIC inter-
preter, the processor speed limitation, machine coded instructions are used when-
ever possible.
These instructions include matrix functions such as matrix multiplication, inver-
sion e.a. and sort and search instructions.
The positive side is the reduction in program development time :
we estimate by experience a 5 to 1 gain in interpreter/classical compiler program
writing time.

The accuracy of the calculations is guaranteed for well-posed problems because
all computations are made with 13 digits. All input is checked for validity and
extensive use of graphical controles has been made.
Equilibrium- and algorithm checking is provided whenever possible.

3. THE STEELSTRAC SOFTWARE : DESCRIPTION OF MODULES

The Steelstrac software can be used without manuals after a few hours of training.
This is obtained by using menu's on the screen which show the different possibi-
lities. (3)

3.1. Software for computation of the total structure (4), (5)

The computational procedures implemented are based on the displacement method
for bar- and beam-structures and on the finite element method for 2-D, axi-
symmetric and plate structures.
Table 3 shows a menu taken from the package as it appears on the screen.
This menu illustrates the different possibilities.

```
E S A - 3 ***   SCIA - Software  ** 2D Frames  * Version 1 *

                                        SCIA s.v.


1 .......... PROJECT DESCRIPTION - TYPE OF ANALYSIS
2 .......... COORDINATES
3 .......... BOUNDARY CONDITIONS
4 .......... TOPOLOGY
5 .......... BEAM HINGES
6 .......... PROPERTY TABLE
7 .......... CORRECTION FOR SHEARFORCE
8 .......... EXCENTRIC CONNECTED BEAMS
9 .......... ELASTIC CONNECTED BEAMS


                 Number   --- (0=NO ) 0

Loaded  :EXAMPLE STRUCTURE -A12 121221 77 /B12
Date    :06.06.82
```

Table 3

The menu approach facilitates input corrections, editing, additions on an inter-
active basis without programming knowledge of the users.

The following modules are in use :
- linear static analysis for two- and threedimensional beam structures
- second order elastic analysis
- plastic analysis
- dynamic analysis with spectral or time response
- module for automatic optimization using linear programming (full plastic design)
- influence lines, load combinations (minimum, maximum)
- generation routines with parameterized input (simple one span, two span portal
  frame buildings)
- finite element plate and 2D elasticity analysis
- mesh generation scheme for geometry, supports and loadings.

To check the results graphical methods are used, as illustrated in figure 1, where the bending moments are drawn.
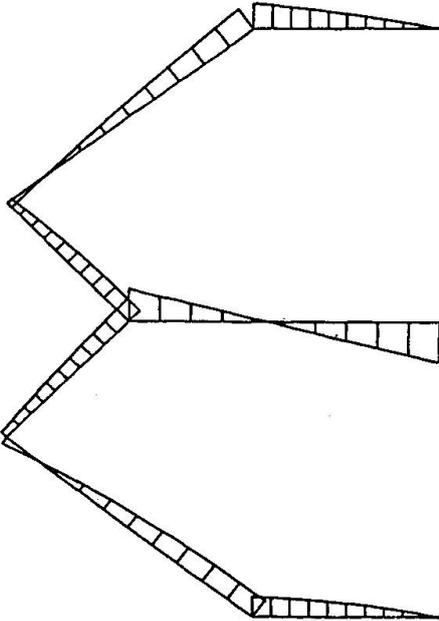
Fig. 1

Other possibilities are : deformed structure, loading schemes, perspective drawings, isolines for finite element results.

3.2. Drawing of the structure

At this stage the structure is drawn to have a complete overview and to present the structure to the designers.
More drawing details can be added : axis lines, girders, wind bracings, purlins, non-structural components.
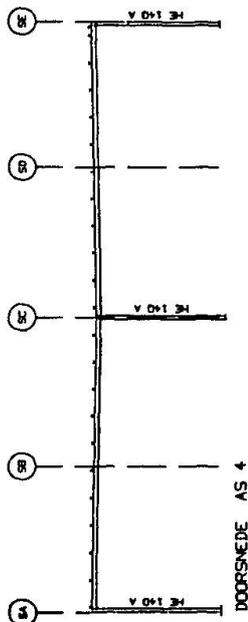
An example drawing is shown in fig. 2 where a section through the building is represented.
Standard foundation connections (anchorage plan, concrete stands) may also be introduced.
All input for the finished drawings can be formulated with the help of the results of the design calculations.
Straight input of columns, girders, ... is also possible.

