OCCASION

This publication has been made available to the public on the occasion of the 50th anniversary of the United Nations Industrial Development Organisation.

DISCLAIMER

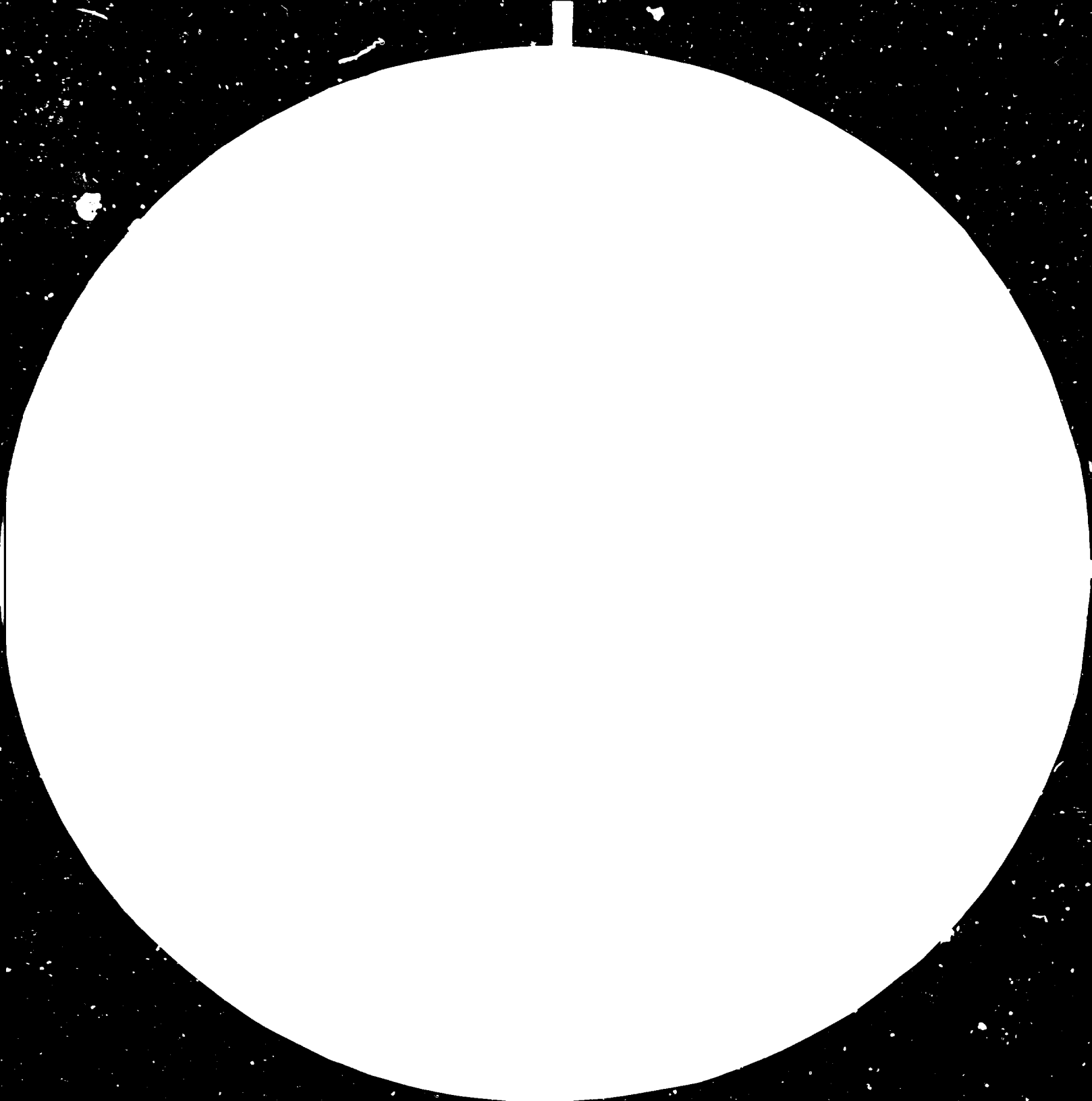
FAIR USE POLICY

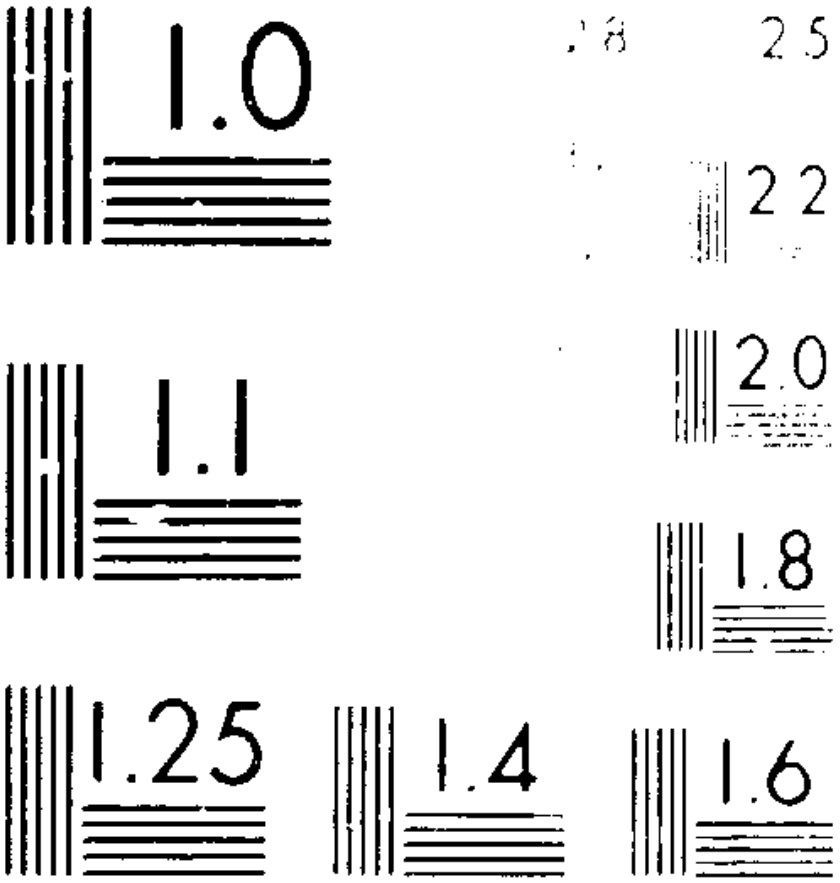
CONTACT

Please contact publications@unido.org for further information concerning UNIDO publications.

For more information about UNIDO, please visit us at www.unido.org

UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna International Centre, P.O. Box 300, 1400 Vienna, Austria
Tel: (+43-1) 26026-0 · www.unido.org · unido@unido.org




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5

12486

UNITED NATIONS
INDUSTRIAL DEVELOPMENT ORGANIZATION

Distr.
LIMITED

UNIDO/IS.383
27 April 1983

ENGLISH

PROBLEMS OF SOFTWARE DEVELOPMENT IN DEVELOPING COUNTRIES*

Prepared by the UNIDO secretariat

509

* This document has been reproduced without formal editing.

