Contributors to This Volume

S. E. GOODMAN
M. H. HALSTEAD
MONROE M. NEWBORN
AZRIEL ROSENFELD
PATRICK SUPPES
STUART ZWEBEN

Advances in COMPUTERS

EDITED BY

MARSHALL C. YOVITS

Department of Computer and Information Science
Ohio State University
Columbus, Ohio

VOLUME 18

ACADEMIC PRESS ■ New York ■ San Francisco ■ London—1979

A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1 7DX

LIBRARY OF CONGRESS CATALOG CARD NUMBER: 59–15761

ISBN 0–12–012118–2

PRINTED IN THE UNITED STATES OF AMERICA

79 80 81 82 9 8 7 6 5 4 3 2 1

Contents

Software in the Soviet Union: Progress and Problems

S. E. GOODMAN

Woodrow Wilson School of Public and International Affairs
Princeton University

1. Introduction

It is only within the last decade that the Soviets have really committed themselves to the production and use of complex general purpose computer systems on a scale large enough to pervade the national economy. This goal has made it necessary for the USSR to devote considerable effort to upgrading and expanding its software capabilities.

This paper is an attempt to provide a broad perspective on software development in the USSR. To this end, it will be convenient to classify loosely the factors that affect the production and use of software in the Soviet Union in terms of four categories:

(1) those that depend on hardware availability;

ADVANCES IN COMPUTERS, VOL. 18

ISBN 0-12-012118-2

(2) those that are related to priorities in the allocation of effort and other resources;

(3) those that are dependent on the nature of Soviet institutions and economic practices, i.e., systemic factors; and

(4) those that involve technology transfers from foreign sources.

Although these categories are neither independent nor mutually exclusive, they provide a useful framework for a survey and analysis.

We will try to show that the Soviets have made substantial progress in removing limitations due to hardware availability, some progress as a result of changes in priorities, and as yet relatively little progress in overcoming an assortment of complex systemic problems that affect the development of software. Consequently, the Soviets will continue to borrow from foreign software technology, and they are now better equipped and motivated to do so.

Soviet software progress and problems cannot be understood on a technical basis alone. Relevant economic and political aspects have to be examined to present a more complete picture. The USSR has permeated technology and economics with politics, and our survey and analysis must discuss software in the context of this overall environment. Although the Soviet situation is extreme, it is not unique. Software engineering and management goes beyond the technical details of program code everywhere. In the last dozen years, Western literature has contained many articles that deal with social and economic aspects of software (e.g., Infotech, 1972; Boehm, 1975; Bauer, 1975; Horowitz, 1975; Buxton *et al.*, 1976; Infotech, 1976; Myers, 1976; Wegner, 1977). The national and international level discussions in this paper are logical extentions of this. We are so used to our own environment that most of us do not think about its advantages or disadvantages relative to other systemic arrangements.

Any serious study of the Soviet software industry[1] must involve some implicit and explicit comparisons with its US counterpart. In most aspects, the Soviets come out a poor second. This is because the peculiarities of the software sector tend to highlight Soviet weaknesses and American strengths. One should be careful not to extrapolate these comparisons over a broader economic spectrum.

It is difficult enough to write about the development and economics of software under the best of circumstances. It is particularly difficult when coupled with the special problems that afflict the study of the USSR. To help come to grips with this combination, an effort has been made to use as many sources as possible. These include a couple of thousand books and articles from the open literature (newspapers, marketing brochures,

[1] We shall use the term "software industry" to denote broadly the totality of a nation's software capacity.

trade journals, research reports, the scientific and technical literature, etc., through Fall of 1978). Of course, spacial limitations restrict the references to a small fraction of these. Unfortunately, the limited availability of Soviet and East European source material in the US necessitates the use of less-than-ideal hand-me-downs of various kinds. It is thus likely that the bibliography contains more "bugs" (e.g., misspelled names) than most. I have also had the benefit of a large number of private communications. Assorted constraints make it necessary to limit most of the discussion to nonmilitary, general purpose computing.

2. A Survey of Soviet Software

During the last ten years the USSR and its CEMA[2] allies have designed, developed, and put into production a family of upward compatible third-generation computers known as the Unified System (ES) or Ryad.[3] This system is an effective functional duplication of the IBM S/360 series, and provides Soviet users with unprecedented quantities of reasonably good general purpose hardware. The development of the Unified System is a watershed in Soviet thinking on software, and it reflects a major commitment by the Party and government to the widespread use of digital computers in the national economy. The appearance of the first Ryad production models in 1972 marks a clear turning point in Soviet software development.

2.1 Soviet Software Before 1972[4]

Although the USSR was the first country on continental Europe to build a working stored program digital computer (the MESM in 1951), and quickly put a model into serial production (the Strela in 1953), the Soviets have been slow to appreciate the value of computers for applications other than small- and medium-scale scientific/engineering computations. Little effort was made to produce large quantities of suitable hardware intended for widespread general purpose use. A business machines industry was essentially nonexistent, as was a body of consumers who had the per-

[2] The Council for Economic Mutual Assistance is composed primarily of Bulgaria, Czechoslovakia, German Democratic Republic (GDR), Hungary, Poland, and the USSR. Cuba, Mongolia, Romania, and Vietnam also have affiliations.

[3] ES is a transliterated abbreviation of Edinaya Sistema, the Russian for Unified System. The Cyrillic abbreviation and an alternate transliteration YeS are also commonly used. Language differences among the participating countries produce other variants; for example, the Polish abbreviation is JS. Ryad (alternate transliteration: Riad) is the Russian word for "row" or "series." The prefix R is sometimes used to designate computer models.

[4] Broad coverage of Soviet software before Ryad can be found in First AU Conf. Prog. (1968); Second AU Conf. Prog. (1970); Ershov (1969); Drexhage (1976); and Ershov and Shura-Bura (1976).

ceived need and priority to obtain such equipment. Before the early 1960s the military and scientific/engineering communities were the only influential customers with an interest in computing. However both were less enamoured with computers than their American counterparts, and the Soviet industry developed only to the extent where it could respond to this relatively limited demand.

By 1971 less than 20 of the approximately 60 known computer models had been serially produced with more than 100 units apiece (Rudins, 1970; Davis and Goodman, 1978). The vast majority of these were small- to medium-scale second-generation machines, some of which were still in production during the Ninth Five-Year Plan (1971–75) (Myasnikov, 1977).

As of 1971, there were less than 2000 medium- and large-scale second-generation machines in the USSR,[5] in contrast with the much larger number and variety in the West. Furthermore, the West had many more smaller computers. For example, by late 1963 IBM had built 14,000 1400 series machines (OECD, 1969), almost twice the total number of computers in the USSR in 1970. Thus the population of experienced programmers in the USSR remained relatively small, and there was a particularly critical shortage of modern systems programmers who had worked on large, complex, multifaceted software systems. This was compounded by the failure of the Soviet educational system and the computer manufacturers to provide the kind of hands-on, intensive practical training that was common in the US.

Two handicaps shared by all Soviet computer models were a lack of adequate primary storage and the state of peripheral technology (Ware, 1960; Godliba and Skovorodin, 1967; Judy, 1967; Rudins, 1970; Ershov and Shura-Bura, 1976). Installations usually had 1–32K words of core memory. The most reliable and commonly used forms of input/output were paper tape and typewriter console. Card readers, printers, and their associated paper products were of poor quality and reliability. Until the mid-1960s alphanumeric printers and CRT displays were essentially nonexistent; printers were numeric and used narrow paper. Secondary storage was on poor quality tape and drum units. For all practical purposes, disk storage did not exist in the USSR until Ryad. Tapes could not reliably store information for much longer than a month. Additional reliability in input/output and secondary storage often had to be bought

[5] Most of these were Ural-14 (1965), Minsk-32 (1968), and M-222 (1969) computers. Performance was in the 30–50K operations/sec range for scientific mixes. All three machines were relative latecomers to the period under discussion. The largest Soviet computer built in quantity before 1977 was the BESM-6 (1965), comparable to the CDC 3600 in CPU performance (Ershov, 1975). Over 100 were in use by 1972. All four machines were in production during most of the Ninth Five-Year Plan.

through duplication of hardware or redundant storage of information. For example, the 16-track magnetic tapes for the Minsk-22[6] had six tracks for data, two for parity checks, and the remaining eight tracks simply duplicated the first eight as an apparently necessary safeguard. Perhaps most importantly, Soviet peripherals did not offer convenient means for software exchange. Punched tape, with its limitations with regard to correcting and maintaining software, was more commonly used than punched cards. Magnetic tapes often could not be interchanged and used on two ostensibly identical tape drives.

One consequence of this was that almost all programming was done in machine (binary) or assembly language. By the late 1960s translators for a few languages were available for all of the more popular computer models, but they were not generally used. A good compiler could take up most of core, and the programmer could not get his program listed on his numeric printer anyway. Thus there was a strong bias that favored the "efficiency" of machine or assembly language programming. Clearly some of this bias arose from real considerations, but some of it reflected the same sort of dubious "professional" factors that perpetuate the use of assembly language in the West. It also helped make a skilled programmer a relatively rare and widely sought after employee in the USSR. Enterprises competing for talent would ingeniously create new job titles with higher benefits.

General purpose data processing and industrial applications were retarded the most by computing conditions. A severe handicap, in addition to those already mentioned, was the lack of an upward compatible family of computers with variable word length. Efforts to create such a family, the Ural-10 (Ural-11, -14, and -16) series and early ASVT models (M-1000, -2000, -3000), did not work out well (Davis and Goodman, 1978). The hardware situation and the use of machine language inhibited the development of software that would permit computers to be used for nonscientific applications by large numbers of people having little technical training. As a result, the hardware that did exist was often underutilized.

The fact remains, however, that by 1970 the USSR contained between 7000 and 10,000 computers and they could not be used at all without software.[7] While this figure may be small when compared to the almost 40,000 installed computers in the United States in 1967 (OECD, 1969), it

[6] The Minsk machines were the yeoman general purpose computers in the USSR before Ryad, with a production span covering the period 1962–1975. In addition to the Minsk-32, there were the earlier -2, -22, -22M, and -23 models (all rated at about 5K operations/sec). Well over 2000 of these machines were built and many of them are in use today.

[7] Our estimates of the Soviet computer inventory tend to be higher than most others, e.g., Berenyi (1970), Cave (1977).

was still large enough to necessitate a substantial effort and commitment of skilled technical people.

Much of the past Soviet systems software effort has been in programming languages. This is reflected in the large proportion of the open publications devoted to this area, and is consistent with the given hardware constraints, the relatively formal academic orientation of Soviet software research personnel, and the historical pattern followed in the West. Something like 50 different higher level languages can be identified from the literature. Many are experimental and have had virtually no impact beyond their development groups.

Most of the more widely used pre-Ryad programming languages were based on ALGOL-60. The popularity of this language is understandable in light of the European role in its creation, the fact that most Soviet programmers have had extensive training in mathematics, and its intended use for scientific/engineering applications. Compiler development began in the early 1960s and ALGOL-60 became available for most computer models after 1963 (Ershov and Shura-Bura, 1976). FORTRAN was also available for at least the Minsk machines, the M-220, and the BESM-6 from the mid-to-late 1960s.

Soviet use of ALGOL-60 has been characterized by a number of home-grown variants (Drexhage, 1976). ALGAMS and MALGOL are designed explicitly for use on slow, small-memory systems. ALGEC and ALGEM have supplementary features that make them more suitable but still not very convenient for use in economic applications. ALGOS appears to have been an experimental language for the description of computer systems. ALGOL–COBOL (Kitov *et al.*, 1968) is a clear hybrid for data processing. ALPHA (Ershov, 1966) and ANALITIK (Glushkov *et al.*, 1971b) are nontrivial extensions, the latter for interactive numeric computations. There was essentially no subsequent revision of these languages after the appearance of ALGOL-68.

A survey of the Soviet open literature on programming languages before 1970 reveals none that were particularly well suited for economic and industrial planning, business data processing, or large integrated systems like airline reservations or command and control systems. Attributes crucial to such applications, like good input/output and report generation capabilities, were just not available. For all practical purposes, the more widely used programming languages in the USSR during this period were only good for scientific and engineering computations.

Interest in the more widely used United States programming languages was not insignificant before Ryad. FORTRAN was used at quite a few installations in the USSR and Eastern Europe. No fully satisfactory reason is apparent, but the Soviet software community was strong in its opposition to the use of COBOL before 1966. However, government interest in

general purpose data processing increased significantly during the Eighth Five-Year Plan (1966–1970), and serious attention has since been paid to COBOL (Myasnikov, 1972). This includes an early effort to set up a minimal compatible COBOL set for Soviet use (Babenko *et al.*, 1968). Other languages, including SNOBOL and LISP, attracted scattered adherents. The Norwegian general purpose simulation language, SIMULA 67, also became fairly popular.

Hardware limitations retarded the development and implementation of economically useful operating systems. Until the appearance of the BESM-6 in 1965, the simplicity and limited flexibility of the available CPUs and peripherals did not necessitate the development and use of sophisticated systems software. This was reinforced by the failure of computer manufacturers to develop and distribute such products and by the lack of support services for either software or hardware (Gladkov, 1970; Novikov, 1972).

As a result, users had to develop all but the most basic utility programs to enable the installation to function adequately in a single program mode. Most programs could not be shared with other computer centers having the same CPU model because of local modifications that were made in the course of hardware self-maintenance and the lack of uniform peripheral equipment.

Gradually, conditions and perceptions improved and a number of packages of utility routines were eventually put together for the more commonly used machines. Later, multiprogramming batch systems were built for the larger computers such as the Minsk-32, the Ural-11, -14, and -16. At least three different operating systems were developed for the BESM-6. The multiplicity of BESM-6 system projects is partially the result of the nontransferability of any one system to all installations, and a lack of communication between installations. Some of these efforts appear to have been "crash" projects that did not permit the full utilization of the software development talent available. All of these systems are primitive by Western standards and did not appear until long after hardware deliveries had begun. We do not know how widely they are used or how well they are supported. Maintenance of even centrally developed systems was largely the responsibility of the user installation.

As might be expected, Soviet attempts to develop time-sharing systems were severely constrained by hardware. The USSR was deficient in every aspect of hardware needed for this mode of computing. A further handicap was the poor state of supporting technology such as ground and satellite communications. Data transmission by telegraph line at 50–150 bits/sec is still common in the Soviet Union (Leonov, 1966; Kudryavsteva, 1976a).

There were a few pre-Ryad time-sharing projects (Doncov, 1971). The

best known of these are the AIST project in Novosibirsk and the Sirena airline passenger reservation system. Neither has done well (Doncov, 1971; Drexhage, 1976; Aviation Week, 1972). The BESM-6 operating system developed by the Institute of Applied Mathematics supported time sharing at the Academy of Sciences' computer center in Moscow (Bakharev *et al.*, 1970; Zadykhaylo *et al.*, 1970). It does not seem to have amounted to much either. Some strange little "time-sharing" systems (e.g., Bezhanova, 1970) were so limited as to be unworthy of the name.

There have been a few experimental multimachine configurations. The best known of these were the aforementioned AIST system and the Minsk-222, which was based on an assortment of Minsk-2 and Minsk-22 computers (Barsamian, 1968; Evreinov and Kosarev, 1970). Both projects were characterized by what could only be described as naive optimism in the form of unwarranted extrapolations and fatal underestimations.

With the exception of work in the area of scientific and technical computing, the open literature was notably lacking in descriptions of significant, implemented, and working applications software systems. No doubt some existed in security sensitive areas, and there is evidence that software was available to help control certain transportation networks, such as the national railway system (Petrov, 1969; Kharlonovich, 1971). However, one gets the strong impression that computers in the USSR were not being used to do much beyond straightforward highly localized tasks. The literature contained papers on the theoretical aspects of such applications as information systems, but this work was generally of a formal mathematical nature and contributed little to the actual implementation of major systems.