Fig. 2

3.3. Price offer

A provisional bill of material is printed covering : a list of the profiles (standard, non-standard) with lengths and sorted in function of range of order, pointing surface, weights.
In a more general word processing software the latter data are assembled and a global text with additional activity descriptions is produced.

3.4. Detailing

The computations include the automatic optimal selection of profiles and joints as well as checking of structural loaded components to see which are conform to the building codes or regulations (e.g. TGB Netherlands, NBN Belgium, DIN Germany).
The modules allow a user to set up, change or extend the list of available profiles.
As an example some screens on the stress calculation are given below.

```
====>       STEEL Package - Dutch regulations     (TGB)  <====
================================================================
1      BUCKLING HE-PROFILES IN UNSHORED FRAMES  - - - - -     1
2      BUCKLING  I-PROFILES IN UNSHORED FRAMES  - - - - -     2
3      BUCKLING HE-PROFILES IN SHORED FRAMES  - - - - - -     3
4      BUCKLING  I-PROFILES IN SHORED FRAMES  - - - - - -     4
5      BUCKLING CORNER-STEEL IN UNSHORED FRAMES - - - - -     5
6      BUCKLING CORNER-STEEL IN SHORED FRAMES - - - - - -     6
7      DETERMINATION MINIMUM HE-PROFILE - - - - - - - - -     7
8      DETERMINATION MINIMUM TUBE-PROFILE - - - - - - - -     8
9      LATERAL BUCKLING OF  I-PROFILES  - - - - - - - - -     9
10     STRESS CALCULATION IN I/HE PROFILES  - - - - - - -    10
11     INPUT AND/OR CORRECTION OF PROFILE-LIBRARY - - - -    11
15     CONNECTIONS  - - - - - - - - - - - - - - - - - - -    15



'0=START / '1=INITIALIZE        GIVE NUMBER AND 'RETURN'? --/
```

Table 4

The results of a static calculation are automatically sent to the detail calculations.
In table 5 the effective buckling length is determined to find the reduction coefficient for the stress calculation.
One can immediately verify if the choosen profile fulfills all requirements. (Table 5 : see next page).

```
*** PROFILE TYPE ***
PROFILE  :  HEA200
DIRECTION :  X

================================================================
I   = 36920000 mm4      Iy = 13360000 mm      Wx = 389000 mm
A   =  5380.00 mm2      ix =    82.80 mm      iy =   49.80 mm
================================================================

Nx-max .....:     45.000  kN   <*>   Sx-max .....:    12.000  kN
Mx-max .....:     23.300  kNm  <*>
Nx-top .....:     38.000  kN   <*>   Nx-bottom ..:    45.000  kN
Sx-top .....:     12.000  kN   <*>   Sx-bottom ..:    11.000  kN
Mx-top .....:     11.200  kNm  <*>   Mx-bottom ..:    23.300  kNm
X-top ......:  5.104E-02  m    <*>   X-bottom ...:  0.000E+00  m
Y-top ......: -6.610E-03  m    <*>   Y-bottom ...:  0.000E+00  m
Phi-top ....: -1.090E-02  rad  <*>   Phi-bottom .:  3.410E-02  rad
L-scheme X .:     3.000   m    <*>   L-scheme Y .:     3.000   m
================================================================
CORRECTION DATA   (1=YES/0=NO/STD 0) :? -/


PROFILE  :  HEA200
================================================================
Rho-top   =      0,42      <*>   Rho-botto =     2.64
Lb-x      =      6.95 m    <*>   Lb-y      =     3.00 m
Lambda-X  =     84.01      <*>   Lambda-Y  =    60.24
Omega-max =      1.68      <*>   Sigma-Eul =   293.65  N/mm2
Nx:(Nx-1) =      1.02      <*>   Theta     =     1.00
================================================================
Sigma-Buck + Sigma-Bx(red.)  =   14.09  +52.40 =   66.50  N/mm2
Sigma-Comp + Sigma-Bx(max )  =    8.36  +59.89 =   68.26  N/mm2
================================================================
'0' = ANOTHER PROFILE/  '1'= NEW CALCULATION    / '2'= PRINT
'3' = CORRECTION RHO-BOT. /  '4'= CORREC. Lb-x   / '0 = MENU
                              ** YOUR CHOICE (STD 0) :? -/ **
```