V.83-54942

ABBREVIATIONS

CPU	central processing unit
EEC	European Economic Community
EPC	European Patent Convention
EPO	European Patent Office
IBM	International Business Machines Corporation
NIC	newly industrializing country
p.a.	per annum
ROM	read-only memory
T + M	time and materials
WIPO	World Intellectual Property Organization

CONTENTS

Preface

This paper brings together those observations and findings on software development which are of relevance to developing countries and which formed a part of other documents prepared for UNIDO on the larger issue of computer technology.

The documents from which material has been taken are listed in annex II.

INTRODUCTION

The use of computers, whether as general purpose information processing systems or as micro components in specific product and systems applications, is growing at a tremendous rate in both developed and developing countries. Likewise, the capacity to produce computer software is also growing - by about 18 per cent annually on a global scale - but this is not sufficient to meet the growing demand since new applications for software are developing at more than double this rate.

While the developing countries at present do not constitute large markets in themselves, some of them like Argentina, Brazil, India and Mexico are already included in the top 40 import markets.[1/]

In considering the benefits of applying computer technology in developing countries, it is recommended that the developing countries do not attempt to reproduce the path already trodden by the developed countries in this field for a number of reasons. This is because the patchwork-like pattern of existing software is far from optimal, given the current (and expected future) state(s) of application requirements and hardware. Furthermore, developing countries require soft- and hardware adapted to their specific applications and purposes. Developing countries ought to place particular emphasis on developing software that will minimize the need to acquire expensive hardware, which is subject to constant technological change and variation.

It should be noted also that in some instances the revenue generated by software products can exceed the revenues generated by associated hardware products.

The sheer size of the market, coupled with its rapid expansion and impact on productivity, employment and industrial "revolutions", make the issue of promoting software self-reliance of significant interest for developing countries today.

1/ B.O. Suprowin, "The World Top 50 Computer Import Markets", Datamation, January 1981.

I. HARDWARE AND SOFTWARE - BASIC CONCEPTS

A computer constitutes a physical framework to manipulate and present information. Information is represented by strings of Os and 1s, which can be received, stored, processed and communicated at the speed of hundreds of thousands or millions of basic operations per second. A computer can also reach simple decisions, on the basis of results obtained earlier, to modify the way data is processed (a feat which essentially distinguishes it from simple electronic calculators).

The central processing unit (CPU) performs the basic arithmetic functions and supervises the entire system operations. In a microcomputer the central processing unit is a microprocessor: a single integrated circuit on a chip of silicon that is about a quarter of an inch on a side. Other silicon chips constitute the computer's primary memory, where instructions as well as data are stored. Other chips govern the input and output of data and carry out control operations. The chips are mounted on a plastic circuit board: a printed pattern of conductors interconnects the chips and supplies them with power.

Data can be entered into the computer by means of a keyboard or transferred into it from secondary storage on magnetic tapes or disks. The computer's output is displayed on a screen called a monitor. The output can also be printed on paper by a separate printer unit. The device called a modem (for modulator-demodulator) can be attached to convert the computer's digital signals into signals for transmissions over telephone lines.

The electronic elements and the various peripheral devices constitute the computer's hardware. The hardware can do nothing by itself; it requires the set of programmes, collectively called software.

Definitions of software differ considerably. This reflects the great variety of functions software is supposed to fulfill. But what its common purpose is and what distinguishes it from hardware could be defined as follows:

"... programmes that modify computer hardware and extend its function beyond the general purpose digital computer. Software includes, but is not limited to control programmes, executive supervisors, teleprocessing and communications monitors, application programmes, programming aides, languages, etc. Software could be delivered as a product, with or without large- or small-scale hardware, as a service through a time-sharing network, as one of the value-added components in a facilities management arrangement etc. I am defining software in the broadest sense possible, and I do so because the current market place requires it." 2/

In the present paper other definitions will be used as follows: 3/

A computer programme is a complete set of instructions to manipulate data during the operation of a processor.

Data is usually defined as information that relates to the outside world. (Data represents information.)

A data base is an accumulation of data that shares one or more common properties (for example, employee records of corporations or extensive compilations of publications and abstracts available for computer access through the world).

A programme is written either in source code or in what is called high level language instruction.

The central processor unit recognizes only a limited repertory of instructions, each of which must be presented as a pattern of binary digits.

It is possible to write a programme in this machine language - but the process is tedious and likely to result in many errors.

2/ Larry Welke, "The Origins of Software", Datamation, December 1980, p.127.

3/ The terminology used by Tipton V. Jennings in his paper "Protection of computer software", October 7, 1981, and quoted in "Licensing computer software", ID/WG.383/3, p.2.

The next higher level of language is an assembler in which symbols and words that are more easily remembered replace the patterns of binary digits. It recognizes each mnemonic instruction and translates it into the corresponding binary pattern. In some assembly languages, an entire sequence of instructions can be defined and called up by name. A programme written in assembler, however, must still specify individually each operation to be carried out and the programmer has to keep track of where in the memory each instruction and item of data is stored.

A high-level language relieves the programmer of having to memorize the instruction set and take into account the hardware configuration. Instead of instructing the processor where in primary memory to find the values to be added, the programmer specifies only the operation, e.g. A + B. The programme, having kept a record of the memory location, of the A and B variables, then generates a sequence of instructions in machine language.

Programmers generally write their programmes in high level languages. Examples of such languages are FORTRAN, COBOL, BASIC and PASCAL. Object Code is the machine-readable counterpart of a source code programme. It contains the strings of ones and zeros meaningful to a computer's electronic circuit and is the result of a compiler or interpreter programme which reads and processes source language instructions.

A flow chart is a computer diagram illustrating the logical progression of the steps and processes performed by a computer executing a programme.

Another item worth defining is the so-called computer firmware which has attributes of both hardware and software. Firmware is a sequence of computer control instructions (like software) which are built into some type of hardware device, e.g. a read-only-memory (ROM), which means its contents usually cannot be changed.

Computer software exists in many different forms of often significantly different attributes. It can be punched into a deck of computer cards, printed on paper, displayed on a tube (cathode ray tube), written as selected polarities on magnetic materials or transmitted as electrical impulses over

telephone lines.

Although the hardware of a computer ultimately determines its capacity for storing and processing data, the user seldom has occasion to deal with the hardware directly. A hierarchy of programmes, which together constitute the software of the computer, mediates between the user and the hardware.

The part of the software that is most closely associated with the hardware is the operating system that controls the computer's operations and manages the flow of data. The operating system mediates as well between the machine and the human operator as between the machine and an application programme that enables the computer to perform a specific task: calculating a payroll or editing a letter. Application programmes are ordinarily stored in secondary-memory media and are read into the primary memory media as they are needed for a particular application.

Application programmes are the ones that ultimately determine how effective a computer is in meeting human needs.

II. THE SOFTWARE SECTOR

The spread of applications for mainframe and microcomputers depends critically on the capacities available to develop, operate and maintain software, particularly applications software. In talking about micro-electronics-related innovations and industrial restructuring, it must be remembered that there is an overall shortage of programming capability in relation to the rapidly growing population of installed computers (mainframe, mini and micro) and that this particularly applies for programmers capable of devising machine codes for microsystems. This shortage of programmes needed to run computers and microelectronic programmable devices, the so-called software bottleneck, is in fact becoming an increasingly important constraint for attempts to expand the reach of microelectronic applications beyond their traditional confines. Thus, an analysis of the software market might permit a better understanding of why the microchip has been so "unexpectedly slow" to become that all-pervasive force as predicted

by the most renowned forecasts for more than 10 years.[4/]

In order to understand today's problems of software engineering, an analysis of the structure of the software industry is needed.

