APPLICATION SOFTWARE DEVELOPMENT FOR MINI AND SMALL BUSINESS COMPUTERS

INPUT EUROPE

ABOUT INPUT

INPUT provides planning information, analysis, and recommendations to managers and executives in the information processing industries. Through market research, technology forecasting, and competitive analysis, INPUT supports client management in making informed decisions. Continuing services are provided to users and vendors of computers, communications, and office products and services

The company carries out continuous and in-depth research. Working closely with clients on important issues, INPUT's staff members analyse and interpret the research data, then develop recomm te ideas to meet client's needs. Clients to data on which analyses are based, an

Many of INPUT's profe ars experience in their areas of speciali ment positions in operations, marketing, INPUT to supply practical solutions to c

Formed in 1974, INPI nsulting firm. Clients include over lly advanced companies.

EUROPE
INPUT EUROPE I
Empire House
414 Chiswick Hig
London W4 5TF
England
London 995-5397
Telex 896739

ITALY
PGP Sistema SRL
20127 Milano
Via Soperga 36
Italy
Milan 284-2850

BELGIUM
CPMI S.A.
75 rue de Caraut
1420 Braine L'Al
Belgium
(02) 736-0054

Coast
ulevard,

4303

Coast

sey 07662

Company, Ltd
2-7 Kita Aoyama

Highland Centre, 7-9 Merriwa Street
P.O. Box 110, Gordon N.S.W. 2072
(02) 498-8199

APPLICATION SOFTWARE DEVELOPMENT
FOR
MINI AND SMALL BUSINESS COMPUTERS

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT'D)

TABLE OF CONTENTS (CONT'D)

1. INTRODUCTION

SECTION I INTRODUCTION

- This report is produced as part of the European Market Analysis Service (MAS/EUROPE).

- The report has five main goals :

  - review the problems facing the manufacturers of small business computer and office computer equipment in supplying application software in support of their hardware products, with particular emphasis on the complications brought about by serving multiple country markets in Europe;

  - analyse the application software strategies vendors are using, and the relative success of the various approaches in fulfilling the need for effective solutions while maintaining costs at an acceptable level ;

  - investigate the structural organisation and relative responsibilities of corporate, country, region and branch staff in developing and supporting applications software ;

  - evaluate the market demand for application software in Europe and typical development costs for an application, plus the overall vendor expenditures on applications development.

  - examine the extent to which U.S.-developed applications can be used by the European distributors, agents or subsidiaries of U.S. products or on similar European products ; (this issue is particularly important to newcomers to Europe, but is also a prime factor in the decision to market U.S. products in Europe). In addition, a number of company profiles have been added to provide an insight into how various suppliers have approached the problem.

- The information for this report was obtained by face to face and telephone interviews with Burroughs, CII-HB, Data General, DEC, H-P, ICL, Kienzle, Matra, NCR, Nixdorf, Philips, Prime, Univac and Wang. The interview form is given in Appendix I.

- In each case a written synthesis of the interview was returned to the respondent to allow for correction, amplification and elimination of company confidential data. In the case of the latter, confidential data has only been used in summary form, i.e. in a format that does not allow it to be traced back to a given vendor.

- Some notable equipment suppliers are absent from this study for various reasons. One vendor, after enthusiastically filling in the questionnaire, found the data so interesting that it was not released to INPUT ! Others were very interested in how others were treating this important problem but didn't wish to disclose what they themselves were doing.

- Fortunately, the vast majority were willing to provide frank discussions on their problems and applications strategies. They have made this report possible. A summary of the company responses, eliminating proprietary data, is shown in Appendix II.

- In order to enhance the scope of the report, data gathered by INPUT on large U.S. user experiences with software development has been included. In many cases the size of these users is equivalent to that of the European vendors listed, and given their preoccupation with applications development, their views are relevant.

- Inquiries and comments from clients on the information presented are invited and welcomed.

II EXECUTIVE SUMMARY

SECTION II EXECUTIVE SUMMARY

- The over-riding difference in the demand for application software in Europe with respect to that of the United States is that customized software not standard applications are predominant.

- This results from a variety of factors, foremost among which are :

  - the different national legislation and tax requirements in effect

  - the variety of national languages used

  - the insistance by most users on a solution for their specific problem.

- To this list, the U-S suppliers of equipment have themselves added the following complications :

  - not all models of a given range of products are sold in Europe

  - often only selected peripherals are offered on each model

  - similarly, they frequently offer a restricted list of options

- Furthermore, there are cases where European-specific requirements for special hardware are sometimes an essential condition for accessing a given market.

EXHIBIT II-1

HIERARCHY OF ANCILLARY APPLICATION SOFTWARE REQUIREMENTS

PROBLEM		NATURE OF REQUIREMENTS	OVERALL TREND TOWARD SOLUTION
RANK	LEVEL/TYPE		
1	INTER - PRODUCT RANGE TRANSFER	• File storage methodology variations • OS variations • Machine language variations • Peripherals variation • Functional extensions	• Separate application products • Separate versions of same product • Programming high level languages only • Specific subset of peripherals and functions supported
2	MODEL VARIATION WITHIN LINE	• Peripheral variation • Functional extensions • Processor incompatibilities	Versions of applications package for functional extensions and/or processor types
3	CONFIGURATION EXPANSION WITHIN MODEL	• Memory size varies • Spectrum of peripherals increases • Levels, versions and number of operating systems may vary	• Limited memory sizes and peripherals supported • Application usually works under variety of operating systems
4	EUROPEAN MARKET DIFFERENCES WITH U.S. PACKAGES	• Line, model, peripherals and options mix available in Europe may vary from those available in the U.S. • U.S. applications do not apply to European requirements	European application development staff for adaptation of existing applications and development of European applications
5	COUNTRY MARKET	• National language for system messages, documentation • National legislature, special tax requirements etc, demand specific calculation methods, documents	• Versions of documentation for main markets • National applications staff adapts European or U.S. - developed packages from source code.
6	COUNTRY SECTOR AND/OR BRANCH	• Sector/branch requirements that may or may not be common with other countries • Special hardware/options needed	• Standard applications only for those rare cases of common requirements with other countries. Licensing and/or one time changes • Customized packages, one-off programs for others, fee paying
7	LOCAL USER	• Own special requirements • On-going support after initial installation	Customized implementation and customized support after installation, both for a fee, (man/days)

- All of these problems result in the specification for a given application package being (a) significantly different from the U.S. counterpart (for U.S.-owned equipment suppliers such as Burroughs, NCR, DEC, H-P etc thereby eliminating the advantages that would accrue from the use of existing packages) and (b) infrequently useable at European level and rarely at all at country level.

- The number of application variations that language, country market differences and sector/branch markets produce is bad enough. However, the increasingly broad variety of models, configurations, functional extensions and options available within the small computer/office computer range poses its own particular set of problems to the application package designer.

- The ability of a given application to satisfy these and other corporate goals is just as important as the requirements for the application to match a set of functional/processing needs. These "non-processing" or ancillary requirements are now examined.

A. HIERARCHY OF ANCILLARY APPLICATION SOFTWARE REQUIREMENTS

- Exhibit II-1 lists these ancillary requirements (i.e.those needs that are supplementary to the actual processing capabilities of the application package itself). They are classified in descending order of interest to suppliers.

- Increasingly important is the ability for the application to cross over from one product to another within the same supplier, i.e. inter-product-transfer. The reasons are obvious :

  - the supplier's investment needed to create a fully developed application is rising rapidly (i) because it is manpower dependent but also (ii) because specifications are being broadened in an attempt to widen the scope of the application's use in terms of users needs and product configurations catered for

- the user's own investment in applying the package to his particular needs (customisation), training his operations staff, establishing the procedures within the company that allow full use of the data/reports produced by the package etc are also increasing

- the speed of introduction of new hardware products is increasing (i.e. the commercial life of the average product is shortening) making it imperative that applications be retained (and developed) for a changing hardware base.

- It is therefore eminently desireable, from both supplier and user standpoints, that change of model, product, operating system data storage media etc on the one hand, or the simple extension of functional ability on the same model/product, operating system data storage etc do not result in a major upheaval of the application package.

- This can be achieved by limiting the vendor's responsibility to limited configurations of certain models, but ensuring that the basic architecture of the application allows for (i) all of the major operating systems offered and (ii) all major models/new announcements as they occur. The investment in maintenance and functional upgrades and/or conversion efforts of an existing package is preferable to the development of a new product.

- For U.S. vendors, there are limited opportunities for implementing U.S-developed applications : European requirements are frequently so different as to make modification/adaptation of existing U.S. products as costly as the development of a European package from scratch.

- Country market differences are dealt with in one of two ways by the vendors interviewed :

- only one version of the application is offered, supported by documentation in one language

- local versions of the application are developed by the national application staff from the original source code, supported by translations (often rudimentary) of the original documentation.

- The approach adopted depends on the ressources available, but the single-version method apparently does not necessarily mean a loss of effectiveness, providing internal communciations are adequate.

- Sector/Branch differences usually result in either (i) customised solutions for each user in each country (ii) repeated use (with or without modification) of an early user-customised package or (iii) the (rare) use of sector-specific standard applications.

B. WIDENING GAP BETWEEN NEED FOR AND AVAILABILITY OF QUALIFIED SYSTEMS STAFF

- The rapid decrease in hardware prices, resulting from the miniaturisation of components, has brought EDP products within the expenditure range of an increasingly large number of neophyte users. Their lack of EDP knowledge in general and inability to design and program their own applications is the main barrier to the successful exploitation of the enormous market they represent.

- Already, available qualified systems staff are becoming a scarce commodity in Western Europe. This applies not only to vendors but to systems/software houses, independents and users. There is a growing backlog of system and application software to be maintained, upgraded and converted to run on new hardware ; this alone is absorbing over a quarter of the total available ressources.

APPLICATION SYSTEMS STAFF USAGE

WESTERN EUROPE

	TYPE	ACTIVITY	%
APPLICATIONS	OLD	● Maintenance of existing packages	15
		● Functional upgrade/improvements	10
		● Implementation at user sites	30
		● Conversion to new hardware	6
		● Other (distribution, file up-keep)	3
		TOTAL	64
	NEW	● Research, Planning and Specification	8
		● Development, original de-bug	8
		● Implementation at user sites	10
		● Maintenance, on-going	5
		● Documentation/Education support	3
		● Other (distribution, file up-keep etc)	2
		TOTAL	36
		OVERALL TOTAL	100

Source : INPUT Estimate
based on vendor
interviews

Exhibit II-2

- The rate of development of new applications is slowing down. Over 80% of existing standard applications, by number, are two or more years old. Small wonder, then, that only just over 30% of application system ressources are being applied to activities related to new applications (see Exhibit II-2). This situation is not going to improve since it is the users who determine the life of an application, (not the vendors who created it), and their reluctance to abandon the tried and trusted in favour of new solutions is very strong in Western Europe.

- Coupled with the predominant demand for customized applications at all levels and sizes of computers, this means that the larger need is for user application "tools", as opposed to fixed-function application packages. While standard applications are obviously the most attractive for the vendor, given their low cost (amortized over large numbers of installations), ease of support (standard documentation, manageable error levels, in-house competence can be developed etc) and speed of implementation (important for vendor cash flow), they cover on-average no more than 30% of the on-going requirements, (from 15 to 40%, see Exhibit II - 3).

- However, the in-built conservatism of European users can be capitalized on by vendors; certain standard applications have become prducts in their own right. The most popular can be put into firmware and offered as plug-in options to the mainline computer models, for a fee. Firmware evaluation capabilities, separately priced, are also attractive. Either would enable the functional freezing of old applications that tie up valuable systems programmer ressources, while generating revenue.

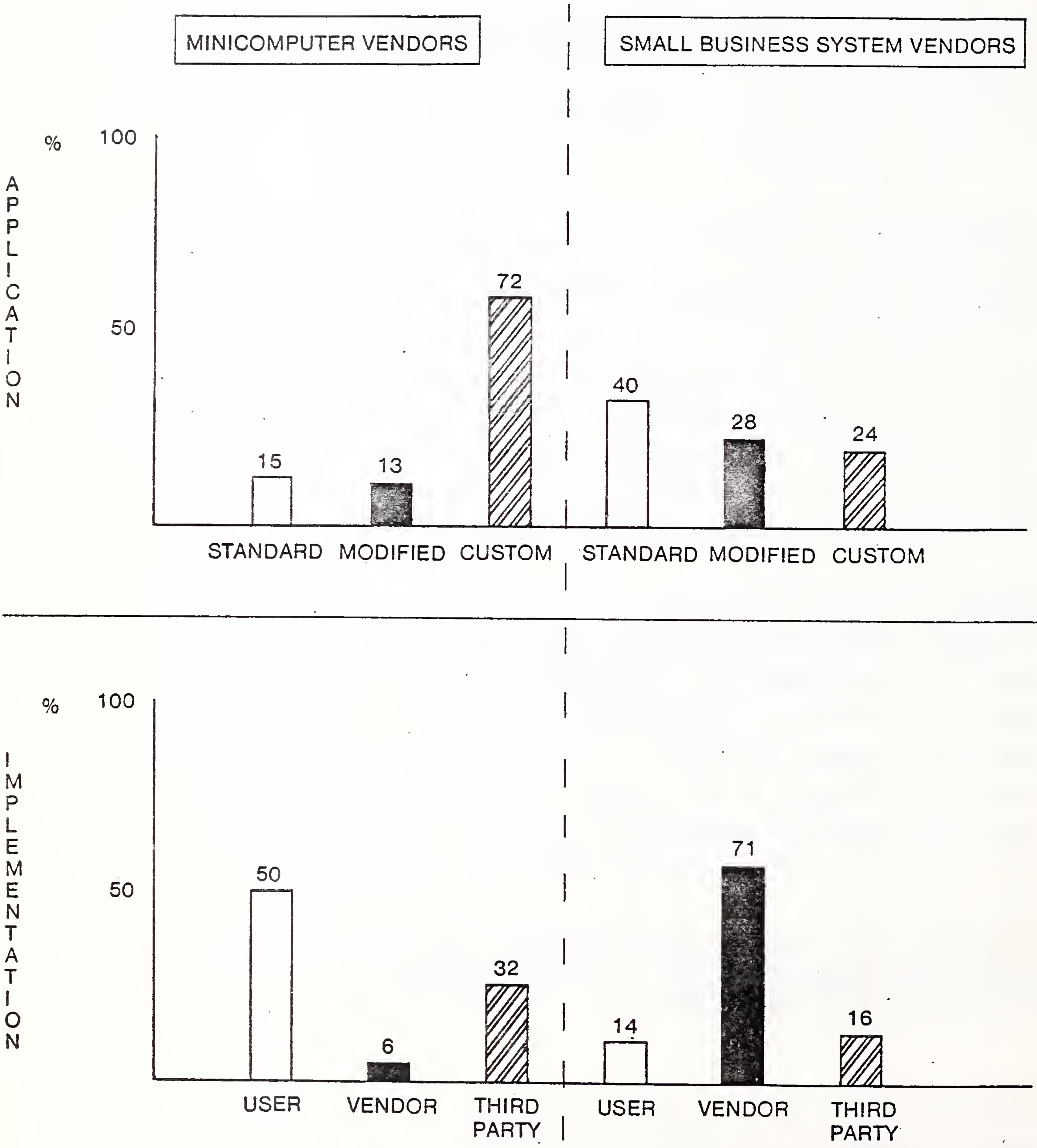
C. DIVERSITY OF APPLICATION SOFTWARE METHODOLOGY OF EUROPEAN EQUIPMENT VENDORS

- There is a wide variance of approach between the major small system vendors in Europe, central to which is the difference in attitude adopted by the minicomputer suppliers (DEC, DG, H-P, Prime and Wang, among those interviewed) compared to the small business system vendors (Burroughs, Philips, NCR etc).

MINICOMPUTER/SMALL BUSINESS SYSTEMS

APPLICATION TYPE/IMPLEMENTATION MIX

WESTERN EUROPE



Sample includes:

Matra, Data General, DEC, H-P, Prime and Wang

Burroughs, C I I-HB, ICL, Kienzle,NCR, Nixdorf, Philips and Univac.

Exhibit II - 3

- Uniformly the minicomputer vendors expect the users (or a third party systems/software house paid by the user) to shoulder the entire application package development responsibility. The vendors themselves provide only basic systems software, in the main.

- In contrast, the traditional small systems vendors attempt to provide all of the application software, with occasional (controlled) support of the third party systems/software house, often retaining the marketing rights and/or copyright.

- Exhibit II-3 illustrates the wide diversity of approach found between the two groups both in the use of standard applications, modified versions of available applications and custom packages, and in the implementation support mix.

D. EFFECTIVENESS OF THE APPROACHES ADOPTED

- That there is no unique method for satisfying user requirements is amply demonstrated by the fact that :

  - DEC is highly successful, relying on third party systems or software houses for 85% of application requirements

  - Philips and NCR are equally successful, but rely on their own internal applications and support staff for 90% of requirements

  - Data General, pursuing an aggressive pricing policy for its equipment, relies almost entirely on the user for the commercial Eclipse series, and on third party system/software houses for the CS/small computer series.

- There is an equal diversion in the intentions of vendors as to how application requirements will be handled in the future. ICL, whose applications development is spread over the users, third party software houses and ICL's own staff, intend to reduce the need for users to support themselves. Kienzle, on the other hand, will provide a variety of user self-help tools for the design and implementation of their applications. NCR's basic policy is to rely on standard packages for nearly all of their requirements, while Prime prefer not to get involved directly and introduce customers to systems software houses.

E. RECOMMENDATIONS

- Each vendor's situation is different, but all have certain standard applications that can be functionally frozen. These should be treated exactly the same as a hardware product (indeed, if implemented in firmware they become "hard") i.e.

  - taken into account in the long term product strategy of the company for OS/peripheral/processor compatibility and staging of the user base

  - charged for separately on a price/performance basis (most already carry a fee, but there is not, as yet, an industry or even IBM standard for price/performance for applications)

  - be sold, rented or leased

  - carry a maintenance charge

  - be upgraded on a planned basis and/or be replaced by new line products

- Software is still only in its third generation (machine/assembly language, high level language, database/non-procedural language) while hardware has progressed to the fourth generation. This imbalance in the relative performance of hardware and software means that software offers the greater opportunity for improvement.

- Improved software productivity is a key requirement, given the explosion in the demand for user-specific solutions. This is dependent on improved specification of requirements, use of automated logic and specification analysis at the earliest stage of product development.

- The new system design and system programming techniques require a broader understanding of the problem to be solved as well as a most rigorous effort by the project staff to define the system environment and its interactions. Initially this does not improve development time : a 20-30% <u>increase</u> is not uncommon. However production failures become rare, Management must be prepared to trade quality for quantity in these early stages, a step which is to be justified only in the later stages.

- There is every incentive to standardize those applications that are offered by the vendor and to encourage the user to expect to pay either the system/software house or the vendor's support staff for any modification which occur, both as a result of the customers own specific needs and as a result of the vendor-initiated modifications for (i) improved performance (ii) expanded peripheral coverage (iii) expanded functional performance.

- In short, the vendors approach should be that of freezing the functional abilities of this systems at both the hardware, system software and application software levels at regular intervals in the products life:

    - at product launch
    - at upgrade announcements (hardware and software).

SECTION III. NEED FOR, ROLE AND COST OF APPLICATION SOFTWARE

SECTION III. NEED FOR, ROLE AND COST OF APPLICATION SOFTWARE

- The original idea behind the development of application software was an attractive one : reduce the cost of systems implementation by creating a series of standard packages each of which covers a high percentage of the processing requirements of a given application area.

- Since this meant pooling resources that would otherwise have been spent on a large number of individual installations, substantial funds could be dedicated to a single application, broadening its scope and improving its performance.

- The intention is still the same, but the ability to implement this goal has lessened due to :
  - the rapid pace of hardware product introduction, expansion, upgrade and replacement
  - the growing variety of configurations to be supported
  - the variety and variable quality of sytem operating software on which applications rely.