But things would soon change. The 1960s was a period of political and economic reevaluation with respect to the need for expanding the general purpose computing capability of the USSR. Soviet economic planners were distressed by falling growth rates and the rising percentage of nonproductive (e.g., clerical) workers. They were also having trouble controlling the sheer immensity and complexity of the economy. The Soviets were becoming increasingly aware of the economic and industrial potential of computing, and they were not oblivious to what was being done in the West. Public discussion of the use of computers, which had been widespread since the late 1950s, began to be supplemented by very high level Party endorsements (Holland, 1971b) and practical measures. Attention was directed toward such esthetically unexciting, but practically important, problems as the standardization of report forms, the elimination of human errors in data reporting, etc. The national economic planning process itself became a prime candidate for computerization (e.g., Glushkov, 1971a). Unlike the United States, which got into data processing through an established business machines industry characterized by a dynamic, fairly low-level, customer–vendor feedback relationship, most of the driving force behind the entry of the USSR came via push from the top of the economic hierarchy.

2.2 Soviet Software Since 1972

The most important necessary condition for upgrading the state of general purpose computing in the USSR was the creation of a modern upward compatible family of computers with adequate quantities of primary memory and a suitable assortment of peripherals. The first public announcement of what was to become the Unified System of Computers (ES EVM) came in 1967 (Kazansky, 1967). Within two years, the Soviet Union had enlisted the aid of its CEMA partners, and the decision was made to try to duplicate functionally the IBM S/360 by using the same architecture, instruction set, and channel interfaces.

The first production units of the Soviet–Bulgarian ES-1020 (20K operations/sec) were announced in early 1972. By the end of 1974, the Hungarian ES-1010 minicomputer, the Czech ES-1021 (40K operations/sec), the Soviet ES-1030 (100K operations/sec; a Polish version never went into serial production), and the GDR ES-1040 (320K operations/sec) were in production, providing the USSR and most of Eastern Europe with about 1000 small- and medium-scale machines per year as of late 1975. The two largest computers in the series were to suffer considerable delays. The ES-1050 (500 K operations/sec) would not go into production until 1975–1976, the ES-1060 (1.5M operations/sec) would not appear until late 1977 (Khatsenkov, 1977; Trud, 1978a). The 1010 and 1021 are not based on the S/360 architecture and are not program compatible with the other models. In addition to the basic CPU models, the CEMA countries have been producing a reasonable range of peripheral devices. Although most of this equipment is at the level of IBM products that existed during the second half of the 1960s, they represent a tremendous improvement over what was formerly available. A much more extensive discussion of Ryad can be found in Davis and Goodman (1978).

The policy to use the IBM instruction set and interfaces was clearly based on software considerations. This was perceived to be the safest and most expedient way to meet the high-priority national objective of getting an upward compatible family of general purpose computers into productive use in the national economy. The Soviets had failed in two previous attempts to produce such a family, and they must have been aware of, and frightened by, the major problems IBM had with S/360 software. There was no serious interest in, or perceived need for, pushing

the frontiers of the world state-of-the-art in computer technology. An obvious course of action was to use a tried and proven system from abroad. The clear choice was the IBM S/360. By appropriating the S/360 operating systems, they would be in a position to borrow the huge quantities of systems and applications programs that had been developed by IBM and its customers over many years. This would do much to circumvent the poor state of Soviet software and permit immediate utilization of the hardware. Although it seems that the Soviets greatly underestimated the technical difficulties of this plan, it has been followed with considerable success and represents one of the most impressive technology acquisitions in Soviet history.

There are several S/360 operating systems (e.g., IBM S/360, 1974), the two most important of which are the disk-oriented system DOS/360 and the much larger OS/360, which consists of several versions that together contain a few million instructions in a bewildering array of modules. A tremendous volume and variety of documentation and training aids are available for these systems. There was no effective way to deny either the software itself or the documentation to the CEMA countries. Much of this is in the public domain and can be sent anywhere without license. Sources of information include IBM itself, tens of thousands of user installations all over the world, and the open literature. Several CEMA countries have legally purchased some of the small- and medium-scale S/360 systems, which include the software and the opportunity to participate in SHARE, the major IBM user group. Soviet and East European computer scientists could also legitimately talk to Western counterparts at meetings, by using Western consultants, through exchange visits, etc. Furthermore, the Soviets have demonstrated that they can illegally obtain entire IBM computer systems if they are willing to try hard enough.

DOS/ES is the Ryad adaptation of the IBM S/360 DOS disk-oriented operating system. From the available literature, we cannot identify any major DOS/ES features that are not part of DOS/360 (IBM/DOS, 1971; ISOTIMPEX, 1973; IBM S/360, 1974; Drozdov *et al.*, 1976; GDR, 1976; Vasyuchkova *et al.*, 1977). Both systems are subdivided into control and processing programs. These further subdivide into supervisor, job control, initial program loader, linkage editor, librarian, sort/merge, utilities, and autotest modules. The DOS/360 system librarian includes a source statement library, a relocatable library, and a core image library, as does DOS/ES. Both will support up to one "background" partition in which programs are executed in stacked-job fashion, and two "foreground" partitions in which programs are operator initiated. Both support the same basic telecommunications access methods (BTAM and QTAM) and the same translators (assembler, FORTRAN, COBOL, PL/1, and RPG).

DOS/360 uses OLTEP (On Line Test Executive Program) to test input/output units; DOS/ES also uses OLTEP. The level of DOS/ES appears to be at or near the level of the final DOS/360 Release 26 of December 1971.

Similarly, OS/ES appears to be an adaptation of OS/360. It has three basic modes: PCP (Primary Control Program with no multiprogramming capability), MFT (Multiprogramming with a Fixed Number of Tasks), and MVT (Multiprogramming with a Variable Number of Tasks) (Larionov *et al.*, 1973; Peledov and Raykov, 1975; 1977; GDR, 1976). All handle up to 15 independent tasks. OS/ES supports translators for FORTRAN levels G and H and ALGOL 60. The levels of OS/ES seem to be around the IBM MFT and MVT Release 21 of August 1972. OS/ES MFT requires a minimum of 128K bytes of primary storage; OS/ES MVT needs at least 256K bytes (Naumov *et al.*, 1975). OS/ES is mentioned much less frequently in the literature than DOS/ES. No doubt this reflects on the fact that the great majority of Ryads are at the lower end of the line. It may also indicate serious problems in adapting OS/360 to the ES hardware and problems with the supply of adequate quantities of core storage (many ES systems were delivered with about half of the planned core memory capacity). It is possible that DOS/ES may have been the only Ryad operating system operationally available for a couple of years.

The ES assembly language, job control language, and operating system macros are identical with those of S/360 (references in last two paragraphs; Larionov, 1974; Mitrofanov and Odintsov, 1977). The Soviet literature preserves the style of IBM software documentation. Assorted error codes, messages, console commands, and software diagnostics were originally in English and identical to those used by IBM. Such things have since become available in Cyrillic, but we do not know if these are standard options. English error codes, etc., still seem to prevail.

Several observers who were very familiar with IBM S/360 systems software have been able to identify fine details in ES software that leave little doubt as to the source of the product and to the degree to which it was copied.

It is as yet unclear exactly how program-compatible the Ryad family members are with each other or with IBM products. Some serious testing by CDC of their purchased ES-1040 indicates a high level of IBM compatibility (Koenig, 1976). IBM systems software could be loaded and run on the 1040 without much trouble. It is not known if the Soviet-made Ryad hardware is as directly compatible with IBM software. The Soviets are investing literally thousands of man-years in the development of the Ryad operating systems (Rakovsky, 1978b), but we really do not know what all these people are doing. Hardware differences between the S/360 and Unified System, and between the different models of the Unified

System, may have made it necessary to adapt the IBM operating systems to each of the ES models.

Now that IBM no longer supports either DOS/360 or OS/360, the socialist countries are on their own as far as the maintenance and enhancement of the two systems is concerned. A recent "new version" is not especially impressive. The Scientific-Research Institute for Electronic Computers in Minsk, the institute that probably adapted DOS/360 to the ES-1020, came out with DOS-2/ES in 1976 (Kudryavsteva, 1976a). The most notable additions to DOS are an emulator for the Minsk-32 and some performance monitoring software. We do not know to what extent these enhancements are built into the operating system.

More generally, all of the ES operating systems have gone through several releases since they were introduced. We cannot really tell to what extent this reflects the addition of significant capability enhancements, academic (i.e., noncost effective) design optimizing perturbations, or simple accumulations of fixes. We suspect that the Soviets try not to tamper with the operating systems unless they have to in order to get them to function adequately. This may have been the case with an announced real-time supervisor known as SRV, an OS/ES coresident program for providing fast response in a real-time environment. SRV seems to be an adaptation of the IBM S/360 Real-Time Monitor (IBM RTM, 1970; Naumov, 1976), but, unlike the situations with DOS and OS, there are substantial differences.

The first USSR State Prize to be awarded for practical software work was announced at the end of 1978 (Trud, 1978b). In some ways it is remarkable that it took this long for the Soviet scientific and technical community to recognize the importance of software. The award was made for the Ryad operating systems. Not surprisingly, neither IBM nor people like F. P. Brooks, Jr., were named as co-winners.

It is important not to underestimate the achievements of the CEMA computer scientists in functionally duplicating S/360. They have mastered the quantity production of reasonably modern hardware and they did succeed in the formidable task of adapting the S/360 operating systems to this hardware. This is not to say that they did not have considerable help from external sources, or that they did a good, or fast, or imaginative job. In fact, the effort took them about as long as it took IBM in the first place, and they have yet to achieve S/360 quality and reliability standards across the Unified System product line. Nevertheless, they had the talent and resources to achieve the basic goals and, relative to their own past, they have acquired a much enhanced indigenous computing capability.

Between 1975 and 1977, the CEMA countries came out with several "interim" Ryad models that are essentially evolutionary upgrades of

some of the earlier machines. These include the Hungarian ES-1012 (another mini), the Soviet–Bulgarian ES-1022 (80K operations/sec), the Polish ES-1032 (200K operations/sec—the "real" Polish 1030), and the Soviet ES-1033 (200K operations/sec). In addition to these new CPU models, the CEMA countries have been producing a small, but steady, stream of new peripheral equipment (CSTAC II, 1978).

Although the current peripheral situation is much improved over the pre-Ryad era, complaints about shortages of peripheral devices and their associated paper products are still common (Lapshin, 1976; Ashastin, 1977; *SovMold,* 1978; Zhimerin, 1978).

The best evidence that the CEMA nations are basically satisfied with the policy of copying the IBM product line is the current effort to develop a new group of Ryad-2 models that are clearly intended to be a functional duplication of the IBM S/370 family (IBM S/370, 1976; Bratukhin *et al.,* 1976; CSTAC II, 1978; Davis and Goodman, 1978). By early 1977 most of the new models were well into the design stage. By the end of 1978, the Soviet ES-1035 was claimed to be in production (Sarapkin, 1978) and prototypes for at least the GDR ES-1055 (Robotron, 1978) and the Soviet ES-1045 (*Kommunist,* 1978) existed. The appearance of other prototypes and the initiation of serial production will probably be scattered over 1979–1982. A Ryad-3 project was recently announced (Pleshakov, 1978), but almost no details are available.

S/370-like features to be made available in the new Ryad-2 models include larger primary memory, semiconductor primary memory, virtual-storage capabilities, block-multiplexor channels, relocatable control storage, improved peripherals, and expanded timing and protection facilities. There are also plans for dual-processor systems and greatly expanded teleprocessing capabilities.

It is not clear if the Soviets intend to use the IBM S/370 operating systems to the same extent as they did those for S/360, or if they plan to build the Ryad-2 operating systems on the Ryad-1 OS/ES base. M. E. Rakovsky, a vice-chairman of the USSR State Planning Committee (Gosplan) and one of the highest ranking Soviet officials to be directly involved in the Ryad project on a continuing basis, has stated that "developing the Unified System's Ryad-1 operating software to the point where it will handle all the functional capabilities of the Unified System's higher-level Ryad-2 system will take between 1600 and 2000 man-years." He goes on to say that this effort will be carried out at "two institutes that employ a total of about 450 programmers" (Rakovsky, 1978b). There is also some reason to believe that GDR Robotron's new virtual operating system OS/ES 6.0 for the ES-1055 may be more of an original effort than was the ES-1040 operating system. Emulators have been announced as

part of the initial offerings for the two most advanced of the Ryad-2 models: one for running programs for DOS/ES on the 1055 (Dittert, 1978), and one for Minsk-32 programs on the 1035 (Kudryavsteva, 1976b).

The Unified System project has by no means absorbed the entire Soviet computer industry although this may seem to be the case since most of what appears in the Communist literature relates to Ryad. The joint CEMA effort has forced the Soviets to be more open about computer developments. The focus is on Ryad because it is by far the largest project and many of the others are officially classified. With respect to mainframe computers, the Unified System has roughly the same relative standing in the USSR as the IBM 360/370 series has in the United States; although most of the Soviet non-Ryad mainframes are smaller second-generation computers, whereas in the United States most of the non-IBM mainframes are technically competitive CDC, UNIVAC, Burroughs, etc., models.

The most extensive, openly announced non-Ryad production is primarily in the form of assorted machines built by the Ministry of Instrument Construction, Means of Automation, and Control Systems (Minpribor). Many are part of the ASVT series: M-4030, M-5000, M-6000, M-7000, M-400, M-40, and, most recently, the M-4030-1 (Naroditskaya, 1977). The medium-scale M-4030 is compatible with the Ryad family at the operating system level (Betelin *et al.*, 1975). The other models are minicomputers, the first of which, the M-5000 and M-6000, appeared in 1972–1973. The USSR also relies on imports from Hungary, Poland, and the United States to meet some of its minineeds. The ASVT line is widely used in Soviet industry and the literature indicates that a considerable amount of software has been developed for these machines. A great deal of substantive minicomputer related R&D is done in the Baltic states (e.g., SovEst, 1978).

A joint CEMA effort is currently in progress to consolidate the scattered member nation minicomputer activities by establishing a new SM (Sistema Malykh–Small System) family (Naumov, 1977). Of the four announced machines, the SM-1, -2, -3, and -4 (SM-5 and -6 announcements are expected in 1979), at least the first three were in production by mid-1978. Early indications are that a substantial amount of general purpose SM software is available, and that some form of ASVT program compatibility is possible (Filinov and Semik, 1977; Rezanov and Kostelyansky, 1977; TECHMASHEXPORT 1978a,b). These minis can be used with much of the peripheral equipment that has been developed for Ryad and ASVT.

Large scientific computers are under advanced development at the Institute of Precise Mechanics and Computer Engineering in Moscow, the developers of the BESM machines. Recently announced were the El'brus-1 and -2 (named after the highest mountain in Europe) (Burtsev, 1978; *MosPrav,* 1978). The El'brus-1 is thought to be based on the Burroughs architecture (Burtsev, 1975). This architecture is particularly well suited for ALGOL programming, the language greatly favored by Soviet computer scientists and scientific programmers. The El'brus-2 may be a loosely coupled collection of El'brus-1 machines. Past experience makes it likely that the Institute of Applied Mathematics in Moscow will participate in the development of its systems software. The new large computers will probably be produced in small numbers and many of these will be used at military and other restricted installations. The majority will eventually displace BESM-6s, so a BESM-6 emulator is likely to be an important element in early El'brus software offerings. By the time El'brus deliveries start, the receiving installations will have been using their BESM-6s for up to 15 yr. There will be considerable resistance to program conversion.

In addition to these large projects, there are a number of scattered smaller efforts that we know about. These include a few complete computer systems like the new line of RUTA models (Kasyukov, 1977) and the Nairi-4 (Meliksetyan, 1976), work on microcomputers [e.g., the S5-11 being built in Armenia (*Kommunist,* 1977)], and some hand-held "programmed keyboard computers" [e.g., the Electronika BZ-21 being built in Kiev (*Trud,* 1977)]. We do not know anything about the software that is being developed for these relatively unimportant machines, but it would not be surprising if the software offerings to early purchasers were very meager. Work on highly modular recursive machines is currently in a rudimentary stage in both the US and USSR (Glushkov *et al.,* 1974; *EE Times,* 1977). We have essentially no information on Soviet efforts to develop software for machines with this architecture.