A first question would concern the origins of the software industry and how the industry relates to other parts of the information processing sector.

There are three people involved in information processing:

- The hardware manufacturer/provider;
- The software manufacturer/provider;
- The end-user.

Originally, these three different functions were united in one and the same person, or at least were part of the same team. The person with a computing problem (the end-user) built a machine (thereby becoming a hardware provider), then modified it to accomplish a specific task (thus creating software).

With the emergence of the computer industry, the end-user (the person with the computing problem) was separated out of the equation. Software was provided either by the hardware manufacturer or by an in-house programming team which the end-user hired specifically for this purpose.

Viewed from the end-user's perspective, this type of arrangement had three basic shortcomings:

First, neither of the two groups of software producers were sufficiently acquainted and in tune with the end-user's needs so that needs and software hardly ever intermeshed completely. In other words, it took a long time and it cost a lot to bring software sufficiently close to end-users' needs

---

4/ "Restructuring world industry in a period of crisis - the role of innovations", UNIDO/IS.2[illegible].

and it was never known whether the system really would work let alone whether it would be able to incorporate necessary adaptations over time.

Second, because of the high salaries of programmers, the cost of software development and engineering rapidly increased to very high levels. That is why both end-users and hardware manufacturers had some common interest in reducing the role of highly paid programmers and in replacing them as far as possible by other arrangements.

Third, it did not take end-users too long to realize that software received from hardware producers always contained significant doses of built-in hardware dependencies, i.e. the need to rely on maintenance and repair services and spare parts, but also to remain loyal to the original system when expanding computing capacity.

Hardware producers also increasingly had reason to feel uneasy with this arrangement. They used to give away software as part of the price paid for the hardware system, but came under increasing pressure to cover the tremendous costs of software development. The famous "unbundling decision" of IBM in June 1969 was the logical outcome of this concern. Some would argue that by explicitly separating software and hardware costs, this strategic move of IBM - probably in contradiction to the motivations prevailing then - did contribute considerably to the emergence of an independent software sector.

Obviously this is only part of the story. In order to understand the causes of the present software crisis which, as indicated above, is one of the key constraints to the application of microelectronics technology to industrial products and processes, further inquiry into the history of the software market-place is needed. Various events could be cited:

- In 1956, IBM started its independent data centre due to a judgement by the US Government that required IBM to treat its Service Bureau Division as a separate, but still wholly owned subsidiary rather than as an activity adjunct to its computer sales.

- In 1959, largely due to the military and space agencies' programming needs, firms working under independent contracts emerged. Their position was strengthened when IBM had difficulties in bringing its third generation computer equipment to the market-place in 1963-1964.

- A further impetus to the development of independent software firms was the rise of the time-sharing industry in 1966-1967.

- Finally, the attempts at taking end-user programmes and modifying them for attempted multi-installation use, a practice that began in 1967-1968, is notable if only for its failure rate.

It is not easy to uncover the logic underlying these events. As L. Welke states: "... the software product market place occurred not because of a grand design, a technological breakthrough, or the genius of any one individual or company. It developed bit by bit and piece by piece, with form chosen only sometimes by technology, with standards dictated by economic necessity, with risk taken out of ignorance, and with the rewards sometimes going to the perverse as well as to those who just persevered."[5/]

Nevertheless, it seems possible to identify some common threads running through each of these events. Focus will be directed on three of them:

First, the hardware manufacturers' concern to reduce the cost of software development and maintenance not only caused changes in pricing policy, but also the development of new modes of software production and engineering (the so-called "software packages") and new forms of subcontracting, starting from software conversion, right up to the development of specific applications packages. In addition, the end-users' concern to reduce or to eliminate the extremely expensive in-house programming teams increased the drive to reduce the importance of highly skilled and lavishly paid programmers.

Second, with the destabilization of the established oligopoly, and the intensifying technological race between hardware manufacturers, the gap between ever new generations of computer hardware on the one hand,

---

5/ Larry Welke, op. cit.

and available software on the other, dramatically increased. This applied to both systems and applications software. Hardware manufacturers were unable to cope with this problem and thus had no choice but to accept and sometimes even to actively promote the emergence of independent software firms.

Third, firms interested in gaining access to the larger information and services market-place soon realized that software could serve as the ideal entry-level business. The reasons are fairly evident:

(a) The relatively low barriers to entry. Compared with starting a hardware company, for instance, software requires little initial investment.

(b) The high rate of return and very attractive profit margins, particularly with multiple sales of software packages. In other words, software is still widely perceived, and probably rightly so, as a "get-rich-quick business".

(c) The great diversity of software requirements and the consequent segmentation of the market-place almost precludes domination by any one firm. The software market has so many needs and demands which furthermore are undergoing rapid change, that a small firm can easily carve a niche for itself. Attempts to homogenize effectively the software market for the sake of subordinating it to oligopolistic control are, of course, under way, but still they have a longer way to go.

III. TRENDS IN MARKET STRUCTURE

The dominant trend up to 1980 has been market segmentation of the software sector (see table). At first sight, it would seem as if very narrow limits exist for a standardization of this market and thus for attempts to increase concentration of control.

Table. The segmentation of the software market - large-scale versus mini-and microcomputer equipment

Large-scale equipment	Mini and microcomputer
Absorbs the majority of software outlays, but hardware markets are approaching saturation limits;	Still a minority market, but large growth potential;
The great majority of this market is geared to IBM machines and IBM-compatible peripherals and CPUs;	Because of great variety of hardware suppliers, exclusive orientation of software firms to a specific hardware firm or a group of products is seldom;
Consolidated patterns of competition and co-operation;	Rules of the game yet to be established, cutthroat competition dominant;
High profit margins per sale; Software is usually leased or licensed, not sold;	Due to aggressive pricing policy (market penetration prices), profit margins per sales are low;
Established rules for securing maintenance and technical support services and for pricing them (maintenance fee normally ranges between 12 and 15% p.a. of the basic lease price);	Deterioriating supply of maintenance and technical support services; no established rules for pricing them;
Highly sophisticated users; Cost burden of software production high, but still under control;	Mostly unsophisticated users. To write easy-to-use programmes is more difficult and thus more costly and time-consuming;
A global market, at least within the dominant economic blocks of the world economy (USA, Japan, EEC, South-East Asia and China, Latin America).	A great variety of local markets rather than a national market.

Yet, closer analysis shows that the reality is much more complex and that the software sector is undergoing significant change. Particularly in certain strategic areas, like "basic systems packages" and "software development systems", the prevailing trend points to increasing standardization and to globalization of the efforts to produce multi-use and multi-system software packages.

Most software firms are specialized in a particular segment of the market.

There are two exceptions to this rule:

- Firms developing "systems software" for various industries;

- A few of the bigger firms, mostly affiliates of major corporations, which strive to develop what they call "a full portfolio of software product offerings".

But overall, market segmentation is the dominant characteristic. This is reflected in the fact that possibilities to homogenize software are still very limited, particularly in applications software. The statement by James A. Unnerstall, General Manager, Indiana Standards Information Services Department, reflects an experience common to practically all industries: "The probability of your programme fitting another company's system up front is practically zero."6/

In fact, taking a programme tailored for one company's operations and revising it for general use requires an enormous investment, a figure that can exceed the original cost of software.7/

The table presents one aspect of the present degree of market segmentation, i.e. the split between large-scale equipment and mini- and microcomputer markets.