  In addition, local languages, industry sector differences and local requirements all complicate the task of the applications designer.

A. GROWING DEMAND FOR SPECIFIC SOLUTIONS

- Since early 1977, few computer manufacturers have had to worry about market expansion - the vast majority have grown as fast as their own capabilities and resources have allowed. The result, on the applications front, has not been positive, however.

ROLE OF APPLICATION SOFTWARE

- Acts as a catalyst for vertical market identification.
- Enhances user perception of system/solution offered.
- Crystallizes vendor's understanding of user needs.
- Provides a link for the user from one hardware product to the next (from the same vendor).
- Widens vendor's understanding of market opportunities through exposure in selling the application.
- Is a product per se, separate from hardware and systems software.
- Focusses energy of vendor's marketing/sales force.
- Reduces system support costs, by providing standard implementation tools.
- Generates revenue.

Exhibit III-1

- An expanding market has meant the proliferation of small business computer vendors, that are either micro-mini, terminal system or office computer based, or mainframe vendors gradually expanding downwards the small compatible mainframe. In addition, the variety of models available from each vendor has grown, such that the potential user of this level of system has a broad choice of supplier.

- As a result, users have become far more demanding in their requirements: generalized, approximate solutions are no longer acceptable ; specific solutions are demanded. This does not automatically signify custom packages, however, since the source of the solution is not important to the user. If a standard package meets the specification, either as it is, parameterized or modified at source code level, all well and good.

B. STANDARD APPLICATIONS - VALUE, LIMITS AND ROLE

- At least two groups of people can specify standard application requirements for a product : the end user and the vendor. While the need for end-user applications is generally well understood, vendor applications needs are not. Usually vendor needs are covered by little more than utility programs (which, as such, are classified as systems software) ; data file creation and database management, in one form or another, are similarly treated.

- The long term value of end-user to applications lies in their ability to act as a communications tool between the end-user and the vendor. This is illustrated in Exhibit III-1, where the first six items listed all relate to the understanding, by user and vendor alike, of what is needed as opposed to what is being offered.

- Nowhere is this clearer than with the minicomputer manufacturers whose systems are largely implemented (i) with customized software (ii) by third party staff. As a result most of these vendors do not have a clear idea of how their systems are being used :

- a multitude of customized solutions is hard to synthesize
- dealing through third party vendors obscures the end user base.

This makes systems planning difficult, so that most minicomputer vendors have to do their product planning in terms of "more hardware for less cost" criteria.

- Originally hardware manufacturers tried offering "do everything" applications for no charge, as a means of reducing user implementation support costs. The manufacturers were rarely close to user requirements and so the "package" concept failed.

- The unbundling of software by IBM created a software products market overnight in which competition ensured that an increasing level of performance, support, documentation and relevance to user needs is now required of each application package offered.

C. HORIZONTAL VERSUS VERTICAL MARKETING

- In the early 70's a new approach developed - Industry Marketing. At the very least, this is a misnomer, since it is difficult to market software products to a whole industry. Even branch level marketing within an industry sector is frequently too broad.

- In addition, there are many instances where two or more entirely separate sub-branch markets have identical needs and can be serviced with the same package.

- Broadly speaking there are two methods of approaching application software development : horizontal and vertical marketing. "Industry" marketing is an example of horizontal marketing, "cross-industry" (i.e. broad-based needs like payroll, stock-control, Accounts payable/receivable etc) is another example.

HORIZONTAL VERSUS VERTICAL MARKETING

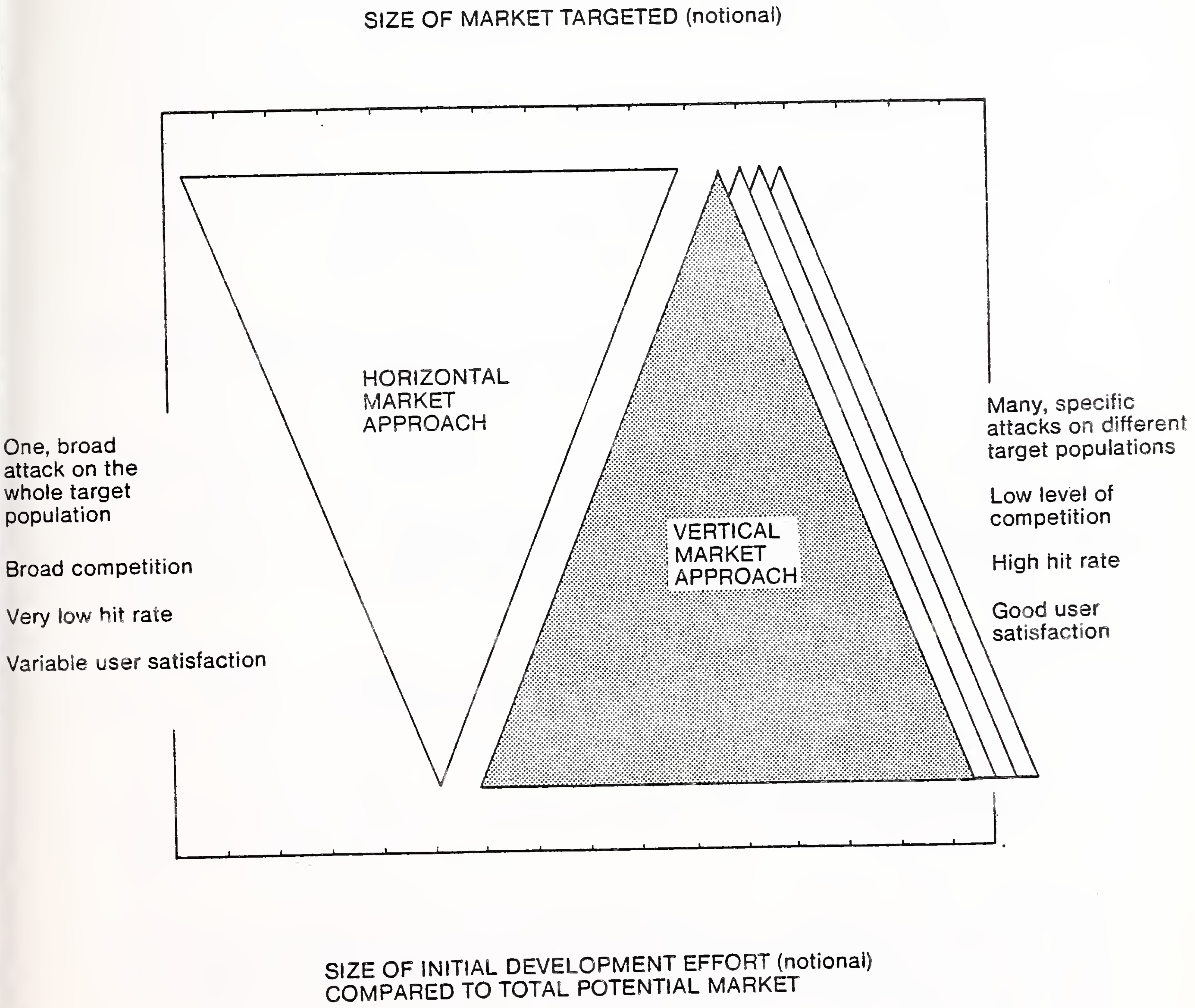

SIZE OF MARKET TARGETED (notional)
HORIZONTAL MARKET APPROACH
VERTICAL MARKET APPROACH
One, broad attack on the whole target population
Broad competition
Very low hit rate
Variable user satisfaction
Many, specific attacks on different target populations
Low level of competition
High hit rate
Good user satisfaction
SIZE OF INITIAL DEVELOPMENT EFFORT (notional) COMPARED TO TOTAL POTENTIAL MARKET

Exhibit III - 2

- In each case, horizontal market packages are easier to identify and require relatively limited development ; (note : exception is made of the multinational packages which will be discussed in section IV A). As a result many competitive solutions are encountered in the marketplace, and this, coupled with the non-specific, "horizontal" approach to a very broad market results in a low hit-rate.

- Vertical market packages are characterised by :

  - a high level of user need specification
  - a large number of separate, potential targets to choose from
  - a high development cost, relative to the total potential market
  - few competitors, once developed.

  All of which results in ease of prospect identification, a high hit rate, good user satisfaction and repeat reference sales.

D. APPLICATIONS AS A HARDWARE SALES SUPPORT-TOOL

- Hardware, in itself, confers no special advantage on a salesman unless it has:
  - unique architecture/modularity
  - price lower than competition
  - a well recognised brand name

- In addition, while salesmen can always be motivated by the attraction of easy sales and quick commission, a small system needs to be distinctive in some other way, if it is to capture the attention of prospect and salesman alike. Application software can do this by narrowing the definition of what is offered (for the user) and specifically targeting the prospective market (for the salesman).

- This ability to focus the energy of the salesman and the attention of the user is best achieved by vertical market products, which offer a unique combination of skills - easily recognised by the prospect and exploited by the salesman.

- However, the horizontal "bread and butter" applications are mandatory for the basic support of any line. The now classic payroll, accounts receivable, accounts payable, general ledger group are generally available from most suppliers. However, a broadly applicable, international stock control package is not easily available due to the combined complication of (i) widely varying industry requirements (ii) national market differences.

E. APPLICATIONS SOFTWARE AS A REVENUE GENERATION TOOL

- While the price of hardware is constantly declining thanks to on-going technological improvements, there are no such technology breakthroughs either available or expected in the foreseeable future in software development. Menwhile manpower costs, the cost of software development are rising.

- As a result, all people-dependent activities such as sytems support, maintenance and implementation represent an increasingly large share of the cost of a user's system, particularly at the minicomputer/small business system level where hardware prices have dropped by half over the last three years.

- Generally speaking it is good pricing sense to match revenues with costs, so that applications software prices should rise over the next few years as will application development costs. Under these circumstances it would be adviseable to treat Applications Software not only as revenue generation tool but also as a cost centre for control purposes.

TYPICAL APPLICATION DEVELOPMENT COSTS
(R & D, Specification, Programming and Documentation)

ITEM	Actual values LOW	 HIGH	Average of those Interviewed LOW	 HIGH
• Application Development Effort	1 man/month*	25 man/years	8.8 man/ month	12.4 man/yrs
• Cost ($M)	0.2	1.9	0.35	1.2
• Total Appl. Dev. Budget/year ($M)	1.5	10.5	1.5	6.7
• % of total company sales	0.01	0.02	0.01	0.02

Notes : - * customizing an existing application

- all other figures are for developing new applications from scratch

Exhibit III-3

- Quite apart from this cost-control concern, there is a further, more fundamental problem to be solved : where will the manpower come from for the implementation of the increasingly high volume of minicomputer and small business systems sold each year ?

- Already the resources of the third party systems/software house are being brought into play, but these resources are limited. Ultimately there is only one manpower resource large enough to handle the requirement: the users themselves.

- While standard applications, (whether fixed-function or parameterized, customized by re-compilation etc) will absorb a significant share of the total requirement (estimated at between 25 and 40%) an increasing proportion of new system sales will be implemented by one-off solutions.

- The user must therefore be provided with the tools for producing his own applications (see section V.C/D/E), the development of which is a long, painstaking process. Planning for these tools in the immediate future would therefore appear to be a reasonable precaution against a shortfall in system implementation resources.

F. TYPICAL APPLICATION DEVELOPMENT COSTS

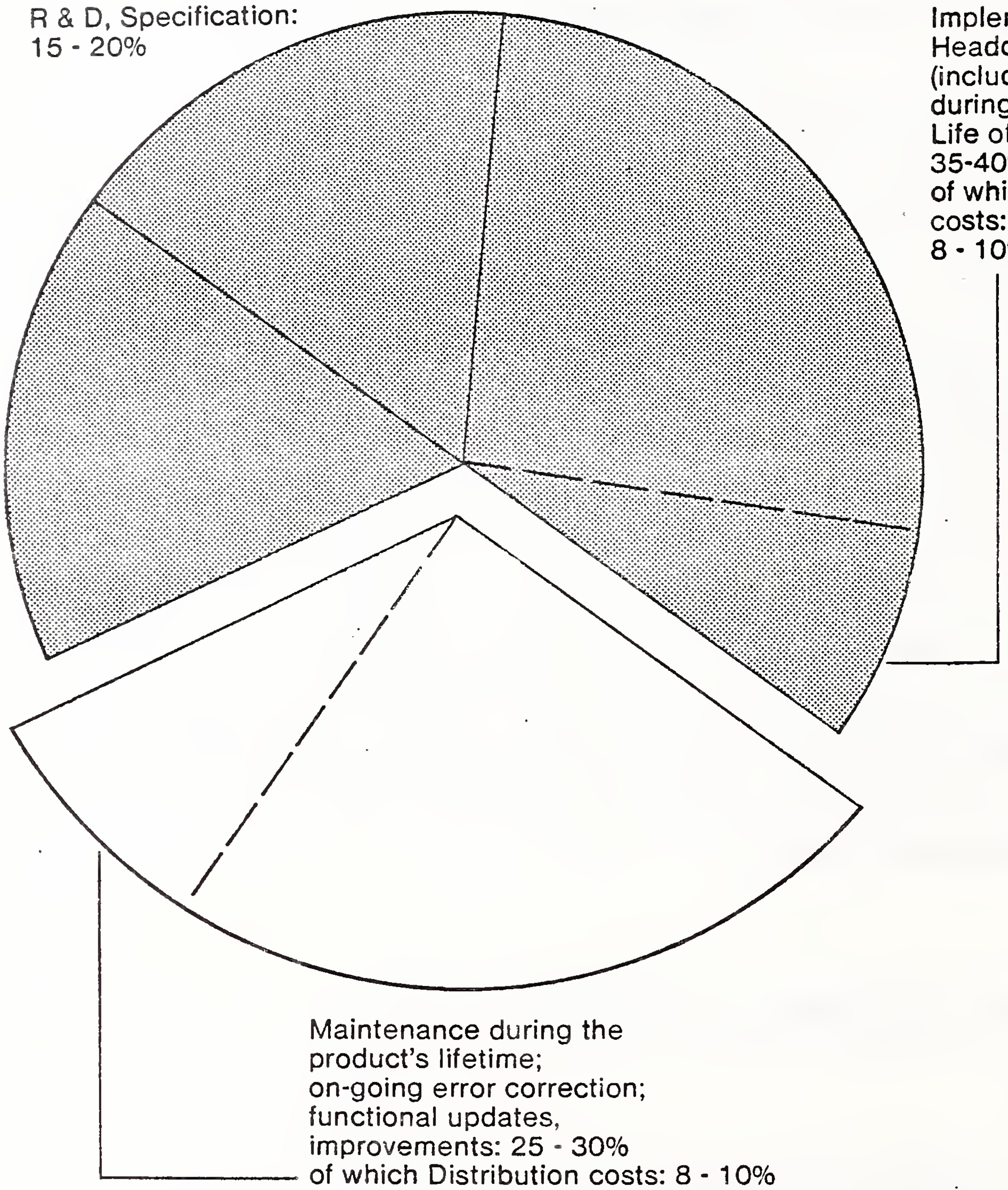
- Quite obviously there is no single value or average for the development of an application package, given the enormous variety of requirements. In addition, none of the companies interviewed know their own average. However, most had a range of low and high values that were used internally, (although INPUT has no way of verifying whether these are accurate).

- The results are listed in Exhibit III-3 in the forms most frequently used:

  - man/months or man/years
  - actual dollar cost (or dollar equivalent)

ESTIMATED AVERAGE APPLICATION COST BREAKDOWN

OVER PRODUCT LIFE-TIME



R & D, Specification:
15 - 20%

Implementation and
Headquarters support
(include training)
during Commercial
Life of product:
35-40%
of which Distribution
costs:
8 - 10%

Initial programming
development debug
and documentation:
15 - 20%

Maintenance during the
product's lifetime;
on-going error correction;
functional updates,
improvements: 25 - 30%
of which Distribution costs: 8 - 10%

TOTAL APPLICATION PRODUCT
LIFETIME COST ESTIMATE: $3.2 - 4.2M

Source: Estimates provided by
small number of vendors.

Exhibit III - 4

- current total applications development budget
- the % of total company sales this represents

- Comments from the vendors interviewed on applications development include :

  - "This is a growing area of investment for our corporation".
  - "The bulk of the development work is done centrally and adapted for use by each country by local staff".
  - "We sell through OEMs mainly, so application development represents a small proportion of the whole R & D effort; each OEM produces majority of software in that country".
  - "On average a team of five specialists is used for an application development ; but on larger systems the total cost can rise to $1.9M".
  - "Demand is increasing ; we spend $10.9M each year on software development and up to $1.3M for an application ; this is higher than some hardware product development costs".

G. MAINTENANCE - THE BIG UNKNOWN COST

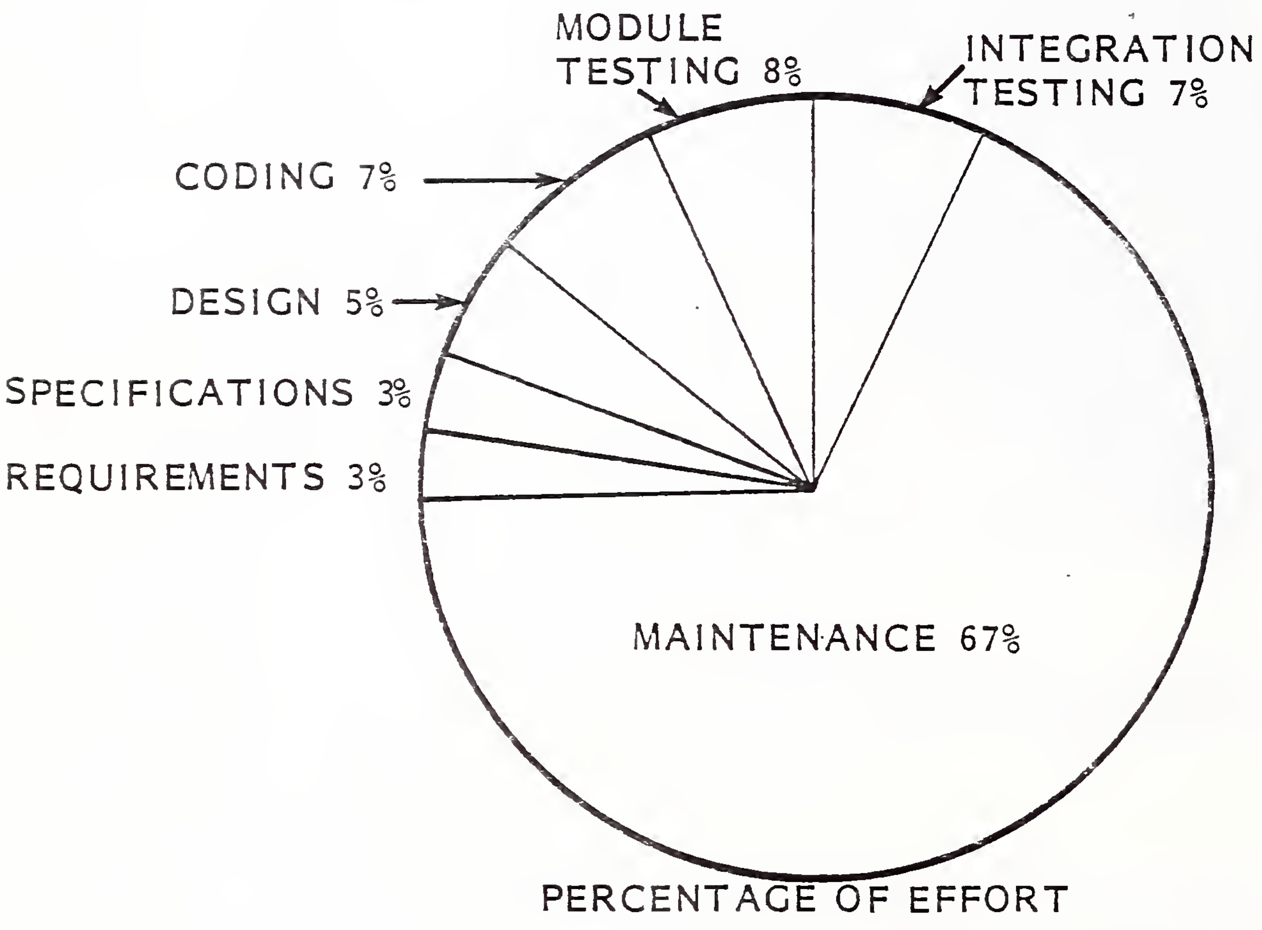
- Whereas nearly all small business system vendors can evaluate, from product inception, the development, tooling, manufacturing and maintenance costs of their systems hardware, model by model and even country by country (where relevant), few appear to be able to do the same for their applications products.