Relative to their pre-Ryad past, the Soviets have clearly come a long way in correcting hardware and systems software deficiencies. There are now 25,000–30,000 computers in the USSR and at least half of them are respectably modern systems. The Unified System and the ASVT-4030, in particular, provide a large, common hardware and systems software base. But how productive have these machines been, and how well have they been integrated into the fiber of the national economy?

There is no question that the Soviets and their CEMA partners have given high priority to the use of computing as an important means to help modernize the economy and increase factor productivity. Indeed, the production of a large number of industrially useful programs began with the delivery of the first ES units. There are visions of great efficiencies to be achieved from the partition of this activity among the member countries (Rakovsky, 1978a), but since the various Eastern European economies differ considerably at the microeconomic level, one might well entertain doubts as to how well this will work out.

The availability of ES and ASVT hardware has resulted in something of

a minor software explosion. But this hardware is still backward by world standards. More important, the experience and personnel base necessary for the development of either large world-standard state-of-the-art software systems or large numbers of low-level everyday data processing programs is not something that can be put together in a short period. And perhaps, in the light of past Western practices, Soviet institutional structure tends to inhibit the customer-oriented design, development, and diffusion of software (see Section 3).

By far, the most extensive and prominent software activity in the USSR relates to what are collectively called automated control/management systems (ASU). The ASU spectrum runs from the simple no-direct-control monitoring of a small production process to a grand national automated data system for planning and controlling the economy of the Soviet Union. A broad range of economic/industrial ASUs is listed in Pevnev (1976). The creation of ASUs has become a major nationwide undertaking (e.g., *Ekongaz,* 1976; Zhimerin, 1978) and there are now literally hundreds of articles and books on ASUs appearing in the Soviet literature. A small sample of recent, general books includes Kuzin and Shohukin (1976), Pevnev (1976, which gives the best overall perspective), Pirmukhamedov (1976), Liberman (1978), and Mamikonov *et al.* (1978). Descriptions of specific ASUs under development and more general articles on the subject often appear in the periodicals *Sotsialisticheskaya industriya, Ekonomicheskaya gazeta,* and *Pribory i sistemy upravleniya.* A large number of industry-specific publications and the public press media also frequently carry articles on ASUs.

Although a great many articles describing a great many ASUs have appeared, by US standards these articles give little substantive information. It is thus difficult to do much more than list a lot of specific ASUs (the reader will be spared this) or present some tentative general observations. The Soviet interest in ASUs at all levels is genuine and serious. ASUs are being pushed vigorously from above, and there is a certain amount of desire at every level of the economic hierarchy to be part of the movement. Two major obstacles to the successful infusion of ASUs into the economy are the resistance of management, who are comfortable in their preautomation environment, and the inexperience of Soviet computer scientists and programmers. The Soviets have been making steady progress in overcoming both problems. Industrial managers are beginning to appreciate the potential of computers for doing tasks that people do not enjoy, but which need to be done, and the software specialists are beginning to think more realistically about simple, useful systems that are within their capabilities to build. This gradual convergence seems to be getting a lot of small systems built and used. With few exceptions (Myasnikov, 1974), it appears that most of this software is not widely disseminated, but used only locally (e.g., Zhimerin, 1978). None of this work is particularly imaginative by US standards, but there is no reason to expect it to be. As we shall discuss at greater length in the next section, the Soviet economic environment is conservative and introverted. The Soviets are cautiously and independently repeating much of the learning experience that took place in the US in the late 1950s and 1960s. It would be surprising if they were doing anything else.

The Soviets continue to expend considerable local effort on software for second-generation machines. Much of what is reported in the open literature is for the Minsk-32 (e.g., Kulakovskaya *et al.,* 1973; Zhukov, 1976; Vodnyy transport, 1977), but this must be true more generally since almost half of the computers in use in the USSR are of pre-Ryad manufacture.

The appropriation of most of S/360's software has eroded the past ALGOL orientation of high-level programming in the USSR. FORTRAN and PL/1 are now widely used. The government has pushed COBOL since 1969 and, given the emphasis on economic applications, it is not inconceivable that it could become the most widely used nonscientific language in the Soviet Union. Assorted CEMA computer centers have used LISP, SNOBOL, PASCAL, etc., and these languages will find their local advocates at Ryad installations. SIMULA-67 is an important simulation language (Shnayderman *et al.,* 1977). So far, we have seen little of the Soviet-designed high-level languages on ES systems, although Ryad translators for some of these do exist. Most of what is done with regard to these languages may be intended to prolong the usefulness of programs written for second-generation computers, or to permit users to remain in the familiar and comfortable environment of these older machines. This would explain why ALGAMS, an ALGOL-60 variant explicitly intended for slow machines with small primary memories, has been made available as an option with DOS/ES (Borodich *et al.,* 1977).

Although frequent allusions to time-sharing systems appear in the Soviet literature (e.g., Bratukhin *et al.,* 1976; Drozdov *et al.,* 1976; Sov-Ross, 1976), it is not clear what is readily available and used. None of the Ryad-1 or interim models has virtual storage, and storage capacities are marginal. Much of the telephone system in the USSR is not up to supporting the reliable transmission of large volumes of information beyond a few kilometers. We have seen no explicit mention of the TSO (time-sharing option) extension of OS/360 MVT, which IBM announced in November 1969. Not one of the 20 large "time-sharing centers" scheduled for completion in 1975 was fully operational by early 1977 (Rakovsky, 1977). Now the goal is to have six by 1980 (Zhimerin, 1978). User demand for time

sharing has only recently become serious enough to motivate more than academic exercises. The development of suitable hardware and software is currently being pursued (e.g., Bespalov and Strizhkov, 1978; Pervyshin, 1978), but most of this seems to be in rudimentary stages of development. Several experimental systems appear to be operational, and the ES-1033 with time-sharing capabilities has been advertised for sale in India (Elorg/Computronics, 1978) using OS/ES. However, widespread time-sharing use seems unlikely as long as most Ryad installations are equipped to use only DOS/ES. The enhanced capabilities expected with the Ryad-2 models should bring further progress.

There is considerable interest in database management systems (DBMS) in the USSR. Much of the work that is described as operational seems to be in the form of very low level, and localized, information retrieval systems. In the past, Soviet work in this area was severely constrained by a lack of disk and other secondary storage equipment, and by the poor state of I/O technology. Ryad and other developments have eased this situation somewhat, but there are still serious limitations. For example, most Soviet installations are still equipped with only 6–8 7.25 Mbyte IBM 2311-like disk configurations that do not allow the interleaving of data transfers. IBM 3330-like disk drives are expected to be available in moderate quantities for nonspecial (i.e., nonmilitary or non-Party) users in 1979–1980. The new capabilities expected with Ryad-2 models, especially block-multiplexor channels, should also be helpful.

Poland, the GDR, and the USSR are developing several DBMS based on Codasyl. The Soviet system is called OKA and was developed at the Institute of Cybernetics in Kiev (Andon *et al.,* 1977). OKA runs on OS/ES 4.0 (MFT and MVT) and has both a batch- and time-sharing mode. OKA is currently being field tested at at least one unknown installation. There is an All-Union working group following Codasyl in the USSR.

The Institute of Cybernetics in Kiev is working on two systems patterned after IBM IMS-2 and the experimental IBM relational DBMS System-R. The Soviet relational DBMS is called PALMA.

The Soviets have been developing several specialized DBMS. Most of the publicly acknowledged work is oriented toward economic planning, including a system that is being field-tested by Gosplan.

Soviet journals are filled with the description of experimental programming systems of various sorts. The relatively new (1975) journal *Programmirovanie* has become one of the most academically prestigious outlets for this work. It also seems to be the only major, regularly published, openly available, Soviet journal devoted exclusively to research in programming and software, although other journals, e.g., *Upravlyayuschiye sistemy i mashiny,* often contain informative articles. Few of these

articles are at the world state-of-the-art in software research (articles on Minsk-32 software appear with some regularity), and the theoretical work being done and the experimental systems being described seem consistent with the overall level of Soviet computing compared to that of the West and Japan. As far as we can tell, none of these products of Soviet research were offered as standard options with the early Ryad computers. Although many of these programming systems are being built to run on Ryads, it is not clear to what extent they are intended to become standard software options.

It is important to emphasize that we currently have a rather poor overall picture of how well or how extensively the Soviets have been using the software they have announced, or even what they have had for a long time. The lack of publications like *Datamation,* the very limited access we have had to Soviet installations, etc., make it difficult to say much more than we have.

3. Systemic Factors

In spite of the real progress and future promise offered by improved hardware availability and official recognition and support, there are some deeply rooted systemic problems that will continue to constrain severely the development of the Soviet software industry.

3.1 Software in the Context of the Soviet Economic System[8]

To a first approximation, the Soviet government/economy is organized in a hierarchical, treelike structure. The highest level node in the tree is the Council of Ministers (COM). The next levels represent a few score ministries, state committees, and other high administrative agencies. Then there are intermediate levels of Republic, branch, and department administration and management. Finally, the lower levels contain the institutes and enterprises that are responsible for R&D and the production and distribution of goods and services. This is a large bureaucratic hierarchy that encompasses every economic aspect of Soviet society. As a result of this vertical structure, and a very long and strong Russian bureaucratic tradition, much of the Soviet economy is unofficially partitioned into assorted domains or fiefdoms. These exist along ministerial, geographical, and personality divisions. People and institutions in this structure gener-

[8] Some general background references for this subsection include Granick (1961), Nove (1969), Bornstein and Fusfeld (1974), Kaiser (1976), Smith (1977), Berliner (1976), and Amann *et al.* (1977).

ally develop behavior patterns that please the higher level nodes in their domains.[9] This behavior may or may not coincide with the goal of providing high-quality service or products to customers.

Superimposed over this vertical hierarchy are a variety of horizontal relationships. The domains are not self-sufficient. In addition to directions from above, they get supplies and services from units in other domains and they, in turn, supply goods and services elsewhere. The centralized planning apparatus, in collaboration with other levels in the hierarchy, establishes suppliers and customers for almost every Soviet institute and enterprise. Although there is some flexibility in establishing these horizontal relationships, they are for the most part beyond the control of lower level management. One of the most important of the self-assigned tasks of the Communist Party is to expedite all sorts of government and economic activity. The Party intercedes to get things done. Although the Party organization is also subdivided into fiefdoms, it is more tightly controlled and operates freely across government/economic domains. Finally, there are the unofficial, sometimes illegal, horizontal arrangements that are often created to enable an enterprise to function successfully in spite of everything else.

In the centrally planned Soviet economy, there is no market or quasimarket mechanism to determine prices, product/service mixes, rewards, etc. For the most part, all of this is worked out at high levels and by a centrally controlled haggling process, although lower level management has been granted some degree of flexibility by gradual reforms since 1965. In this system quantity is stressed over quality, and production is stressed over service. Enterprises are told what to do. Failure to meet these imposed commitments can bring stiff penalties. Success is rewarded, but there is little opportunity for the high-risk, big-payoff, innovative entrepreneurial activity that is common in the US. The central planners do not like much activity of this sort because it is difficult to control.

The business practices that have evolved in this environment are not surprising. Enterprises are oriented toward the basic goal of fulfilling the performance indices that are given to them. These are usually narrowly defined quantitative quotas. Thus, for example, a computer producer's most important index may be the number of CPUs manufactured and a less important index may be the number of peripheral devices built. Rewards are paid for meeting the basic goals and for overfulfillment. Lists of suppliers and customers are provided by the planners. Plant management

[9] Of course, this behavior is not unique to the Soviet bureaucracy. It is characteristic of many bureaucracies, including most (if not all) of the US Government. However, in the USSR it is much more pervasive and there is no alternative to being part of this system.

will obviously give first priority to meeting the CPU production norm, then priority goes to the peripherals. They do not want to overdo things, because this year's successes may become next year's quotas. Furthermore, it is clearly in their own best interests to haggle with the planners for low quotas. Since customer satisfaction is of relatively minor importance (particularly if the customer is far away or under another ministry), management is not going to divert its resources to installation and maintenance unless it absolutely has to. There is also an obvious incentive to try to retain the status quo. Once a plant operation has started to function smoothly, there is no market pressure to force innovation, improved service, and new products, All these things mean finding new suppliers, changing equipment, and retraining personnel. They involve serious risk, and local management cannot control prices or suppliers to balance the risk.

There are strengths in this system. Central control and the powerful expediting role of the Party allow national resources to be concentrated in high-priority areas. The differences between the Minsk machines and Ryad show that much can be done on a respectably large scale once the high-level decisions have been made. Apathy disappears and labor quality improves on priority undertakings. Of course, the government and Party do not have the resources and cannot maintain enough pressure to do this sort of thing across the entire economy. Furthermore, it can be argued that some of this high-priority success occurs because these projects are really removed from the economic mainstream.

Software development would seem to circumvent some of the systemic difficulties that plague other products. Once the basic hardware exists at an installation, software work does not depend to any great extent on a continuing and timely flow of material supply from outside sources. Not surprisingly, Soviet enterprises have a tendency to avoid intercourse with and dependence on the outside. It would seem easier to develop an in-house software capability than one for spare parts or raw materials. It would also seem that commercial software houses would be able to provide better service than, say, a hardware maintenance group. The software house is not in the middle of a supply chain, the hardware maintenance group is. Since the software industry does not involve the distribution of material products, more casual horizontal vendor–customer relationships would be expected to be less troublesome for the central planners. Finally, the problem of the mass production of copies of a finished product is reduced almost to the point of nonexistence.

It would thus seem that software has been singularly blessed at both the macro- and microeconomic levels in the USSR. But high-level policy statements are not always easy to translate into practice, and the firm-

level advantages just described may be less advantageous than they appear. The development of a broad national software capability is not like the development of a capability to build computing hardware or armored personnel carriers. The nature of software development places considerable emphasis on traditional Soviet economic weaknesses and is not well suited to the "annual plan" form of management that is dominant in the USSR.

Before Ryad, hardware manufacturers did little to produce, upgrade, or distribute software. Few models existed in sufficient numbers to make possible a common software base of real economic importance. Repeated attempts to form user groups produced limited successes. Soviet security constraints restricted participation in sharing software for some models. Enterprises rarely exchanged programs. Contracts with research institutes to produce software products were often frustrating for the customer (e.g., Novikov, 1978). The research institute staff would be content with a prototype system that was not well tailored to the customer's needs. Most users had little recourse but to modify and maintain the programs on their own.

Conditions are gradually improving, but changes take time even where they are possible. One promising reform has been the establishment of the corporationlike production associations (Berliner, 1976; Gorlin, 1976).[10] These support the creation of relatively large and efficient computer centers that should be able to better serve the needs of the association and its component enterprises. The association may contain a research institute with its own software group. On the surface, at least, an association appears to be a more viable unit for the production and utilization of software, and one that might be able to deal more effectively with other firms. However, seemingly reasonable reforms in the past have actually produced results opposite those that were intended (e.g., Parrott, 1977). It is as yet too early to evaluate the impact of this reorganization, either in general or with respect to software development.

In the US there are a large number of companies that provide professional software services to customers. They range in size from giants like IBM to one-man firms. Some build systems and then convince users to buy them. Others ascertain customer needs, and then arrange to satisfy them. A variety of other services are also offered. Basically they are all trying to make a profit by showing their customers how to better utilize computers. To a considerable extent, the software vendors and service bureaus have created a market for themselves through aggressive selling and the competitive, customer-oriented, development of general purpose

[10] It is worth noting that enterprises engaged in the development of computer hardware were organized in loose research–production associations before they became generally fashionable.

and tailor-made products. There is probably no other sector of the American economy with such a rapid rate of incremental innovation.[11] The best firms make fortunes, the worst go out of business. Adam Smith would have been overjoyed with this industry.