6/ "The Risky Business of Selling Home Grown Programmes", Business Week, 9 March 1981, p. 64.

7/ Ibid.

There are other types of segmentation. Two of them are of particular importance:

- Basic versus non-basic systems;
- Individually tailored software versus software packages.

Basic systems is a shorthand for describing those software systems which are in great demand. They include:

- Hardware utilization/performance measurement and accounting systems;
- Payroll and personnel information systems;
- Financial planning and profit analysis programmes;
- Project management and control systems;
- General accounting and integrated financial reporting systems.

It is on these basic systems that the greatest part of presently available programming capacities is focused. The main tendency in software business is specialization in a few fast-growth systems; the capacity to develop and market commercially attractive software packages and to guarantee reliable technical support.

Software Packages

Software packages are systems which have been developed to overcome the shortage of skilled personnel, especially programmers. Software packages are increasingly used by banks, insurance companies, but also by manufacturing firms. According to a recent review by the American Bankers Association, in 1981 for instance, more than half of all software in United States banks was provided in package form.[8/]

8/ Frederic G. Withington, "The Golden Age of Packaged Software", Datamation, December 1980, p. 131.

Although the use of software packages is increasing in manufacturing firms, the limitations to a wide-range application of packages still seem to be considerable.[9/] In absolute terms, this market is of considerable importance. In the United States, for instance, users of information processing systems spent about $US2.5 billion in 1979 on software packages (excluding contract software). About two-thirds of this sum went to the computer manufacturers and one-third to the independent software industry.[10/]

There are two types of software packages:

- Application packages
- Packaged systems programmes.

Application packages include, in order of sales volumes presently realized in the United States, general ledger packages, payroll and personnel records. For these basic management functions, reprogramming is a periodic necessity. The traditional approach would be to delegate this job to in-house programming teams or to subcontract it to independent programmers. Yet even for big firms, the costs of this approach rapidly becomes prohibitive so that a package is more attractive. Very small user firms acquiring their first computers have no professional programmers; at most they might hire a couple of part-time ones. Without packages (or systems so easy to use that professional training is unnecessary), the small user could never consider acquiring a computer at all.

Manufacturing packages, on the other hand, with fewer sales so far, seem to have a considerable growth potential in the future, for instance in the field of testing, measuring and analytical instruments; industrial electronic equipment such as motor controls, numerical controls, inspection systems, sequence controls; robots and semiconductor production equipment.[11/]

9/ For a detailed discussion see "Picking and Perfecting the Packages. Banking, Insurance and Manufacturing Executives Talk about Selecting, Modifying and Using Software." Datamation, December 1980, pp. 139-148, especially the interventions by Larry D. Woods, Manager of Special Purpose Computing with Deere & Co., Moline, Illinois.

10/ Figures are taken from Frederic G. Withington, op.cit., and include captive development of software packages by IBM and other mainframe computer manufacturers.

11/ Frederic G. Withington op.cit.

Packaged systems programmes include data bases, data dictionaries, programme development aids and schedules, and protocol translations that permit interlinking of computer systems and terminal complexes (including word processors) from various vendors.

The largest independent companies in the package business are all primarily producers of systems programmes. Demand for system packages is huge and is bound to increase: very few users would even think of preparing their own systems programmes any more, and the computer manufacturers can satisfy only a minority (at best) of the diverse demands of their customers.

Further, new opportunities for systems programmes keep appearing which means that systems packages will continue to be an essential component of the software market.

IV. A REVIEW OF THE EXPERIENCE OF SOFTWARE PRODUCTION

The present state-of-the-art in data processing in industrialized countries is not worth imitating.[12/] Through rationalization of hardware production the hardware/software-cost-ratio is now 20:80 and is predicted to be 10:90 in 1985. The programming crews in industry and public administration are spending 80/90 per cent of their time on software maintenance and correction. The value of installed applications software worldwide is about $US50 billion. Nearly all of these software systems are badly structured and difficult to maintain. These systems could be called "polluted" software.

To reduce the maintenance costs and to facilitate adaptation to the market and changes in technology and law the polluted software has to be restructured and partly rewritten.

This polluted software was not recognized as such when applications software began to be developed. It is now known from empirical and theoretical evidence, that there are inherent limitations which have not been overcome.

12/ Recommendations on measures to be taken to organize software houses and production of software in developing countries prepared for UNIDO by Hans-Jochen Schneider in May and June 1982.

There are indications that the best software created has an average failure rate of about 0.15 per cent, i.e., that existing operating systems, compilers and data base management systems, holding 300,000 to 600,000 statements, contain on average not less than 500 to 1,000 mistakes.

The experience of industrialized countries in producing complex software systems reveals the dangers of imitations. Developing countries still have the chance not to copy the same failures and not to run into the same troubles as the industrialized countries. It is emphasized that a government strategy and policy is urgently needed to prevent developing countries producing totally polluted software. The governments of developing countries have to react very fast because of the penetration of microelectronics and related software that has now started in these countries.

In order to cope with this situation, systems should be disaggregated, if necessary in an artificial way, into different components running on different computers in so-called distributed or loosely coupled systems.

User participation in the problem solution process should be as early as possible, i.e., that the user for whom the system is designed should be involved in the problem solution process from the beginning. The question is how to do this in an efficient and effective way. The system designer and programmer must have the software tools to be able to produce quickly a prototype system. With this so-called rapid prototyping strategy the designer can give the user an idea of the future system after only 10 to 15 per cent of the time required to develop the whole system. In the past acceptance tests showed far too late the failures and misunderstandings originating from the design process.

A skeleton of interfaces between different software components should be defined for the following reasons. If the data processing environment over a period of 20 to 30 years is considered, it can be seen that reliance cannot be placed on operating systems, programming languages or data base management systems. The only thing that can be relied on are so-called virtual interfaces, the building skeleton of interfaces. An example is the so-called

virtual terminal interfaces. The functions of the virtual terminal are specified without regard to a specific terminal. The keyboard functions, the function codes, the characters, the digits and the special characters etc. are specified without allocating special signals. The adaptation of the interfaces from the virtual interface to the existing physical interfaces is done by software. The skeleton of interfaces allows easy change of equipment and software components.

V. PROTECTION OF SOFTWARE

The development of the computer software industry was paralleled by the development of intellectual property protection of software.

From the outset, the computer software industry in the United States sought suitable legal protection of its property embodied in the software, with the following three basic goals:

(i) Adequate protection of financial investment in software development;

(ii) Technological progress resulting from full dissemination of software information;

(iii) Public benefit from new applications of computer technology.[13/]

In view of the above, three basic means have been explored for the most effective protection: trade secrecy, patent law and copyright law.

Patent Law

The US Patent and Trademark Office (PTO) released on 14 October 1980 Guidelines on Computer Protection which provide for the possibility of obtaining patents and copyrights for computer programmes.

The Guidelines indicate that the rejection of claims for the application of computer programmes are to be limited to cases in which the claims pertain

---

13/ See, for example, "Intellectual Property Protection for Computer Programmes, are Patents now obtainable?" 26 Cath U.L. Rev.835/1977. "Computer Programme Protection: the need to legislate a solution" 54 Cornell L.Rev.486 (1969).

solely to a mathematical algorithm or formula, a method of calculation, a method of doing business, an abstract intellectual concept or a collection of printed matter.[14/] The Guidelines include, as an example, a specific claim reciting a "base set" of programme instructions which would be rejected as defining nothing more than the abstract intellectual concept of a programmer.