Table 5

In the same sense the connection details are designed.

```
====> Index CONNECTION DESIGN following plasticity       <====
================================================================
1       Column - foundation Connection (method 1)  - - - -    1
2       Column - foundation Connection (method 2)  - - - -    2
3       Bolted hinged connection (L profiles) - - -  - - -    3
4       Bolted CROSS connection - - - - - - - - - - - - -     4
5       Bolted TEE   connection -- - - - - - - - -  - - -     5
6       Bolted KNEE  connection - - - - - - - - - - - - -     6
7       Welded CROSS connection - - - - - - - - - - - - -     7
8       Welded TEE   connection - - - - - - - - - - - - -     8
9       Welded KNEE  connection - - -- - - - - - - - - - -    9
10      Initialisation basic design parameters - - - - - -   10
11      PROFILE LIBRARY - EDITING / INPUTTING / CONSULT. -    11
12      Standard dimensions table 1 - check - - - - - - -    12
13      go to stress calculation - - - - - - - - - - - - -   13
================================================================
'0=START / '1=INITIALIZE        Give Number &  'RETURN'? --/
```

Table 6

Table 6 shows the program menu with selection of type of connections.
In table 7 the results of a calculation for the bearing plate under a column is shown.
High strength bolts and weld dimensions are designed.

```
=============== P R O F I L E   P O S I T I O N I N G ============

      ------------------------------------------------    +
  I                                             I   I  20
  I              II            II               I   +
  I              II            II               I   I
  I              II            II               I   I
  I              IIIIIIIIIIIIIIII               I   I  100
  I              II            II               I   I
  I              II            II               I   I
  I              II            II               I   +
  I                                             I   I  20
      ------------------------------------------------    +
          10              200         ? 30
         +----------------+--------------+------------+
```

```
==============================================================
*** COLUMNPROFILE ** PROFILE   -   IFE200          DIRECTION - X
==============================================================

f'bd      =       13.50  N/mm2 <*>  f'vd       =      18.00  N/mm2
Depth     =      390.00  mm    <*>  Height     =     215.00  mm
free span =       95.00  mm    <*>  free span  =      57.50  mm
Amin      =     5555.27  mm2   <*>  Sigma  vd  =       1.19  N/mm2
M(I-I)    =       81.22  kNm   <*>  t(I-I)     =      45.06  mm
M(II-II)  =       90.83  kNm   <*>  t(II-II)   =      47.65  mm

==============================================================
'0' = ANOTHER PROFILE/  '1'= NEW CALCULATION    /  '2'= PRINTING
'3' = OTHER PLATE DIMENSIONS  /  '4' = BEGIN  /  '5' = ANCHORS
                              ** CHOICE   (STD 0) :? -/ **
```

'0 = INDEX <*> '1 = INITIALIZE BASIC PARAMETERS <*> '7 = BEGIN

Table 7

3.5. Production drawings

The details are ready for preparing the monoplans.
Figure 3 shows a detail of the column-foundation connection.
Each of the parts can be drawn separately and various lists of materials are prepared.
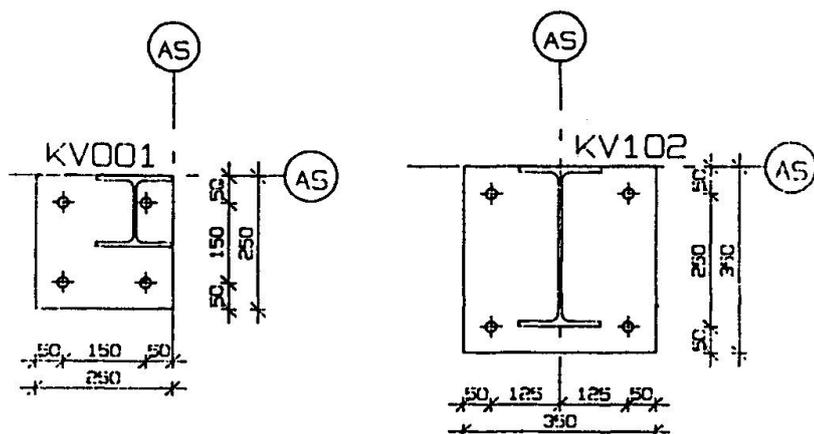(Fig. 3 : see next page)

Fig. 3

To have a minimum effort in designing the connections, it is preferred to use standardized connection types with parametric values of bolts, plate dimensions, etc. In principle, the software has no restriction in drawing complex assemblies.