The Soviets appear to have no real counterpart to these firms for the customer-oriented design, development, diffusion, and maintenance of software. One enterprise, the Tsentroprogrammsistem Scientific-Production Association in Kalinin, has been publicly identified as a producer of ES user software (Izmaylov, 1976; Ashastin, 1977; Myasnikov, 1977). This organization is under Minpribor. We assume that the Ministry of the Radio Industry, the manufacturer of Ryad in the USSR, has some central software facilities available because of legal responsibilities. Some research institutes, computer factories, and local organizations develop and service software, but complaints about their work is common (e.g., Zhimerin, 1978) and praise is rare. We know little about what any of these places are doing or how they function. The average Soviet computer user does not seem to have many places it can turn to for help. This is particularly true of installations that are not near major metropolitan areas (e.g., Davidzon, 1971; Letov, 1975; ZarVos, 1976).

The mere fact that we know so little about Soviet software firms is strong evidence that the volume and pace of their activities must be much below that of the American companies, or at least that benefits to users are limited by a lack of readily available information. Most American computer users are not very sophisticated and need to have their hands held by vendors and service companies. Most Soviet users are less sophisticated. It is inconceivable that the USSR has anything comparable to the American software companies that we do not know about, because then there is no way for the thousands of computer users in the Soviet Union to know about such services either. It is simply not the sort of thing that can be successfully carried on in secret. It must advertise in some way or it will not reach its customers.

Soviet installations are now pretty much on their own with regard to applications software. The open literature seems to confirm this with articles on how "Such-and-Such Production Enterprise" built an applications system for itself. There are few articles on how some research institute built something like a database management system that is now being used at scores of installations in a variety of ways. Currently, Soviet installations are building lots of fairly obvious local systems. This pace may actually slow down once these are up and running because there are few effective mechanisms for showing users what they might do next.

[11] Unfortunately, there appears to be no study of the US software industry that would enable us to be more specific.

Considerable potential for improvement exists. Although there do not seem to be many commercially developed software products in widespread, operational use, there have been quite a few articles on ASUs that are being developed with this goal (e.g., Bobko, 1977). Many of these are for management information systems intended for general or industry-specific users. There is high-level push for standardization of ASUs and the increased commercialization of software (Myasnikov, 1976; Zhimerin, 1978). Sooner or later, as they gain experience, some of the industrial and academic institutes that are doing software work will evolve into viable software houses. There are other possibilities. Right now computer installations are building up in-house software capabilities to meet their own needs. After a while there is bound to be some local surpluses of various kinds. We might see the gradual development of an unplanned trade in software products and programmers among enterprises. This sort of trading goes on all over the economy, and there is substantial opportunity for software. Finally, it is not inconceivable that a little unofficial free enterprise might evolve, as it does in plumbing and medicine. Small groups of bright young programmers might start soliciting moonlighting tasks.

The extent of the software service problem may go beyond applications software. We know little about how new operating systems releases are maintained or distributed to users, although in 1976 the All-Union Association Soyuz EVM Komplex was established, along with local affiliates like Zapad EVM Komplex in the Ukraine and Moldavia, to service centrally both hardware and software (Trofimchuk, 1977). We do not know who produces the new releases or how changes are made. The Soviets are not in the habit of soliciting or seriously considering a broad spectrum of customer feedback. The research institutes that maintain the ES operating systems may only communicate with a few prestigious computer centers. New releases are probably sent on tape to users[12] who are not likely to get much help should local problems arise. New releases may well necessitate considerable local reprogramming, particularly if the users modify the systems software to their own needs. Once an installation gets an operating system to work, there is a tendency to freeze it forever (Reifer, 1978).

There is a widespread users' attitude that accepts the software service situation and is thus a major obstacle to progress. The legendary tradition for endurance of the Russian people, and the vertical structure and shortage of resources that strongly favor the vendor's position, makes poor service a chronic and pervasive feature of life in the USSR. Improvement in the service aspects of the computer industry are taking place more slowly than are improvements in production. Most Soviet users can do

[12] This is actually an optimistic assumption. There is no evidence that new releases are not sent in a printed form that might require a major effort by users to put up on their machines.

little more than complain (complaints that would get at the core of the problem are politically unacceptable), and wait until the leadership perceives that the problem is serious enough to do something constructive. The Soviet Union has no counterparts to the market power of the average consumer and the flexibility for creating mutually desirable business arrangements that have built up the impressive commercial software industry in the United States.

The introduction of computers into Soviet management practice has been coming along slowly. Conservative applications, like accounting systems, seem to be the rule. The use of simple management information and process control systems is gradually increasing. Although there is some Soviet research on the utilization of computer techniques for decision analysis and modeling management problems (Ivanenko, 1977), little seems to be put into practice. Soviet managers tend to be older and more inhibited than their American counterparts. The system in which they work stresses straightforward production rather than innovation and marketing decisions. Soviet economic modeling and simulation activity stress the necessity of reaching a "correct socialist solution," and is not oriented toward being alert for general and unexpected possibilities in a problem situation. Furthermore, Soviet industry has learned not to trust its own statistics, and there may be a big difference between "official" and actual business practice. What does one do with a computer system for the "official" operational management of an enterprise when actual practice is different? Does one dare use the computer to help manage "expediter" slush funds, under-the-counter deals with other firms? A recent case indicates that these are serious problems (Novikov, 1978; *WashPost,* 1978).

Soviet programmers may be in an odd position with respect to industrial management. It is not clear that the managers know what to do with them. Firms are oriented toward plan fulfillment; they are not as information oriented as their American counterparts. The work of a programmer is often not directly related to the enterprise's plan, nor is his function as readily perceived as that of, say, a secretary or janitor. Management has to figure out what to do with these people and somehow measure their value to the enterprise. This is a big burden, and many of the older, highly politicized industrial managers are probably not up to doing this well. It will take the Soviets at least as long to learn to use their machines effectively as it took us.[13]

The USSR can claim what is potentially the world's largest management application—an ASU for planning the entire Soviet economy

[13] Americans should be reminded that some US management groups behaved similarly during the 1950s. The insurance industry, now among the largest and most committed computer users, is a notable case in point.

(OGAS). The Soviets have been talking about a network of computer centers for this purpose since the late 1950s. An often cited plan calls for a hierarchy consisting of a main Gosplan center in Moscow, 80 regional centers, and 4000 local centers (Chevignard, 1975). Data will be consolidated upward and plans will be passed downward in this treelike structure. The literature on the subject is large, and this is neither the place to review nor to analyze the project except to comment briefly on some software-related aspects.

On the surface, of course, it is ridiculous for the Soviets to talk about such an undertaking when data communication between computer centers often takes the form of someone carrying a deck of cards crosstown on a bus. The Soviets do not understand the operation of their own planning practices well enough to write down a useful set of specifications for the super software system that would be necessary to support such a large, highly integrated, and comprehensive network. The system is primarily a political football that is being struggled over by Gosplan and the Central Statistical Administration. From a software standpoint, it has helped them to start thinking, in some detail, about important problems like standardization, documentation, data-reporting procedures and formats, and the usefulness of their own statistics (*Ekongaz,* 1977). It has also spurred considerable investment in an assortment of data-processing systems. These products are useful and the experience is desperately needed.

3.2 Internal Diffusion

Before Ryad, the dissemination of software products and services was accomplished through a variety of mechanisms including national and regional program libraries, user groups, and informal trades. None of this was particularly effective or well organized [see references listed on p. 112 of Davis and Goodman (1978)]. For example, some libraries were little more than mail-in depositories that were not properly staffed, indexed, or quality controlled (Dyachenko, 1970; Galeev, 1973). The development of the Unified System was accompanied by a greater appreciation of the limitations of part practices. Ryad hardware would be pitifully underutilized if each user installation were left with an almost empty machine and expected to do all its own programming. This would have defeated the whole purpose of the new system.

The creation of the Unified System, with its common hardware and software base, is a major step in the alleviation of the technical difficulties of portability—the transfer of software from one installation to another. The hardware mixes and self-maintenance practices of the pre-Ryad days were severe limitations to portability. It should be noted however that this

in itself does not guarantee portability of systems. Programs developed at one IBM 360 installation in the West are not necessarily trivially transferable to another. Local differences in hardware and software—including differences in operating systems—may make this difficult.

Ryad marks a singular development in Soviet computing history: Its vendors are providing complete and modern operating systems and utility programs to all users. We do not know what the vendors are doing beyond this to promote standardization and diffusion. Standardization is an important form of diffusion since it facilitates portability and centralized maintenance. In the US, software standards exist primarily through the activities of important vendors; government efforts have had some success (notably with COBOL) but tend to be less effective (White, 1977). With their hierarchical system, one would think that the Soviets are in a particularly strong position to promote standardization and diffusion. For example, the detailed specifications for a programming language can be incorporated in an official State Standard (GOST) that has the force of law. Compilers that conform to this GOST could then be built for widely used computer models by centralized software groups and distributed to the users of these models. It would literally be against the law to change the syntax at an installation. Such a standard exists for the ALGAMS language (GOST, 1976). We do not know to what extent the Soviets are trying to standardize software in this way. We do not even know how this has affected the use of ALGAMS, a language that has been in use since the mid-1960s. Many programs must have been written in a lot of local variants of ALGAMS during this time. Are they being rewritten to run on compilers for the standardized version? Does the State Standard effectively encourage future programming, on the new computers, in this language that was specifically designed against the limitations of Soviet hardware of the mid-1960s? The Ministry of the Radio Industry, which has a legal near-monopoly over the production of mainframe computers, is in a strong position to push this kind of standardization and diffusion, but seems to have little motivation to work very hard at it. To some extent Minpribor acts as a competitive and mitigating influence. The Minpribor Minister, K. N. Rudnev, has been a dynamic force in promoting standards and customer service, and Minpribor has established the only publicly announced national customer software service.

Since the Soviets currently seem to be doing better with hardware than software, perhaps one way to gauge software service is to see what is happening with hardware service. In 1977 the Council of Ministers "obliged" all ministries and departments to provide for centralized technical service for computers (Trofimchuk, 1977). Although it is not clear what these obligations are, it is clear that the extent and quality of this service

leaves much to be desired (Fadeev, 1977; Perlov, 1977; Taranenko, 1977; *Izvestiya,* 1978). We find situations where a Ministry X has a geographically centralized service facility for its own enterprises using certain computer models. An enterprise in that area with that model, but under a different ministry, cannot use the service. This kind of bureaucratic fragmentation pervades all computing services and is a major obstacle to diffusion.

In addition to the software services provided by the hardware vendors, diffusion in the US is greatly facilitated by independent software outlets. We would conjecture that relatively few of the successful independent software ventures in the US were started and principally staffed by people with only an academic background. IBM and other computer companies have been the real training grounds for these entrepreneurs, not the universities or government facilities like the National Bureau of Standards. It is, however, primarily the academics that the Soviets seem to turn to for help with software problems. This does not appear to have done them much good, and it is difficult to see where, in the Soviet institutional structure, they will be able to create an effective substitute for the American computer companies to train and diffuse aggressive and imaginative software specialists. As we noted earlier, the Soviets are in the early stages of developing their own counterparts to these firms, but it is as yet too early to do much more than speculate on the possibilities and their chances for success.

User groups are also vehicles for software diffusion. Before Ryad, the Soviets tried several user groups. Lack of interest, the lack of sufficiently large user bases, poor communications, large geographical distances, a lack of hardware vendor support, and assorted bureaucratic aggravations severely hampered these efforts. Furthermore, the existence of many installations were secret, membership in some groups required security clearances, and lists of centers using the same models were probably not readily available. The BESM-6 and M-20/220/222 user groups seem to have been the most successful. These machines were particularly favored by the military and other high-priority users, and the importance of the clientele and their applications had to be a significant factor in these relative successes. These two groups hold regular technical meetings and have built up respectable libraries over the last 10–20 yr. It is likely that both had active support from the hardware developers and manufacturers. Most of the other user groups do not seem to have worked out as well.

There is a Ryad-user group, but current indications are that it is not much more effective than the others (Taranenko, 1977). To be really successful, the Ryad users would have to be broken down into specific model

groups and each of these would have to be supported by the specific enterprises that developed that model's hardware and systems software. Even then, a group's effectiveness might be geographically confined.

The Soviets have a respectable number of conferences and publications on computing, although efforts in this direction are handicapped by a lack of professional societies that are as active as the ACM, SIAM, and the IEEE. The Soviet Popov Society for electrical engineers does not engage in the same level of activity. In the USSR, the ministries and some particularly active institutes, such as the Institute of Cybernetics in Kiev, sponsor conferences and publications. Each year, they hold a few large national-level conferences and perhaps a couple dozen small, thematic conferences. Occasionally, the Soviet Union hosts an international meeting. Conference proceedings are neither rapidly published nor widely disseminated. Until 1975, with the publication of *Programmirovanie,* there was no generally available software journal in the USSR. Articles on software were rare, theoretically oriented, and distributed over an assortment of other professional journals. Few journals are widely circulated or timely. At least two relatively substantive journals, *Elektronnaya Tekhnika Ser. 9* and *Voprosi Radioelektroniki Ser. EVT,* are restricted. In the West, some of the most timely information appears in periodicals like *Datamation* that are sustained by vendor advertisements. Soviet vendors do not have the motivation, outlets, or funds for advertising. They seem to have little interest in letting anyone know what they are doing.

The Soviets claim to have "socialized knowledge" and it is thus easier to diffuse scientific and technical information in the USSR than it is in the capitalist countries. "Soviet enterprises are all public organizations, and their technological attainments are open and available to all members of society, with the exception of course of information classified for military or political reasons. The public nature of technological knowledge contrasts with the commercial secrecy that is part of the tradition of private property in capitalist countries. Soviet enterprises are obliged not only to make their attainments available to other enterprises that may wish to employ them but also actively to disseminate to other enterprises knowledge gained from their own innovation experience. The State itself subsidizes and promotes the dissemination of technological knowledge through the massive publication services of the All-Union Institute for Scientific and Technical Information [VINITI]" (Berliner, 1976).[14] This sounds better in theory than it works in practice. While services like those provided by VINITI and efforts to establish national programming libraries (Tolstosheev, 1976) are unquestionably useful, they do not provide the

[14] Not surprisingly, VINITI is at the forefront of Soviet work in large information retrieval systems.

much broader range of diffusion services available in the US. Capitalistic commercial secrecy is overstated; very little remains secret for very long. The Soviets have no real counterpart for the volume and level of Western marketing activity. By comparison, lists of abstracts of products that have not been properly quality controlled for commercial conditions, that have no real guarantees or back-up service cannot be expected to be as effective a vehicle for diffusion. The Soviet incentive structure not only does not encourage dissemination of innovation particularly well, but it also often promotes the concealment of an enterprise's true capabilities from its superiors.

The vertical structuring of the Soviet ministerial system works against software diffusion. Responsibility is primarily to one's ministry and communication is up and down ministerial lines. It is much easier to draw up economic plans for this kind of structure than it is for those with uncontrolled horizontal communication. Furthermore, each ministry appears determined to retain full control of the computing facilities used by its enterprises. In the West, software diffusion is a decidedly horizontal activity. Data processing and computing personnel and management talk to each other directly across company and industry lines, and people are mobile in a wide-open job market. This communication is facilitated by active professional organizations. Such arrangements do not exist to anywhere near the same extent in the USSR.

It is not only the ministerial system that mitigates against the really effective encouragement of direct producer–customer horizontal economic activity. Often the various layers of local Communist Party organizations perform the role of facilitating horizontal exchanges. The Party needs visible activities that justify its existence and authority, and this is one of the most important. No serious erosion of this prerogative is possible. However, it is much easier for a local Party secretary to get a carload of lumber shipped than it is for him to expedite the delivery of a special purpose real-time software system. He can take the lumber away from a lower priority enterprise, but what can he do to get the bugs out of the software? He can throw extra people on the job, but that will probably only make matters worse. Software projects tend to react badly to the "Mongolian horde" approach often favored by the Soviets. The detailed enterprise level software transactions cannot be managed by politicians.

