PART I
CODING COLLECTIVES
In the early 1990s, in a typical middle school located in an industrial neighborhood of the city of Novosibirsk, we had an “informatics” class where we learned about the principles of hardware and programming and could play computer games. The machines were not called “computers” but “evm” (electronic calculating machines); they had a gray and green interface and were all connected to the main computer controlled by the teacher.

My high school was an experimental school endowed with additional funds, and was where I first experienced a modern computer class with “real” personal computers. The classroom itself often stayed closed behind iron doors and barred windows—during the “wild” 1990s the robbery of school computer classrooms supplied with expensive foreign machines was common. This classroom was closed because the informatics teacher I had met during the admission tests left for Israel; she was greatly missed by older students, who said she was very competent. Eventually, the administration found a replacement and we began to learn how to use a text-editing application. By this time it was generally understood that Word and Excel were what informatics classes should be about. In those days, I was busy discovering French existentialism and Russian semiotics and remember cheating on the Excel assignment.
Among other things, the collapse of the Soviet Union wrecked the national education system and opened the country’s frontiers: a calamity turned into an opportunity when I got a chance to study abroad. Moving from Russia to France and then to the US, my own personal trajectory impacted my research subject: the history of Soviet computing from a transnational perspective. Working on my PhD thesis and book manuscript (Tatarchenko 2013), I uncovered in the history of computing itself explanations and connections that shed light on what I experienced in my computer classes as compared to those skills taught to my American friends. In this connection, it is worth pointing out that claims stating the Soviets had missed the “Computer Revolution” were at best misleading and that the relative rarity of personal computers in Soviet homes did not represent the absence of a computer industry or professional programmers. I learned that Western and Eastern IT histories were entangled on many levels and that the Iron Curtain simultaneously isolated and connected these two worlds.

To discern the depth of the transnational connections, we need to consider multiple facets of Soviet IT, including: hardware and software as complex technological artifacts; the emergence of a new mathematical discipline called “computer science” in English and informatika in Russian; a set of localized practices; and machines as emblems of political legitimacy. The Cold War military and space race was the chief driving force behind the miniaturization of electronic components encapsulated by what is termed “Moore’s law.” The military origins of American networking systems and the parallel Soviet efforts to computerize their economy into a single “big data” network are other well-known cases of contemporaneous IT systems that had Cold War origins. If electronics and computer network technology were the material embodiments of competition between the East and West, the capitalist and socialist versions of modernity were equally rooted in a techno-utopian imaginary that led to different visions of the “Information Age.” Accordingly, the curricula for school computer education reflected two versions of an “information society”: the American one was predicated on a proficient instrumental use of the personal computer as a basic commodity and a data-processing device; the Soviet one aimed at inculcating thought habits and programming skills in an effort to enable self-control and self-expression for a new kind of responsible individual.

The collapse of the party state and the ensuing political transformations put an end to the project of creating a “socialist information society.” The invasion of global IT products following the opening of the Russian markets during the 1990s dramatically altered the material landscape of computing
in the New Russia. Yet a half century’s worth of Soviet experience with computing did not just disappear; instead, important continuities exist across the 1991 fault line. In this chapter, I take a synthetic approach to the history of Soviet programming in order to provide context and genealogy explaining the distinctly national dimensions of the contemporary IT landscape. First, I provide an overview of the pioneering stage of Soviet programming efforts, as shaped by early Soviet hardware and cybernetics. Next, I focus on the commodification of programming work and analyze the professionalization efforts led by Soviet programming experts who came to claim that programming was a form of human and machine brotherhood. I then conclude with reflections on the philosophy behind the 1985 educational reform, which introduced compulsory programming classes within a context where the cloning of Western hardware became the norm.

EXCLUSIVE AND ILLUSIVE: EARLY PROGRAMMERS BETWEEN ENGINEERS AND CYBERNETICIANS

The specificity of Soviet computing history is inextricably linked to key features of the socialist state: its planned economy and the party’s ideological guidance. Centralized power and the planned economy did not lead to an absence of inventiveness or competition. On the contrary, from the first days of Soviet digital computing efforts in the late 1940s, the development of Soviet machines was marked by a rivalry between different groups of specialists. Similar to the Anglo-American debates on the “firsts” embroiling the epithets of “digital,” “programmable,” and “fully operational,” a controversy surrounds the status of the “first” Soviet computer. The chief protagonists in the battle for public memory are M-1 and MESM machines (the former built in Moscow, the latter in the suburbs of Kiev), both important less for their performance characteristics than for their influence on the design and training of the workforce involved in two larger, also competing, hardware projects: Strela and BESM. Operational by the mid-1950s, these two computers engendered a new kind of occupation: professional programming.