3.6. NC machinery & production planning

The drawings generated in 3.2. and 3.5. are used to prepare the production.
The structural data-bases are consulted to extract machine-operations.
A typical application is the preparation of punched tapes for drilling holes in standard profiles.
The sequence of operations and bore positions is automatically programmed.
Other operations are classified and listed in sorted order to be used in the management decision & planning.

4. CONCLUSIONS

The STEELSTRAC software offers a complete solution to steel constructors.
The software is developed for WANG computers and is written in BASIC.
Starting from the design, it allows the user to make drawings and strength calculations of the complete structure and details.
It also makes a link to management and cost-control.
Each module can be used separately or can be used in connection with the other modules because it is based on a unique data-base accessible by the standard modules or by user written special purpose programs.

5. ACKNOWLEDGEMENTS

6. REFERENCES

1. BACKX E., The implementation of a dynamic memory allocation and database management system on Wang Computers, in preparation.

2. BACKX E., RAMMANT J.P., SCHYMKOWITZ G., SAP Runs on a 16K Desk Computer, London Third SAP Users Conference, Los Angeles, 1978.

3. BACKX E., RAMMANT J.P., SCHYMKOWITZ G., Interactive Engineering software for desktop SAP computers, Engineering Software, Pentech Press, 1979, pp 161-179.

4. SCIA, Program sheets

5. BACKX E., RAMMANT J.P., Structural Dynamic Interactive Analysis in Basic on Micros, Engineering Software, Pentech Press, 1981, pp 818-827.

Leere Seite

Blank page

Page vide

SESSION III

DISCUSSION

October 7, 1982 - Afternoon

Chairman: E. ANDERHEGGEN (Switzerland)

F. STEIGER - I have a question to Mr. Kruisman. I think The Netherlands are not so a great and big market in the building sector and, if all the mentioned 240 companies accept and buy your system, I assume everyone can do nearly everything. And the question is: how long would the most of these companies still exist because they are all so strong in the market with these powerful tools and the number of clients will not increase?

G. KRUISMAN - This question could be answered fairly simple because,as I indicated, the total group amounted to about 2400 companies, and we only took into account ten percent of this number. So, if this ten percent survives,the problem is solved. But,of course,these companies will not act in the same way. I indicated already that we are trying to bind developments together, to stimulate cooperation between technical disciplines. In each discipline several subjects will be covered. With help of the Nemesis system you can build CAD systems and each CAD system will be connected to one product. So there is a lot of work to do in order to cover all products and I think we will be working for many years.

D.P. GREENBERG - Not without risk, but with the deliberate intention of trying to start a controversy, there were two themes which really came across. One with respect to the need for,not only general databases or relational databases, but database management systems to make it effective in the building industry. The other theme promotes the uses of microcomputers. There is surely a paradox in the fact that the microcomputer can support these large database management systems. I would like to provoke the argument between the panelists on this.

S. FENVES - There are several hardware developments which will make this boundary disappear. A number of manufacturers are producing back-end computers which handle database access. At an even smaller scale, a number of companies are coming out with database chips. INTEL has announced one of these as a product intended to be connected to other micros. Thus, as far as hardware is concerned, I don't think there is any problem: there is a continuity of available hardware to do it, and there is going to be more of it in the future.

H. PIRCHER - I have to say that, if microcomputer configuration is very small, there is no place for database management,but there is a need for some functions coming out from the database. So, in a good system for microcomputer configuration, it is necessary that the program includes some function of the database. If I think of our program for prestressed concrete, for example, it is necessary to organize and store a lot of data and I think that we have various construction stages,each giving us seven or eight lowcases; we must add all the stresses for few construction stages that we have, to store them for later drawing and so on. If all this is functioning in one system of program running on a micro-computer, I think we have some part of the database management system into this

program. And it is clear that future development in hardware will produce some possibilities to include database: possibility also in microcomputer environment, as a program is necessary to define how we can connect these features, if they will be later available. But, actually, we have to do it on a small configuration, if we supply programs for it; because there is always the question of calculation speed, which is limited also by the peripheral devices of a microcomputer and the very cheap hard disks of five or ten megabytes (already available) are slow.