The claims that define a process, apparatus (machine or article of manufacture) or composition of matter, or an improvement of any of these, and involve the operation of a programmed computer, are acceptable (under §101) so long as they do not directly or indirectly recite a mathematical algorithm.[15/]

Clauses that directly or indirectly recite mathematical formulae or algorithms are to be accepted "if claims implement or apply the formula in a structure or process which, when considered as a whole, are performing a function which the patent laws were designed to protect, e.g. transforming or reducing an article to a different state or thing".

Finally, the Guidelines point out that clauses in a patent application must be considered as a whole and can no longer be "directed" into old and new components for the purpose of analysis under §101.

Copyright Law

A new amended law signed on 12 December 1980 significantly clarifies the scope of copyright protection for computer programmes.

According to this copyright law a computer programme is: "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result". By such a definition, protection is extended to both source code and object code, and thus owners of copyright on computer programmes could prevent the unauthorized copying of the programme, including

---

14/ Richard H. Smith and Robert J. Gaybrick "Rules for Safeguarding Computer Programmes Clarified", Legal Times of Washington.

15/ Ibid.

the right to prevent the making of derivative works of the programmes.

It should be mentioned,however, that under United States law, a reproduction of a computer programme which is not fixed in a tangible medium does not represent a copy of that programme.16/

Furthermore, copyright laws have been criticized for protecting only against copying the expression and for not preventing the unauthorized use of a programme to control the operation of a computer.

In view of this, many owners of computer software have turned to using proprietary markings and non-disclosure agreements as the preferred mode of protection. Unfortunarely, trade secret protection is unsuitable for mass-distributed software.

Member countries of the European Economic Community are also parties to the European Patent Convention (EPC) which provides for the establishment of a single European Patent Office (EPO), and a uniform procedural system for centralized filing, searching, examination and opposition as well as a European Patent, which, when granted, results in a series of individual patents under the laws of the EPC countries designated by the applicant.

From the point of view of obtaining patent protection for computer software, two articles of the EPC are crucial, that is Article 52 (2) defining categories not included in the definition of an invention: (a) discoveries, scientific theories and mathematical methods;(b) aesthetic creations; (c) schemes, rules and methods for performing mutual acts, playing games or doing business and programmes for computers; and (d) presentation of information;and Article 52 (3) stating that the above categories are excluded only to the extent that the application relates to such subject matter or activities "as such".

Although the examinations in the European Patent Office began only in June 1979 and practice has not yet evolved, the Guidelines for Examination in the EPO indicate the likely results.

16/ See "Protection of Computer Software" by Tipton D. Jennings and Data Case System, Inc. v. JS and A Group, Inc., 480F. Sapp (N.D. I 11. 1979).

Chapter IV of the Guidelines states that "computer programmes need not necessarily be an abstract entity but may also appear in terms of a process for operating a computer or a record, for instance on magnetic tape. As regards the mathematical method, it is said that, for example, a shortest method of division is not patentable, but a calculating machine designed to operate accordingly may well be patentable. All in all, for the time being, one must come to the conclusion that computer-related applications which are not of evident technical substance will be thoroughly revised against the article 52 (2), prohibitions".17/

Finally, it should be mentioned that the resolution of patentability of software products, computer programmes or data bases under EPC does not end with a granting of a European patent. That patent must then be enforced in the various national legal systems.

In Japan, on the initiative of the Ministry of International Trade and Industry, it was decided that legal protection of software products should be fully ensured.

In addition to the statutory requirements contained in the Patent Law, the Patent Office in Japan has established a set of "Examination Standards for Inventions relating to Computer Programmes" which has been in effect since March 1976.

The WIPO Model Provisions on the Protection of Computer Software is aimed at assisting countries in introducing certainty into their existing legislation and in harmonizing their national legislation with that of other countries.

The WIPO Model Provisions essentially adopt the combined patent, trade secret and copyright approach.

Section Five of the Model Provisions sets forth the type of protection needed for computer software. The owner of the rights in computer software can prevent any person from disclosing the computer software or from aiding

---

17/ Robert T. Reiling, "Patentability of computer programmes, a worldwide view".

in its disclosure before the programme is made public. The computer software owner may also prevent any person from allowing or aiding someone to have access to any apparatus storing or reproducing the computer software before the computer software is made public.

Under the Model Provisions, a proprietor of computer software is also given the right to prevent the copying of computer software, including the right to prevent the making of derivative works.

Furthermore, the Model Provisions permit the owner to prevent the actual use of a computer programme to control a machine with information processing capabilities and to prevent the storage of the programme in the memory of a computer. The owner can also prevent the sale, lease or licensing of computer software or objects storing the software, such as ROMs.

The Model Provisions are intended to supplement rather than provide protection of computer software under the patent, copyright or unfair competition laws of subscriber states.

VI. POTENTIALITIES FOR SOFTWARE PRODUCTION IN DEVELOPING COUNTRIES

Industrialized countries have about $US100-150 billion worldwide for system and applications software development. Hardware costs are still decreasing by a factor of 10 every 10 years; costs of highly skilled personnel, however, are increasing. The only way out of the so-called software crisis is to standardize software and support software development by software tools. The industrialized countries will have to reorganize and reconstruct their polluted software in the next decade. The developing countries now starting software production have the chance not to repeat all the mistakes made in the industrialized countries. They could use software tools for software production like machine tools were used in the industrialization process of the developed countries. One way is to define a so-called skeleton of software interfaces into which all software building blocks developed within a time period of 5-10 years could fit.

Developing countries could begin by supporting software development teams in government, university and industry environments. These teams could start with improving user interfaces of existing installations. As a follow-up, software tool-boxes should be developed and/or imported to produce well structured (and not polluted) applications software.

How should such activities in different types of developing countries be started? First of all governments have to define and implement an overall strategy for software production.

Observation of the data processing market in developing countries has revealed several remarkable points which, to some extent, are also valid for the industrialized countries:

- There are few software companies or other so-called third party (consulting) companies working successfully in the market;

- Hardware manufacturers and distributors are mainly active in the data processing market;

- Hardware is oversold and there is little appreciation of software and system (hardware and software) thinking;

- Complete solutions to problems tend to be imported into companies through hardware manufacturers and distributors in standard software packages;

- Normally, standard software packages do not totally fulfill user requirements;

- As a consequence it is the organization that has to adapt to data processing and not vice versa;

- Highly educated people tend to leave industry and academia and join hardware sellers;

- There is no motivation to establish third party companies (software houses, consultancy companies) due to the belief that software development is too difficult.

It should be realized that software production is a manufacturing, rather than a service activity nowadays. At the same time, software production is extremely labour-intensive and in developed economies the cost of software production is as high as 80 per cent of the average cost of the computer-based application. Thus, some developing countries have a substantial advantage in the software production field since, with properly qualified manpower available, they could produce software at approximately one fifth of the cost of producing it in developed countries. From this it can be surmised that software production in developing countries may be highly competitive. 18/

The considerations given above do not apply to all developing countries, but only to those where a certain level of programming education already exists, i.e. where there are universities, polytechnics or equivalent schools offering some education in programming which permits the introduction of advanced training in software production.