- When pressed, some can produce approximate costs for development and programming for a given package but very few can begin to approximate the maintenance cost, particularly if the product is a mature one. Yet all agree that the cost is high.

- As with hardware products, this is a fundamental issue. Few manufacturers would argue with the conclusion that, as the maintenance costs of a product

EXHIBIT III-5

TYPICAL ALLOCATION OF EFFORT BETWEEN NEW DEVELOPMENT AND MAINTENANCE OF SOFTWARE



SYSTEM LIFE CYCLE OF FIVE YEARS OR MORE

exceed its maintenance revenue, it should be replaced as soon as possible (and dropped from the catalogue). Without knowledge of this cost for each application, how can logical decisions be made ?

- Some "guestimates" are provided in Exhibit III-4, based on vendor's personal convictions rather than actual monitored data. These suggest that product maintenance creates over one quarter of the total cost of an application during its life time, or $0.8M to $1.26M.

- This also suggests that, if revenue is to balance cost in all areas of an application, then maintenance should represent 25-30% of total applications revenue, and charges to end-users adjusted accordingly.

- Early this year INPUT conducted a survey of large user installations in the U.S. to ascertain where the user's software costs occurred in the lifetime of the business application systems over a useful life of five years or more. The results show that over two thirds of the effort occurs after the systems development is completed. The new design and programming techniques are aimed at this cost balloon. (See Exhibit III-5).

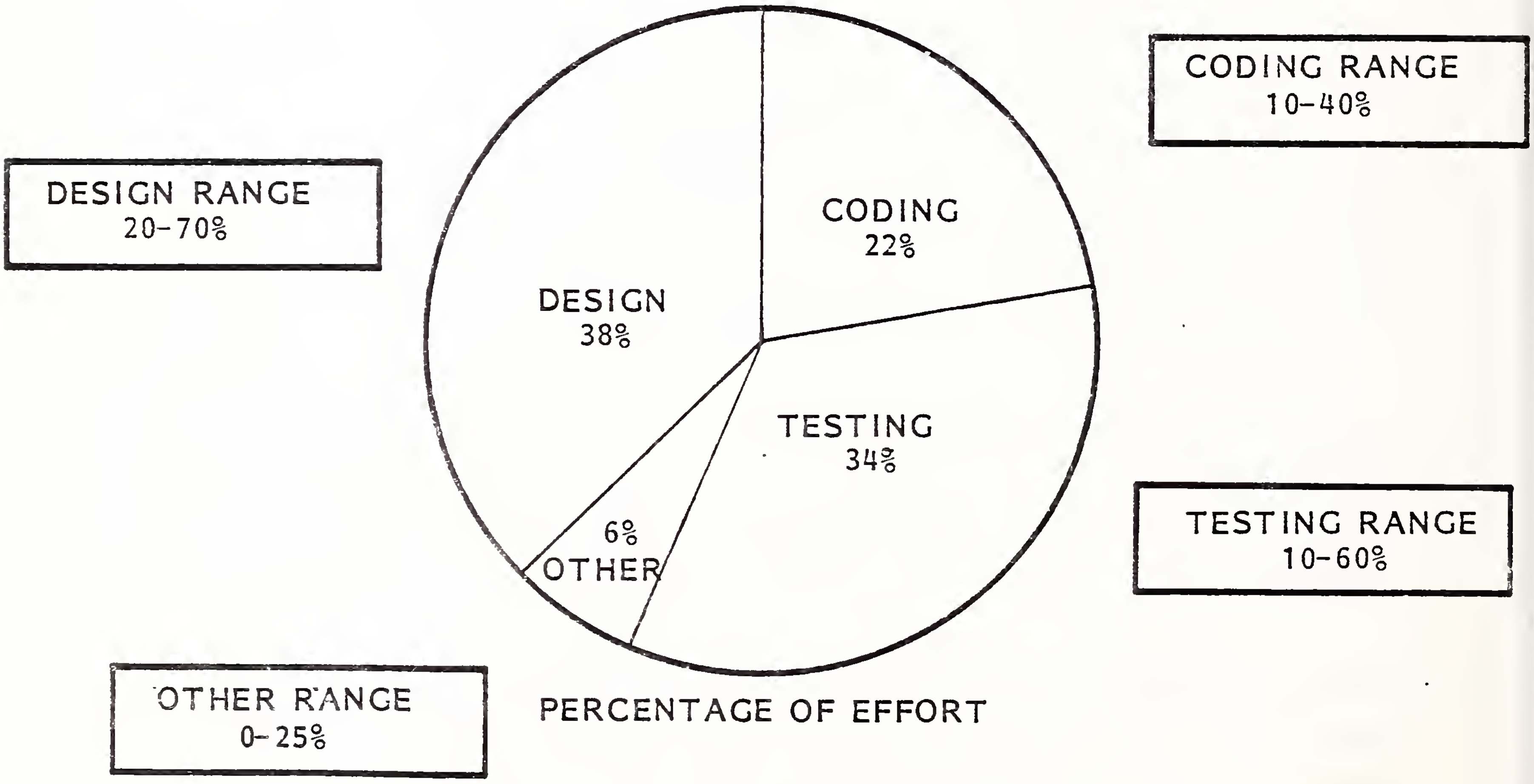
- This also suggests that vendors' estimates of the maintenance cost of their own systems are either optimistic or that vendors have discovered design methods they have not passed onto their users. However, both users and vendors agree that forecasting maintenance costs is very difficult.

- Another significant analysis to emerge from the U.S. user survey is given in Exhibit III-6. This is the chart that users and vendors alike recognise as their rule of thumb. The key point to remember is that this concerns only the development cycle, i.e. the most visible stage, cost-wise, of an applications development. Thereafter, cost monitoring usually ceases.

- Finally, whereas the U.S. users had a fairly clear idea of how man days, were spent in the maintenance cycle, the European vendors had little idea. Frequently conversion, adaptation and enhancement are activities which run

EXHIBIT III-6

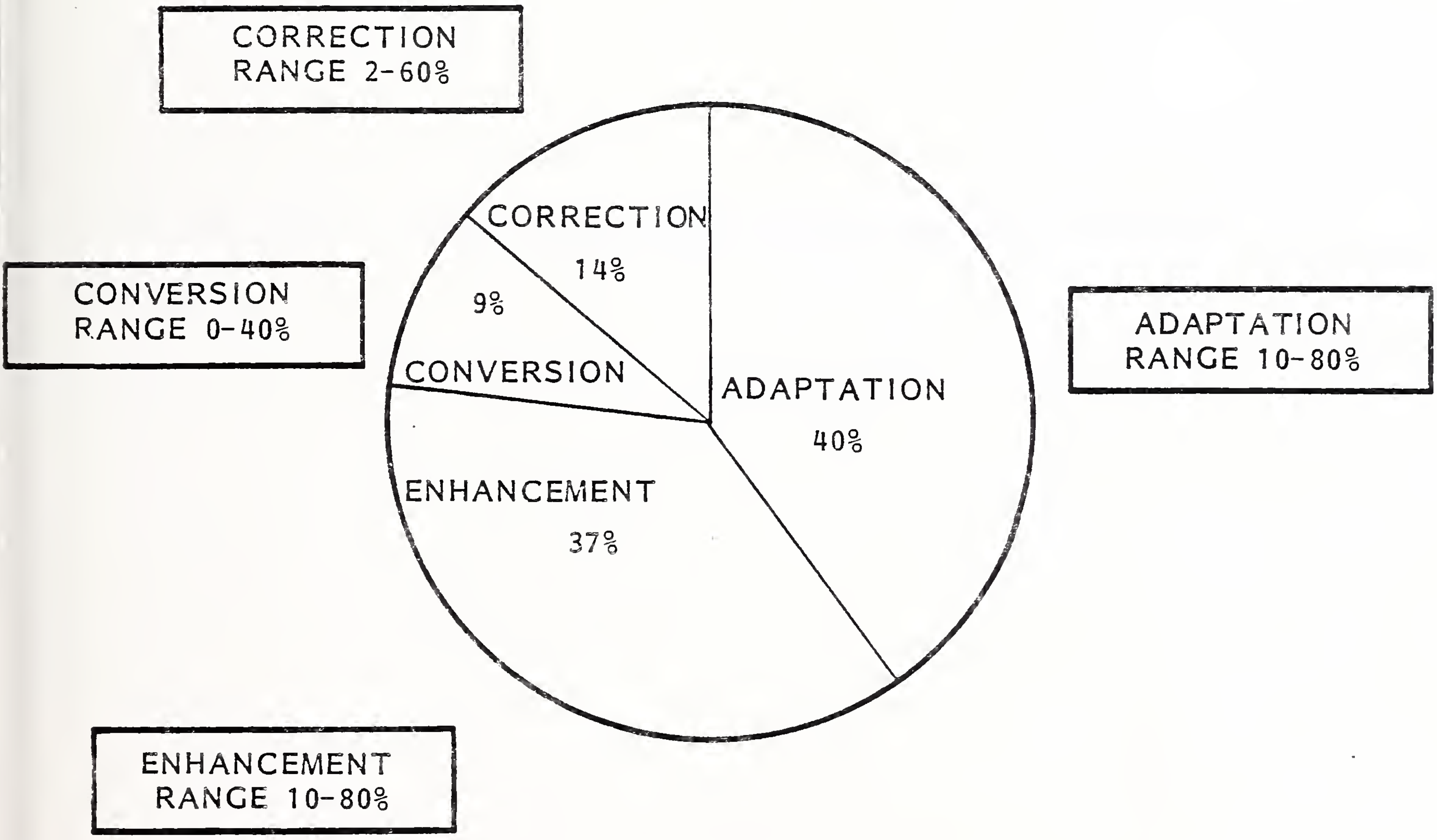
RESPONDENTS' ALLOCATION OF EFFORT WITHIN SOFTWARE DEVELOPMENT CYCLE



NEW SYSTEM DEVELOPMENT

EXHIBIT III-7

RESPONDENTS' ALLOCATION OF EFFORT WITHIN SOFTWARE MAINTENANCE CYCLE



PERCENTAGE OF EFFORT

in different cost centres, even different product lives than those in which the initial application development began. This is as it should be, except that the product's cost files rarely follow it through those cycles.

SECTION IV SYSTEMS DESIGN AND PROGRAMMING TECHNIQUES

SECTION IV SYSTEMS DESIGN AND PROGRAMMING TECHNIQUES

- Treating application software as a source of revenue by pricing it separately places two constraints on it :

  - Functional performance must be competitive and must fulfill hardware support requirements

  - Development and maintenance costs must relate to the revenue the application can generate

  For both of these reasons there is a need for improving productivity of software development and ensuring that costs are closely controlled.

- Performance improvement in software development requires :

  - the isolation and identification of potential problems at the <u>beginning</u> of the software development cycle, where they are easier and less costly to fix or prevent

  - trading increased design effort (and possibly increased short term operational costs) for lower <u>lifetime</u> systems costs, stemming from lower maintenance requirement and effort.

- The objective is clear, and the strategies are straightforward. Unfortunately, whether the results will justify the costs cannot be determined accurately until enough years have passed to measure any reduction that may have occured in maintenance costs. For many organizations, this point only occurs after two to three years.

- The problem that the software designer/manager faces is how to define program performance in terms of competitiveness and in relation to the price the vendor intends to place on the final product. There is no established guideline as to how price and performance relate in a given application area, and even in those instances where vendors feel they know "what users will stand", their knowledge is usually based on the history of what they themselves, and often in the same field, have done, rather than on specific market research on the prospective user base.

- Nevertheless, almost anything that constrains design and programming into more disciplined and rigorous channels has the potential to produce some improvement in performance, and certainly controls if not constrains costs.

A. SYSTEM DESIGN DISCIPLINES

1. TOP-DOWN DESIGN

- Top-down design as a concept dates back to the mid 1960s and was first proposed by Larry L. Constantine under the title of "Structured Design". It was an outgrowth of modular programming that took this concept back to its logical beginning. The intent was to formalize and simplify the design of complex systems by decomposing them into manageable subsystems.

- Other theoreticians and researchers joined in the search to strengthen this process and within five to ten years a number of variations on the top-down design theme had appeared.

- As the name implies, the top-down procedure begins to consider a system at the highest level of generality and consciously defers making any decisions about implementation details until the latest possible stage in the design process.

- In a simple business data processing system, this procedure would identify the first level of component systems as :

- Initiation.

- Edit.

- Update.

- Report.

- Termination.

- Each of these subsystems in turn would then be broken down into its subcomponents producing a hierarchically structured design that considers increasing levels of detail as one proceeds lower down in the hierarchy.

- This design can then be implemented and tested in the same manner by using higher level modules to call the lower level modules which initially simply return the desired output without performing any processing or transformation of data.

- Once the designer is certain that the highest level of the system is performing its task according to specifications, the next lower level of design can be implemented.

- The process continues in an interactive manner until the lowest level of detail function has been implemented and tested. At this point, there is no need for further testing since the whole system has been integrated as it was developed.

2. BOTTOM-UP DESIGN

- Contrasted with the top-down strategy, method of design is to proceed in the opposite direction or "bottom-up".

- This method considers the production of a number of detailed components which may be subroutines or larger units that are then collected into increasingly more comprehensive functional systems.

- Under this strategy, entire functions may be completed before other functions are begun or even before their design is completed.

- The underlying strategy is one of vertical integration, rather than horizontal integration. The process thus defined tracks more closely the "traditional" sequence of design, code, unit test, systems test.

3. FUNCTIONAL ABSTRACTION

- Addressing a different dimension of the design process, and especially useful for those situations where splitting functions produces programme components that are still too complex to be easily understood, is the principle of abstraction.

- Abstraction attempts to simplify systems complexity by considering only a limited amount of detail at one time.

  - this principle is implemented by (temporarily) eliminating from the set of design factors those attributes of events or objects that are not relevant to a given situation.

- The abstraction principle is particularly applicable to the analysis of data and data structures. It thus leads to a theory which states that data structure should define program structure. This theoretic construct has been combined with a general top-down design approach to produce techniques such as the "Jackson Methodology" and the "Logical Construction of Programs," also known as the Warnier-Orr approach.

4. DIFFICULTIES

- At the root of top-down principle and the data abstraction principle is the hypothesis that program correctness can be "proved" algebraically by establishing the proof of the smallest component structure and logical constructs, and then by proving the correctness of their collation into systems.

- Considerable progress has been made in establishing the formal proof of certain functions and structures, although the proof is likely to be much longer than the implementation itself. However, many other functions have not yet been proved, and there is some question as to whether they are provable.

- Pragmatic considerations also work against a rigorous application of some of the decomposition and abstraction principles.

  - none of the vendors in this study use a single methodology for all of their new design applications, but most had a preferred technique or combination of techniques that they employed more frequently in their own organization.

  - strict application of the top-down approach does not take into account the possible later identification and combination of common low level routines. Since it consciously defers decisions on detailed implementation until as late as possible in the design process, the top-down approach cannot adequately deal with the existence of low level routines whose implementation involves a high risk factor.

  - top-down designing does not usually produce the best result on the first attempt. Respondents report having to redo an entire design once the total system implications became clear.

- The top-down design process is a lengthy one and frequently results in higher, not lower operating costs. However, operating costs for many systems are not the critical constraint, since they decline due to reductions in hardware costs, (while the cost of software maintenance increases due to increases in salary and benefits).

5. SOME U.S. RESULTS

- The vendors interviewed uniformly had great difficulty in assessing the practical value of structured design techniques. The results produced here

EXHIBIT IV-1

RESPONDENTS' RATING OF EFFECTIVENESS OF STRUCTURED DESIGN TECHNIQUES

CRITERION	EVALUATION
● STAFF MORE PRODUCTIVE?	* * *
● TESTING/DEBUGGING MORE EFFICIENT?	* * ½
● MORALE BETTER?	* *
● DOCUMENTATION BETTER?	* * ½
● DEVELOPMENT COST LOWER?	* ½
● OPERATING COST LOWER?	- 1
● IMPLEMENTATION FASTER?	* ½
● QUALITY BETTER?	* * *
● MAINTENANCE EASIER, LESS COSTLY?	* *
● RESPONSE TO USER REQUESTS MORE FLEXIBLE, TIMELY?	* * ½
● USER COSTS LOWER?	½
● USER INTERFACE EASIER?	* * ½

EVALUATION KEY

* * *	VERY MUCH
* * ½	MUCH
* *	SOME
* ½	A LITTLE
½	BARELY
- 1	WORSE

relate to INPUT's study of large users, referred to earlier (see Exhibit IV-1).

- None of the respondents felt that structured design techniques per se had made the largest or the critical contribution to their overall performance improvement.

  - In most cases the specific contribution of structured design was difficult for respondents to separate from the contribution made by structured programming since these techniques were usually used in tandem.

  - The relative contribution made by structured walk-thru or design review is generally held to be higher and easier to achieve than that made by structured design.

- This differential in effectiveness can be attributed to the stronger requirement for training and experience that is attached to structured design and programming, whereas structured walk-thrus require almost no prior training or experience to be successful.

- A further reference to Exhibit IV-1 reveals that respondents regard improved quality and overall productivity as the highest benefits achieved through structured design. Additional side benefits include better staff morale and easier maintenance.

- As to lower costs or faster implementation resulting from structured design, most respondents feel these effects are neglibible or even negative. But total development costs may be slightly lower, and total life cycle costs are expected to be considerably lower.

- Comparative costs between traditionally designed systems and structured design systems are not available, mainly because it is not considered worth the effort to determine these costs.

B. STRUCTURED DESIGN TECHNIQUES

- In the ten years or so that have passed since structured design was first proposed as a concept, at least half a dozen variations have been proposed, each emphasizing a different aspect of the design problem.

- All relate to the premise that environments and systems change over the course of their lifetimes. Systems which have been designed to adapt more easily to these changes manifest "good" design.

- "Good" design produces systems which are highly robust (that is, they can correctly process a wide variety of input data, both true and false, without impairing the system). This objective is not often met in complex environments because of the difficulty of anticipating all aberrant types of data and sequences of events.

- A summary of characteristics of the major design alternatives or aids is shown in Exhibit IV-2. Each is discussed in more detail below.

1. STRUCTURED DESIGN

- The original proponents of structured design, Constantine, Myers, and Stevens, elaborated a set of guidelines, naming conventions, documentation formats, and analysis techniques which endeavour to follow the flow of data through a system and identify each transformation in sequence.

- Advantages of the technique are :

  - It aids in the definition of data flows

  - It produces improved documentation

EXHIBIT IV-2

CHARACTERISTICS OF MAJOR DESIGN TECHNIQUES

DESIGN TECHNIQUE	CHARACTERISTICS
SADT-STRUCTURED ANALYSIS AND DESIGN TECHNIQUE	Systematic decomposition of modules. Diagramming technique to show component parts, their interfaces, and hierarchic structure. Teamwork. Written comments.
STRUCTURED DESIGN	Data flow and transformations through a system define the system. Top-down, iterative approach. High modularity, careful definition of interfaces and boundaries.
JACKSON METHODOLOGY	Data structure defines program structure through resolution of input and output structure's. Most useful with serial files.
LOGICAL CONSTRUCTION OF PROGRAMS-WARNIER-ORR METHODOLOGY	Structuring input and output data in bracketed, hierarchical formats. Naming conventions, instruction sequences produce pseudocode which is easily understood and translated into program code.
META-STEPWISE REFINEMENT	Starts with multiple simple solutions, successively adding detail and retaining best solution on each iteration to produce level-structured, tree-structured programs.
HIPO-HIERARCHY PLUS INPUT-PROCESS-OUTPUT	Hierarchically oriented graphic technique for defining functions and sub-functions, and separating control flows from data flows.
PSL/PSA-PROBLEM STATEMENT LANGUAGE/ PROBLEM STATEMENT ANALYZER	Formal, automated language for describing a system in terms of objects, properties, and relationships to produce specifications. Formally proves adequacy and consistency of problem definitions.