E. ANDERHEGGEN - So we are really expecting an other computer generation, which is probably coming very soon.

G. KRUISMAN - And there are already several database systems running on 8 bit, 64 Kb memory micros.

H. PIRCHER - But, if we include it in a complete calculation system, we will get unbelievable calculation-times, not calculation, but waiting-times.

E. ANDERHEGGEN - If I may add a comment, I think database makes sense if there are multiple accesses from different applications to the same database. If you have only one program that accesses the database, this is not a database any more, it makes no sense. You can save your data on tape every week or something like that.

H. PIRCHER - But there are already microcomputer systems available, where it is possible to connect a lot of microcomputer systems in one network, that excess the common disk-space.

S. SHIMADA - In our research laboratory, students are very fond of gaming on TV screen using the microcomputers and they are gaming without making the report of their home task. But, during this gaming, we found some problems on the use of computers. And one is reliability for some accident. Yesterday, for instance, we had a very strong thunder storm and some of the thunders caused accidents. The memory is attached sometimes and changes the data during the computation. When the students were gaming on TV screen, the game works were found wrong. I think this is one of the most important problems for the professional calculators.

E. ANDERHEGGEN - I think it is a problem of data security, can anybody comment on that?

H. PIRCHER - Yes, that is the same problem. Then for each other computer (you have the same problem on a big unirecord on a VAX), if you look at reliable micro computers, you can see that there are some for the price of thousand dollars and there are others for the price of ten thousands dollars, all based on the same CPU chip. The main difference between both machines is the quality of the power supply into the machine and there are very very cheap personal computers, which are only cheap and, if you go to a special floor and you put the hand on it, you can switch it off immediately. But there are other machines, with well insulated housing and with a good power supply, so that you have the same quality than a big computer, and also the same possibilities to avoid such events.

S. SHIMADA - Yes, but in our research most of the study is carried out in the field and the field works are very often attached by several thunders, because we are working in the open structure-sites and sometimes the cables are extending to some thousand meters from the local working station; these cables sometimes are attached. So we are now trying to ensure them most reliable transmission for the automated and robotic measurements.

J. BLAAUWENDRAAD - I have two questions. The first question is directed to the people who were speaking on database management systems. A couple of years ago, at a symposium of NASA in Washington, Lockeed and Boeing both had a strong plea, why they did not use the available administrative management database systems. They argued that they had to make specific database systems which were oriented to engineering. The question is: is airplane industry so different from structural engineering that we have to do it in an other way, or have we changed our minds in the last three or four years? The second question is (that is to all panel members, I think): we are speaking of the impact of technology of hardware and software to our field, the economical recession in western community might be a stronger impact in this moment. What do you feel about this impact on what we are doing?

K. VAN DER WERFF - As a matter of fact I don't think that there is very much difference in aerospace industry. Months ago I heard of the Dutch National Lucht - en Ruimtevaart Laboratorium, they had installed a big database system which covers all their activities, it was from CDC and I think it was a quite normal database system.

S.J. FENVES - The Lockeed system is not generally available. The database management system that cames out of ICAM is the one that I mentioned. It is called RIMS, it is distributed by Boeing, and now private companies are also marketing it. The one substantial change they have made was to add matrices and vectors as additional attribute types.

J. BLAAUWENDRAAD - As for the answer of Dr. Van der Werff, I agree that the Dutch Aerospace Institute did use an available database management system, but when I asked for the reason, it was just financial. They had their problems using it, but they could not effort to do an other new effort themselves. Therefore I asked: wasn't it possible for Boeing then to use one of the available administrative systems?

S.J. FENVES - It is a very small package; they admit that it was initially developed for their own internal education. I listed a minimum of four things that I feel administrative databases don't contain and of those four RIMS only implemented one.

K. VAN DER WERFF - It is also not very clear to me. On the IFIP Conference on CAD databases in 1981 it was stated that the requirements we have for engineering databases are exactly the same as they have for administrative databases. The only difference may be that some people say that engineering database should have a flexible structure, so it should dynamically be changeable, but I am not quite sure about this requirement. Dynamically means that you can dynamically change the logic of your data.

E. ANDERHEGGEN – Does the application program play the role of the classical database administration someway?

K. VAN DER WERFF – Yes it is possible in relational database. You can just relate the most strange things with each other. That's dynamical but that is also dangerous because you might get up to inconsistent database.