Western specialists studying Soviet technology during the Cold War era were little concerned by these priority disputes. Instead, they grappled with the issue of technology transfer, asking questions such as: Were the first Soviet computer developments independent of one another? Were the snippets of information in the form of publicity and published overviews really all the Russians had? As the evidence relating to these questions remains partial at best, it seems sensible to turn our attention to the well-known case
of the Soviet nuclear bomb project. According to the latest analysis by historian of science and technology Michael Gordin (2009), even such sustained information-collection efforts as were organized by Soviet intelligence under the secret police chief Lavrentiy Beria could not solve the major problem of information fragmentation, management, and trust. In fact, it was the public knowledge about the technical feasibility of the project and various published reports that were most responsible for the astonishing speed of Soviet nuclear efforts. These observations help elucidate the issue of transfer in the case of Soviet digital computing: feasibility was no doubt the most crucial piece of Western knowledge for early Soviet projects.

If the circulation of scientific overviews stimulated the efforts of Soviet engineers and mathematicians interested in calculation problems, the relationship between early Soviet and early Western computer technology was not limited to a unilateral flow of technical information and soon became shaped by an ideological confrontation over technology’s place in society. The speed, size, and cost of early computers attracted the attention of the media and fascinated the Western popular imaginary, famously associating computers with “giant brains.” However, in Russia what became known as an anticybernetic campaign was founded in a series of publications that appeared in the Soviet press from the spring of 1950 to the winter of 1955. The first publication was a witty analysis of the militarism implied in the anthropomorphic representation of the Harvard Marc III machine on the pages of Time magazine, but later articles attacked a specific enemy: the new American science of cybernetics (Peters 2012).

In his book Cybernetics: Or Control and Communication in the Animal and the Machine (1948), the American mathematician Norbert Wiener introduced the term and gave it a vague definition as “scientific study.” The interdisciplinary and metascientific ambitions involved in drawing connections between biological and man-made systems, as well as the explicit analogies between machines and human institutions articulated in Wiener’s bestseller, The Human Use of Human Beings: Cybernetics and Society (1950), made cybernetics an inviting ground for ideological contestation. The Soviet publications labeled cybernetics a pseudoscience, a triumph of semantic idealism, and the newest form of mechanical philosophy, all the while stressing its role as a tool of Western militarism. In the wider context of rising geopolitical tensions and Soviet domestic efforts to exert a hold over the creative and scientific intelligentsia (Zhdanovshchina), the anticybernetic campaign was peculiar because unlike the infamous case of Lysenkoism and the antigenetic campaign, it was not directed against any Soviet individuals.
or institutions. The most direct result of the anticybernetic publications—the withdrawal of Wiener’s books from Soviet libraries—did not mean that Soviet experts ignored this new Western development.

The Soviet ideologues were not the only Soviet specialists to observe the military roots and implications of the new calculating technologies and cybernetic notions. Wiener’s works were held in closed libraries and several unofficial translations circulated among experts. One reader was a young graduate of the Dzerzhinsky Artillery Academy, Anatoliy Kitov. Employed as a military representative at the special construction bureau—the SKB-245—and responsible for the design of the Strela computer, Kitov became an early Soviet programming expert and a proselyte of cybernetic ideas among the Soviet military and scientific elites. Reading and being interested in such ideologically suspicious works was not an act of rebellion but a logical step within a belief system that postulated employing Western technology in a battle against capitalism. Working on secret military projects, early Soviet experts fascinated by cybernetics and digital computing were eager to fight against an imagined Western aggressor, but their immediate threats were much closer—the competitors for state funds offering alternative technical solutions.