- It works best in situations where input and output can be clearly distinguished from each other.

- Disadvantages of the technique are :

  - It is simpler in appearance than in actual practice

  - It does not necessarily produce a satisfactorily detailed specification at each level of the hierarchy.

2. SADT - STRUCTURED ANALYSIS AND DESIGN TECHNIQUE

- This variation of how to functionally separate programmes into tasks was developed and is taught by SofTech, Inc. It includes analysis techniques and a linking process to associate those techniques to produce a graphic, layered model of a system. Each layer of the model defines its component parts, their interfaces and relationships, and the events which occur at this level.

- A graphic convention comprising boxes and arrows assists in relating layers of the model to each other in an exact logical representation of the system.

- Data decompositions and event decompositions are cross-referenced to assure that the use of individual elements is both consistent and complete.

- Emphasis is placed on the interdependence of the seven basic design concepts, the organization of the project team, and the documentation process that describes the defined system.

- Advantages claimed for the method are :

  - The written comments and documentation techniques which provide an up-to-date record of the status of the system.

- The ability to integrate the method with an automated problem definition language.

- Disadvantages are :

  - The requirement to use all aspects of the method to gain reasonable benefits from the technique.

  - The necessity to save all versions of all documentation since each includes certain written comments which may later be apropos.

3. JACKSON METHODOLOGY

- This technique was originated in England by Michael Jackson and is now taught by Infotech and others. It is based on the premise that data structures should define program structures. Since input and output data structures are not alike, it is their resolution which defines the program.

- Because the principles for resolving data structure differences are very specific, the program designs which result should be the same for each analyst who designs them.

- Advantages claimed for the method are :

  - It is easy to understand and to learn.

  - It produced uniform results no matter who applies the technique.

- Disadvantages are :

  - If the data are not well structured to begin with, the programs will not be well structured either.

- Non-sequential data files are difficult to handle.

- Error processing is not provided for within the normal structures.

4. LOGICAL CONSTRUCTION OF PROGRAMS

- Also known as the Warnier-Orr technique, LCP was originated in France by Jean-Dominique Warnier and promoted in the U.S. by Kenneth T. Orr.

- It is similar to the Jackson Methodology in assuming data structures define program structures.

  - The emphasis of LCP is placed more on the sequence of procedures and tasks.

  - Unique graphical symbols (a set of nested brackets) are used to order both the data and their associated tasks.

  - Interpretation of the resulting diagrams produces a language -independent pseudo-code that is easily translated into any common programming language.

- Advantages of the method are :

  - It rapidly produces program designs for data that are already hierarchically structured.

  - Resulting diagrams are easily interpreted directly in pseudocode.

- Disadvantages are :

  - Data that are not already hierarchically structured end up in a contrived hierarchy that may be logical, but not resemble the actual structure used.

- The method does not address such topics as file access methods or operating environments and, thus, only covers part of the system designer's concerns and responsibilities.

5. META-STEPWISE REFINEMENT

- This technique flows from concepts developed by Wirth Dijkstra, Mills, Schneiderman, and Ledgard. It is an attempt to formalize the iterations that always occur in systems design by successive addition of detail on each iteration.

- Initial solutions are kept simple, structured by levels.

- Sewveral independent solutions are developed at each iteration, but only the best one is kept for the next round.

  - Details are postponed until the later iterations.

- Advantages of this method are :

  - It produces elegant, well-refined designs.

  - Results are strictly tree-structured, with no lower level module calling a higher level module.

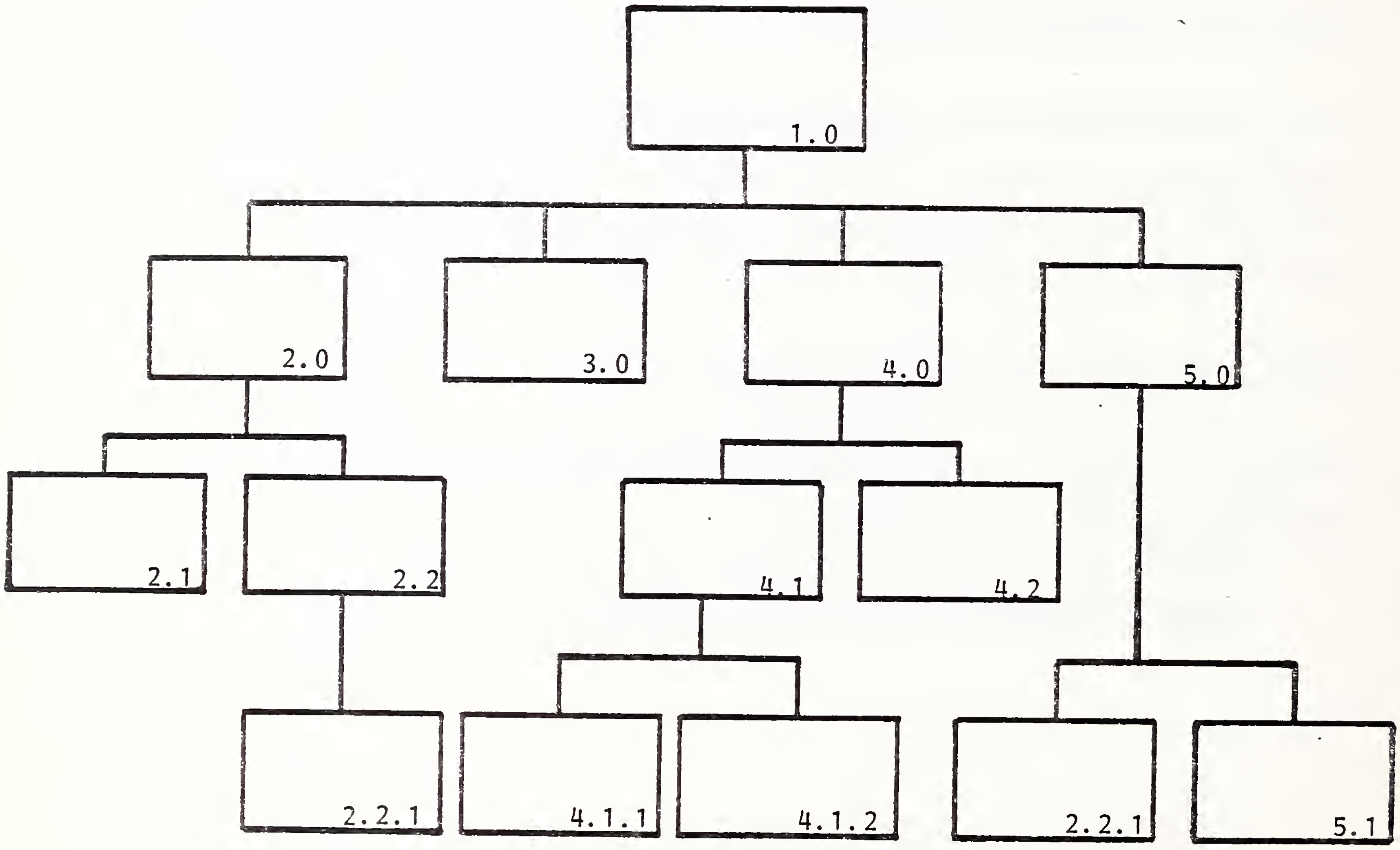
- Disadvantages are :

  - Few analysts can produce fundamentally different solutions at each iteration.

  - Few problems are so sufficiently stable and well-defined at their

EXHIBIT IV-3

THE "H" CHART OF HIPO



outset that later stages do not require retroactive changes to earlier stages.

6. HIPO - HIERARCHY PLUS INPUT-PROCESS-OUTPUT

- More a documentation technique than an independent design alternative, HIPO was developed and is taught by IBM as part of its set of Improved Programming Technologies.

- This method provides two distinct phases :

  - A "visual table of contents" (the hierarchy, or "H" part of HIPO - see Exhibit IV-3).

  - An "Input-Process-Output" (overview) chart that shows the transformation of data from input to output.

- Specified symbols and formats are used in detail diagrams and tables to define data flow separately from the flow of control. Development of the design proceeds from the top down, and from the more general to the more specific. Numbering and naming conventions are used to relate levels to each other.

- Advantages of the method are :

  - It provides a specific framework to relate detailed diagrams to the more general overview, since each box on the "H" chart represents an entire "Input-Process-Output" chart.

  - Its inherent rigorous structure is an aid to comprehensive analysis of each component function of a system.

- Disadvantages are :

- For complex systems, the charts become cumbersome.

- Keeping charts updated with changes that occur later in the design process is awkward and time consuming.

- There is no obvious indication within the method of the sequence in which tasks are performed.

7. PSL/PSA - PROBLEM STATEMENT LANGUAGE/PROBLEM STATEMENT ANALYZER

- Developed by the ISDOS Project at the University of Michigan, this tool provides a computer-aided capability to build, analyze, and update a cumulative data base of systems specifications.

- The PSL portion provides a rigorous syntactic capability to build a model of a system by naming and describing objects and relationships in a conceptual manner, independent of any particular programming language or computer hardware.

- This information about a system may then by processed by the PSA to determine similarities, redundancies, logical conflicts, and gaps in the systems specification, as well as to provide a historical record of any modifications that are made to the system.

- Advantages of the method are :

  - It uses the power of the computer to perform much of the tedious and error-prone cross-checking of specifications.

  - It becomes cumulatively more effective and valuable as the number of specifications entered into the data base increases.

  - It relieves the project team of much of the clerical effort associated with documentation of a system.

- Disadvantages are :

  - It is presently limited to the description of functional requirements and specifications and must, therefore, be used in conjunction with some other procedural analysis technique.

  - It is rather complicated to use, requiring a three day training course plus three to six weeks of use to become proficient.

C. STRUCTURED PROGRAMMING TECHNIQUES

- While there may be disputes about who initiated structured analysis and design, there is no dispute that structured programming should be attributed to Edsger W. Dijkstra in his well known attack on the "GO-TO" statement in 1968.

- Subsequently, in 1972, Dijkstra, Dahl, and Hoare published their volume entitled Structured Programming which elaborated upon the thesis that a restricted number of control structures were sufficient to traverse any flowchart path.

- The elimination of the "GO-TO" statement from all program codes is not the real objective, however.

  - Neither does the imposition of a restricted set of control structures on existing program codes automatically produce improved programs.

- Improvement in programming is derived as much (or more) from improvement in the programming environment as from improvements in programming structure and style. In pragmatic terms, much greater short term

EXHIBIT IV-4

USE OF PROGRAMMING IMPROVEMENT TECHNIQUES

	EXTENSIVE USE	SOME USE	LITTLE USE	NO USE
CHIEF PROGRAMMER TEAM	▨	□	░	▒
AUTOMATED SOURCE PROGRAM CONTROL	▩	□	□	▨
STRUCTURED CODING	▩	▨	░	▩
TOP-DOWN IMPLEMENTATION	▩	▨	□	▩
STRUCTURED WALK-THRU	▩	░	□	▩
DECISION TABLES	▨	░	□	▩
PROGRAMMING STANDARDS	▒	□	□	▨
ONLINE CODE DEVELOPMENT (TSO, ROSCOE)	▒	░	□	▨
OUTSIDE COMPUTER SERVICES/CONSULTANTS	▨	░	□	▩
PROGRAM OPTIMIZERS/PRE-PROCESSORS	▩	░	□	▩

▒ MORE THAN 2/3 OF RESPONDENTS

▩ 1/3 TO 2/3 OF RESPONDENTS

▨ LESS THAN 1/3 OF RESPONDENTS

░ RESPONDENTS

□ NONE

returns can be achieved from changes in the programming environment, e.g. when using TSO or another on-line development technique. Other environmental factors, such as the organization and procedures used to communicate within the project team, also have a large impact (positively and negatively) on programming productivity.

- Exhibit IV-4 reports on U.S. users' use of ten techniques or aspects of programming improvement.

- The most frequently used techniques, both in terms of the number of organizations using them as well as the extent to which they are used within those organizations are :

  - Programming standards

  - On-line code development.

- The least frequently used technique, judged by the same criteria, is the "chief programmer team". Certain techniques (for example, structured coding) are much more powerful when used in conjunction with related techniques (top-down implementation, structured walk-thru). Others, such as the use of programming standards or on-line code development, can clearly stand on their own.

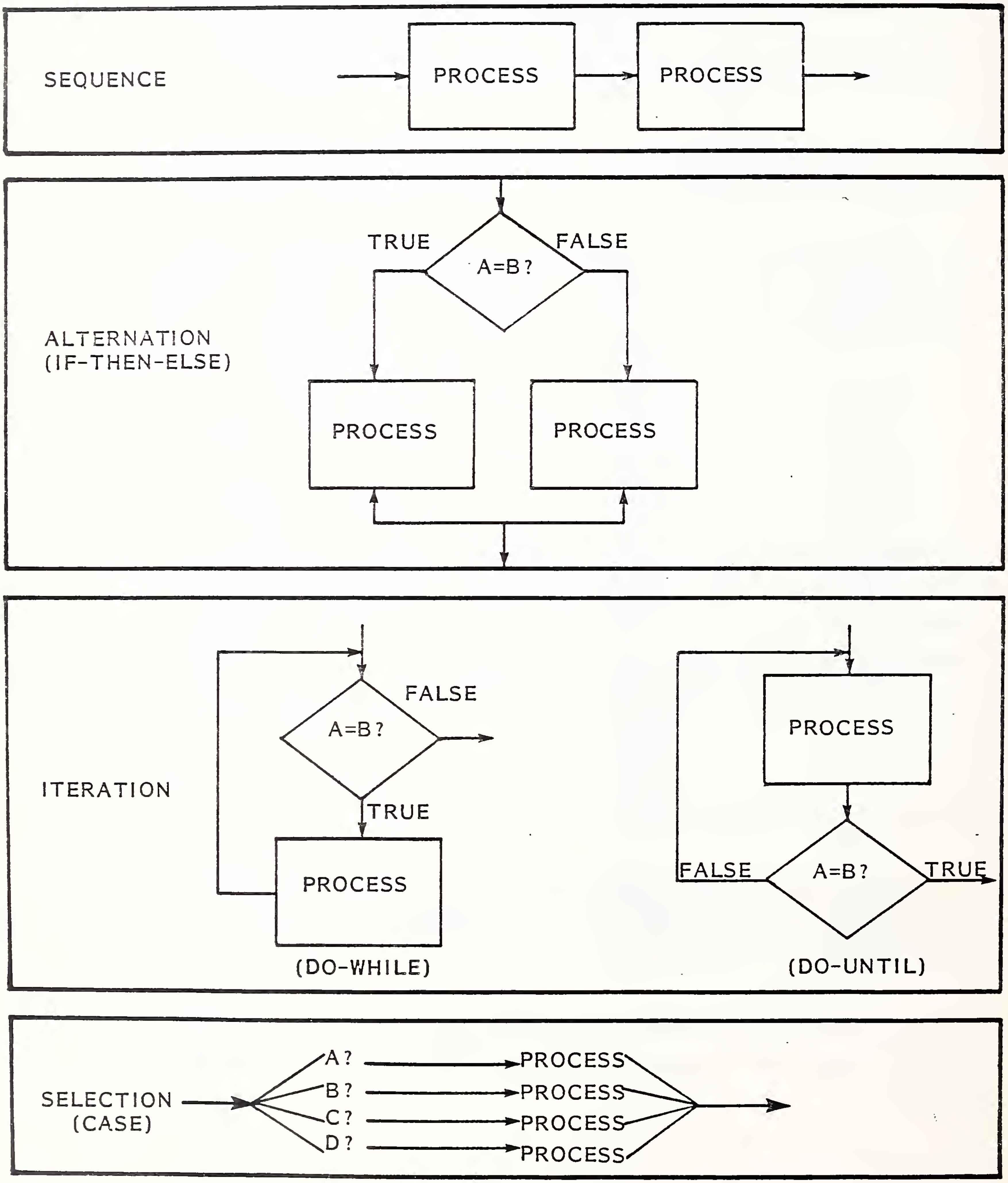
1. SYNTACTIC STRUCTURE

- Before there was structured programming, there was the concept that good programming practice included using the principles of subroutines and modularization.

  Each subroutine or module performed a single task or a small set of tasks and was invoked when necessary from some higher level or "mainline" routine.

EXHIBIT IV-5

STRUCTURED CODING ELEMENTS



- Researchers who were examining the question of whether it was possible to develop a formal proof of correctness of certain common processing algorithms were able to demonstrate that only three basic syntactic structures are essential :

    - Composition, or sequence.

    - Condition, or alternation.

    - Repetition, or iteration.

  A fourth structure, Case, is a variation of Condition which involves multiple exclusive alternatives. Illustrations of these basic control structures appear in Exhibit IV-5.

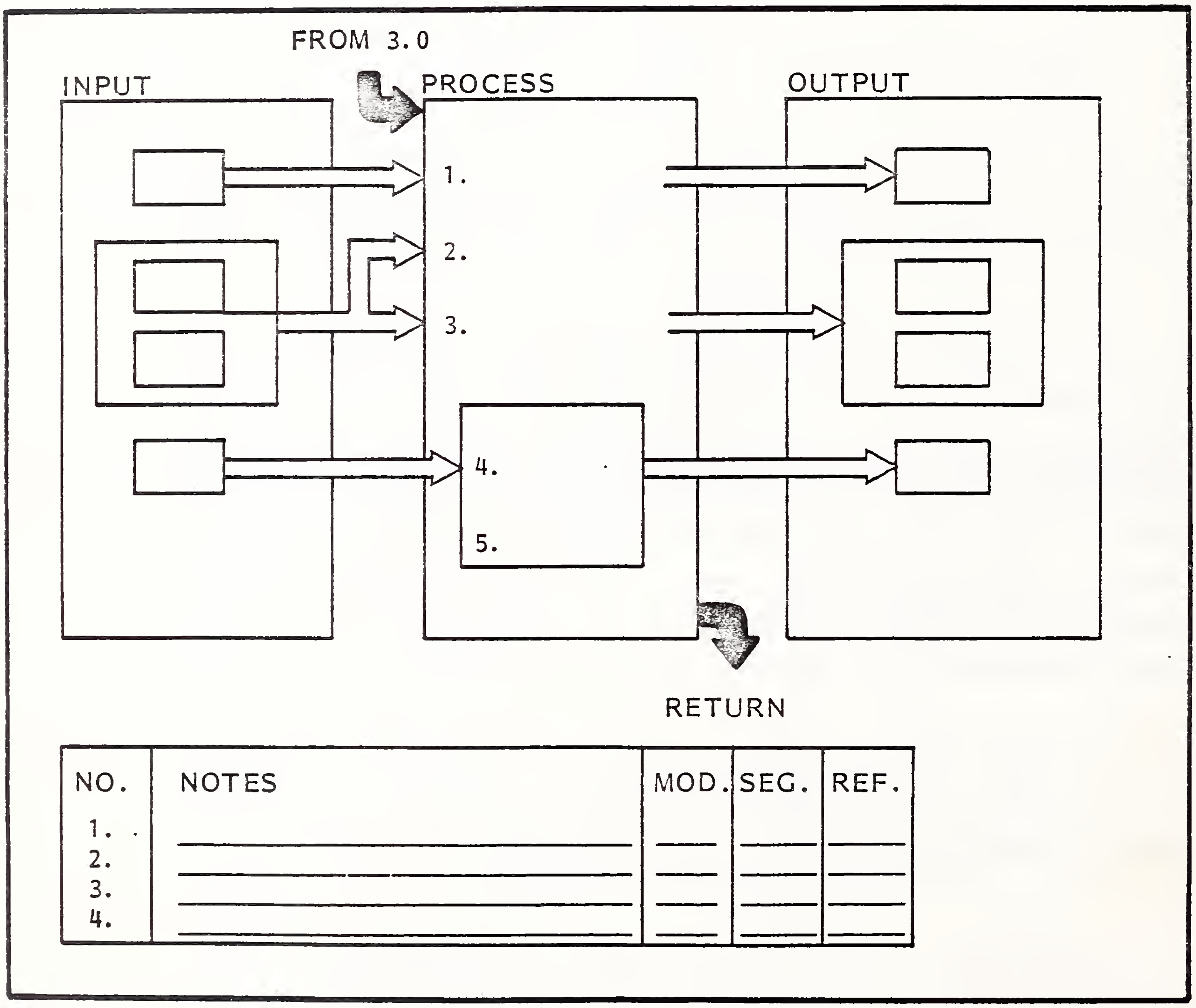
- An earlier section discussed the organization of the "H" chart of HIPO which describes the overall logical organization of a system and its component programs. Exhibit IV-6 is an example of the "Input-Process-Output" chart which comprises the other half of HIPO. The value of the "I-P-O" chart is that it clearly distinguishes the flow of program control (the solid black arrows) from the flow of data (the hollow arrows). The "Extended Description" box at the bottom of the chart elaborates upon the processing steps listed in the Process box.