E. ANDERHEGGEN – And, if there are multiusers to the same database, it might run into problems with other users as much as I understand (I am not a database specialist). So we have other questions? Oh, I forgot the political question. Who wants to say something about that?

P. LENGYEL – I think this has a lot to do with yesterday's very interesting discussion concerning who should take the lion's share in CAD development: the universities or the software houses? I feel that here the computer manufacturers have to be very active. I work at a computer factory in Hungary producing basically mini computers and we think that we have to help the users in this field. I try to sum up shortly what I see here as a way out concerning economic difficulties of the bases of program development.

E. ANDERHEGGEN – If I understood the question correctly, it was about the european recession and now you say you feel responsible for this. Well, I don't think you are responsible. I just want to make sure that you understand each other correctly. The problem was that we all feel in many european countries a recession and this might have some consequence. May be people have more time to develop software, in this sense it could be an advantage, but I don't want to interrupt you, just to get the contact between you two.

P. LENGYEL – What we do – and I think this way is very advantageous to be followed by other manufacturers too – is developing program packages by ourselves or make them made by software houses in the following way. We get into contact with real or potential users and then, according to their wishes, we shape our program library being developed partially by software houses, universities departments or by ourselves. The hardware for all this is developed by us, the manufacturer. We feel that, if all the three parties concerned are working together, it contains scientific and economic advantages as well. Namely one gets good brains from software houses and universities departments and a direct contact between them and the industry. Thus there is also more money provided for the development. So I can say this is a solution which works well in this field.

E. ANDERHEGGEN – If I can make a comment, I think one of the few industries we have, which we ignored in this session, is probably the hardware industry.

H. PIRCHER – I can make some comment on this question. I have the experience that, especially in times when economics is more down, it is more important for industry to make rationalization and I think software development is a deem which should help rationalization. Infact, especially in the years with bad economics, we got some orders only due to the fact that rationalization was necessary and, especially in software development, the time now is not so bad as for other parts of economics.

H. WERNER - Mr. Lengyel indicated that we will have a lack of practical applica-
tion in the '80s. Bearing this in mind, I would like to ask Dr. Pircher, or
other gentlemen who are involved in finite element program development or appli-
cation: Let us suppose you implement a finite element program on the micro, and
that micro with this program is bought by a small consulting firm. How can you
ensure that this powerful tool is used in a correct, in a proper way?

H. PIRCHER - That is a nice question, but first statement is that I know very
small engineering offices having a lot of very specialized knowledge about some-
thing, may be finite elements. For example, we have three or four customers
using finite element programs on very small machines. There is only one man,
but one man who knows what he does. But it is clear that very often there is
people buying a program, may be a finite element program, with the illusion that
this program replaces the knowledges and I have to repeat my statement that the
computer should support an engineer, but it cannot replace him. And it is really
very dangerous to give sophisticated software to people unable to understand
it, and sometimes, if we recognize that the situation will be bad, we refuse
delivery of programs. We did it, because in our prices we include the maintenan
ce for one year and, if we give a program to a customer which calls us by tele-
phone every time he switches on the machine, this is the end of business. And
that is the problem: how to give the needed knowledge not only to students, but
also to engineers being in practical work for ten years.

J.P. RAMMANT - May I propose three solutions to the question? First you should
give the customer as much graphics as possible. Help him to use graphics,
that's what we try to do. Second solution: keep the price of the programs high,
what do you think? Third solution: we give regularly seminars on the applica-
tion of finite elements and therefore we use people from the university coming
to practice.

P.J. PAHL - With your permission, Mr. Chairman, I would like to return to the
question of government support. And I would like to specifically ask Mr. Kruis-
man the reason of the statement he made this morning, that he considers some
support from the government as an essential precondition for realizing his pro-
ject. If we look at the experiences we had in the Federal Republic of Germany,
we find that about 15 milion marks a year for support in the CAD area have been
completely stopped within the past two years because of economic developments.
This implies that, if we rely only on government, we become extremely dependent
on government. Could you comment on this?

G. KRUISMAN - Fear is a bad leader, but a good incentive. And there is fear
among companies for their continuity, because work is going down. There also
is fear within government that does not have a proper policy towards new techno-
logy. In Germany the government has stopped giving money. The same applies
to England. Both governments have already given in the past. In the Netherlands
this is not the case. So Dutch government has started now and - maybe in some
years, after this project is finished - they will stop too. I think that the
economic recession generates fear and that's why good ideas are supported now
by Dutch government.