In comparing software production with other modern technology products that could be manufactured in developing countries, the low investment level needed for software production gives it a substantial advantage. The technological equipment required for the production of software consists mainly of a computer system or systems which, in many cases, may be rented from the supplier with the cost of rental even being offset by the income derived from the sales of the software.

The one, but substantial obstacle for software production in developing countries is the market for the products, since:

(a) the local market is usually limited, resulting from the limited scope of computer applications in developing countries, and cannot create the broad demand for full-scale production;

18/ For a comparison of costs in four developing countries, see figures 1 and 2 from an article by Ian Palmer in Computer Weekly reprinted in Microelectronics Monitor, no.3, July 1982, p. 37.

(b) the export market, mostly in developed countries, is difficult to enter without a proper marketing and product dissemination network. Besides, the users in developed countries have, in many cases, an unfounded prejudice concerning the quality of software produced in developing countries.

Observation mainly of the computer market shows the possibility of introducing application systems in developing countries in an inexpensive way. These systems can support, in a systematic manner, industry, commerce, public administration, banking and insurance companies in the management of masses of data (text processing, data base systems, information systems, statistical computation), control of processes (manufacturing, quality control, optmization procedures) and governmental and industry planning (model building, simulation, optimization).

In order to reach the appropriate software supply in developing countries, several objectives should be met. These are as follows:

(a) To establish self-supporting software production in developing countries;

(b) To train local staff in advanced programming technology;

(c) To create a basis for developing the local utilization of computers to solve optimization problems, inter alia, of small and medium-sized industry as well as other applications; and

(d) To promote the export of software from developing countries.

Indigenous software production is more immediately attractive than hardware manufacture. As has been pointed out elsewhere, the industry does not have a high demand for capital investment either directly or for physical infrastructure (with two exceptions, telecommunications and educational facilities) and is expected to grow extremely rapidly in the foreseeable future.[19/] In developing countries the special informatics applications requirements involving hardware and software could be met by importing the hardware (or components of it if

---

19/ D.P. Hanna in a paper prepared for Dublin Conference on Development: Informatics and Industrial Development, 9-13 March 1981.

local systems integration, etc. is appropriate) and developing the software locally. These special requirements derive from the following: high unemployment and resulting poverty levels, the small scale of industry allied to the fact that most people live in villages; the lack of adequate infrastructure to support large industrialized centres; the need to concentrate on rural development as a prerequisite of industrial development so as to generate adequate demand.[20/] The potential of foreign-produced hardware (when augmented with suitable locally produced software) for innovative local applications can be best judged by persons fully conversant with local conditions. Governments can take a number of measures to promote an indigenous software capacity. These might include the following: providing finance for acquisition of computers, training, etc.; tax exemptions on reserves set aside by software houses for use in connection with the inevitable modifications required on software as a result of experience in the field or of material changes in circumstances; setting up a programme register to avoid duplication of effort; making legislative provision for copyright of software, although not much progress has been made in this area as yet.[21/]

The two pivotal considerations for an indigenous software industry are: personnel training and identification of appropriate applications.

Suitable institutional arrangements to manage and co-ordinate indigenous software production (including supporting personnel training arrangements) and to ensure that the software produced is of a high quality and that it is fully compatible with national application priorities are a prerequisite.

Singapore is one country in which measures have been taken to restructure the economy to aim specifically at developing high-technology industries and the Government of Singapore has planned the development of a software industry in the next decade.[22/] Steps have been taken to ensure a steady supply of high quality personnel. Education curricula have been reviewed and new training institutions established. The development plan calls for the production of

---

20/ P.D. Jain, Appropriate Informatics. Paper prepared for Dublin Conference on Development: Informatics and Industrial Development, 9-13 March 1981.

21/ R.E. Kalman in a paper prepared for Dublin Conference on Development: Informatics and Industrial Development, 9-13 March 1981.

22/ Robert Ian, The Computer Knowledge Industry - a Look at the Economic Rationale of a New Phenomenon from the East. Paper prepared for Dublin Conference on Development: Informatics and Industrial Development, 9-13 March 1981.

10,000 highly trained persons by the mid-1980s. A new Institute of Systems Technology, jointly supported by the Governments of Japan and Singapore, will train software personnel and provide the related management courses. In the shaping of its informatics policy and identifying a niche in the informatics industry for a market product, Singapore recognized the impossibility of competing with Japan and the industrial giants of the developed nations in established products and marketing networks.

VII. CONSIDERATIONS IN THE PRODUCTION OF SOFTWARE IN DEVELOPING COUNTRIES

For developing countries to develop software production with the aim of attaining future self-reliance a number of factors should be taken into consideration.

The first is an awareness of the software problem of education and training. The above-mentioned example of Singapore shows the way: education curricula have been reviewed and new training institutions established. The software policy should be implemented in that way as software production can begin only where a certain level of programming education exists i.e. where universities, polytechnics or any other equivalent schools offer some training in programming. This level of education need not be high; but to produce software, a basic knowledge of programming is expected.

The next step is the creation of team(s) to produce software. This could be done either by identifying an existing organization (e.g. a university) or creating a new one.

These objectives can be reached by the creation of multi-function software centres.

Apart from software production the centres should be able to provide a wide variety of viable, sustainable and high quality services, such as training, consultancy, software services and even hardware product development. Such centres also could be usefully linked to an organization producing special purpose, custom-made microelectronic components.[23/]

23/ See also J.M. Oliphant "Microprocessor Applications in Developing Countries" (UNIDO/IS.351), Vienna 1982.

A centre should start with one general manager and one technical manager, several system analysts and programmers, one hardware maintenance person, an operator and a secretary. Government and international organizations should support the training and the software technology transfer and help the centre to be commercially independent after some years. After having reached a stabilized phase the centre could try to export software and services. The centre could function as a publicity organization to spread awareness of the services it offers and to spread information to decision makers. A group of experts, consultants and researchers should permanently support the centre

On the other hand specialists from one centre should support the users in all phases of the problem-solution process: fact finding, requirements and restrictions, specification, design, feasibility study, economic analysis, design inspection, programming, testing, documentation, installation, evaluation, maintenance and tuning. The users have to be involved in the problem-solution process from the beginning in a participatory manner. The centre and consultant companies,the so-called third parties, mediate between hardware companies and users. Normally they are much closer to and much more important for the users than the hardware companies.

Accordingly, the centre has to carry out systems development work and management functions. In general the centre should employ 5 to 50 persons; or up to 300 to 500 persons as an exception. As a rule, one quarter of the staff p forms back-up work, another quarter is occupied with management work and 50 per cent works on consultancy-oriented and software engineering problem solutions. A general manager or president and a vice-president form the top management. Project managers responsible for several projects together with project leaders responsible for one project form the middle management. The system-oriented personnel hierarchy contains the positions of the senior specialist, senior programmer, programmer and junior programmer.

The centre should be structured into at least four departments: software engineering; software development for mainframe and middle-size computers; software development for mini-and microcomputers and sales and customers support. Within the departments, responsibility should be distributed according to the special commercial branches, to the subject areas within the branches and types of computers of different manufacturers.

Internal standards for problem solutions in general, for project management and for all development phases of the software life cycle have to be fixed and software tools should follow these standards, so that the employees adhere to these recommendations. This channelled behaviour will guarantee that the software produced fulfills standards which ease the maintenance and updating of the programmes after the installation phase and prevents production of badly structured, polluted software.

To maintain a high standard in the quality of the software products staff of the software centres should regularly participate in training courses to improve their knowledge of software engineering and production and dealing with customers.