- The choice of processing algorithm described by the "I-P-O" chart (or by any other appropriate technique) should ideally be a choice of an algorithm and not just a sequence of steps to solve the problem.

- Unfortunately the inertia of many programming organizations and the tendency of programmers to "re-invent the wheel" combine to inhibit the most effective implementation of structured programming. The most frequently raised objection to a rigorous application of all of these structures and principles is that "they lead to inefficient code".

EXHIBIT IV-6

THE "IPO" CHART OF HIPO



- To a large extent, these criticisms belong more properly to the programming language in which the constructs are implemented. COBOL and the various data base languages are not structured languages, yet they are by far the most frequently used languages. It is the critical "3%" of program code that most affects the operating time of a program, and rewriting or hand coding of those portions of a program (i.e. inner loops, table search routines, etc) could totally eliminate these objections. Yet COBOL is often strictly enforced as the sole applications development tool.

- A more enlightened, though still controlled, policy on programming language flexibility is necessary to balance operational efficiency with developmental and maintenance efficiency.

2. PROGRAM STYLE

- Besides restricting the set of legitimate programming control structures to the basic three mentioned above, structured coding is concerned with the overall organization of a program, often called programming style.

  - Kernighan and Plauger propose that well-structured programs should exhibit :

  - Clear, intelligible expression.

  - Simple, straightforward control and data structures.

  - Robustness, the quality of producing correct results and rejecting invalid data under a wide variety of conditions.

  - Efficiency.

  - Self-documentation.

- The implication of this set of characteristics is that good programming practice is at least as concerned with readability and simplicity as it is with accuracy and efficiency. However, the key to readability and simplicity is through better analysis, not through comments, multiplication of flowcharts and lengthy program descriptions.

- This prescription for good programming style does not match very well with that which respondents reported concerning the nature and extent of their programming standards.

D. PROGRAMMING ORGANIZATION

- Frederick Brooks' well known principle, "Adding manpower to a late project will make it later" has to do with the geometric growth of communication paths within a project team.

- The "chief programmer team" organization model attempts to reduce this communication complexity via two tactics :

  - Use the chief programmer as the major communication mode.

  - Off-load as many clerical duties as possible onto a project librarian /project secretary.

- However, organization of the analysis/programming staff tends to follow corporate functional lines, such as marketing systems, financial systems, and specific product line systems. Within these broad categories, project teams may be organized more or less permanently according to specific applications, or they may move in and out of particular projects on a matrix management basis.

- The aspect of organization which was most appealing to the U.S. user

EXHIBIT IV-7

STRUCTURED WALK-THRU PRINCIPLES

- Technical reviews called by person whose work is to be reviewed.
- Objective and review materials distributed in advance.
- Meeting attended by peers, not by managers (usually 3-6 people).
- Purpose is to find errors, not to correct them.
- Notes and minutes distributed to participants after the meeting.
- Responsibility for fixing problems that were found is that of the person who called the review.
- Responsibility for the problems that were not found is that of the entire group.

respondents and used most widely by them was the structured walk-thru.

- Junior team members at respondent organizations found the structured review to be the device which made them feel truly participants on a team, rather than "cogs on a wheel". Senior members in turn find that the structured walk-thru helps them :

  - Pace their work better.

  - Ease the debugging task.

  - Anticipate problems better.

- Exhibit IV-7 summarizes the major principles of the structured walk-thru procedure.

  - While the scheduled reviews may occur as often as necessary most respondents felt that the minimum scheduling should include:

    - After specification is complete, but before design begins.

    - After the hardware configuration has been determined.

    - After each major phase of detailed design.

    - After each major phase of coding.

- At the maximum, a frequency of every ten days to two weeks was felt to be appropriate; but this can vary widely depending on :

  - The complexity of the system.

  - The experience level of the participants.

- The familiarity of the participants with the application.

E. PROGRAMMING ENVIRONMENT AND MEASUREMENT

- By far, the most widely used programming improvement used by the U.S. user respondents is the use of on-line terminals to write and/or test code. Most respondents had implemented TSO or CMS, depending on their operating system. Terminals were installed by one respondent in a "terminal room" to which programmers could go when they were ready to compile and test their programs. In this case, initial coding was entered by data entry clerks, not by the programmer. Other respondents have installed terminals directly in the programming areas at a ratio of one terminal to every two or three programmers.

- None of the respondents in this study had yet installed the Programmer's Workbench or Maestro, although several respondents have them under consideration. Aerospace companies who are writing software to run on mini or micro computers have supplied minis or development systems to each programmer on projects where this is feasible.

- The advantages of on-line code development and testing are both physical and psychological. One respondent devotes one entire 3032 (or a pair) to programming development between 8.00 a.m. and 7.00 p.m., while the other machine runs the on-line data entry and retrieval system. At night both machines are coupled for batch updating of the IMS files. This respondent feels that their on-line programming development facility is the competitive advantage in retaining their programming staff - a not inconsiderable benefit. In addition, programming thruput of the installation has been approximately doubled.

- Respondents who are using ROSCOE feel that it gives them all of the convenience and capability of TSO at a much lower cost. The difference in turnaround time compared to TSO does not have any noticeable impact on programmer attitudes.

- Evaluation procedures for most respondents occur prior to the adoption of a new product or technique, as part of a cost/benefit justification. Once the decision has been made to install the new product or technique, barring any bad experiences during an initial trial period, the previous evaluation is considered to be confirmed without further measurement.

- So it appears to be with programing productivity improvements. No respondent had any useful base of "before and after" statistics gathered in a controlled manner and matched to eliminate factors that were not being evaluated.

- Performance characteristics were either considered to be unessential, or had been gathered in casual, impressionistic ways. The true evaluation criterion becomes a matter of whether or not the new product or technique is accepted and used, and whether management by consensus feels that it represents an improvement. Thus, most performance expectations turn out to be in the nature of self-fulfilling prophecies.

- Attempts to be more specific are apparently more generally impelled by political factors at the respondent organization. Interpretation of results under these circumstances depends upon "which side you are on".

- A recent example of the impact of structured design, programming and improved environment, is given by a measured evaluation by Elf Acquitaine (France) in the development of a payroll/administration application.

- A structured design close to SADT was chosen and the HIPO documentation technique was added. PL/1 was used as the language in conjunction with PL/1Checkout and SPF/TSO as the test environment. The project size was approximately 90 man months or 17000 source code instructions.

- The result was to transfer more than half of the total development effort to the design phase, reduce the effort needed for both coding and module integration/testing and reduce the type of errors found to mainly minor language syntax problems as opposed to the habitual mix of design/logic and minor syntax errors.

- The main benefits obtained were better reliability of the final product, deadline met, reduced costs and better productivity in relation to traditional programming projects.

SECTION V. APPLICATION SOFTWARE PRODUCTION COVERAGE & SUPPORT

SECTION V APPLICATION SOFTWARE PRODUCTION, COVERAGE & SUPPORT

- Application software for a given system or range of systems can be developed either :

    - prior to the development of the hardware
    - as an integral part of the hardware design
    - after the system hardware design has been completed (see Exhibit V-1).

- The advantages for allowing the application to influence system hardware design (in all of its aspects) and the sytems software construction are many, but are usually considered by vendors only in those markets where the end-user has significant influence (the banks and the retail/wholesale companies are examples). This is usually because the hardware vendor does not want to limit his product to a single market.

- Yet every time this approach has been adopted, the outcome has been significant unit sales. This does not always mean significant profit, (as demonstrated by the Point of sale market where the number of vendor failures or withdrawals has been very high), due to the failure of the vendors to assess key system costs, such as initial and on-going systems implementation support and maintenance.

- The more classical approach of the hardware vendor is that of the production of "state of the art" hardware (i.e. systems that have (i) an adequate manufacturing margin, given the market going rate price, (ii) reasonable expansion potential in memory, peripherals and connection ports (iii) competitive performance). This hardware and "state of the art" system software, or the frozen specification is then put in the hands of marketing for promotional packaging in as many different user markets as ressources allow.

EXHIBIT V-1

SMALL BUSINESS SYSTEM APPLICATION SOFTWARE DEVELOPMENT SCHEDULE

APPLICATION DEVELOPMENT TIMING	% OF TOTAL		USUALLY OCCURS AS ...	IDEALLY WOULD OCCUR AS .....
A. PRIOR TO SYSTEM DESIGN BEING FROZEN	25	15	• The result of old applications being applied to new hardware "as is".	• An influence on the systems architecture, systems software, peripheral capacity and speed, and ergonomics
		10	• Old applications modified for new hardware	
B. INTEGRAL TO SYSTEMS DESIGN AS IT IS DEVELOPED	5	4	• Very rarely exc. in vertical terminal markets e.g.-POS retail terminals - Banking installations - some Office comp. • Not at all in minicomputers	• Firmware for small dedicated processing units capable of single task or single application processing
		1	• Rarely in SBC	
C. AFTER NEW SYSTEMS DESIGN IS FROZEN	70	35	• New Standard vendor applicat.	• As for A above
		15	• Standard and custom third party packages	• Either as an assist to hardware vendor's initial package developments or for specific vertical markets where vendor is not present
		20	• User developed packages	• In standard high level languages to safeguard user investment

Source : INPUT estimates

A. IN-HOUSE DEVELOPMENT BY THE MANUFACTURERS

- With the exception of the minicomputer vendors, who expect (and prefer) the user to handle his application software requirements himself, either directly (own ressources) or indirectly (third party), the small business system vendors are all trying to increase the proportion of end user support provided by standard applications.

- In addition, they would like the majority of such application software to be developed and produced in-house, mainly to avoid the problem of transference of know-how which arises when packages are developed externally.

- Their problem is that the minicompter vendors, with the price advantage that their cheaper hardware allows, can offer the user a customized (and therefore more attractive) solution for the price of an SBC with a standard application.

- So far, the market has been so buoyant that both have been able to operate successfully and profitably. However, (i) the minicomputer vendors have a far greater profit margin (ii) not all of the OC/SBC product lines are profitable, because of the support costs.

- The application guidelines that most international OC/SBC vendors are following can be summarized as follows :

  - use high level languages for all new applications, combined with tools for the user to effect limited modifications/extensions himself.

  - use one operating system for the entire OC/SBC range to standardize the software interface for all applications and provide common file handling

- target limited flexibility or fixed functions for each application to limit support requirements and allow country customisation of the packages and their documentation

- centralize development and maintenance of the applications, using country level staff as front line support

- extend each application to provide support for the OC/SBC range, not just for one or several models

- separately charge for applications for a "reasonable" fee (without clear identification of the users level of acceptance of the price, nor a detailed evaluation of how revenues compare with costs).

B. THIRD PARTY SOURCES/POLICIES ADOPTED

- There is no standard approach by the OC/SBC manufacturers with regard to third party sources of application software. Some are open to their use where a demonstrated expertise or market know how is available, others limit their intervention to specific application requirements defined by the internal corporate software staff (usually where a lack of ressouces has occurred or a deadline exceeded), still others refuse to have anything to do with external sources of any kind.

- The full gamut of policies can therefore be found :

  - third party sources discouraged

  - third party assistance accepted (and the copyright for packages developed remains with them)

  - third party assistance encouraged freely at contry level and individual/ tailored agreements sought actively by hardware vendor (corporate) for OEM type activities.

- The latter approach, until now, was limited to minicomputer vendors, but there are changes being implemented that will bring the OC/SBC vendor into the same (OEM) market.

C. HELPING THE USER HELP HIMSELF

- In view of the enormous volume of mini/OC/SBC units being shipped each year and the growth rate of the market (25%-40% according to the country market), the time is not far off when support demand will exceed the combined staff ressources of the vendors and the third party system/software houses.

- Exhibit V-2 takes a quick look at some of the possibilities. It must be emphasized that no reliable data is available from vendors on :

  (i) their total staff dedicated to small systems
  (ii) the number of systems that each man, on average, can install
  (iii)the number that need support (the rest presumably being installed either with standard applications without outside help or programmed by the users themselves)
  (iv) the total number of systems shipped.

- Nevertheless, INPUT has made some broad assumptions on the capabilities of each major company in each product category and assumed the level of user self-sufficiency for each, with the results shown in Exhibit V-2.

- The values in some of the columns are doubtful, but however the numbers are juggled, one fact emerges : at the rate of growth of small system shipments, demand will exceed support before the end of the 80s.

- In the case of the minicomputer vendors, this stage has already been reached, of course; but, thanks to the third party support available from systems and software houses, the upper limit of shipments possible is still 5-6 years away.

EUROPEAN SUPPORT STAFF ESTIMATES

Source of Staff by Category	Estimated Field Support Staff		Systems Support per Man per Year	Equals Systems Supported per Year	Assume % Needing Support	Total Shipments Possible (units) per Year	At Current Growth %, Year when Demand Exceeds Support
	Total	Max. Avail Small Syst					
● Office Computer Vendors (Nixdorf, Philips, Kienzle, IBM etc.)	3,000	3,000	10	30,000	60	50,000	1982
● Small Business Systems Vendors (IBM, CII-HB, ICL, Burroughs, NCR, etc.)	10,000	3,000	8	24,000	80	30,000	1984
● Mini Computers Vendors (DEC, DG, H-P, MAI etc.)	1,000	800	6	4,800	50	9,600	Exceeded but... ↓
● System/Software Houses	15,000	8,000	8	64,000	100	64,000	1985
TOTALS	29,000	14,800		122,800		153,600	

Source : INPUT estimates

Exhibit V-2

- In the light of this situation there is an urgent need to review the means available for encouraging and enabling the user to help himself to a greater degree (and the greater the better). There will always be a requirement for user support and unfortunately, the smaller the user, (as system prices drop) the greater this requirement becomes.

- From this (rapid) analysis it would seem that, for small business systems

  - simple, user programming tools are a must

  - vertical market-oriented languages (i.e. problem oriented syntax) would be useful

BUT- dedicated, fixed application systems, (black box application systems) look very attractive and offer the best approach/solution to the need for narrowly targeted systems required little support.

D. MULTI-COUNTRY, MULTI-SECTOR APPLICATIONS

- This concept is still alive in many vendor's corporate applications despite the small number of such packages available in comparison with the total requirements. Certainly standard applications are a fundamental support tool, and many successful packages exist. But stretching the application to several different European country markets is a difficult task at best.

- There is one approach, however, that apparently can claim a degree of good results. This is the idea of making source code of basic commercial applications available to local country staff for adaptation. The responsibility for the resulting package is, of course, transferred at the same time.

- The best standard package results have been achieved in the scientific/engineering domain, where the laws of science are fixed and the resultant end-user requirements vary only slightly. In the commercial world, where the local tax, business and legal requirements are enormously varied, standard requirements are few and far between.

- Multi-sector applications, at country level, are an entirely different proposition : it is very common to find basic, standard commercial requirements that can ben satisfied by a standard application package (Payroll, A/P, A/R, General Ledger etc etc).

- This is where the third party system/software houses are best employed, since :

  - they usually operate in one country market (very few truly international groups)

  - they normally have several distinct areas of expertise/local knowledge.

E. CHANGE REQUEST, ERROR CORRECTION AND MAINTENANCE

- Once the application is created and installed in a sufficiently large number of installations, there is every incentive to extend its commercial life as much as possible :

  - users are more easily tied into an application than they are to a particular type of system

  - a heavy investment will have been made by the vendor, sometimes the equivalent of that of a new hardware product, so that it is desireable to obtain a good return

  - the application product represents the crystallisation of the vendors knowledge of that particular application area ; it is better to extend/improve than to abandon or replace the application

  - documentation and internal education on the system represent a high cost that is tied to the application : each new application means a long period of internal training of salesmen and support staff.

- This means that application maintenance will be (indeed, usually is) a large part of the total expenditure by the vendor. This is a cost that can, and should, be offset by the user, who benefits from the improved performance.

- Asking users to pay for error correction is obviously <u>not</u> viable, but where the change brings an improvement in function or performance, users should expect to pay their share of the costs.

- Charging for this can take the form of (i) a one time charge (which would offset the development costs) or (ii) a gradual increase in the rental for the application (which would off-set the constant increase in man power costs, the main element of application software costs)

- The actual cost to the user can be minimal : a $200,000 improvement of an existing application spread over 1000 sites, is good value for the users.

SECTION VI. RESPONSIBILITY OF VENDOR COUNTRY BRANCHES/SUBSIDIARIES

SECTION VI. RESPONSIBILITY OF VENDOR COUNTRY BRANCHES/ SUBSIDIARIES

- The allocation of responsibilities and workload at each stage of the development of an application plays a crucial role in the efficiency of the end product and the level of support it provides.

- On the surface it would appear that few vendors have spent much time evaluating this aspect of application development, and consider it only at the implementation/support phase. Of those that have thought about it, most like to maintain the omnipotence of the corporate groupe anyway, if only for political reasons.

A. ROLE OF THE CORPORATE DEVELOPMENT GROUP

- In smaller companies it is acceptable to find all applications development decisions, funds and staff centralised at the corporate headquarters. For these companies, the small local representations at country level have enough to do selling, installing and maintaining the products, without having to concern themselves with developing products of their own.

- In the larger companies there is little justification for this same approach being maintained - and yet there are few examples of true delegation of application development responsibility.

- The true role of corporate application staff lies in :

  - the establishment of company-wide application support policies for example :"vertical markets where we have or intend having a 10% or

DISTRIBUTION OF RESPONSIBILITIES

CORPORATE/COUNTRY/BRANCH

RESPONSIBILITY /FUNCTION	COMPANY SIZE				
	SMALL (up to $ 300M)		LARGE (over $ 300M)		
	Corporate	Country	Corporate	Country	Branch
●Define policies					
- application support	P	-	P	-	-
- application develp.	P	-	P	-	-
- pricing	P	S	P	S	-
●Define Market Targets	P	S	-	P	S
●Define Application Needs					
- identification	P	S	-	P	S
- specification	P	S	Pi	Pc	-
●Development					
- funding	P	-	P	S	-
- design	P	-	P	S	-
- programming	P	-	Pi	Pc	-
- QA	P	-	Pi	Pc	-
- documentation	P	S	Pi	Pc	-
●Distribution/Support					
- International	P	-	P	S	-
- Country market	-	P	-	P	S
●Maintenance					
- error correction	P	-	Pi	Pc	-
- up grade	P	-	Pi	Pc	-
- hardware transfer	P	-	P	S	-
●Termination	P	-	P	S	-

Source : INPUT
Analysi

Exhibit Vl - 1

P = primary responsibility; Pi = primary for international;
Pc = primary for country level;
S = secondary responsability (i.e. as assist to primary)

more share which must be supported by at least one standard application package")

- the establishment of company-wide application development policies (e.g. "fixed function/limited flexibility" ; "high level language(s) only" ; "structured programming" etc)

- the evaluation and establishment of pricing policies for application products

- the funding and design (or at least design overview) of all application development

- the international distribution and support of all applications that are valid on an international basis

- the transferance of existing applications from one product (line) to a new product (line)

- the decision to discontinue the application product(s)

B. RESPONSIBILITIES DELEGATED TO THE COUNTRY BRANCH/ SUBSIDIARIES

- Exhibit VI-1 offers a suggested pattern for the gradual decentralisation of responsibilities as the size of the company increases.