Created in 1948 under the auspices of the Soviet Academy of Sciences, the Institute of Precise Mechanics and Computational Technology (ITMiVVT) had its mission to develop computational technology inscribed in its very name; tellingly, however, computation did not yet imply “electronic” or “digital” but rather “mechanical.” This situation changed in 1950, when the mathematician and explosives expert Mikhail Lavrentyev took over the directorship of the new organization. In his memoirs, Lavrentyev (2000, 57–60) detailed his shrewd solutions for dealing with material scarcity by returning to patronage networks, revealing that no tactic was too dirty. Lavrentyev recounts how he accused leaders of the analogue technology projects of machinations with bureaucratic documents (a common Soviet practice at which Lavrentyev excelled) in order to force them out of the institute. Competitors out, old friends in. Returning to the capital from the Ukrainian Academy of Sciences, Lavrentyev also transferred his protégé and the designer of the MESM computer, Sergey Lebedev, and his team of engineers from Kiev to Moscow.

Lavrentyev’s reliance on his prewar Moscow networks also brought to the ITMiVVT the mathematician Lazar’ Lusternik, an old companion from the famous Luzitanya, a group of mathematicians formed around Nikolay Luzin in the 1920s, and a colleague at the Central AeroHydrodynamic Institute.
(TsAGI), the cradle of Soviet aviation during the 1930s. A leading figure in early Soviet programming efforts at the ITMiVT, Lusternik recreated the practices with which he was familiar from his time at the TsAGI: a close co-operation between mathematicians and engineers as well as scientific forms of interaction. In 1950, he organized a seminar on programming where the available literature was read and discussed in a scientific fashion. Lusternik's seminar led to the publication of an influential overview volume: The Solution of Mathematical Tasks on the Automatic Numerical Machines, a collective work, with the subtitle Programming for High-Speed Electronic Calculating Machines (Lusternik et al. 1952). Over three hundred pages long, it covered all aspects of programming, from a brief introduction to digital computers and numerical systems, to detailed examples of programming techniques for a “hypothetical” three-address machine, which was in fact the besm computer, which was under development. Circulated under conditions of restricted access, this was the book that introduced most early Soviet programmers to their new craft.

While mathematicians in Lusternik's group considered the problem of how to solve mathematical tasks using computers before any operational Soviet machines even existed, the Soviet science administrators began to work on the crucial question of who would operate them. In 1948, the same year as the creation of the ITMiVT, Moscow State University (MGU) created a new chair in computational mathematics. Here, at the most prestigious school in the country, the mathematics department had few volunteers among its student body willing to abandon their aspirations in pure mathematics for the unknown perspectives of machine mathematics. Assigning students to the chair was the typical top-down solution to the problem of who but not the how of turning them into specialists of the new machine computation. The logic behind the curriculum—freed from many of the traditional subject areas in mathematics to make room for classes like “Algebra of Relays” and “Theory of Machines and Mechanisms”—implied that in order to program one needed to understand the mechanisms of machines. After struggling through the eclectic curriculum, the first graduates of the computational mathematics chair were to learn their jobs on the fly.

According to graduates' memoirs, they spent their last year as interns at the ITMiVT learning to code on the besm computer. The actual experience of interaction with the new machine was immersive: its twinkling lights, sounds, and heat combined with the very size of the installation impressed its operators with a sense of almost mysterious power. But it is the human element of interaction, the shared learning and competition between peers
on how best to control the machine, that had a deep impact on one’s sense of self: “Programmers were counted on fingers, and joining this tribe filled one with a feeling of exclusivity” (Podlovchenko 2003, 372). This tribe, a small group of pioneers bound by the unique experience of working on the first Soviet machines, would influence Soviet programming for several decades to come. The members of the group would go on to lead software projects, consult for new hardware development, and teach many generations of programmers. Unlike the designers of the first machines, they are not in the spotlight of public memory but act as a less visible network transmitting disciplinary mythologies.