This problem affects the diffusion of technical R&D to production enterprises in general. Software is an extreme case because it is so difficult to manage under any circumstances. One mechanism that has evolved to facilitate technical work is the emergence of very large enterprises and research institutes that are capable of handling most of their own needs in-house. Thus one finds many enterprises who own and operate computing facilities entirely on their own.[15] This is basically a defensive reaction that improves local viability in a highly constrained environment. Globally, the wide distribution, limited use, and hoarding of scarce resources, particularly personnel, in bloated organizations is counterproductive. The Party and government do recognize this and have shown themselves prepared to give up some control to obtain increased efficiency in innovation. Most of these changes have related to highly technical R&D matters over which they have had little effective control anyway. Changes include the already discussed corporationlike associations and R&D contract work, and also reforms in innovation incentives and prices for new products (Berliner, 1976). This represents progress and will help the development and diffusion of software.

3.3 Stages in the Software Development Process

The Soviet literature is missing the detailed articles on software engineering that are so abundant in the Western literature. This would seem to indicate a lack of widespread appreciation of and serious common concern about the technical, economic, and management problems that plague the stages of development of large software systems. As they gain more experience, this situation is likely to change. Articles on programming methodology are beginning to appear in East European publications (e.g., *InforElek,* 1977), and the Soviets should soon follow. Such articles will become more common and, in time, there will be papers on case studies, programming productivity experiments, chief-programmer teams, etc. Until such studies are published, we have to content ourselves with a cursory description of some of the problems they are probably having with the various phases of the software development process.

There are several nearly equivalent breakdowns of these stages. We will use the following list: producer–client get-together; requirements specification; system design; implementation; testing; maintenance; and documentation. Of course, the software development process is not a single-pass through this list. There are assorted feedback loops, iterations, and overlaps. In particular, documentation is not a distinct stage, but an activity that should pervade every stage. Nevertheless, the list suits our purposes.

Producer–client get-together. This can obviously happen in one of two ways. Either the software producer seeks out the client or vice versa. The Soviets have trouble both ways. Producers in the USSR are not in the

[15] Computer rental seems to be nonexistent. Rental arrangements would complicate service obligations for the hardware manufacturers. There is a serious effort to establish large, "collective-use," computer centers, and these may eventually prove successful.

habit of seeking out customers. On the other hand, most Soviet enterprises are still naive customers for software. They do not know what they want or need or what is available. We know almost nothing about how Soviet firms negotiate software work, but they must be having even greater difficulties than we have in the US in negotiating price, time, and manpower needs. In general, the Soviets themselves do not know how they determine prices for new products (Berliner, 1976).[16] The annual plans of both the producer and client must limit the flexibility of the arrangements that can be made, and there is a serious shortage of experienced software specialists.

Requirements specification. This refers to the translation of customer needs into a statement of the functions to be performed by the software system. The specifications should be at a level of detail that will make it possible to test the product unambiguously to see if they have been met. They serve the producer by making its task clear. This stage clearly demands good communications between the producer and client, something Soviet enterprises are not noted for in general. This stage also requires a great deal of patience and sympathy on the part of the software firm, something that is in short supply at most Soviet research institutes. Experience shows that software specifications change almost continuously as a result of the changing needs, better perception on the part of the customer, or because of problems encountered by the producer. It is important that the client regularly monitor system development progress and that the producer be receptive to client input. If not, then it is almost inevitable that the wrong product will be built.

Given their highly centralized economic and political structure, the Soviets are in a position to take requirements specifications quite a bit further than any of the developed noncommunist countries. As we noted earlier, they can specify national (or lower level) standards that would be legally binding. Some serious effort to do this has been undertaken by the State Committee for Science and Technology and other agencies for ASUs (Myasnikov, 1976; Zhimerin, 1978). However, the rigidity of these requirements are being resisted by the enterprises, who want systems that are tailored to their individual desires (Bobko, 1977). As time goes on, and more and more individually tailored systems are built by the enterprises themselves and outside contractors, it will become more difficult and disruptive to impose requirements specifications from above. One can

[16] The Polish ELWRO-Service firm uses simple formulas based on unit prices for assembly language instructions. Price appears to be determined primarily by the number and type of instructions in the object code of the software (Mijalski, 1976). The USSR has been slow to appreciate the economic aspects of software development. It came as something of an initial shock to the Soviets when they learned that Western companies expected to be paid more than simple service fees for the software that they had built.

easily imagine the attractiveness of such uniform standards to the central planners and the opportunities they provide to overcome some of the systemic difficulties that affect Soviet software development and diffusion. However, it is one thing to have the power to impose standards, but quite another to do it well. The technical problems are enormous. It will be very interesting to see what becomes of these efforts.

System design. A good design is usually put together by a few talented people. The Soviet Union does produce such people. Right now, for the reasons discussed earlier and others yet to be noted, they lack experience and number. Their design options are also more restricted than those of their American counterparts since they have far fewer software and hardware tools and building blocks available.

Implementation. This generally refers to coding the design and getting the system up to the point where there are no obvious errors or where such errors are unimportant and can be patched. It is the most straightforward of the stages. However, it can be made unpleasant by a lack of hardware availability and reliability. Ryad has eased both of these problems considerably. It can also suffer from a lack of well-trained programmers and of available installation user services. These problems are not deeply systemic and we should see a steady improvement in the Soviet ability to handle this phase of software development.

Testing. This is the verification that the coded programs and other components of the system satisfy the requirements specification. This stage generally ends with customer acceptance of a supposedly error-free or error-tolerant system. It involves program testing and consultation with the client as to the satisfaction of his needs. Testing often accounts for almost half of the total preacceptance development effort of large software projects. Soviet strength in mathematics and their interest in programming theory may eventually place them among world leaders in the field of formal proofs of program correctness. However, this is an abstract area that currently has little practical impact. Testing large complicated systems or real-time software is a completely different matter. We have seen little in the Soviet literature that realistically and specifically comes to grips with these problems. They do use a commission to approve computers for production and use, but we do not know if there is a counterpart for software. Software testing is also not the sort of activity that would be expected to show up on any of their measures of institute or enterprise productivity and is thus likely to suffer accordingly. Good system testing is a difficult and complex activity that requires highly skilled people. However, it is a frustrating and low profile thing to do. In light of common Soviet personnel utilization practices, it is likely to be assigned to the lowest ranking neophytes.

To a considerable extent, Soviet problems with this stage are basically a

matter of acquiring experience in building large software systems. It has taken the US a long time to learn to struggle with these difficulties, and the Soviets will have to go through the same painful learning experiences. One place where systemic considerations might be important again relates to customer docility. If the software developers can get away with not taking responsibility for the errors that are passed on to the user, then this is what will happen. The effort devoted to checkout is directly related to customer power.

Maintenance. This refers to the continued support of a program after its initial delivery to the user. It includes the correction of errors that slipped through the checkout phase, the addition of new capabilities, modification to accommodate new hardware or a new operating system, etc. Good maintenance clearly extends the lifetime and economic value of software. Maintenance costs in the West are now running around 40–60% of the total life cycle cost of major software systems (Boehm, 1977). As one extreme example, some Air Force avionics software cost about $75 per instruction to develop, but the maintenance of the software cost almost $4000 per instruction (Trainor, 1973). Maintenance can either be done by the original developer, the customer, or a third party. Extensive third-party arrangements currently seem out of the picture in the USSR, but could become important if software standardization becomes a reality to any appreciable extent. Vendor/producer maintenance requires a high quality of customer service and will be slow to develop there. It appears that the usual procedure has been for the customer to do its own maintenance. This could result in local modifications that would eliminate compatibility and lead to the resistance of centrally supplied updates or improvements.

Documentation. Documentation encompasses design documents, comments in the code itself, user manuals, changes and updates, records of tests, etc. To be most effective and accurate, it should be done concurrently with all the other stages. This is not a particlarly interesting activity, and is often slighted unless there exists pressure on the software development group to do it. Good documentation can make checkout and maintenance much easier; poor documentation can cause terrible problems. It is difficult to see where serious pressure for the documentation of ordinary software would come from in the USSR. It is another activity that does not show up well in the measures of productivity. Customer pressure is not likely to be effective. Pressure in the form of State Standards will get software documented; but without strong customer involvement there is really no way to control quality and poor documentation can be a lot worse than none at all. This is likely to remain a long-term problem.

The almost total lack of convenient Xerox-like services in the USSR is a factor that adversely affects all the stages of the software development process. This is a means to quickly and reliably record and distribute changes in specifications, documentation, test data, etc. This capability is particularly important for large projects involving frequent updates that need to be seen by many people. The absence of fast photocopying facilities can lead to unnecessary delays and costly and dangerous loss of sychronization among the project subgroups. In a similar vein, there is a shortage of good quality user terminals.

3.4 Manpower Development

The training of technically competent software personnel and raising the computer consciousness of management is an important task in the development of a national software capacity. This diffuses and enhances the capability to produce and utilize software effectively, and is the ultimate source of products and services. The USSR trains more mathematicians and engineers than any other country. Both the quantity and quality of mathematical education in the Soviet Union, from the elementary school level (Goldberg and Vogeli, 1976) through postgraduate training, is at least as good as that in the US. For the most part, Soviet managers have engineering rather than business degrees (Granick, 1961). One might think that, with this personnel base, they would be in an unusually good position to rapidly develop a large-scale national software capacity.

However, it is one thing to develop a strong national mathematics curriculum. It is quite another to train and utilize, say, a quarter million professional quality programmers and systems analysts (about half the number in the US) and a couple million scientists, engineers, administrators, and businessmen who do applications programming as part of their professional activities.

This requires equipment. One does not become a skilled programmer unless one spends a lot of time programming. Schools and industrial training centers are generally low on the priority list for computer allocation. By 1976, Moscow State University, a school comparable in size to UC Berkeley, but with a curriculum much more oriented toward science and engineering, had among the best central computing facilities of any university in the USSR. This consisted of two BESM-6 machines, one of which was to be used in a new time-sharing system with 25 terminals. They were expecting to augment this with two ES-1020s by early 1977. The first ES-1030 to go to a higher educational institution went to Leningrad State, another large prominent university, in 1975 (Solomenko, 1975). A major engineering school, the Moscow Aviation Institute, was

still limited to a Minsk-22, a BESM-2, and two Minsk-32 computers in its computing center as of early 1976. These three universities are at the top of the educational hierarchy. The vast majority do much worse.

As a result of this situation, there are many students still spending time learning to write small applications and utility programs in machine language for the medium-scale Minsk and Razdan computers and a host of small second- and third-generation computers such as the Mir, Nairi, and Dnepr lines. This may not be as fruitless as it seems, since a lot of these models are still in use in the general economy. The situation is currently changing. The important objective should be to get respectable numbers of the smaller Ryad models into the educational system. Once this is done, students will be trained on the dominant national hardware/systems software base, and their immediate postgraduation value will be increased considerably. Ryad production capacity is such that this is likely to happen by the early 1980s.

The software side of computing as an academic discipline went through an extended infancy that started in 1952 with A. A. Lyapunov's first course in programming at Moscow State University (an interesting account of the early days can be found in Ershov and Shura-Bura, 1976), and lasted until the end of the 1960s. Not surprisingly, the new Soviet perspective on computing that emerged by the late 1960s included an appreciation of the need to train a much larger number of programmers and systems analysts. To help meet this need, separate faculties in "applied mathematics" were established around 1970 at universities in Moscow, Leningrad, Novosibirsk, and Voronezh (Novozhilov, 1971). In addition to these, and other more recent (e.g., Sabirov, 1978), separate faculties, computer science is also taught under the auspices of mathematics and electrical engineering departments.

The Soviet academic community has a strong theoretical tradition. Peer group status considerations, and a shortage of hardware, tend to reinforce this bias. Thus there is considerable pressure to do esoteric computer science to maintain respectability among colleagues (Novozhilov, 1971). Many instructors have had little practical training of their own. So, for example, computer science under a mathematics faculty would be strongly oriented toward numerical analysis, formal logic, and automata theory. There was essentially no opportunity for a student to learn about such things as practical database management systems. Industrial cooperation programs have had only limited success in establishing a better theory/practice balance. Soviet university students getting on-the-job training at research institutes and industrial enterprises are often given menial tasks.

The quality of university level education in the USSR varies considerably across subject lines. Outstanding centers of learning in mathematics exist at many places. Training in mathematics and in some of the mathematically oriented science and engineering fields is as good there as anywhere in the world. On the other hand, the academic study of history and politics is severely circumscribed, rigid, and pervasive (the degree requirements for all technical fields include heavy course loads and examinations on Soviet ideology). Education in the range of subjects that lie between mathematics and the ideologically sensitive areas, including all of the engineering disciplines, seems to be more narrowly focused and rigid than it is in the US [see Granick (1961) for some interesting first-hand observations]. We do not have a good picture of how CS education is evolving in the Soviet Union, but it is likely that it is some kind of hybrid between mathematics and engineering. By US standards, it is probably heavy on mathematics and light on practical programming work. As more hardware becomes available at schools, as instructors gain more practical experience themselves, and as Soviet industry pushes to have its needs met, we can expect to see CS education move closer to US models.

Although there are frequent complaints about the shortage of programmers and software specialists, there is little quantitative information on the output from the higher educational institutions or the shortfall that is perceived to exist. In addition to university-level training, there is also substantial activity in the large number of vocational institutes and night school programs. One thing is certain, there is currently an unprecedented effort under way to expand the base of people who can make use of the new computers. Where once 10,000 copies of a programming or software text was a large printing, now books on the ES system are appearing in quantities of 50,000 (Khusainov, 1978), 80,000 (Naumov *et al.*, 1975), and 100,000 (Agafonov *et al.*, 1976). Considerable efforts continue to be expended on software for second-generation machines, especially for the Minsk-32 (Zhukov, 1976—43,000 copies).

The problem of raising the computer consciousness of management is only part of the more general task of modernizing Soviet management structure, training, and practice. The magnitude of the problem is enormous. "Soviet sociologists have estimated that 60% of all administrative personnel in industry—including directors, deputy directors, chief engineers, heads of service departments, and shop foremen—are in their 50s and 60s. It is estimated that in the next 5–10 yr, when 30- and 40-yr-olds will move into responsible positions, approximately four million people will have to be trained for administration. This will amount to 40% of all such positions in industry. The number of managerial specialists (presumably above the shop level) to be brought into industry is estimated at 1.5 million" (Hardt and Frankel, 1971). In spite of much talk about improving

managerial training along the lines of American models, little is apparently being done in practice (Holland, 1971a) and certainly nothing is being done on the scale just described. It is difficult to imagine how the American models would be effective in the context of Soviet economic institutional structure. Most consciousness raising will have to evolve on the job.

4. Software Technology Transfer[17]

For the most part, the influence of the West on Soviet software development by the mid-1960s was via the open literature. Although this influence was very important (Ershov and Shura-Bura, 1976), the level of technology transfer was weak and there was not much product transfer. The reasons for this include the lack of suitable hardware, an underdeveloped interest in nonnumeric computing, the theoretical orientation of Soviet computer scientists, and the weak position of computer users.[18]

With the change of perception of computing that led to the Ryad undertaking, there came a commitment to produce and install complex general purpose computer systems in large enough numbers to make it necessary to upgrade general software capabilities. During the last decade, the rather low-key, localized, almost academic, Soviet software community has evolved into a serious industry with a long term and intensive program to acquire software products and know-how from abroad.

There are several reasons to think that software technology would be particularly easy for the USSR to obtain from the rest of the world. This is an extraordinarily open technology. Most of the basic ideas and many of the details necessary to produce a functionally equivalent product are available in open sources. It is much more difficult to hide "secrets" in the product itself than is the case with hardware, and the distinction between

[17] Parts of this section are adapted from Goodman (1978). A more complete discussion of the nature and control of this problem is in preparation (CTEG, 1979).