- One notable point is that, ideally, the identification of (i) market targets and (ii) application packages needs should not come from the corporate group but from a census of the country marketing group. This is a difficult process since the needs of one country rarely match the requirements of another.

- However, it is important that the priorities be established by this group assisted by their own branches, so that the line groups are all involved with the decision process from top to bottom - something that rarely occurs in today's vendor organisations.

- Country market distribution of the application products and their support is, of course, the responsibility of each country group and its branches. This is usually the case today for the large and small companies alike.

- The important criterium to be used is that of equating delegated authority, with responsibility : if a country marketing manager is going to be held responsible for a product sales budget then he must decide where key ressources like application support are best placed.

C. LOCATION OF DEVELOPMENT STAFF

- On the face of it the location of the actual development staff is also a clear cut decision : put them where the development need is, so that the product expertise is available where the product is being marketed.

- In practice this is not often the case, with the exception of the use of local third party systems/software houses. In most instances the applications staff are mainly located at corporate headquarters (e.g. the U.S.) or in a country market which is assigned the "mission" for software development (most popular is the UK).

- This has to be wrong. While it may appear reasonable to argue that the cost of a good programmer in the UK is half that of his German counterpart, in reality it is not the best long term solution. Marketing the resulting application on the German market when the product knowledge is in the UK will rapidly make this clear.

- If a product is to be marketed internationally, however, there may be every reason to have it developed by the best (expertise, programming capability), cheapest (efficiency, man/day cost) and quickest source. The problem then becomes one of ensuring the gradual training and internal education of all of those groups who will be called upon to market, install, support and maintain the product in the local markets.

D. TIMING THE TRANSFER OF KNOW-HOW FROM CORPORATE TO BRANCH/SUBSIDIARY

- It should be noted that if responsibilities are delegated in the fashion suggested in Exhibit VI-1, this problem only arises for the truly international packages, which usually have a life time of at least ten years. Thus it should be an infrequent problem. In reality it is a problem facing more than half of the products produced, and would concern a higher percentage still, if it were not for the fact that many packages are ignored at Branch (and sometimes country) level.

- The earliest possible moment should be chosen for the transfer of know-how, and can occur at several distinct time periods according to the function concerned :

  - Marketing should be informed once the specification has been frozen and the development work begun. This allows the inception of marketing litterature.

  - Sales should be advised of the role and capabilities of the package once it has entered the QA phase; at this stage the precise function of the application is known in detail and a reasonable estimate of true availability can be given ; at this stage the draft marketing litterature should be available also ; ideally this should be handled by a combined development project leader/marketing team, to make a powerful impact on the sales group.

  - After completion of QA the source code and systems documentation can be transferred to the country support groups (if such exist) for language customization ; this should be supervised by members of the programming team who produced the product (or from the third party vendor if the application was produced externally).

- Country launches of the same product need not be simultaneous and depend on local circumstances, which may include such matters as the setting up of the administrative procedures designed to cope with the collection of the revenues generated by the product, or the more mundane procedures to handle error correction and change requests.

E. VARIATIONS AT COUNTRY LEVEL

- There are two major variations that can occur at country level :

  - the product proposed is not valid and is not launched

  - the local market conditions allow a far higher price than the simple translation of the corporate price into local currency suggests.

- The former decision should obviously be made by the country market management. The latter needs to be carefully reviewed by corporate applications staff, in the light of corporate goals, as well as in view of the local country competitive environment since a mistake at this level could seriously damage the hardware product sales that are linked with the application.

- The main variations all concern levels of detail :

  - operator message and support documentation translation (neither of which affect the logic of the programme)

  - marketing strategy in view of the local competition

  - benchmarking, as a prospect "warmer"

  - etc

- Of fundamental importance, however, is the fact that each country market manager should feel responsible for the product lines sold on his territory and that application products are given equal weight with the hardware products.

F. THE FIXED FUNCTION VERSUS FLEXIBILITY TRADE-OFF

- This is a key issue that forms part of the corporate application staff's responsibilities, but that can be adapted to include local country responsibilities.

- It should be permissible for each contry to determine whether it promotes and uses all or a subset of the applications produced elsewhere, thereby reducing the support requirements down to a level consistent with local staff availability.

- This should normally consist of "freezing" the functional flexibility of a package at a given, acceptable level so that it becomes a fixed function package: easier to promote, support and maintain.

- Of the vendors interviewed, few could justify the use of "flexible" packages in terms of identifiable benefits in customer support, popularity of the package, wider market application etc., whereas those that advocate the use of fixed function packages all benefit from the simplified support and clearer profile that their applications offer.

SECTION VII. MAJOR SMALL BUSINESS SYSTEM COMPANY PROFILES

A. BURROUGHS

BURROUGHS

- As a company, Burroughs is a contradiction of :

  - High volume success in the small business system and accounting machine markets, and medium/small volume sales in the larger compatible main frame market.

  - Loyal customers (up till now) and an unsatisfactory level of service (from Burroughs).

  - High technology innovation and on-going maintenance problems in many of its products.

  - A reputation for having a good understanding of small business applications and a reluctance to invest in this area, which is the core of Burroughs' strength and success.

- In addition, Burroughs' on-going difficulties with its sales, branch manager and national executive staff seem to be escalating, at a time when the small system market is growing by leaps and bounds. As a result Burroughs may lose the market share for the worst reason of all - inadequate staff to cope with demand.

- However,with the Business Management System (BMS) suite of application programs developped for the B700/1700 now moved onto the B80/800/1800, rebaptised CBMS II, Burroughs is making the level of investment needed to (i) support its small systems range (ii) standardize basic business applications across the three small system lines.

- Burroughs has been the pace-setter among IBM's competitors for profits, leading Honeywell, NCR, and Univac and growing revenue around 15 per cent per year. Despite this, and the sound financial base that has resulted from it, Burroughs had not been paying adequate attention to the fundamental needs of the small business system market in the late 1960's; very high reliability, trouble free hardware; user-oriented, proven application software development tools; good quality implementation and support staff.

- Since then the B700/1700 has been used as the springboard for a very ambitious suite of Invoicing/Sales Analysis, Inventory Control/Management, Accounts Receivable/Customer reporting, Accounts Payable/Cash Forecasting, Payroll/Labour Analysis and General Ledger/Financial Reporting packages.

General Approach Used in Supplying Application Software

- At present, standard SBC and OC application software is produced by both Burroughs, third party software/system houses and the users themselves. There is a desire, however,to eliminate the user from this area, to allow Burroughs greater control. Ideally third parties operate under contract to Burroughs, (e.g. Computer Analysts and Programmers-CAP).

- No clear policy exists with regard to third party consultancies and software/system houses. Burroughs has used external sources for software development but prefers to be self-sufficient. In the small system area the biggest third party contract was with CAP who assisted in moving the BMS suite of applications from the B700/1700 to the B80/800/1800.

- Small system users are now offered good, reliable system software (particularly operating systems like MCP), a full complement of compilers and utilities and broad-ranging standard application packages that offer cross-line compatibility of function.

- Application software is intended both as a support of H/W sales and as a generator of revenue. The gradual decline of H/W profitability means that S/W profitability must increase and Burroughs is aware of this.

- All of the major decisions on application software selection and development are taken in the U.S., so that the viability and responsiveness of the standard packages offered, vis a vis the wide-ranging European requirements, is not always adequate. This varies from country to country.

Application Support and Maintenance

- Formal procedures for problem reporting operate locally as opposed to centrally. Usually all of the applications offered can be transfered to other Burroughs hardware.

- A worldwide catalogue of licensed program products exists and is distributed internally within the company. Requests for modifications or enhancement of these functions are fed directly back to the U.S.

Application Software Production Methods

- Burroughs covers the entire range of SBC/OC requirements with three product ranges, (B80, B800, B1800) although distinct models tend to alternately overlap one-another according to competitive announcements. For example, Burroughs has most recently re-vamped the B1800 to produce the B1885 in response to the IBM System/38, overlapping part of the B1800. Equally the larger B80s overlap the smallest B800s.

- The strong advantage that Burroughs holds over most other manufacturers is the user of a single operating system (MCP) over almost the entire range of Burroughs' products from the B80 to the B7800.

- The weak point of upgrading the earlier B700 users is that they must either run their programs in the old SCP environment or carry out detailed program changes for running under CMS (Computer Management System). The latter is most likely since the full capabilities of the replacement system could be otherwise largely wasted.

- In general one version of an application product is offered to cover a given product range, but functional/capacity extensions are possible after initial installation. This is an almost mandatory requirement given the wide range of capabilities offered by each range.

- According to European sources there is no fixed policy as to whether application products should offer limited flexibility/low customisation (for rapid implementation) or greater flexibility (to provide greater effectiveness). The latter seems to be preferred, however.

- Traditional, high level language (e.g. COBOL) are preferred for applications development but tools are marketed to enable users to change programme products. Burroughs maintains a number of development centres in the world, the main one being Detroit. Some product development is sub-contracted to external third parties but the ownership distribution and support are all assured by Burroughs.

B. CII-HONEYWELL-BULL

B. CII-HONEYWELL BULL

- From its beginning as an independent European manufacturer of electro mechanical data processing products, through its two stages of ownership by U.S. vendors (GE, Honeywell) this group has always maintained a strong European and International outlook. With the creation of CII-Honeywell Bull through the merger with CII, a measure of independence has been gained, tinged with a re-inforced influence of French interests.

- In all, CII-HB covers some 27 countries. Despite the frequent changes of ownership, the Honeywell-inspired move to unite incompatible products under the 60-series logo has been broadly successful.

- At the low end of the series, the level 61 and level 62 business systems have now been added to the Mini 6 minicomputer to form the product base for a new "DPGD" division, the equivalent of IBM's GSD. This group now serves the small/medium sized centralised user and the decentralised/-distributed processing requirements of the large user.

General Approach Used in Supplying Application Software

- CII-HB currently absorbs the main responsability for application software development, and compliments this with third party system and software houses. There is no intention of this approach being modified in the near future.

- Third party system/software houses stand a good chance of placing application products with CII-HB where they can demonstrate a clear/acknowledged expertise in a given area. In general CII-HB uses outside help of this nature on a country by country basis. The main problem is the transfer of know-how from the third party to CII-HB staff.

- Standard packages, parameterised packages and program modules are all offered, some of which are marketed under a licence agreement.

- The Warnier method of structured programming has made good in-roads in CII-HB, and there are plans to make broad use of it for application development.

- With such a large number of country markets to consider, the difficulties are many in defining the order of priorities, how each country's requirements can be best covered, how national language differences are resolved etc. Chief among these difficulties are :

  - major markets in one country are not as important in other countries, (i.e. Retail/Wholesale may be of paramount importance to Germany, the main requirement in France may be Banking etc)

  - for common applications, operator messages source coding comments and national documentation usually have to be in the local language.

- CII-HB's approach has been to provide application source code (after the QA phase) to each affiliate for local adaptation of both specific requirements, operator messages in the local language and translation of the support documentation. For some packages this can mean as much as 50% of the code being modified.

- CII-HB treats application software, in order of priority, as (i) a source of revenue, (ii) a source of hardware sales and support (iii) for maintaining or gaining expertise in user needs in a given area.

Application Support and Maintenance

- Although not yet achieved across the board, CII-HB's goal is to offer applications that provide transferability to follow-on hardware and can be used across the SBC range. (This presents a problem in the new organisation in that the level 61 is manufactured by CII-HB and the level 62 by HIS Italy).

- Change request procedures and error correction/maintenance have been formalised both for CII-HB's own applications and those that are licensed products. These procedures are partially controlled locally (National markets) and partially centrally.

- A centralised catalogue of licensed products exists as a picking list for each affiliate. It applies to almost all of the countries in which CII-HB sells, is published locally but is managed centrally by CII-HB.

Application Production Methods

- The specification process is not the beginning of the product definition. Instead it is a market goal that needs to be satisfied that initiates the development. A standard (confidential) development cycle exists, but as yet CII-HB has not found a satisfactory general approach to product development.

- Many variations occur at country level, as already explained, but there are plans for multi-national application ; in particular a manufacturing suite and a retail/wholesale suite.

- For hardware support, CII-HB uses the technique of "re-implementing" existing applications. This means that some application statements are modified as required and recompiled using high level language compilers. In this way one "version" of a package can cover the total SBC range. Functional and capacity extensions of the package are possible after initial installation.

- The overall trend is towards limited flexibility of the package ("to avoid high costs"), as opposed to greater flexibility and effectiveness. New applications are all developed in high level languages.

- To complement this trend of standardisation, tools are sometimes offered to allow users to change/extend applications (with varying success).

C. DATA GENERAL

DATA GENERAL

- Although a young company, (11 years old in April of this year), DG now employs more than 4200 people in thirty countries and had 1977 sales of slightly less than $255M. In 1978 sales were expected to be of the order of $370M. Originally based entirely on the NOVA line of mini computers, DG introduced a medium scale ECLIPSE line in 1975, and in 1978 added the 9045 and 9070 small transaction oriented systems.

- Particularly noteworthy in the growth of DG has been the gradual departure from OEM sales towards end-user markets. The Eclipse line is the main vehicle for this, and is equipped with data management software (INFOS), COBOL (ANSi 74), RPGII, PL/1, FORTRAN, BASIC and Assembler. The 9045/70 are equipped with FORTRAN IV and BASIC, enhanced by file management capabilities.

- The significance of this entry into the end user market is that DG has had to face a massive demand for application software. In response DG has adopted a very flexible attitude, allowing software houses and system houses to design turnkey systems around the Eclipse, as well as around the traditionally OEM CS family (CS/20, 40, 60).

General Approach in supplying Application Software

- The main responsibility for the production of application software is left to the user himself (in the case of the Eclipse line) and to third party systems/software houses (for the CS series). DG see no change in this approach for the foreseeable future.

- The responsibility for the resulting applications products lies entirely with the producer (user or systems/software house); DG's responsibility is limited to the hardware.

- The basic tools supplied to the user for developing applications are the compilers COBOL'74, Fortran IV and V, PL/1, DG/L (similar to Algol), Business Basic and certain utilities (e.g. IDEA - interactive data entry/access).

- The overall attitude towards application software is that it is a means of selling hardware not a source of revenue. DG does sell operating systems software, however.

Applications Support and Maintenance

- DG offers a limited number of program products, towards the top of their range. Source code is not provided to users, but error correction and maintenance are offered.

- Errors found by clients in these programs products are documented to the Engineering Dept., for correction. Major errors are notified to DG's Boston headquarters for processing and distribution of the definitive modification.

- Licensed program products are handled by Westboro Massachusetts. Formal procedures for problem reporting and change request for these products are controlled centrally from there. A catalogue of these licensed program products is held internally.

- The Paris France headquarters acts as a filter for applications problems, before forwarding them to Westboro.

- DG has no plans for providing multi-country applications for Europe.

Applications Software Production methods

- The SBC/OC product range is covered by two product lines, the CS family and the Eclipse Data Systems family (commercial systems). Within the CS family there is total compatibility, but there is limited compatibility between the CS and Eclipse lines.

- As a result, several versions of applications are necessary to cover the different hardware configurations. Functional/capacity extensions are possible after installation.

- New applications are being developed with traditional programming languages (mostly COBOL, for the commercial Eclipse) including RPG II, PL/1, FORTRAN, BASIC and Assembler. This is to enable end users to do their own adaptation/modifications or extension of program products. Outside organisations are not used for software development subcontracting.

D. DIGITAL EQUIPMENT CORPORATION

D. DIGITAL EQUIPMENT CORPORATION

- Now twenty two years old and an established billion dollar plus company, DEC has built the position of the world's supplier of minicomputers on fast, rugged, real time systems with minimal software, relying on users and OEMs to customize their hardware to fit the end-users applications.

- User self reliance is still an integral part of DECs corporate philosophy in approaching the markets they serve, and with close to 120,000 computers installed worldwide, the policy has a lot to recommend it. Typically between 50% and 80% of shipped systems are handled by third party OEMs and system houses in European countries.

- DECs main application markets include timesharing (for Computer Services Vendors, universities, laboratories etc) workstation products (e.g. the VT78 DEC station, graphic terminals, DEC writer II and III etc), process control and/or data collection in manufacturing environments, network products, small business systems (the DEC DATASYSTEMS series) and many specialised applications areas (e.g. automotive, aerospace, cartography etc).

- Word processing is a new opportunity for DEC and is being approached, with the same thoroughness as DECs data processing activities. However, this new market is looked upon more as an application for existing hardware, rather than a new product design requirement.

General Approach Used in Supplying Application Software

- In most of the fifteen European country markets where DEC is active, the majority of applications products are produced by software/systems houses and OEMs. There is no fixed policy in this regard and each new software requirement is reviewed at branch level with the OEM or Software house known to have the relevant industry application expertise.

- DEC usually enters into individually tailored software house agreements, based on an internal (five-volume) guide called "Consultant Reference Guide" designed to assist in the writing of such agreements.

- There is no fixed or generalised application development technique: compilers, parametrized packages, standard packages and program modules all contribute support according to availability.

- DEC has reached the stage where it does not want to antagonize the very large numbers of OEMs that it deals with, so that standard applicational support tools are only to be provided in "focussed" markets, not across the board.

- DECs strength and success has been built on applications with a high technology/technical/scientific content where the requirements are common across all country markets. Any application which has a business practice content usually means enormously varied and non-standard requirements, which demand specific, customised solutions for each country sector or even branch.

- Non-portable applications, of this type, include for example financial applications (where tax laws and accounting procedures/practices vary from country to country). In contrast engineering computations are usually international.

- DEC began using application software as a generator of hardware sales. Now it sells systems, and software is gradually becoming a business of its own. It remains a small proportion of DEC's business at present, however.

Applications Support and Maintenance

- DEC provides a world-wide catalogue of licensed program products that are applicable to all of the 15 European countries where their hardware is sold. This assists in making available the many third party programmes that are constantly being developed.

- Since the whole area of software support is (i) largely out of the DECs hands (ii) a politically sensitive area for third party software/system houses and OEMs it is not meaningful to discuss the support/maintenance aspect of the applications running on DEC equipment, (done largely by the third parties themselves).

Application Software Production Methods

- The new applications under development are mainly written with traditional programming languages. Some effort is being made to improve the level of standard software support available for each of the major product areas offered. However, there are far more packages available from third party sources (including end-users) than those developed by DEC themselves.

- DEC has benefitted from the fact that its large installed base attracts third party support by its sheer size. In this way the vast support requirements are off-loaded to third parties leaving DEC free to concentrate on the hardware.

E. HEWLETT-PACKARD

E. HEWLETT-PACKARD

- In the last ten years, H-P's mix of products has altered dramatically. From a predominantly electronic test and measurement equipment company (86% in 1968), and little involvement in EDP (4% in that same year), H-P's electronic data processing products are now the major source of sales. (45%). This includes computers, electronic calculators and computer/calculator peripheral products.

- H-P also has a considerable technology strength, which has led to the development of new families of products through the constant improvement in density, cost, speed and reliability of components. This has gradually led H-P to produce, in-house, more and more of the microprocessors it uses in its products.