B.A. SZABO - The problem identified by Prof. Werner is a very important one.

Abuse of the technology is certainly widespread. It has been said that "it is almost impossible to make anything foolproof, because fools are so ingenious". Nevertheless, there is work going on at the University of Maryland and at Washington University in St. Louis, for example, to make finite element analysis as foolproof as possible. As a result of this work, adaptive finite element computer codes will become available in a few years. Such computer codes will not only yield data which are of interest to users, but will also provide reliable and close error estimates in various norms. In his presentation Dr. Peano described certain steps in this direction. As we have seen, a p-version code is already in existence, which makes it possible to change the number of degrees of freedom by an order of magnitude with minimal user intervention and permits assessment of the effect of that change on the computer data.

H. WERNER - I have a specific question to Dr. Peano. Dr. Peano, you showed us very impressive high order elements. I have two questions; the first one is: what was the main reason for the development of this element and what about computer time for running high order elements in relation to many simple elements? The second question concerns the accuracy: is it checked by the user just looking at the results or have you automatic accuracy checks implemented? If yes, which?

A. PEANO - The first motivation for development of high order adaptive Finite Element techniques was data reduction. By reducing the number of elements, you reduce the amount of data in input and in output and save manpower in data preparation and in evaluation of results. The computer time is also reduced, because you end up using less degrees of freedom. In this regard, the first problem I showed is very illuminating. That problem was solved by using NASTRAN and more than five thousand degrees of freedom. However the result obtained using quadratic elements, with only six hundred degrees of freedom, was better than the one obtained with five thousand degrees of freedom and linear elements. The reason why the user selected many linear elements is that the simpler solution with six hundred degrees of freedom, is useless unless he has a way to know whether the results are reliable or not. This is exactly the capability provided by my approach: by going to the next higher approximation, which required about fourteen hundred degrees of freedom, I am able to validate the results just by simple comparison. And the cost of the two analyses is lower than the cost of one analysis with five thousand degrees of freedom. So I think that the first area of saving is due to the fact that, when there is no way of checking the accuracy, the user is forced to use as many elements as his budget permits. The second point is that there is now the mathematical proof, as well as the practical evidence that, increasing the polynomial order over the same mesh, is more efficient than subdividing the mesh. If you increase the order of interpolation, the convergence rate is twice as much, so you need much less degrees of freedom to reach the same accuracy. Moreover larger element matrices may be computationally advantageous provided you exploit it, for istance, by an array processor or by vector computers. I didn't show here how to automate adaptivity for reasons of time and because it has already been published by "Computers and Structures". Basically a sensitivity analysis is performed by computing locally at each point of the mesh the gain in strain energy expected, if more degrees of freedom are provided in that area. This error indicator is used to locate new degrees of freedom.

G. SCHMIDT-GONNER - I have a question to Mr. Peano. I am involved in the analysis of concrete structures - even with three dimensional finite element models - and I found that, in nonlinear analysis, more elements with simple formulations gave the best results. If there are cracks and other discontinuities in the element, it is not very useful to have high order elements with a high integration order. I am surprised that you use this large high order elements. Are you not looking for nonlinear problems or not dealing with structures with discontinuities?

E. ANDERHEGGEN - Can I also make a comment? I also believe that, when making a nonlinear analysis, you should try to stick to very simple models, so the question comes from both of us. Infact, if you make a non linear analysis, you are more interested in the overall behaviour of the structure, not in stresses which you will never get anyway.

A. PEANO - The point is that 90% or more of finite element applications are still based on linear elastic stress analysis. Non linear analyses are limited in number and are attempted by the most sophisticated users only. In many situations, nonlinear analyses are used to validate design criteria based on linear analysis. For simplicity of application, real design always tends to be based on linear elastic analysis, even when not strictly applicable.
Moreover I expect no difficulties in developing elastic plastic analysis capabilities with high order elements. Of course it is more difficult to model cracking. Still it is possible to try and either subdivide the element or add discontinuous functions. Various levels of sophistication are avaiable but in any case the cost of software development is certainly larger for higher order elements in problems with material nonlinearity.

Leere Seite

Blank page

Page vide