[18] On rare occasions, influential users would take matters into their own hands. An important use of FORTRAN in the USSR stemmed from interest in Western applications programs on the part of physicists at the Joint Institute for Nuclear Research in Dubna and the Institute of High Energy Physics in Serpukhov. They had had considerable exposure to the CDC applications programs at CERN in Switzerland and other research centers. Their interest and influence led to the purchase of a CDC 1604, including software, that was installed at Dubna in 1968 (Holland, 1971c). The CDC FORTRAN compiler was translated, line by line, into the machine language of the Soviet BESM-6 so that the applications programs could be run on this machine [the result has become known as "Dubna FORTRAN" (Saltykov and Makarenko, 1976)]. Here is an instance where active contact with the West produced a real stimulus to go out and get some useful software. However, this was a transfer that was not diffused much beyond BESM-6 users.

product and technology transfer is often blurred. Relatively little software is proprietary and much that is can still be obtained. Sources of information are abundant: conferences, journals, books, manuals, discussion panels, program listings, software libraries, consulting groups, and vendors. The Soviets have a large trained scientific/engineering manpower base[19] that should be capable of absorbing the contents of foreign work and putting together similar products of their own. The successful appropriation of the complex IBM S/360 operating systems is proof that they can do this on a large scale.

On the other hand, there are reasons why software technology transfer may not be as easy as it appears. Direct product transfers often run into problems at hardware interfaces. Even small differences in donor and borrower hardware can make conversion difficult. The Ryad hardware is effectively a functional duplication of S/360, but it is not identical to it. It may have taken the Soviets and their CEMA partners almost as long to adapt the DOS/360 and OS/360 operating systems to their Unified System hardware as it took IBM to build these systems in the first placé. Furthermore, it is possible for an unwilling donor to make it painful and time consuming to copy its products, e.g., by only releasing software in object code form or by inserting "time bombs" (code that destroys critical portions of the system after the passage of a certain amount of time or after a preset number of uses). Some of our most advanced software products cannot be transferred because the Soviets lack appropriate hardware. Most importantly, it is extremely difficult to effectively master the techniques and skills of software engineering and management.

4.1 Mechanisms for Software Technology Transfer

This subsection describes the active and passive mechanisms by which software technology is transferred. We adopt the definitions used in the Bucy Report (Bucy, 1976):

> Active relationships involve frequent and specific communications between donor and receiver. These usually transfer proprietary or restricted information. They are directed toward a specific goal of improving the technical capability of the receiving nation. Typically, this is an iterative process: The receiver requests specific information, applies it, develops new findings, and then requests further information. This process is normally continued for several years, until the receiver demonstrates the desired capability.
>
> Passive relationships imply the transfer of information or products that the donor has already made widely available to the public.

The term "passive" is used primarily in reference to donor activity. The receiver may be very active in its use of passive mechanisms.

[19] They claim 25% of the world's total of "scientific workers" (Ovchinnikov, 1977).

An illustration of how the terms "active" and "passive" will be used in the context of software transfers might be helpful. There are two kinds of proprietary software: that which is generally available to the public and that which is not. The purchase of a publicly available system, perhaps with some basic training and maintenance service, is passive, even though the buyer might become very active in distributing or duplicating the software. The sale of software that is not publicly available would be considered a more active relationship. The donor is clearly contributing more than what is normally and widely available. If sale is accompanied by advanced training, then the donor relationship is that much more active. "How to build it yourself" lessons from the donor will be considered very active even if such serviees are publicly available.

Listed below are a sample of mechanisms that can be used to transfer software products and know-how. They are roughly ranked by the level of donor activity. (One can easily imagine specific examples that might suggest some reordering, but this list is adequate for our purposes.)

Joint ventures	More
Sophisticated training (e.g., professional-level apprenticeships) ←	
Licenses with extensive teaching effort	active
Consulting	
Education of programmers and systems analysts	Donor
Sale of computing equipment with software training	
Detailed technical documents and program listings	activity
Membership in Western user groups	
Documented proposals	
Conferences	More
Academic quality literature	→
Licenses and sale of products without know-how	passive
Commercial and exchange visits	
Undocumented proposals	
Commercial literature and exhibits	

The term "license" needs to be defined here since normal patent considerations do not apply to software (Mooers, 1977). We will take it to mean the provision of a copy of the software to a receiver who then has the recognized right to distribute it extensively within some domain. The distinction between this and a simple product sale may be a matter of a paragraph in a contract, but the distinction is worth making. It is easy to produce multiple copies of software products and the Soviets have control of a large, and economically isolated, domain of computer installations.

Of course, some categories of software are more transferable than others. The following four rough (partially overlapping) categories are listed in order of decreasing ease of transferability:

(1) Applications programs written in higher-level languages.

(2) Systems and utility programs in machine or higher-level language form.

(3) Large, highly integrated systems (e.g., multiprogramming operating systems, real-time air traffic control systems).

(4) Microprograms and other forms of "software" that are very closely interfaced with and dependent on the hardware on which they are run and which they control.

Although it is difficult to quantitatively merge our two lists because the effectiveness of software transfer is so strongly dependent on such highly variable factors as local programmer talent, there is a clear qualitative merge. As one goes down the list of transfer mechanisms, their effectiveness decreases for all software categories. For any given mechanism, its effectiveness decreases as one goes down the list of software categories.

If any of the listed mechanisms should be candidates for US Government control, they should be the top four listed. An example will illustrate the third mechanism. In their efforts to adapt DOS/360 and OS/360 to the Ryad-1 models, it would have been of considerable help to the CEMA countries if they had had a deal with IBM, or with a company that had considerable experience in the business of making non-IBM hardware compatible with IBM software, which would have included a license for the software and a teaching effort that would have showed them how to adapt it to the Ryad hardware.[20] This effort might have gone further and included help in designing the hardware with the compatibility goal in mind. Such an arrangement could conceivably have substantially reduced the time it took the Soviet Bloc to acquire and adapt the systems on their own, and it could have provided a tremendously valuable transfer of know-how.

Simple product transfer should be of much less concern than know-how transfers that will enable the Soviets to build up their indigenous software capabilities. The top four mechanisms transfer considerable know-how and short circuit the painful experience of learning through time-consuming and costly trial and error. The delay of the acquisition of indigenous capability is a major goal of antitransfer measures.

The lesser forms of licensing and product sale on our list are not as important. For example, IBM might have sold the Soviets a "subscription" to the S/360 operating systems. This could have taken the form of supplying one copy of each of the operating systems on tape plus informa-

[20] No such arrangement actually existed.

tion on new releases, etc., and a license for distribution to all Ryad users. They would have had to adapt the software to the Ryad hardware themselves. This would have saved them the effort of obtaining it through other legal channels or by covert means, and IBM would have been able to cultivate good will and get some compensation for the use of its products. There was no effective way to deny the CEMA countries access to copies of this software; it was simply available from too many sources. The time that the Soviets could have saved through such an arrangement would not have been great. The time it took to adapt the software to Ryad must have been much greater than the time it took to acquire copies of it.

But the importance of the passive mechanisms to software technology transfer to the USSR should not be underestimated. We think they contributed significantly to the massive appropriation of IBM S/360 software for the Unified System. They also affect training programs at all levels. Much written and oral material is available on subjects that relate to the management of software projects and on software engineering. These are areas where the Soviets are particularly weak. Passive material is publicly available in huge quantities. The Soviets have been using these sources for almost three decades and their influence is obvious in almost all Soviet software work. Before Ryad, hardware problems limited the use of direct product transfer. Now, of course, direct product transfer is an important source of useful software. However, it is important to point out that passive sources are of limited value for several of the most important phases of the software development process. These include the customer/developer relationship, certain aspects of specification and design, the higher levels of testing and integration, and maintenance. All of these stages become particularly important for the construction and support of large, highly integrated systems.

Active sources are also abundantly available in the West. In contrast to a hardware area, such as disk manufacturing technology where there are only a few really valuable potential donors, there are literally thousands of places in the US alone that have something to offer the Soviets in software products and know-how.

The Soviets do not use these active mechanisms to the extent that they could (but there has been substantial improvement since the mid-1960s). USSR restrictions on foreign travel by its citizens is a severe constraint. The people they send out are helpful to their effort, but they are too few. They would have to send several hundred software specialists to the West each year, and most of these for extended study, to affect continuously and broadly their software capabilities. The leadership is very unlikely to do this. It might be politically and economically more acceptable for them to import Western experts who would spend extended periods showing

them how to manage large software projects and how to upgrade computer science education. They might also buy full or part ownership in Western software firms, and use the Western talent employed there to develop software for their use. The ELORG centers in Finland and Belgium represent moves in this direction. A more unlikely form of long-term joint venture would be to permit partial Western ownership and management of a Soviet enterprise. Some of the other CEMA countries allow this, but so far the USSR has not. On the other hand, the internal political situation in the USSR may change to militate against both the import and export of computer scientists after the death or retirement of Brezhnev (Yanov, 1977).

4.2 External Sources

The S/360–Ryad software transfer was facilitated with considerable help from Eastern Europe, particularly the GDR. It is hard to avoid the impression that the "per capita" software capabilities of the GDR, Hungary, Poland, and Czechoslovakia exceed that of the USSR. This is probably the result of many factors, not the least important of which is the greater contact these countries have with the West European computing community. They have also had much more direct and indirect experience with IBM products. We would not go so far as to conjecture that the indigenous capacity of the USSR may have been such that the S/360–Ryad software transfer would have failed without help from Eastern Europe, but the role of these countries should not be underestimated.

Hungary, the GDR, Poland, and Czechoslovakia are not only important conduits for facilitating software technology transfer from the West to the USSR, but they are also valuable sources of products and know-how in their own right. They have potentials for providing active mechanisms for personnel training, consulting, etc. As communist countries using a common hardware base, they are the best external source the Soviets have for many industrial- and management-related software products. They are also external sources that can be used directly in the development of military software systems, such as those used for command, control, and communications, for the Warsaw Pact. Problems that inhibit active involvement with the West, such as travel restrictions and a lack of hard currency, are much less important.

Perhaps the greatest value of the Eastern Europeans to the USSR is as models for institutional arrangements and economic practices. In particular, Hungary and the GDR seem to be much more effective in the areas of software customer service and systems software support than the Soviets. Marxist theory may be opposed to an uncontrolled gaggle of profit-

hungry, privately owned firms operating outside of a central plan, but it is hardly opposed to the development and maintenance of products that benefit the economy. The Hungarians and East Germans are showing that it is possible for communist economies to provide minimum basic software services to general users. The Soviet Union might learn much from them.

Western Europe is both a conduit for US software technology and a source of innovation in its own right. Not surprisingly CEMA has easier access to US multinational corporations through their European companies than through US-based enterprises. The shared culture and language across East and West Germany makes for a particularly low barrier. Notable West European developments of direct value to the USSR include: the Europe-based ALGOL project, CERN in Geneva, SIMULA-67 (Norway), the Aeroflot airlines reservation system (France), and the International Institute of Applied Systems Analysis located in Austria.[21] The most important sources are West Germany, England, and France. Others are Belgium, Denmark, Holland, Norway, and the politically neutral Austria, Finland, Sweden, and Switzerland. Joint ventures with firms in these countries may become an important transfer mechanism.

The US remains the ultimate source of software technology. In addition to the IBM–Ryad connection, Soviet interest stems from the facts that more R&D is done here than anywhere else and that we are the largest repository of software products and information. The US is clearly the world leader in the development of large military-related software systems. English is an effective second language for almost all Soviet computer scientists. Finally, there is the nontrivial matter of prestige and the "Big Two" psychology. From the standpoint of career enhancement, it is more desirable for a Soviet citizen to come here than to travel anywhere else. Russian pride also seems to suffer less when they borrow from us than when they have to go to the Hungarians or Germans for help.

The Soviets make less extensive use of the Japanese as a source of software technology transfer. This is partially because Japan has not developed as much software, although their potential is high. However, Japanese software institutional arrangements and development/maintenance practices may be even less suitable for Soviet emulation than those of the US. In general, it would appear that cultural and language barriers make Japan a less attractive source than the West.

A distinction should be made between commercial software, which is produced for sale, and noncommercial software, which is used only by its developers or distributed free or at a nominal cost. The latter is usually

[21] It should be noted that all five of these important examples involve substantial US participation.

produced by nonprofit organizations (e.g., universities, government labs) and may be of high quality, but most of it is not tested, maintained, or protected to the same extent as commercial software. Commercial software has become a multibillion dollar business in the West. Over the last 10–15 yr, the companies in this industry have become increasingly aware of protecting the proprietary value of their products. The protective mechanisms include a variety of legal and technical options that appear to be reasonably effective, although in such a dynamic industry it is usually only a matter of time before a competitor comes up with an equivalent or better product.

We do not know how well people who have been trained in the West, or in jointly operated facilities in Eastern Europe, are actually used. It is not clear if they are used in any particularly effective way to promote the internal diffusion of know-how.

It is important to recognize that technology transfer will not solve the most basic Soviet software problem. The Soviets may be able to import enough turnkey plants for manufacturing automobiles to satisfy their perceived need for cars, but they are going to have to develop the internal capacity to produce most of their own software. There are thousands of computer centers in the USSR and they all need large quantities of software. Contacts with foreign sources are limited to only a very small fraction of the Soviet computing community. The orifice is too small to import the volume of software technology required, and internal systemic problems prevent its effective diffusion and use. Finally, these computer installations have their own special software needs that reflect their way of doing business and Western commercial applications software products may be unsuitable for these needs.

4.3 The Control of Software Technology Transfer

In terms of in-depth understanding and the avoidance of repetition of mistakes, the Soviets do not seem to have profited much, so far, from the Western experience. They consistently make the same mistakes and suffer from the same growing pains as we did. These are often exacerbated by difficulties of their own making. The Soviets have been making extensive use of Western software technology, but they currently seem satisfied with the short-term goals of recreating selected Western systems at a rate that may actually be slower than that with which the West built these systems originally.

It is inevitable that the Soviets will significantly improve their software capabilities as they acquire more experience and as their perception of the role of software matures. Their interest in software technology transfer as

a means of acquiring both products and know-how is likely to continue indefinitely. Furthermore, as their own indigenous capabilities improve, they can be expected to make more extensive and more effective use of transfer mechanisms and opportunities.[22]

We could make life more difficult for them through various forms of control. Unfortunately, software control is more complex than the control of the kinds of technology that were used as examples in the Bucy Report (1976). The range of software products and know-how is enormous. Some of it, such as microprograms and sealed-in software (Mooers, 1977), can be controlled in much the same way as hardware. Some of it, such as numerical and combinatorial algorithms, is essentially mathematics and beyond any effective control (although the translation from algorithm to program is often nontrivial). Most software lies somewhere between hardware and mathematics, and we do not know how to protect this part of the spectrum.

There are several different ways to try to control software. We could try to focus on those categories that are most amenable to control. For example, we might attempt to control the large, highly integrated systems, and give up on the applications programs in high-level languages and the small systems routines. Another approach would be to try to control the mechanisms of transfer. Thus we might regulate licenses with extensive teaching effort, joint ventures, etc., and ignore the mechanisms at the lower end of the list. A third approach would be to base controls on the potential military uses of the software. We could try to regulate software for pattern recognition, communications networks, test and diagnostic systems, command and control. Finally, we might use some form of "time-release" control over many products. All four approaches have serious definitional and enforcement problems. For example, where does technology transfer for management information systems end and transfer for command and control uses begin?

Not the least of the problems faced by efforts to regulate software transfer is its huge number of sources. There is nothing that can be done to seal up all the ways to obtain noncommercial products and know-how from universities, laboratories, and the open literature. One of the largest single sources of readily obtainable software is the US Government, including the Department of Defense. Assorted US Government

[22] We should not forget that transfers can go both ways. The Soviets will someday develop software products and ideas that American firms or the US Government would want to use. Systemically, we are capable of more effectively exploiting and diffusing software advances than are the Soviets. There is potential for a two-way flow of software technology transfer. Although the flow into the US would be much smaller than the outflow, we would probably make better use of what we get.

agencies literally give it away to the Warsaw Pact countries (CSTAC TT, 1978).