Another development possibility would be to link up the centre with a developed (or newly industrializing) country (NIC) partner for a two-way flow relationship. The developed country partner would identify specific software applications that could be channelled to the centre for development. The applications would be graded to match the developing capacity of the centre, full specifications and quality control would be provided. In addition, an initial training programme and supervisory assistance would be provided by the developed country/ NIC partner on a continuing basis over a one to two-year build-up period. Over time, centre would progress from the production of applications-project-specific software to the development of software packages of a more general nature in response to the need for development and the opportunities for initiatives at the country level (including locally conceived products).

It should be realized that software production is only one of many aspects of modern technology development, aimed at the development of computer applications.

Developing countries have stressed the need for computer applications to solve their development problems. These applications are one of the leading modern technologies because of their wide ranging impact and contribution to productivity. In order to create a methodology and policy resulting in the systematic introduction of computer applications to the economies of developing countries, some appreciation of the new technology should be created first inside the countries and at the same time a partial assessment of the needs of the countries should be made. It is a recognized fact that any action resulting in the application of computers should match the potentials and problems existing in developing countries.

The software industry requires integrated development facilities. Research and development, public purchase policies, training of managers and technicians and effective co-ordination between centres of education and research, industrial circles and government agencies are of great importance for the development of this industry.

Mechanisms for such a co-ordination and the creation of integrated development should be created at the national level, preferably in the form of a programme. This would result in the software industry being given a higher priority in national development plans and help to allocate adequate resources towards its development.

Thus to outline the programme the following procedure should be adopted:

First, the application problems to be solved should be determined. This includes a sectoral development approach based on the particular economy, aspirations and the long-range development priorities of the country. In a realistic approach it would appear necessary to prepare an inventory of the computer capabilities needed within the country. This, together with reliable and comprehensive information on the state-of-the-art and emerging trends in global informatic technology would create a base upon which to construct the programme.

Some centres (e.g. Mexico) even are introducing special laws and regulations. In any case, it should be borne in mind, that at each stage in the national development of computer applications from the initial creation of policy to the actual implementation of operational systems, a country can obtain a great deal of information, guidance and sharing of experience through international agencies. General referral points would be the Industrial Information Section of UNIDO and the Department of Co-operation, Intergovernmental Bureau for Informatics.

Also, as already stated elsewhere, group action on the part of developing countries may be recommended:

"Group actions may be in the form of exchange of experience and sharing of tasks, such as assessments and monitoring efforts. More importantly, group actions will provide an opportunity for adoption of group strategies. A formal or informal institutional effort in group strategies by developing countries may be required just as several developed countries are also proceeding to adopt group strategies."[24/]

VIII. PRODUCTION AND PROTECTION OF SOFTWARE IN DEVELOPING COUNTRIES

As regards legal protection, as far as can be ascertained, the degree of legal protection available to computer software, either by means of patent, trade secret or copyright in developing countries is very limited and is basically embodied in their national patent and copyright laws which with few exceptions (Brazil, India, Mexico) have not been amended since their adoption and enactment.

This situation raises a fundamental question, namely, whether it is in the interest of developing countries to extend legal protection to computer software.

UNIDO's preliminary investigation has shown that in the developing countries such as Argentina, Brazil, China, Egypt, India, Malaysia, Mexico, Republic of Korea and a few others, the potential for the development of a computer software industry exists; this industry may, eventually, become internationally competitive

---

24/ "Policy Responses to Technological Advances: Some Illustrative Cases" ID/WG.384/3, Vienna, 1982, p. 20.

thus requiring, in its own interest, measures of protection similar to the ones employed by the leading software producers in the industrialized countries.

The premature introduction of protection of software may cause more harm than good, particularly for the development of a software industry (if such an industry is to be developed). The Model Provisions of WIPO may be used as guidelines in those countries.

ANNEX I

SOFTWARE CONTRACTS

The growing software industry has adopted for its purposes a variety of contractual forms to cover the use of the computer software.

For reasons described earlier, licensing under patents, trade secrets or copyrights have become the most appropriate vehicle for utilizing software. This annex deals with the basic types of agreements, describes their main features and recommends options for technology regulatory agencies when dealing with these types of transaction.

Custom Software Contracts[a/]

Custom-made software contracts deal with any procurement of computer software, either alone or in conjunction with the acquisition of computer hardware and related products and services, which involve either the development of new products and services or the substantial modification of the existing programmes (supplied by either the vendor or the user).

The most important part of a contract for custom-made software is the development of a complete set of functional specifications for the software, i.e. a set of documents which describes the business functions that the software must perform in the context of the overall data processing system in sufficient detail so that the functional specifications can provide the basis for the standards of performance that will be used to evaluate the vendors' performance.

The functional specifications will generally include:

(a) Functional description of the package, that is:-
(i) all tasks the package must accomplish; (ii) all inputs;
(iii) all outputs; (iv) all processing requirements;
(v) all data files; (vi) volumes of activities and files;

---

a/ See "Licensing Computer Software: Basic Considerations as to Protection and Licensing of Computer Software and its Implication for Developing Countries" (ID/WG.383/3), Vienna, 1982, p.10.

(b) description of the hardware environment in which the package must operate, including: (i) storage restrictions; (ii) peripheral equipment restrictions; (iii) data transmission procedures and (iv) communication interface;

(c) description of the software environment within which the programme must reside including (i) specifications of the operating systems; (ii) the programming languages; (iii) other programmes with which the customized software must properly interface; (iv) any specific nomenclature system which must be used for programmes;

(d) statements concerning the performance of the software in relation to (i) its internal organization; (ii) its execution speed; (iii) its capability for enhancement and modifications; (iv) its error deduction properties; (v) its error correction and recovery properties and (vi) any restriction of the activities which the user must avoid;

(e) programming and documentation standards, including details as to (i) documentation content; (ii) quantity; (iii) forms, and (iv) the nature and extent of coding.

An important issue, and one which is specific to this type of agreement, is that of pricing. The least desirable form of pricing is a pure "time and materials" (T+M) contract, as in this type of agreement there is an increased risk that it will take longer than anticipated to deliver custom-made software.

Sometimes T+M contracts provide for the overall ceiling of the amounts the vendor can charge to the user; in those sitations the formula is close to a fixed price contract,which is usually the best formula from the user's point of view.

It is quite common that part of the fixed price (or T+M price) is held back by the user in order to encourage the vendor's co-operation.

In custom-made software agreements, the concept of liquidated damages as an incentive to performance is automatically and frequently used. It may be applied,for example,to: (a) unliquidated credits for late performance;

(b) delayed payments; (c) free machine time; (d) increased level of service; (e) temporary back-up personnel; (f) substitute processing.

Another feature of the custom-made software contract is the quality of the personnel. This should be specifically spelled out, as should be the responsibilities for project management and control. At present the most complicated software system requires extensive documentation and training necessitating extensive provisions.

As software develops, at least in some of the countries, a degree of legal protection, title to the software and related information (including design aspects) and rights to use such systems should be included in the contract.

The following are basic issues which should be clarified in the contracts (as the need arises):

(a) whether title and/or unlimited rights to use software should remain with the vendor;

(b) whether exclusive title to the software should remain with the user;

(c) the possibility for joint ownership;

(d) sole ownership by user with limited marketing rights granted to the vendor;

(e) sole ownership by vendor with limited use/marketing rights granted to user;

(f) sole ownership by vendor with royalties payable to user;

(g) sole ownership by vendor in return for reduced development charges, future services, etc.