- H-P's image is one of quality and performance, particularly in the engineering/scientific environment, but increasingly in the personal computing/ small computer system area, thanks to the 1000-Series and the 9800-Series.

General Approach in Supplying Applications Software

- H-P does most of its OC/SBC selling in Europe through OEMs, and the responsibility for producing the software rests with the end user himself or a third party system/software house.

- In the near future, HP hopes to have the user accept full responsibility for his software. This is because the policy is to avoid the first time user and to concentrate on experienced computer users who do not require support.

- For this reason H-P does not sub-contract software, since there is either a direct HP/customer relationship with no third parties involved or an

OEM/user relationship where the software responsibility lies with the OEM. The vast majority of HP's sales of OC/SBC goes through system houses.

- As a result there is no software writing done by HP in Europe. User training and consulting does occur through the H-P systems engineer, but the only standard applications that are offered are all produced in the U.S.

- There are plans to sell certain application packages to the OEMs and experienced end users, but in all of these dealings the objective is to sell hardware systems. No list of the software houses that H-P has dealt with is maintained and distributed.

Applications Support and Maintenance

- As seen by the above, this chapter is not particularly relevant to H-P's mode of operation.

Application Production Methods

- In a recent move HP acquired a small German software house in Karlsruhe to do quality assurance of applications developed outside and to carry out modifications and adaptations to the U.S. developed application products.

- H-P's overall policy for applications is one of great flexibility by the development of modular tools/modules that can relate directly to the architecture of the hardware products, (these are known as "intrinsics").

- Where possible H-P likes to cover the entire range of their computer products with the same application. This is the case of IMAGE, which was developed as an information management system for the systems/3000 and which has been modified for the HP 1000.

- Application development represents a very small portion of the whole R & D effort.

F. INTERNATIONAL COMPUTERS LTD

F. INTERNATIONAL COMPUTERS LTD

- In 1978 ICL produced a turnover of £509.4M ($1.01B approx) almost as much as CII-HB ($1.04B). While the rank of number one European manufacturer still eludes ICL, they have grown considerably over the last three years and are now totally independent of U.K. government aid.

- In the small business system area, before the Singer takeover, ICL had had a strong success with its own 2903 small business system. With the takeover came the System Ten and the Cogar-developed 1500 Series of terminals/small systems, which ICL has marketed with great success.

- More recently ICL announced the 1505 which rounds out the 1500 Series (1501, 1503, 1505) which now services requirements ranging from data entry to full blown small business system applications in satellite or remote batch mode.

General Approach Used in Supplying Applications Software

- ICL prefers to provide application software support (and software support in general) itself or through third party sources. As a rule they do not assist customers in the design and implementation of custom requirements, however, judging that the customer should "stand on his own two feet".

- None of the standard applications are marketed through licence agreements. ICL will, however, commission software houses to write software, but will expect to take the worldwide proprietary and marketing rights to the final product.

- An alternative approach is that of buying an existing software house product and (optionally) paying royalties for its use. ICL may also suggest a system/software house to one of its clients, as a source of applications development.

- Plans for improvement include :

  - more support of the software community, encouraging them to write programmes for ICL machines

  - joint development of applications with (large) users

  - an increased number of agreements with local software houses.

- Application software is now being treated primarily as a source of revenue.

Applications Support and Maintenance

- The major problems are : (i) that of locally supporting each application; (ii) adapting a given application to a customers requirements (iii) integrating a standard application into ICL's total software offering.

- One of the solutions being considered is the use of local software houses to set up customer support centres (in order to customize standard packages and support them locally).

- Problem reporting and change requests are controlled locally for locally acquired products and centrally for centrally acquired packages. Change requests are logged on a centrally recorded (Slough U.K.) log which serves as the basis for enhancements.

- Errors in standard software are handled by a customer interface/maintenance centre at Wokenham U.K. where temporary patches are issued while faults are referred to the development units. The permanent correction is issued later.

- Modifications can include, for example, discount procedures or tax rates etc. These are factored in by the local country staff. ICL offers practically no industry orientation modifications for cross-industry packages. The tendency is to parameterize the packages so that they can be adapted to each industry.

Application Software Production Methods

- The process begins with a statement of market requirements which is jointly prepared with the national market directors (who sign off each document). The package is then written and local territories are encouraged to produce their own version.

- Versions of each package with restricted or extended functional ability are necessary to cover the different hardware configurations. When these extensions are made after installation, the customer pays for the service.

- Development plans are moving in the direction of limited flexibility/low customization (for rapid implementation). The "greater flexibility for greater effectiveness" approach was tried in the past, with poor results.

- New applications are being developed with traditional programming languages (COBOL, RPG, Assembler etc) for the sake of portability. Development staff are usually located centrally.

- Locally developed products are issued locally by the local company branch offices.

G. KIENZLE APPARATE GmbH

G. KIENZLE APPARATE GmbH

- The Kienzle group is divided into two main groups :

  - measurement devices (taximeters, automated petrol dispensers etc)

  - office computers/small business computer systems

- Kienzle's basic strength lies in the office computer systems market, in which it has an established position in the West German market, (just over 20%). The same devices are used as banking terminals and small business systems, or as satellite processors connected to a network.

- Kienzle's position elsewhere in Europe is far less emphatic, but the Kienzle products are sold across the board in all Western European countries.

General Approach in Supplying Applications Software

- The basic application software approach is for Kienzle to provide the system software from the central headquarters in Villingen, for customisation by the local Kienzle office, a software house or the client himself, depending on the users in-house ability.

- However, Kienzle takes full responsibility for the total system that is ultimately provided to the user : thus if a software house is being used, Kienzle will select, among the bidders, the software house that it feels is capable of doing the best job, checks their product when it is ready and delivers it to the client.

- To assist customers in the design and implementation of their own applications, Kienzle provides a modular mix of compilers, decision table procesors, parameterized and standard packages and programme modules.

- A one-time licencing fee is charged for applications bought outright and there are similar fees for programming tools that are provided to those users intending to write their own programmes.

- The post sale assistance is provided on a per diem basis up to a month's elapsed time, beyond which a flat fee is charged.

- The demand for customized/dedicated packages for a given customer's special requirements is charged for.

- Software is both a generator of hardware sales as well as a source of revenue. But Kienzle expects that the amount of revenue coming from software will gradually increase in the future.

Applications Support and Maintenance

- Kienzle's control over how the local country representations handle applicaation support is flexible : policies enunciated by the headquarters are not binding, but are given as guidelines.

- Licensed software packages are catalogued and the list is maintained and supplied to all country markets. There are some multi-country packages, and these are modified in the field by the local offices.

- Industry-specific needs are met by modifying the basic application to the industry requirement.

- Formal problem reporting, change request and error correction procedures are operated.

Application Software Production Methods

- Software versions of the same application are produced to cater for the extended functional abilities of the various hardware configurations, since Kienzle covers the OC/SBC range with several product lines.

- Although functional extensions of a package are possible after initial installations these are not volunteered and only occur in response to a specific customer requests.

- Development plans call for effective user-oriented languages that can handle the flexible user requirements, for greater effectiveness in user support. Currently the tools available are mostly generators.

- New applications are being developed both with traditional high level languages and with parameterized languages. Every year Kienzle spends in the order of DM 20M ($ 11M) on software production. DM 5M ($ 2.75M) or more can be spent on a single application and the demand is constantly rising.

- Already software development costs exceed hardware development costs. The development staff are located both in the countries served and centrally (Villingen, West Germany).

- To face up to the demand for software, Kienzle has developed working relationships with sophisticated international software houses, who either sell their software products direct to the end user or have it marketed by Kienzle.

- The relationship policy is flexible; in some cases they can also sell turnkey systems based on Kienzle hardware. In the case of software package sales, the developer either receives a percentage of the package price or a licencing fee for each package sold.

H. MATRA INFORMATIQUE

MATRA INFORMATIQUE

- Owned 55% by Matra, and 45% by TRW, Matra Informatique (MI) operates in France, Benelux and Italy (through an agency). Regional centres in eight major cities in France and in Brussels serve some 3,500 terminals worth $45M installed as of year-end 1977. The main product is the Matra GCS 430 multistation, disk-based data capture and preprocessing device. Rapid growth in the Datapoint line of 1100, 1150, 1170, 1200, 2200, 5500 and 6600 products is altering this picture, however. MI also offers Scandata OCR equipment, and offers its own minicomputer (based on either Zilog, Mostek, Motorola or SEMS products).

- MI is an interesting example of a European national distributor of (US) (TRW) small systems, which is largely left to its own devices for the provision of software, support and maintenance/service to the end user and which must be careful in the use of its limited ressources in assisting users to install and programme their small systems.

General Approach in supplying Application Software

- The main responsibility for application software production lies currently with the users, with some assistance from systems/software houses where necessary. The trend is for MI to place the emphasis on the systems/software houses, but MI will also be making greater efforts themselves.

- TRW, the manufacturer of the Datapoint systems, has developed a number of program modules for assisting users in designing/implementing their own applications, particularly for I/O. An example of these is the DSGEN formatted data entry module written in the high level DATABUS/DATASHARE language.

- The main languages provided are COBOL (ANS-based), GAP-II/RPG II, Dataform/Multiform (monostation and multistation data manipulation), BASIC SCRIBE (text composition and editing), and DATABUS (commercial applications only). Also provided are interpretors, emulators (IBM 2741, 2780, CII-HB, VIP etc) and an autonamous telecommunications link (Multilink).

- Most ambitious of all (and the most impressive) is the Attached Resource Computer (ARC) system. This relies upon software-controlled, specialised processors interlinked by a hardware/software Interprocessor Bus. The individual Datapoint units (6010/20, 5500, 6600, 3810/20) are divided into those dedicated to the applications(s) and those that manage the data ; the Interprocessor Bus allows high speed data interchange without imposing telecommunications overheads on each of the processors.

- In this way each department of an organisation can grow its processing requirements modularly and independently of each other, while being able to tie themselves in to the overall system when necessary.

- Some parameterized applications are available (e.g. The FORD car dealers package) and a limited number of standard applications, (accounting functions, agriculture). None are marketed under a license agreement yet, but there will be developments in this area, particularly for user-developed packages.

- Attempts to improve the application package coverage will centre on vertical market areas (i.e. narrow markets where standard functions are identifiable).

- Up till now, MI has treated application software (and software in general) as a source of support for hardware sales, not as a revenue generator. The principal goal is profit centre management and software is unlikely to become a major revenue issue for some time.

Application Support and Maintenance

- With few standard or parameterized applications available, the problems of their support and maintenance are not great. In general MI tries to guide the user towards using standard applications (if one exists) in its unmodified form. Where this is not possible, the user is on his own, (own staff, systems/software house) or must contract with MI for (paying) assistance.

- MI has no plans for multi-country applications but has run into difficulties trying to apply standard applications to different industry sectors. No particular effort has been made to adapt or modify the existing applications to resolve this, and each user requirement is treated separately.

- MI does provide, however, tools to enable users to change/adapt/extend those program products that are available.

Application Production Methods

- MI's (and TRW's) goal is to cover the Small Business Computer/Office Computer range with one (compatible) product line. The development of ARC is an obvious step in that direction since it allows most of the current Datapoint devices to communicate with each other while executing their own specific applications.

- The availability of SCRIBE provides an entry into text applications from the same base of products - a not inconsiderable advantage for attacking the combined text/data processing market in a highly modular yet interconnected fashion.

- In MI's view it should be possible for one application to cover this range of products and configurations, but with extended-function versions for the higher models and largest configurations. Functional extensions should always be possible after the initial installation. At present it is too early for MI to talk about their (and TRW's) plans in this area.

- The main objective in developing new applications is to provide users with a high level language, but not necessarily a traditional one.

- The level of effort involved in application software production varies from 1 man month (for a local development) to 7 man/years (for an international package developed by TRW).

- TRW itself very rarely uses outside organisations for developing standard software, but each of the country distributors makes its onw decision in this respect.

I. NCR

NCR

Note:

NCR did not give INPUT approval to publish details of the application software development interview or a description of its application product development and support procedures and strategy. What follows is INPUTS' own view of NCR, ommitting software development details.

- Only seven years ago, two thirds of NCR's revenues came from electro-mechanical cash registers and accounting machines. At that time (1972) the company was showing a loss of $60m although revenues were the highest among the IBM competitors, in front of Burroughs, Sperry Univac and Honeywell.

- Since then, NCR has swiftly improved its profit position and can now fund expansion from its own internally generated cash. The big change has been the emergence of NCR as a leading supplier of terminals (mainly banking and point-of-sale) and the revival of its position as a manufacturer of small and medium-sized computers. The acquisition of Data Pathing in 1976 expanded the specialized terminal coverage to the manufacturing industry.

- But the most important change in NCR has occurred in its own manufacturing. This reduced the work force to one tenth of its former size and decentralized manufacturing - a move which Olivetti would like to emulate, but for the Italian labour laws.

- NCR has also taken steps to increase the standard application package support available for small entry level computers by high spending on a regular basis. This task is somewhat simplified by the clean cut sector orientation of many of NCR products.

- On the marketing front, NCR is organised into "vocational" groups selling to retail, commercial, industrial, medical, educational, financial, and government prospects. The main thrust is to develop a detailed understanding of the customers problems and of the industries served.

- In the field engineering area, NCR has been innovative in allowing engineers to sell add-ons and additional systems to established clients that they visit for maintenance service. In addition, the maintenance function has been set the goal of becoming a profit generator - something that IBM has been alone in achieving so far. Field productivity, improved diagnostics and higher service rates are seen as the main tools for achieving this.

J. NIXDORF COMPUTER

J. NIXDORF COMPUTER

- The number one office and small business computer supplier in West Germany (behind IBM) with well over 30,000 installations, Nixdorf also ranks as the fourth largest European computer manufacturer. Worldwide installations are well over 50,000.

- The acquisition of Entrex in the U.S. expanded Nixdorf's marketing and support coverage in the U.S., headquaartered in Chicago. The move stretched Nixdorf's financial ressources, however, resulting in the forced participation of a German bank in the capital of the company.

- As its main product line, Nixdorf markets two versions of the 8870 small computer, one for West Germany (where demand is for remote batch or intelligent terminals) and one for the international markets (which has transaction-oriented interactive processing abilities). The two are incompatible.

General Approach used in Supplying Application Software

- User application support occurs in two ways :

  - Nixdorf takes direct responsibility for the application development (the preferred mode of operation)

  - a direct customer/software house agreement is reached in which Nixdorf is not involved.

- Nixdorf prefers the in-house development process because it maintains the know-how of the application within Nixdorf. Future plans call for Nixdorf developing standard packages either directly or through system/software houses. In the latter case Nixdorf tries to retain the copyright of the packages produced (unless, of course, the software is paid for directly by the customer).

- In general Nixdorf retains the source code and delivers only the executable programs. This applies to compilers, parameterized and standard packages. All are delivered from Paderborn.

- Application software is looked upon as a source of hardware sales and support, not as a source of revenue.

Application Support and Maintenance

- In the past, the differing tax systems and statutory laws in force in Europe have created problems for standardisation in bookkeeping applications (stock control, payroll, order processing and invoicing etc).

- Formal procedures exist for problem reporting (the Fehler Melde Verwahren or Error Communication Warning System) managed centrally in Paderborn.

- All support, in the error correction sense, is centralised in Paderborn also. The country branch groups provide the usual front-line support at the local level.

- There is no catalogue of licensed programs since Nixdorf has only 6 or 7 packages of this nature.

- Nixdorf plans to produce multi-country packages in Padenborn which will be adapted to the needs of each individual country by the Paderborn staff.

- With regard to cross industry packages, Nixdorf produces a version of each package for modification for the industry sector requirements, through the use of parameters. To do this a parameter generation tool (CHeck list Input and Customer oriented Output or CHICO) is available.

Application Software Production Methods

- In order to cover the different hardware configurations, several different packages with restricted functional capacity are prepared. Extensions of the functional ability are possible after initial installation.

- Development plans are moving in the direction of greater flexibility for greater effectiveness, and the new applications are being developed with traditional high level programming languages.

- Development staff is located in Paderborn, although some development is subcontracted to systems houses, including the testing of the resulting package.

- On average, the customization of a standard application package requires one man week per program and between two and three man/months are spent on an installation for system implementation.

K. PHILIPS DATA SYSTEMS

K. PHILIPS DATA SYSTEMS

- With over 20,000 office computers, mostly of the P350 Series, Philips has gained an enviable customer base of first time users who, in general, are well satisfied with the Philips product and service.

- The parent company is the fourth largest industrial concern in Europe with a turnover of some $1.6B, but only 4% of that is contributed by the Data Systems group.

- The P400 Series, the follows-on upgrades for the P350 is rated by users as simple to use, reliable and quiet. In addition Philips boasts extremely good support staff for the system implementation phase.

Approach used in Supplying Application Software

- The general approach is that of relieving the user of most of the burden of applications software development, relying mainly on (a) Philips 'library of standard application packages (b) the support staff that Philips can place at the user's disposition.

- Some third party system/software houses are brought in to complement this approach, but usually only on request from the user and in those instances where packages are required that are not available in some form from Philips. This is unlikely to change in the future.

- Compilers (mostly COBOL, IDEAL-Philips's own Interactive Data Entry and Access Language, and PHOCAL-Philips Office Computer Assembly Language) are also provided. PHOCAL is intended for monoprogramming the P300 Series.

- Application software can also be generated from a library of modules based on customer specifications. The future plan is for these modules to be standardised between country groups.

- There are some applications products that are marketed under a license agreement; this occurs only in the U.K. and Austria however.

- Philips is currently aiming at tying different country groups together that have similar accounting laws (e.g. France and Spain, the German-speaking countries etc).

- Application software is treated both as a source of revenue and as a generator of hardware sales and support.

Application Support and Maintenance

- In general, program products are offered that are transferable from one hardware line to the upgrade or follow-on line. However, it is recognised that there is a significant difference in application requirement between to OC market and the SBC market.

- Problem reporting and change request for application software are both handled locally by the country groups and policies vary widely between them.

Application Production Methods

- Horizontal application packages for more than one country are developed by a group representing several different countries with like requirements. An extension of this "country group" approach is currently being considered. The effort includes specification, design and testing of the application which is then adapted locally as needed.

- Normally, applications cannot be adapted from the P300 to the P400 range, but can be within either of the two ranges. Function extensions are possible after initial installation.

- New applications are being developed with traditional programming languages. Development staff are located in the countries served, although some risk on multi-country projects.

- Outside organisations are seldom used for software development sub-contracting on a local basis and never on an international basis.

- Development plans are moving in the direction of limited flexibility/low customization for rapid implementation. However some countries have jointly developed applications with great flexibility, for customization with generator techniques.

L. PRIME COMPUTER INTERNATIONAL

L. PRIME COMPUTER INTERNATIONAL

- Founded in 1972 by ex-Honeywell engineers, Prime has hitherto concentrated on the scientific and engineering market where hardware could be sold with a minimum of system software and next to no application software.

- Gradually, however, standard compilers were added (COBOL, BASIC, Algol 60, RPG, Coral 66) and networking requirements were addressed by the development of Primenet, allowing Prime systems to interface with IBM, Univac, ICL and CDC mainframes.

- In January 1979, Prime announced its intention of competing in the business computer market with the launch of the 450, 550, 650 and 750 systems, complemented by PL/1 and Fortran '77. An interactive data management package and a COBOL program generator were the main application-oriented support tools offered.

General Approach Supplying Application Software

- Like all minicomputer manufacturers of its size, Prime relies on excellence of hardware performance and reasonable price to convince the prospect. From then on, the business computing user is largely on his own. Prime will introduce their users to systems and software houses but prefers not to get directly involved.

- Prime's scientific and engineering packages, over the years, have grown in scope and number, some of which are marketed under licence agreements. Commercial applications, however, are largely inexistent to date.

- Prime treats software in general as a generator of H/W sales and for speeding up customer acceptances. Software is not a source of revenue, it is a support tool.

Application Production Approach

- Prime is confident that the entire OC/SBC product range can be accommodated within the PRIME minicomputer range. The recent thrust towards the business computer market is partially a result of this and the growing attractiveness of this vast market.