It is more realistic to try to control commercial software. Commercial software houses distribute products that are usually better tested, maintained, and documented than noncommercial products. Regulation may delay acquisition or discourage the Soviets from obtaining as much software as they might if there were no regulations. The best specific forms of control might be the protective mechanisms the commercial software producers use against their market competitors. With their growing appreciation of the cost and value of software has come the desire and effort to protect it more effectively. The trend with the IBM operating systems is a case in point. With S/360, almost all the software was available to anyone who wanted to take it. With S/370 and the 303X models there is a continuing tendency to collapse the "free" software around the nucleus of the operating system, and give the user an option to purchase the rest. Unfortunately, some marginal US companies might be willing to let the Soviets have more than they would their market competitors. Thus government regulation would be necessary to supplement company practices.

One of the best forms of control of software transfer is the control of hardware. Sophisticated software systems often require sophisticated hardware. Soviet general purpose hardware has reached a 360-level plateau and it will not be easy for them to develop advanced telecommunications and real-time processing hardware for widespread use. Software is basically an evolutionary technology. The closest it comes to revolutionary developments results from opportunities presented by major advances in hardware availability. Control over hardware technology transfer may be an effective way to delay acquisition of advanced capabilities.

A basic problem in the formulation of controls is that we really do not understand what benefits past transfers have given the Soviets or how well they utilize transfer mechanisms. Did the CEMA countries learn more by adapting the S/360 operating systems to Ryad than they would have if they had built new operating systems? Would the latter have taken the same time as the former? Did they use fewer people than it would have taken them to do something more innovative? They devoted many man-years of many of their best people to the piecemeal debugging of the huge S/360 operating systems on the Ryad hardware. This time might have had a higher payoff, from the standpoint of enhancing their indigenous software capabilities, if these people had invested the effort in acquiring experience in large system design, integrated test design, and planning for maintenance.

Perhaps the best statement on software technology transfer was made by Edward Teller:

> The Russians know all of our secrets; they know what secrets we will develop two years in advance. We are still ahead in electronic computers because there are no secrets. Without secrets we are advancing so rapidly that the Russians can't keep up.

Although this statement was made in reference to computer hardware, and in that context it may be a bit exaggerated, there is no better short appraisal of the software situation. Ultimately the diversity, openness, and high rate of incremental innovation of the American software industry is the best protection it has.

5. A Summary

By and large, the development of Soviet computing has tended to follow closely the US technical pattern, but it has differed considerably in terms of timescale, philosophy, institutional arrangements, capital decisions, and applications. In particular, the USSR was slow to appreciate data processing, and to develop the technology to support the widespread use of digital computers for such applications. It is only within the last ten years that the Soviets have given the priority and resources necessary to the production and installation of complex general purpose computer systems in large enough numbers to make it necessary to improve greatly their software capabilities.

Prior to this, computer use in the USSR was limited primarily to small- and medium-scale scientific and engineering computations. There was no well-developed business machines industry, nor was there an important clientele with a perceived need for such equipment. The Soviet military and technical communities were less enamoured with computers than their US counterparts, and the Soviet computer industry developed only to the extent that it could meet the relatively limited needs of these users. As a result, Soviet computing went through an extended infancy, with its first-generation hardware/software period lasting to the mid-1960s, and the second generation continuing into the early 1970s. Very few machines large enough to necessitate a real operating system were built. Storage and peripheral limitations restricted the use of high-level languages. The Soviets did not build the software that allowed computers to be used by many people who had not had much technical training.

The shift to the production of large numbers of general purpose computers was forced by internal economic pressures and, most likely, by the greater needs of the military. A substantial commitment necessitated the

development of much improved hardware capabilities—most important, the creation of an upward compatible family of computers with a respectable assortment of peripherals. The Ryad-1 family, an effective functional duplication of the IBM S/360, provides the Soviets and their CEMA partners with a reasonably modern mainframe capability. The computers of this family have been produced in considerable quantities and give Soviet users an unprecedented assortment of peripherals and level of reliability. Soviet satisfaction with this hardware can be inferred from their continued development of evolutionary upgrades of the early Ryad models, and their further commitment to the development of the Ryad-2 series, based on the IBM S/370. There has been a parallel, although somewhat smaller, major effort devoted to the development of minicomputers: first to the ASVT models, and more recently to the CEMA SM family.

This new, and substantial, base of mainframe, minicomputer, and peripheral hardware has done much to give the Soviets a broad general purpose national computing capability. Although backward by the current US state-of-the-art, it seems clear that it was never the intention of the Soviets to try to push the frontiers of either hardware or software technology. The overall plan was to put a large number of respectable, compatible computers into productive use as expeditiously as possible.

To this end, it was not surprising that the Soviets decided to use an already proven technology in the form of the IBM S/360. Although they seriously underestimated many of the difficulties of trying to duplicate a sophisticated foreign technology, they felt that the appropriation of the S/360 systems and applications software was the safest and quickest way to achieve their primary goal.

The Soviets have been making extensive use of Western software products, particularly in the area of systems software. They currently seem satisfied with the goal of recreating selected Western software systems at a rate that may actually be slower than that with which the West built them in the first place. In terms of in-depth understanding and the avoidance of repetition of mistakes in their own work, the Soviets do not seem to have profited much from the Western experience. They consistently make the same mistakes and suffer from the same growing pains as we did. These are often exacerbated by difficulties of their own making.

The Soviet economic system, with its vertical hierarchical structure and lack of opportunity for developing flexible horizontal relationships, seems ill-structured to support many of the software development practices that have worked well in the US. A strong hierarchical bureaucratic environment and a conservative incentive system effectively discourages entrepreneurial innovation. Enterprises are severely constrained with respect to

finding both suppliers and customers. By US standards, there is very little consumer pressure exerted on vendors, except in the case of special (e.g., military or Party) customers. The net result is that most Soviet computer installations have to rely on their own internal assets for most of their software needs. It is not even clear if they get much outside help with the systems software supplied by the hardware vendors. There is a long standing users' attitude that accepts this situation and is thus a major obstacle to progress. These difficulties exist in many other sectors of the Soviet economy, but they appear to be especially serious in the sophisticated service-oriented software industry.

In spite of these problems, Soviet software has come a long way during the last decade. The appropriation of IBM software for the Unified System was a substantial technological achievement. The volume, level, and intensity of software development and use has risen greatly over this period. The indigenous software capacity of the USSR has become respectable by world standards. Furthermore, as their own capabilities improve, they can be expected to make more extensive and more effective use of technology transfer mechanisms and opportunities.

The Soviet software industry will need some systemic changes to function more effectively. It is not clear to what extent such reforms will be allowed to take place. As the Soviets gain more experience, and as their perception of the value and problems of software matures, we can expect to see considerable improvement take place within the present economic structure. Past reforms, such as the establishment of the corporationlike associations and the expansion of contracting arrangements, seem likely to benefit software development. But improvements within the existing economic environment would still appear to leave the Soviet software development/user community well short of the systemic advantages enjoyed by its US counterpart. Since software is such a widely dispersed and pervasive technology, it would seem impossible to permit major reforms here without also permitting them elsewhere in the economy. It is doubtful if the needs of computing alone could build up enough pressure to bring about broad reforms in the economic system.

The USSR has lots of potential software talent and lots of need. The two have to be brought together in some effective way. Various forms of technology transfer from the West might serve as catalysts to help bring this about. However, the changes that will come will take time and have to fit in with the way things are done in the Soviet Union. Simple foreign transplants will not work. No reforms in a country that is as self-conscious as the USSR can be successful if they are divorced from Russian and Soviet traditions. But the history of Soviet computing shows a strong dependence on Western, and particularly US, technology and social/economic practices. Effective solutions to Soviet software problems will have to have a hybrid character.

Acknowledgments and Disclaimer

Various forms of support are gratefully acknowledged. These include a NSF Science Faculty Fellowship, a Sesquicentennial Associateship from the University of Virginia, and a research fellowship from the Center for International Studies at Princeton University. Other support has come from the US Army Foreign Science and Technology Center, Department of Defense, and FIO/ERADCOM, Ft. Monmouth, New Jersey. Continued collaboration with N. C. Davis of the CIA has been particularly valuable.

A couple dozen scattered paragraphs have been excerpted from Davis and Goodman (1978) and Goodman (1979). Permission has been granted by the ACM and the Princeton University Press. Some duplication was necessary to keep this article reasonably self-contained. Permission to use the quotations from Berliner (1976) in Section 3.2 and from Hardt and Frankel (1971) in Section 3.4 was granted by the MIT and Princeton University Presses.

The views expressed in this paper are those of the author. They do not necessarily reflect official opinion or policy of the United States Government.

References*

Agafonov, V. N. *et al.* (1976). "Generator of Data Input Programs for ES Computers." Statistika, Moscow.

Amann, R., Cooper, J. M., and Davies, R. W., eds. (1977). "The Technological Level of Soviet Industry." Yale Univ. Press, New Haven, Connecticut.

Andon, F. I. *et al.* (1977). Basic features of data base management system OKA. *Upr. Sist. Mash.* (2).

Ashastin, R. (1977). On the efficiency with which computer equipment is used in the economy. *Plan. Khoz.* May (5), 48–53.

Aviation Week (1972). July 31, 14.

Babenko, L. P., Romanovskaya, L. M., Stolyarov, G. K., and Yushchenko, E. L. (1968). A compatible minimal COBOL for domestic serial computers. Presented at *AU Conf. Prog., 1st, 1968.*

Bakharev, I. A. *et al.* (1970). Organization of teletype operations and debugging in the IAM operating system. Presented at *AU Conf. Prog., 2nd, 1970.*

Barsamian, H. (1968). Soviet cybernetics technology: XI. Homogeneous, general-purpose, high-productivity computer systems—a review. Rand Corporation, **RM-5551-PR.**

Bauer, F. L., ed. (1975). "Advanced Course on Software Engineering" (Munich, 1972). Springer-Verlag, Berlin and New York.

Belyakov, V. (1970). How much does a computer need? *Izvestiya* March 1, 3.

Berenyi, I. (1970). Computers in Eastern Europe. *Sci. Am. Oct.*, 102–108.

Berliner, J. S. (1976). "The Innovation Decision in Soviet Industry." MIT Press, Cambridge, Massachusetts.

* Foreign publication titles translated.

Bespalov, V. B., and Strizhkov, G. M. (1978). The equipment complex of the Unified System for teleprocessing of data. *Prib. Sist. Upr.* (6), 9–12.
Betelin, V. B., Bazaeva, S. E., and Levitin, V. V. (1975). "The ES-ASVT Small Operating System." Order of Lenin Institute of Applied Mathematics, Academy of Sciences USSR, Moscow.
Bezhanova, M. M. (1970). The Tenzor system program. Presented at *AU Conf., Prog. 2nd, 1970.*
Bobko, I. (1977). Testing. *Sov. Rossiya* July 12, 2.
Boehm, B. W. (1975). The high cost of software. *In* (Horowitz, 1975), 3–14.
Boehm, B. W. (1977). Software engineering: R&D trends and defense needs. *In* (Wegner, 1977), 1.1–1.43.
Bornstein, M., and Fusfeld, D. R., eds. (1974). "The Soviet Economy: A Book of Readings" (4th ed.). Irwin, Homewood, Illinois.
Borodich, L. I. *et al.* (1977). "ALGAMS-DOS ES Computers." Statistika, Moscow.
Bratukhin, P. I., Kvasnitskiy, V. N., Lisitsyn, V. G., Maksimenko, V. I., Mikheyev, Yu. A., Cherkasov, Yu. N., and Shohers, A. L. (1976). "Fundamentals of Construction of Large-Scale Information-Computer Networks" (D. G. Zhimerin and V. I. Maksimenko, eds.). Statistika, Moscow.
Brich, Z. S., Voyush, V. I., Deztyareva, G. S., and Kovalevich, E. V. (1975). "Programming ES Computers in Assembly Language." Statistika, Moscow.
Bucy, J. F. (1976). An analysis of export control of U.S. technology—a DoD perspective. Defense Science Board Task Force Report (Feb. 4) on Export of U.S. Technology, ODDR&E. Washington, D.C.
Burtsev, V. S. (1975). Prospect for creating high-productivity computers. *Sov. Sci.* **45.**
Burtsev, V. S. (1978). Computers: relay-race of generations. *Pravda* April 4, 3.
Buxton, J. M., Naur. P., and Randell, B. (1976). Software Engineering: Concepts and Techniques Proc. NATO Conferences, Garmish, West Germany, Oct. 7–11, 1968; Rome, Oct. 27–31, 1969. Petrocelli/Charter, New York.
Campbell, H. (1976). Organization of research, development and production in the Soviet computer industry. RAND Corporation, **R-1617-PR,** Santa Monica, California.
Cave, M. (1977). Computer technology. *In* (Amann *et al.,* 1977), 377–406.
Chevignard, D. (1975). Soviet automation and computerization effort. *Def. Nat.* (*Paris*) Feb., 117–128.
CSTAC TT (1978). Transfer of computer software technology. Jan. 20 Report of the Technology Transfer Subcommittee of The Computer Systems Technical Advisory Committee (CSTAC), U.S. Dept. of Commerce.
CSTAC II (1978). COMECON Ryad-II Report (Rev. 1, Feb. 22). Foreign Availability Subcommittee (CSTAC), U.S. Dept. of Commerce.
CTEG (1979). Computer Networks: An Assessment of the Critical Technologies and Recommendations for Controls on the Exports of Such Technologies. Computer Network Critical Technology Expert Group (CTEG), U.S. Dept. of Defense May.).
Davidzon, M. (1971). Computers wait for specialists. *Sot. Ind.* Dec. 25, 2.
Davis, N. C., and Goodman, S. E. (1978). The Soviet Bloc's Unified System of computers. *ACM Comp. Surv.* **10** (2), 93–122.
Del Rio, B. (1971). Cybernetics: To a common denominator. *Pravda* Jan. 5.
Dittert, W. (1978). ES-1055 computer. *Szamitastechnika* (*Hung.*) Jan.
Doncov, B. (1971). Soviet cybernetics technology: XII. Time-sharing in the Soviet Union. Rand Corporation, **R-522-PR,** Santa Monica, California.
Drexhage, K. A. (1976). A survey of Soviet programming. *SRI Tech. Rep. Proj.* **3226.**

Drozdov, E. A., Komarnitskiy, V. A., and Pyatibratov, A. P. (1976). "Electronic Computers of the Unified System." Mashinostroyenie, Moscow.
Dyachenko, A. I. (1970). Ukrainian Republic fund of algorithms and programs. *Mekh. Avtom. Kontrola* (1), 61.
Efimov, S. (1970). Horizontals and verticals of control. *Izvestiya,* March 8, **3.**
Ekonomicheskaya Gazeta (1976). Sept. 1.
Ekonomicheskaya Gazeta (1977). April 15.
Electrical Engineering Times (1977). Nov. 28.
Elorg/Computronics (1978). Growth of Soviet computers and Indo-Soviet cooperation: new high rate performance third generation computer ES-1033 from the USSR. May-June advertisement by V/O Elektronorgtekhnika, a Soviet foreign trade organization, and by Computronics, India, its marketing agent in India.
Ershov, A. P. (1966). ALPHA—An automatic programming system of high efficiency. *J. ACM* **13,** 17–24.
Ershov, A. P. (1969). Programming 1968. *Avtomat. Program.* (*Kiev*) 3–19.
Ershov, A. P. (1970). Problems of programming. *Vestn. Akad. Nauk SSSR* (6), 113–115.
Ershov, A. P. (1975). A history of computing in the USSR. *Datamation* Sept., 80–88.
Ershov, A. P., and Shura-Bura, M. R. (1976). Directions of development of programming in the USSR. *Kibernetika* **12** (6), 141–160.
Ershov, A. P., and Yushchenko, E. L. (1969). The first All-Union conference on programming. *Kibernetika* **5** (3), 101–102.
Evreinov, E. V., and Kosarev, Yu. G., eds. (1970). "Computer Systems." Akademiya Nauk, Novosibirsk; translated and published for the National Science Foundation by Amerind Publ., New Delhi, 1975.
Fadeev, V. (1977). Who is to answer for computer servicing? *Sots. Ind.* Sept. 4, 2.
Filinov, E. N., and Semik, V. P. (1977). Software for the SM-3 UVK. *Prib. Sist. Up.* (10), 15–17.
First AU Conf. Prog. (1968). "First All-Union Conference on Programming" (11 vols.), Kiev. Excerpts translated in *Sov. Cybern. Rev.,* July 1969, pp. 20–65.
Galeev, V. (1973). The collection is large but the benefit is small. *Pravda* Jan. 8.
GDR (German Democratic Republic) (1976). Ryad Overview. *In* "Rechentechnik Datenverarbeitung".Memorex, McLean, Virginia (distr.).
Gladkov, N. (1970). A help or a burden? *Pravda* Oct. 16 (2).
Glushkov, V. M. (1971a). The computer advises, the specialist solves. *Izvestiya* Dec. 15, 3.
Glushkov, V. M. *et al.* (1971b). ANALITIK (Algorithmic language for the description of computational processes with the application of analytical transformations). *Kibernetika* 7 (3), 102–134.
Glushkov, V. M., Ignatyev, M. B., Myasnikov, V. M., and Torgashev, V. A. (1974). Recursive machines and computing technology. *Proc. AFIPS Conf.,* pp. 65–70. North Holland, Amsterdam.
Godliba, O., and Skovorodin, V. (1967). Unreliable by tradition. *Pravda* Aug. 27, 3.
Goldberg, J. G., and Vogeli, B. R. (1976). A decade of reform in Soviet school mathematics. *CBMS Newsletter* Oct.–Nov.
Goodman, S. E. (1978). The transfer of software technology to the Soviet Union. Presented at "Integrating National Security and Trade Policy: The United States and the Soviet Union," a conference held June 15–17 at the U.S. Military Academy, West Point, New York.
Goodman, S. E. (1979). Soviet Computing and Technology Transfer: An Overview. *World Politics* **31** (4).