As with other licensing agreements, this type of contract will usually include provisions related to the protection of software from intentional or inadvertent disclosure, third party infringement and acceptance testing which includes test procedures, acceptance testing, acceptance criteria and

the ultimate measure of suitability of software functions in relation to:

(a) the hardware and software system environment;

(b) the test data;

(c) the time period for testing;

(d) the degree of reliability

(e) the degree of accuracy;

(f) the response time and the turn around time for error correction.

Finally, as with other licensing contracts, the following provisions should be included:

(a) limitation of assignments;

(b) termination procedures;

(c) choice of law and venue;

(d) arbitration vs. litigation;

(e) limitations of liability;

(f) force majeure;

(g) offset rights;

(h) users' access to vendors, work product;

(i) future modifications and enhancements.

Agreements for Packaged Software[b/]

The so-called "packaged" software is ready-to-use software developed for use by more than one customer, usually with only minor adjustments to the users' individual needs. As a rule, packaged software is licensed rather than sold and is customarily non-exclusive and non-transferable.

There are four types of packaged software, depending on parties to the agreements, as described below.

---

b/ Ibid.

A. Developer (of the software) - end-user contract

The following would be the provisions included in this type of agreement:

(a) description of software (including provisions for updates and new versions);

(b) price and payment schedule;

(c) taxes;

(d) terms of agreement (this may include termination provisions in a perpetual licence);

(e) maintenance;

(f) proprietary protection (including third party infringements);

(g) Escrow arrangements for source (to secure services in case the vendor ceases to do business);

(h) ownership of user-made changes;

(i) documentation;

(j) training (of varying duration and scope depending on the complexity of the software);

(k) limitation of use (limiting the use of the programme to a single processing unit or a single location, or inside the user company);

(l) acceptance criteria;

(m) liquidated damages (these are not generally used in packaged software, however the concept may be useful in the case of "leak" of the software to third parties);

(n) warranties (may or may not be included, depending on the nature of the software);

(o) limitation of remedies (usually consequential and indirect damages are excluded).

B. Vendor (Licensor) - Original equipment manufacturer (OEM)

In addition to the provisions provided for the type of agreement described under A, the OEM type of agreement provides for volume price discounts and

authorization for sublicensing. The agreement spells out under which key conditions OEM is required to sublicence or cause the sublicence to be executed.

C. Vendor-Distributor Agreements

In addition to many of the foregoing conditions, the vendor-distributor agreements are contracts which usually contain provisions for pre-distribution inspection and post-distribution returns and may also include conditions not to compete by one or both parties; guaranteed order levels and production levels, etc.

D. Vendor-service bureau agreements

The additional clauses may include the establishment of a basis for payment as a function of amount of use. However, there may be minimum payments or flat rates. Furthermore, the vendor will usually require access to the licencee accounts and security arrangements. Continuous training will be of a more extensive and substantial nature. The licences cases may be of an exclusive nature.

IV. Suggestions as to how developing countries should approach the licensing of computer software.

The brief overview of the current status of protection of computer software and of current practice regarding licensing suggests technology registries, in countries where they exist, might deal with licensing agreements for computer software.

In developing countries, one is primarily concerned with non-protected computer software, and protection may only be available (either in a form of patent or copyright) in the next few years. This lack of legal protection in the user country leads to an important consideration by technology registries in terms of their attitude and position vis-à-vis:

- duration of the agreements;
- rights of use after expiration of the agreements;

- limitation of use;
- payment level.

These are the basic contractual elements that should be considered by technology registries.

It is suggested that agreements for use (licence) of computer software should be subject to scrutiny by technology registries in developing countries.

By local legislation technology registries in these countries are empowered to scrutinize such contracts: Argentina,[c/] India,[d/] Mexico,[e/] Philippines,[f/] Portugal,[g/] Spain.[h/] Although in other developing countries computer software contracts as such are not necessarily covered yet by the activities of technology registries, due to the increase in this type of contracts, registries will have to give them attention in future.

In terms of types of agreements, technology registries should deal with either packaged computer software agreements (which are probably more common) and/or custom-made software contracts.

The following are some basic suggestions as to how to approach the main contractual provisions. The suggestions cover both types of agreement.

1. Duration

In both cases, whether custom-made software or packaged software contracts, the duration of the licence should be limited and be equal to the minimum period of time required by the user (licencee) to absorb and use the software transferred. No perpetual agreement should be permitted since the technological development of this field is moving very fast.

---

c/ Law 21,617.

d/ Guidelines on Foreign Technology Collaborations.

e/ Law on Technology Transfer dated 11.01.1982.

f/ Decree 1520 of 1978.

g/ Decree 53/77.

h/ Decree 2343.

2. Payments

With respect to custom-made software agreements, it is suggested that the fixed price formula should be used, combined with very precise performance standards. Concerning packaged software, a one-time payment may be prefered, which should include, however, additional (improved) software.

3. Maintenance

In both types of agreements the extent and frequency of maintenance should be spelled out precisely including the payments for such services.

4. Training

The provisions for training especially in custom-made software agreements, should be extensive; in a packaged licence it is also essential.

5. Title to the Software

In the case of custom-made software, technology registries should insist on the users' sole title to the software (eventually with limited marketing rights by the vendor). In packaged software agreements, however, the title may be with the vendor for the duration of the agreement; and the users may have the right to use it freely in the scope originally granted them.

6. Third Party Infringements - Property Protection

Since in most developing countries, no legal protection can be granted to computer software, (except through trade secret), the licensors/vendors have to ensure that their software does not infringe third party rights.

7. Acceptance Criteria

These conditions are particularly important for custom-made software agreements; they are also of significance for packaged software and in both cases the criteria should be extensive and worked out in detail.

8. Liquidated Damages and Warranties

Both provisions are of significance, particularly for custom-made software; therefore a good deal of time and effort should go into the preparation of these clauses.

9. Documentation

This clause is of crucial importance for custom-made software.

10. Future Modifications and Enhancement

The licencees should secure rights of access to future modifications, particularly in the case of packaged software.

11. Rights of Use after Expiration of Contract Term

It is recommended that users have unlimited rights in using the software after the expiration of an agreement.

12. Limitations of Use

Particularly in packaged software agreements, many vendors try to limit the use of their software to the users' plant and/or location. It is suggested that in principle, such limitations should not be acceptable.

ANNEX II

List of documents, prepared by UNIDO, utilized for this paper

1. Draft notes on a mission to Thailand to examine prospects for the development of a software exports programme, by M. Radnor.

2. Informatics in the service of industrial development, Draft of a paper by R.J. Nolan.

3. Licensing Computer Software, Basic considerations as to protection and licensing of computer software and its implication for developing countries, (ID/WG.383/3).

4. Policy responses to technological advances: some illustrative cases, (ID/WG.384/3).

5. Proposals for the creation of software houses in developing countries as a part of information application network, prepared by K. Fialkowski.

6. Recommendations on measures to be taken to organize software houses and production of software in developing countries, prepared by Hans-Jochen Schneider.

7. Restructuring world industry in a period of crisis - the role of innovation, prepared by D. Ernst. (UNIDO/IS.285).

8. Selective micro-eletronic applications in developing countries, Mission report by K. Fialkowski.