- The PRIME approach to the business system market is, however, identical to that used for the earlier scientific/engineering markets : namely that hardware excellence allied to good basic system software will be adequate attraction for prospects and that users and software houses must jointly assume the responsibility of the applications development and systems implementation.

- It is therefore not meaningful to talk of business application software production methodology with PRIME, since this area is essentially left in the hands of third parties. The vast majority (80%) of PRIME's current applications, for historical reasons, are written in FORTRAN, but now that a wider spectrum of high level languages is available, there will be a higher proportion of business-oriented language based applications.

- Given the current low level of European penetration by PRIME, the software support staff is centrally located pending the development of local market demand. This latter stage is not anticipated to occur for the next 2-3 years.

- As a general rule, PRIME is concerned with satisfying the customers system requirements without getting involved in the development and production of specific application software. Thus outside organisations are usually not contracted to develop software for PRIME, but deal directly with the end-user.

M. SPERRY UNIVAC

M. SPERRY UNIVAC

- Univac's re-entry into the small business computer market was a very late move, beginning with the February 1977 announcement of the BC/7. Aimed at the manufacturing and distribution markets, the product got off to a slow start, despite attractive standard application software and a new programming language (Escort).

- Univac's position in Western Europe is a strong one and centres on the UK, France, West Germany, the Nordic countries, Italy and Spain. The fastest growth has been registered by Saab Univac which aims to capture 25% of the Nordic market by 1980 and looks capable of achieving it, from a 9% share at the company formation in 1975.

- Until recently Univac had little or no involvement with small business computers and is not instantly thought of as a source of small computers. This is a problem that will not be easily solved.

General Approach Used in Supplying Application Software

- The current mix of applicational support to end users is composed as follows :

	USER	UNIVAC	THIRD PARTY
U.K.	30	40	30
France	35	30	35
W. Germany	40	20	40

The above appears to be mainly due to two factors :

(i) Univac is more strongly implanted in the UK than in other countries (International headquarters is located there).

(ii) West German users are far more self sufficient than in other countries.

- In the future it is hoped that users will absorb a higher percentage of their own application development requirements.

- Although Univac does not encourage the development of OEM agreements, they do try to work with software houses that have already developed packages in the areas of greatest support demand. When collaboration does occur it does not entail Univac taking the copyright (retained by the software house).

- The benefits that derive from a vertical marketing approach have not escaped Univac, who has historically organized their sales and support efforts along market areas that are close to industry groups. Univac intends to (i) move into specialized branch areas with small business systems (e.g. Hotels/Motels) (ii) enhance their manufacturing packages and (iii) develop a suite of construction packages.

- In general applications software is treated as a hardware support tool, although there is a recognition within the company that it can be a source of revenue also.

Application Support and Maintenance

- All affiliates receive the same standard package for adaptation to the local language, (which is carried out locally). The logic of the package does not usually change, however.

- Error correction and fault reporting follows the inverse path :

  - the initial attempt at solving the problem is made by the local office on the basis of problem reports (in the local language)

- if the fault persists, the description of the essor is translated into English and forwarded to the International headquarters.

- Licensed packages are not used, nor are they catalogued for reference when available from third party sources. The problem seems to be that there aren't that many third party sources interested yet in developing software for an anticipated growth of the Univac SBC/OC market.

Application Production Methods

- New applications are designed to cover 80% of the overall Univac SBC/OC requirement ; the intention is to be as comprehensive as possible, while remaining within a standard package. The traditional applications languages (COBOL, RPG II) are used for this, in general, but complex parameterizing is done in Assembler.

- Univac's approach has been to try to provide users with the tool(s) to help themselves as much as possible, through the use of the ESCORT language and extension/adaptation tools like PIXIE. ESCORT is a macro language that has been translated into French and German to allow nationals to write programs in their own language.

- The International Division spends approximately $1.5M per annum on application software development, with staff located in London (HQ), France and Germany. All of this has been spent in-house until now, but this may change in the future.

N. WANG EUROPE

N. WANG EUROPE

- Established in 1951 Wang has a background not unlike that of Hewlett-Packard, beginning as a specialist instrument maker and progressing to programmabale calculators, desk top computers and minicomputers. Unlike HP, Wang continued into word processing.

- Originallly Wang concentrated on the scientific and engineering/technical markets, but has since moved into commercial applications markets, in particular with turnkey minicomputer systems in the $50,000 range.

- Wang's turnover is approximately $200M. They try to avoid using OEMs in the small business area.

Approach Used in Supplying Application Software

- The responsibility for systems implementation lies mainly with the user who often deals with a software house. In certain instances Wang does provide installation support.

- Specialised software houses are used for industry expertise : e.g. in France, Informatique Medical for medical applications, Sogema for insurance etc. This is the trend of the future with more and more application being developed by the third parties.

- The policy towards these third parties varies at the moment. Ideally, Wang would like to control the contacts in a similar way to that of Kienzle by helping the clients to select the best house for the particular application considered.

- Wang provides compilers, BASIC, COBOL, RPG II, parameterized packages, standard packages and sometimes programme modules. Their cost is included in the total systems price. None of this is marketed under license agreement.

- Plans for improvement include standardisation of applications. Basic packages are to be customised, (for example, medical laboratories and Insurance companies).

- The company treats application software both as a generator of hardware sales and as a source of revenue. The objective is always to offer a total service, but presently it is mostly a generator of hardware sales.

Application Support and Maintenance

- Problems encountered centre on the accounting procedures (which vary from one country to the other). As a result all applications software for a given country is developed in that country. There is no transferability.

- In the U.S. the situation is much simpler. Management Planning, Manufacturing Management, Patient Billing systems and Dollar Park (which enables the customer to control an inventory of spare parts for cars) are all available. None of this can be used in Europe, but Wang also has a U.S. developed General Business Systems, (payroll, inventory etc) which can be partially used.

- The problem reporting and change requests are all handled centrally, when the product is produced by Wang. The majority of packages are not, however, and the responsibility lies with the software house, which also takes into account any industry sector variations that occur.

Applications Production Methods

- The SBC/OC product range is covered with several products, the PCS-II, the 2200VP and the 2200VS. As a result, different application software versions with restricted/extended functional capacity are necessary to cover the different hardware configurations. Functional capacity extensions are possible after the initial installation.

- Development plans for more broadly based application program product exist, and Wang is moving in the direction of greater flexibility for greater effectiveness. The service companies/houses will determine which tools will be used for the purpose of customizing the applications.

- New applications are being developed with traditional programming languages, and tools are marketed to enable end users to change/adapt/extend program products.

- The level of effort for the development of applications is very small in local countries; typically five to six people are involved: a support manager and a manager of Applications Software with a small, locally distributed staff. Wang spends approximately $5M per year on R&D, most of which is for hardware. No outside organisation is used for standard applications software development sub-contracting.

APPENDICES

I. Questionnaire Used

II. Summary Results

III. List of Vendor Contacts Interviewed

IV. Definitions

APPENDIX (I)

SURVEY OF APPLICATION SOFTWARE TECHNIQUES AND PRODUCTION METHODS

CO. NAME

ADDRESS

Tel. No

CONTACT(S)

I. GENERAL APPROACH USED IN SUPPLYING APPLICATIONS SOFTWARE (S/W)

I.1 In which of the following countries does your company deliver Small Business Computers (SBCs) or Office Computers (OCs) directly to end users ? :

□ UK
□ Benelux
□ Switzerland
□ France
□ W. Germany
□ Austria
□ Spain
□ Italy
□ Scandinavia

Others : ____________________

I.2. Whose is the main responsibility to produce application S/W ?

Now : □ User □ Ourselves □ Third party (C/SWH)
□ Other

Future □ User □ Ourselves □ Third party (C/SWH)
□ Other

I.3. If third party consultancies/software houses/systems houses are utilized what policy is used (describe) :

____

____

____

____

I.4. To what extent do you assist customers with design and implementation of their applications ?

□ Compiler only (C) □ Dec Table Process (D) □ Parameterised Packages (P) □ Standard Packages □ Program Module (M)

Other (X) ____________________ (Y) ____________________

Are any of the above marketed under licence agreement ?

____

What plans for improvement do you have ?

I.5. What problems (and solutions) have you found in using given application S/W in more than one European country ?

____

____

____

Which applications are notable in this respect ?

____

____

I.6. Does your company treat application S/W solely as a generator for H/W sales, as a source of revenue or for other purposes ?

____

____

2. APPLICATION AREAS COVERED

2.1. Does your organisation offer program products which offer :

□ transferability to follow-on hardware __________

□ applicability to SBC/OC products (same manufacturer) ______

□ change request procedures ________________

□ error correction and maintenance ____________

2.2. Do you have formal procedures for problem reporting and change requests for licensed program products ?

□ yes □ No

Are these procedures controlled :

□ locally □ centrally

2.3. Do you have a catalogue of licensed program products ?

□ yes □ No

For which countries does it apply ?________________

2.4. What specific plans do you have for multi-country applications S/W and how will these be implemented ? (Investigate the process of specification of program products and how national differences are handled)

2.5. What variations occur in your application software in its use for different industries ? (Investigate the process of specification and how national differences are handled).

3. PRODUCTION METHODS

3.1. Do you cover the SBC/OC product range with :

□ one product range □ several (name)________________

____

Could one application program product cover this total range or are versions with restricted/extended functional capacity necessary to cover the different hardware configurations ?

____

____

____

Are functional/capacity extensions possible after initial installations ?

□ Yes □ No

3.2. Do you have development plans for this type of program product If so in which direction are you moving ?

□ limited flexibility/low customisation for rapid implementation

□ great flexibility for greater effectiveness (flexibility means ability to cope with increasingly detailed specifications late in implementation period).

→ which tools are used for this purpose ?

____

____

3.3. Are new applications developed

□ with traditional programming languages (COBOL, RPG, Assembler, etc)

□ with new parameterized languages

→ ________________

Are tools marketed to enable end users to change/adapt/extend program products ? □ yes □ No

3.4. What level of effort is invested in application S/W production (man/months per application, number of staff).

____

____

Are the staff located in the countries served or centrally ?

____

____

3.5. Are outside organisations used for S/W development sub-contracting ? If so how is broad/international distribution arranged ?

____

____

____

Thankyou for your time. Would you like to receive a summary of this survey ? □ Yes □ No

APPENDIX (II)

	RESPONSE														
Q. NO.	QUESTION	BURROUGHS	CII-HB	DATA GENERAL	DEC	H-P	ICL	KIENZLE	MATRA	NCR	NIXDORF	PRIME	UNIVAC	WANG	COMMENTS
1.1.	In which countries do you sell SBC/OCs :														
	● Austria	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● Benelux	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● France	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
	● Germany	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	
	● Italy	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● Scandinavia	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● Spain	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● Switzerland	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	–	✓	
	● United Kingdom	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	
	● Others	–	✓	–	✓	✓	–	–	–	✓	–	✓	–	✓	
1.2.	Main responsibility for producing application S/W Now is :														O = either supplier only or user + third Party
	● Ourselves	✓	✓	–	✓	–	✓	✓	–	✓	O		✓	✓	
	● Users	X	–	✓	–	✓	✓	✓	✓	–	✓	✓	✓	✓	
	● Third Party	✓	✓	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	X = rarely
	● Others														
	In future will be :														
	● Ourselves	✓	✓	–	–	–	✓	✓	✓	✓	✓	–	✓	✓	
	● Users	–	–	✓	–	✓	–	✓	–	–	✓	✓	✓	✓	
	● Third Party	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓	✓	
	● Others														
1.4.	Application development tools available to users are :														
	● Compilers	✓	–	✓	✓	–	–	✓	–	–	x	✓	✓	✓	
	● Decision Table	–	–	–	–	–	–	✓	–	–	–	–	–	–	X = only in execution
	● Standard Packages	✓	✓	–	✓	✓	–	✓		✓	✓	✓	✓	✓	O = Utilities
	● Parameterised	✓	✓	–	✓	–	–	✓	✓	–	✓	–	–	✓	
	● Program Modules	–	✓	–	✓	–	–	✓	✓	–	–	–	✓	✓	
	● Others	–	–	O	–	–	–	–	–	–	–	–	–	–	
	Are any marketed under a licence agreement ? Y = Yes N = No	Y	Y	N	Y	N	N	Y	N	Y	N	Y	N	N	

SPONSE

QUESTION	BURROUGHS	CII-HB	DATA GENERAL	DEC	H-P	ICL	KIENZLE	MATRA	NCR	NIXDORF	PRIME	UNIVAC	WANG	COMMENTS
Is S/W a revenue or H/W sales generator S = SW ; H = H/W	S + H	S + H	H	S + H	H	S + H	H + S	H	S + H	H	H	S + H	S + H	
Do you offer application with :														
• Transferability	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
• Applicability to SBC/OC	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
• Change request proc.	✓	✓	-	✓	-	-	✓	✓	✓	✓	-	✓	✓	
• Error correct/maint.	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Do you have change requ. and pb. reporting for licensed program product? Y = Yes N = No	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	N	N	
Are procedures controlled locally (L) or centrally (C)	C	C + L	C	L	C	L	L	-	O	C	C + L	C + L	C	O = informal procedures
Do you have a catalogue of licensed prog. prod.?	Y	Y	Y	Y	N	Y	Y	N	Y	N	Y	N	Y	W = Worldwide E = Western Europe R = Rest of the World
• Applies to which countries ?	W	W	W	W	-	W	E	-	W	-	W	-	W	
Is SBC/OC range covered with one product range (1) or several (S)	I	I	S(2)	S	I	S	S	I	S	S	I	I	S	
Could one applic. prod. cover this range (Y) or are several versions necessary (V) ?	Y	Y	V	Y	Y	V	V	Y	V	V	Y	Y	V	
Are functional/capacity extensions possible after initial installation ?	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y = Yes
Do you prefer limited flexibility (L) or high (H)?	H	L	*	H	H	L	H	*	L	H	H	O	H	O = can go either way * = company confidential
Are new applications developed with traditional lang. (L) or new ones (N)?	T	T	T	T	T	T	T	S	T	T	T	T	T	S = Super language, not a traditional one
Can users change program products ? Y = Yes N = No	Y	Y	Y	Y	Y	N	Y	Y	Y	N	Y	Y	Y	
Is development staff loc. centrally (C) or in count. served (S) ?	S	S + C	C	C	C	C	C + S	L	C + L	C	C	C + L	L	

INPUT

APPENDIX (III)

LIST OF VENDOR CONTACTS INTERVIEWED

COMPANY	TOWN/COUNTRY	FUNCTION
• BURROUGHS	Feltham (U.K.)	Mgr. Applications Systems Development Centre
• CII-HONEYWELL-BULL	Paris	Mgr. Applications Development Division
• DATA GENERAL	Paris	Mgr. Software Development
• DIGITAL EQUIP. CORP.	Geneva	Software Development Manager
• HEWLETT PACKARD	Geneva	Marketing Manager
• " "	"	Systems Engineer
• ICL	Slough (U.K.)	Product Line Requirements Manager (Software)
• KIENZLE APPARATE	Villengen (W.G.)	Product Planning Mgr.
• MATRA INFORMATIQUE	Montrouge (France)	Manager, Studies and Product Development
• NCR	Brussels	Management Office
• "	Brussels	Small EDP Systems Manager
• NIXDORF	Lauzanne (Switz)	Sales Dept
• "	Zurich (Switz)	Software Mgr Ind./Commerce
• "	Paderborn (W.G.)	Software Production
• PHILIPS	Apeldoorn (Nether.)	Business Planning Mgr
• PRIME	Hounslow (U.K.)	Mgr. Scient. & Operations Systems
• "	Hounslow (U.K.)	Intern. Marketing Support
• "	Hounslow (U.K.)	Senior Consultant Scientific Applications
• SPERRY UNIVAC	London	Technical Operations Director (Business Systems Marketing)
• WANG EUROPE	Brussels	Mgr. Software Development

APPENDIX IV

DEFINITIONS

Acceptance Test: A specified review operation for assuring the completeness, validity, and reliability of a developed system before it is put into production.

Automatic Flowcharting: A software system that constructs program flowcharts from source programs themselves. May also provide other documentation, such as file layouts and program code cross-referencing. The most widely used example is Autoflow II.

Automated Source Program Control: Provides audit trail of history of program changes, and ensures that revisions are made systematically. May provide security from unauthorized access to source programs. Examples are: LIBRARIAN, PANVALET, SLICK.

Chief Programmer Team: A hierarchically organized project team, with specialized roles for team members. Chief programmer (highly experienced) designs system, codes top level modules and other particularly difficult modules, and reads all coding produced by the team. Assistant chief programmer shares these duties and provides backup. Librarian provides clerical support, maintains files and libraries, submits jobs, keypunches. Members at various experience levels enter and exit team as required.

Data Base Language: A software system intended to manage and maintain data in a non-redundant structure for the purpose of being processed by multiple applications. Examples are: ADABAS, DMS, IDMS, IMS, Model 204, System 2000, TOTAL.

Data Dictionary: A means for locating data elements in a data base structure, and for determining what the desired item's relationships are to other data elements in the file, and what resources are used by the various tasks employing that data.

Decision Table: A matrix technique for systematically relating actions to conditions, singly and in combinations.

File Management Language: Generalized software to permit access to, and retrieval from, already existing files, usually for a single application. Examples are: Mark IV, Easytrieve, Data Analyzer, Inquire, Ramis, Quickjob.

HIPO Charts: "Hierarchy plus Input-Process-Output" is a graphic technique for separating control flow structures from data flow structures, using top-down design techniques and specified documentation standards.

Logic Analyzer: Examines interactions between objects, properties, and relationships of a systems specification to determine redundancies and inconsistencies. An example is PSL/PSA, discussed in the text.

On-Line Coding: Allows the programmer to interact directly with the computer via a CRT terminal during program development and testing, eliminating turnaround time associated with batch processing.

Pre-Processor: Expands a shorthand version of a programming language into the full version of the language. Optionally, may also optimize code, produce test data, and/or enforce programming standards. Examples are: Meta Cobol, Work Ten, Optimizer III.

Program Auditing: The technique of incorporating an outside reviewer into the program development team at an early stage to assure that corporate standards and specifications are met, and that the final product will be auditable.

Program Execution Monitor: A testing aid and capacity planning tool that analyzes the use of hardware system resources by the program segment that is using them, producing frequency statistics to aid in optimizing the highly used sections of code or in reducing operating bottlenecks. Examples are: LOOK, TSA, CUE, QCM (all software-driven). Hardware-driven examples are: MS and DYNAPROBE.

Programmer Workbench: A PDP-11 based system developed by Bell Laboratories that provides a set of support routines for module development, library maintenance, documentation, and testing.

Programming Standards: Specification of a restricted subset of language and/or design capabilities and formats that are permitted to be used, thus helping to assure compatibility of systems.

Structured Coding: "GO-TO-Less" programming; a program whose modules contain only: sequences of two or more operations; or, conditional branches to one of two operations and returns (IF-THEN-ELSE); or, repetition of an operation while a conditon is true (DO-WHILE). Program modules are restricted to a single entry and a single exit.

Structured Compiler Languages: Languages designed to support structured programming techniques. Examples are: ALGOL, PASCAL, C, EUCLID, ALPHARD.

Structured Walk-Thru: A series of schedule reviews, each with different objectives and each occurring at different times in the application development cycle. Emphasis is put on error detection rather than correction.

Test Data Generator: A file generation program that translates a generalized description of the data base, file, record, and field characteristics into files meeting the desired specifications. Examples are: DATAMACS, PRO/TEST.

Top-Down Design: Organizes a system into a tree structure of program modules. Development and testing commences with the highest hierarchical level of generality and proceeds downward to as detailed a level of functional definition as necessary to produce a manageable work break down structure. Also known as step-wise refinement.

LEFT

彩

照

姓名：

Name

职务：

Post

单位：

Unit

No:

Date:

002

装得快 文具

APRIL2013