Gorlin, A. C. (1976). Industrial reorganization: the associations. *In* (Hardt, 1976), 162–188.
GOST 21551–76 (1976). "USSR State Standard for the Programming Language ALGAMS." Standartov, Moscow.
Granick, D. (1961). "The Red Executive." Anchor Books, Garden City, New York.
Hardt, J. P., and Frankel, T. (1971). The industrial managers. *In* (Skilling and Griffiths, 1971), 171–208.
Hardt, J. P., ed. (1976). "The Soviet Economy in a New Perspective." Joint Economic Committee U.S. Congress, Washington, D.C.
Holland, W. B. (1971a). Kosygin greets first class at management institute. *Sov. Cybern. Rev.* May, 7–11.
Holland, W. B. (1971b). Party congress emphasizes computer technology. *Sov. Cybern. Rev.* July, 7–14.
Holland, W. B. (1971c). CDC machine at Dubna Institute. *Sov. Cybern. Rev.* July, 19–20.
Holland, W. B. (1971d). Comments on an article by M. Rakovsky. *Sov. Cybern. Rev.* Nov., 33.
Horowitz, E., ed. (1975). "Practical Strategies for Developing Large Software Systems." Addison-Wesley, Reading, Massachusetts.
IBM RTM (1970). "Introduction to the Real-Time Monitor (RTM)." GH20-0824-0, IBM Systems Reference Library.
IBM DOS (1971). "Concepts and Facilities for DOS AND TOS." DOS Release 25, GC 24-5030-10, IBM Systems Reference Library.
IBM S/360 (1974). IBM System/360 Models 22-195. *In* "Datapro Reports" (70C-491-03). Datapro Research, Delran, New Jersey.
IBM S/370 (1976). IBM System/370. *In* "Datapro Reports" (70C-491-04) Datapro Research, Delran, New Jersey.
Informacio Elektronika (Hung.) (1977). Three articles on structured programming and program correctness verification. **12** (4).
Infotech Information Ltd. (1972). "Software Engineering." International Computer State of the Art Report. Maidenhead, Berkshire, England.
Infotech Information Ltd. (1976). "Real-Time Software." International Computer State of the Art Report. Maidenhead, Berkshire, England.
ISOTIMPEX (1973). English language description of the **ES-1020.** Bulgarian State Trade Enterprise ISOTIMPEX, Sofia. (Untitled, undated, assume issued 1973.)
Ivanenko, L. N. (1977). Imitation and game simulation of human behavior in technological and socioeconomic processes. Report on a conference held in Zvenigorod, May 27–June 1, 1977. *Kibernetika* **13** (5), 150.
Izmaylov, A. V. (1976). Software system for the 'Tver-ES' automated control system. *Ref. Zh. Kibern.* (8), Abstract No. 8G603.
Izvestiya (1978). March 14, 2.
Judy, R. W. (1967). Appendix: Characteristics of some contemporary Soviet computers. *In* "Mathematics and Computers in Soviet Economic Planning" (J. Hardt *et al.*, eds.), pp. 261–265. Yale Univ. Press, New Haven.
Kaiser, R. G. (1976). "Russia: The People and The Power." Atheneum, New York.
Kasynkov, I. (1977). *Izvestiya* March 4, 2.
Kazansky, G. (1967). Moscow *Nedelya* Dec. 4 (7).
Kharlonovich, I. V. (1971). Automated system for controlling railroad transport. *Avtom. Telemekh. Svyaz* (8), 1–3.
Khatsenkov, G. (1977). Instantaneously subject to computers. *Sots. Ind.* April 24, 1.
Khusainov, B. S. (1978). "Macrostatements in the Assembler Language of the ES EVM." Statistika, Moscow.
Kitov, A. I., Mazeev, M. Ya., and Shiller, F. F. (1968). The ALGOL-COBOL algorithmic language. *In AU Conf. Prog., Ist,* 1968.

Kmety, A. (1974). Demonstration of the R-20 at the capital city office for construction operations and administration. *Szamitastechnika (Budapest)* April–May, 1–2.
Koenig, R. A. (1976). An evaluation of the East German Ryad 1040 system. *Proc. AFIPS Conf.*, pp. 337–340.
Kommunist (Yerevan) (1977). Nov. 29, 4.
Kommunist (Yerevan) (1978). Dec. 31, 1.
Kryuchkov, V., and Cheshenko, N. (1973). At one-third of capacity: Why computer efficiency is low. *Izvestiya* June 14, **3.**
Kudryavsteva, V. (1976a). *Sov. Beloruss.* April 25, 2.
Kudryavsteva, V. (1976b). *Sov. Beloruss.* July 18, 4.
Kulakovskaya, V. P. *et al.* (1973). "Minsk-32 Computer COBOL." Statistika, Moscow.
Kuzin, L. T., and Shohukin, B. A. (1976). "Five Lectures on Automated Control Systems." Energiya, Moscow.
Lapshin, Yu. (1976). Maximizing the effectiveness of computer technology. *Sot. Ind.* Sept. 1.
Larionov, A. M., Levin, V. K., Raykov, L. D., and Fateyev, A. E. (1973). The basic principles of the construction of the system of software for the YeS EVM. *Upr. Sist. Mash.* May–June (3), 129–138.
Larionov, A. M., ed. (1974). "Software Support for ES Computers." Statistika, Moscow.
Leonov, O. I. (1966). Connecting a digital computer to telegraph communication lines in a computer center. *Mekh. Avtom. Proiz.* (8), 40–42.
Letov, V. (1975). Computer in the basement. *Izvestiya* Aug. 22, 3.
Liberman, V. B. (1978). "Information in Enterprise ASU." Statistika, Moscow.
Mamikonov, A. G. *et al.* (1978). "Models and Methods for Designing the Software of an ASU." Statistika, Moscow.
Meliksetyan, R. (1976). *Nedelya* Dec. 27, 3.
Mijalski, Czelslaw (1976). The principles, production and distribution of the software of MERA-ELWRO computers. *Informatyka (Warsaw)* Nov., **27.**
Mitrofanov, V. V., and Odintsov, B. V. (1977). "Utilities in OS/ES." Statistika, Moscow.
Mooers, C. N. (1977). Preventing software piracy. *In* "Microprocessors and Microcomputers" (selected reprints from *Computer*), pp. 67–68. IEEE Computer Society.
Moskovskaya Pravda (1978). April 8, 3.
Myasnikov, V. A. (1972). Need for improved computer technology. *Izvestiya* May 27, 2.
Myasnikov, V. A. (1974). Automated Management Systems Today. *Ekon. Organ. Promyshl. Proizv.* (6), 87–96.
Myasnikov, V. A. (1976). *Sov. Ross.* Dec. 24, 2.
Myasnikov, V. A. (1977). Results and priority tasks in the field of automation of control processes in the national economy of the USSR. *Upr. Sist. Mash. (Kiev)* Jan.–Feb. (1), 3–6.
Myers, G. J. (1976). "Software Reliability." Wiley, New York.
Naroditskaya, L. (1977). New computers are running . . . We audit fulfillment of Socialist pledges. *Pravda Ukr.* Nov. 18, 2.
NASA (1977). Standardization, certification, maintenance, and dissemination of large scale engineering software systems. NASA Conference Publication No. 2015.
Naumov, B. N. (1977). International small computer system. *Prib. Sist. Upr.* (10), 3–5.
Naumov, V. V. (1976). Real-Time Supervisor (SRV). *Programmirovanie* May–June, 54–60.
Naumov, V. V., Peledov, G. V., Timofeyev, Yu. A., and Chekalov, A. G. (1975). "Supervisor of Operating System ES Computers." Statistika, Moscow.
Nove, Alec (1969). "The Soviet Economy" (2nd ed.). Praeger, New York.
Novikov, I. (1978). They put their AMS up for sale. *Pravda* March 13, 2.
Novikov, N. (1972). Idle computers. *Pravda* Aug. 21.

Novoshilov, V. (1971). The levels of mathematics. *Izvestiya* Jan. 17, 3.
OECD Report (1969). Gaps in technology—Electronic computers. Organization for Economic Cooperation and Development, Paris.
Ovchinnikov, Yu. (1977). Science in a nation of developed socialism. *Izvestiya* Nov. 18, 2.
Parrott, Bruce B. (1977). Technological progress and Soviet politics. *In* (Thomas and Kruse-Vaucienne, 1977), 305–328.
Peledov, G. V., and Raykov, L. D. (1975). The composition and functional characteristics of the software system for ES computers. *Programmirovanie* Sept.–Oct. (5), 46–55.
Peledov, G. V., and Raykov, L. D. (1977). "Introduction to OS/ES." Statistika, Moscow.
Perlov, I. (1977). The ASU—Its use and return. *Ekon. Zhizn (Tashkent)* (6), 83–86.
Pervyskin, E. K. (1978). Technical Means for the Transmission of Data. *Ekon. Gaz.* June (25), 7.
Petrov, A. P. (1969). "The Operation of Railroads Utilizing Computer Technology." Transport, Moscow.
Pevnev, N. I. (1976). "Automated Control Systems in Moscow and Its Suburbs." Moskovsky Rabochy, Moscow.
Pirmukhamedov, A. N. (1976). "Territorial ASU." Ekonomika, Moscow.
Pleshakov, P. S. (1978). Utilizing Automated Management Systems Efficiently: Computer Hardware. *Ekonomicheskaya gazeta,* July 31, 15.
Rakovsky, M. (1977). Computers' surprises. *Pravda* March 2, 2.
Rakovsky, M. (1978a). According to a single plan. *Pravda* Feb. 3, 4.
Rakovsky, M. (1978b). On a planned and balanced basis. *Ekon. Gaz.* June (23), 14. (Quotations from a translation in *CDSP* Vol. XXX, No. 24, p. 24.)
Reifer, D. J. (1978). Snapshots of Soviet computing. *Datamation* Feb., 133–138.
Rezanov, V. V., and Kostelyansky, V. M. (1977). Software for the SM-1 and SM-2 UVK. *Prib. Sist. Upr.* (10), 9–12.
Robotron (1978). EC-1055 electronic data processing system. VEB Kombinat Robotron Brochure, May 25.
Rudins, George (1970). Soviet computers: A historical survey. *Sov. Cybern. Rev.* Jan., 6–44.
Sabirov, A. (1978). Specialty: cybernetics. *Izvestiya* March 12, 4.
Saltykov, A. I., and Makarenko, G. I. (1976). "Programming in the FORTRAN Language" (Dubna FORTRAN for the BESM-6). Nauka, Moscow.
Sarapkin, A. (1978). To new victories. *Sov. Beloruss.* Jan. 4, 1.
Second AU Conf. Prog. (1970). Second All-Union Conference on Programming, Novosibirsk. (Translated abstracts in *Sov. Cybern. Rev.* May, 9–16).
Shnayderman, I. B., Kosarev, V. P., Mynichenko, A. P., and Surkov, E. M. (1977). "Computers and Programming." Statistika, Moscow.
Skilling, H. G., and Griffiths, F., eds. (1971). "Interest Groups in Soviet Politics." Princeton Univ. Press, Princeton, New Jersey.
Smith, H. (1977). "The Russians." Ballantine, New York.
Solomenko, E. (1975). Machines of the Unified System. *Leningradskaya Pravda* May 15.
Sovetskaya Estoniya (1978). March 15, 2.
Sovetskaya Moldavia (1978). Jan. 1, 2.
Sovetskaya Rossiya (1976). Sept. 11, 4.
Tallin (1976). First IFAC/IFIP Symposium on Computer Software Control, Estonia. Paper titles published in *Programmirovaniye (Moscow)* (5), 100–102, and *Vestn. Akad. Nauk SSSR* (11), 1976, 93–94.
Taranenko, Yu. (1977). How to service computers. *Sots. Ind.* July 19, 2.
TECHMASHEXPORT (1978a). SM EVM Minicomputer Family: SM-1, SM-2. Marketing Brochure. Moscow.

TECHMASHEXPORT (1978b). SM EVM Minicomputer Family: SM-2. Marketing Brochure. Moscow.
Thomas, J. R., and Kruse-Vaucienne, U. M., eds. (1977). "Soviet Science and Technology," pp. 305–328. National Science Foundation, Washington, D.C.
Tolstosheev, V. V. (1976). "The Organizational and Legal Problems of Automatic Systems of Control." Ekonomika, Moscow, pp. 49–50.
Trainor, W. L. (1973). "Software—From Satan to Savior." USAF Avionics Laboratory, Wright-Patterson AFB, Ohio. Referenced in (Boehm, 1975).
Trofimchuk, M. (1977). How do you work, computer? *Pravda Ukrainy* Sept. 7.
Trud (1977). Jan. 14, 2.
Trud (1978a). Jan. 4, 1.
Trud (1978b). Nov. 7.
Vasyuchkova, T. D., Zaguzoba, L. K., Itkina, O. G., and Savchenko, T. A. (1977). "Programming Languages with DOS ES EVM." Statistika, Moscow.
Vodnyy Transport (1977). Riga ship repair plant to use ASU with 'Tver' software system. Sept. 24, 4.
Ware, W. H., ed. (1960). Soviet computer technology—1959. *Commun. ACM* 3 (3), 131–166.
Washington Post (1978). The battle of Minsk, or socialist man beats computer. March 28.
Wegner, P., ed. (1977). Proc. Conf. on Research Directions in Software Technology. Final version to be published 1978, MIT Press, Cambridge, Massachusetts.
White, H. (1977). Standards and documentation. *In* (NASA, 1977), 20–26.
Yanov, A. (1977). Detente after Brezhnev: The domestic roots of Soviet foreign policy. Policy Papers in International Affairs, No. 2. Institute of International Studies, University of California, Berkeley.
Zadykhaylo, I. B. *et al.* (1970). The BESM-6 operating system of the USSR Academy of Sciences' Institute of Applied Mathematics. *In AU Conf. Prog., 2nd, 1970.*
Zarya Vostoka (1976). July 28, 2.
Zhimerin, D. G. (1978). Qualitatively new stage. *Ekon. Gaz.* May 22, 7.
Zhimerin, D. G., and Maksimenko, V. I., eds. (1976). "Fundamentals of Building Large Information Computer Networks." Statistika, Moscow.
Zhukov, O. V. (1976). "Generation of Programs For Data Processing." Statistika, Moscow.
Zhuravlev, V. (1973). Translators for computers. *Pravda* Feb. 20.