The important events that shaped the professional representation of this core group were closely associated with cybernetics, which had radically changed its status from a tool of imperialism to a mathematical metascience in the service of communism. By the fall of 1955—when the existence of Soviet computers was first officially announced in conjunction with an international conference in Darmstadt, West Germany—the scientific reputation of cybernetics had already been publicly redeemed through the appearance of a seminal publication (Sobolev, Kitov, and Lyapunov 1955) titled “Osnovnyye cherty Kibernetiki” (The main features of cybernetics) appearing in the key Soviet ideological journal, Voprosy filosofii (Questions of philosophy). The text of the article was drafted by the young colonel Kitov and cosigned by his former teacher from the Dzerzhinsky Artillery Academy, mathematician Aleksey Lyapunov, and Sergei Sobolev, an academician and mathematical prodigy who contributed to the Soviet nuclear bomb project. The article focused on explaining the subject and methods of cybernetics and stressed the legitimate scientific nature of the discipline. According to Slava Gerovitch (2002), the almost decade-long delay in the introduction of cybernetics to the Soviet public had an impact on its very content: unlike the servomechanisms that inspired Norbert Wiener, computers became the machines of reference in the Soviet version of cybernetics, which began to gain popularity in the late 1950s.

I argue that these crucial insights entail yet another observation: by 1955, Kitov, Lyapunov, and Sobolev not only presented computers as the machines of reference for cybernetics but also believed that programming itself was mathematical and a part of a machine’s self-regulative processes and amenable to automatization. In addition to their highly visible and celebrated role in changing the status of Soviet cybernetics, the three authors played key roles in spreading this vision of programming as pioneer practitioner, mentor, and patron, respectively. Aligning programming with
cybernetics by highlighting its mathematical foundations, their publication became the most visible national instantiation of the general international trend toward the development of so-called high-level computer languages and automatic programming systems. The relationship between programming and cybernetics in the 1960s was complicated by the mathematics-based cybernetic orientation on eliminating programming labor by developing system software and the parallel growth of programming as a mass profession. This tension would eventually be resolved with the establishment of a separate disciplinary and professional identity for programmers. However, the visions stipulating the automation of programming labor still animate policy discourses such as that of Dmitry Marinichev (Biagioli and Lépinay, this volume).

IN SEARCH OF IDENTITY: INTEGRATING INTERNATIONAL COMMUNITY, PRODUCING A NEW SOCIALIST COMMODITY

Up to the mid-1950s, programming practice, coding, and notations were highly localized: the craft of an individual programmer was dependent on the intimate knowledge of specific features of a machine and on devising clever tricks to use them efficiently. Proliferation of computers and their transformation from military and scientific calculators into business data-processing machines brought forward the problem of coordinating human efforts and introduced the difficulties of transmitting programs between machines. Across the Atlantic, “hardware” became the colloquial term for computer equipment during the 1950s. The term “software,” on the other hand, was initially invented in 1959 and came to denote everything that was not hardware: notation, consulting, and the new programming tools such as assembly systems, compilers, and operating systems. Running parallel to this process of the commodification of programming—encapsulated in the “ware” part of software—the professionalization of programming involved the creation of the first computer science departments in American universities and ongoing debates about their pedagogical mission. Confusion over the meaning of “software” and “program” reflected the ill-defined ontological status of a programmer’s work, astutely observed by the historian of computing Thomas Haigh (2002, 6): “Not all ’software’ was programs, and not all programs were software.”

These Western developments are important to Soviet programming for two reasons—comparison and contextualization. For instance, the fluidity of the better-known American case helps to clarify the paradoxical nature
of the Soviet term similar to “software,” which came into use in the early 1960s: “mathematical supply” (obespecheniye). Starting in the late 1950s, the serial production of computers—such as the lamp-based m-20 and its transistor-based modifications—led to the same problems of coordination from human to human and from machine to machine familiar to the West. In 1960, the Soviet military and scientific organizations that used the m-20 machines met at a conference that resulted in the first user association created to facilitate the exploitation of the machines, and a crucial part of that process became the exchange of programs. Such bottom-up initiatives were soon institutionalized at the state level with a commission on mathematical supply attached to the State Committee for Science and Technology (GKNT). Similar to the vagueness of the English-language “software,” the Russian-language “mathematical supply” did not provide for a clear distinction between product and service. However, the epithet of “mathematical” helps trace the direct relationship between Western and Soviet developments in programming—a relationship based on a shared belief in the mathematical nature of programs.

In the political chronology of Cold War interactions, 1955 (the year of the Geneva Summit) appears as an important turning point and a moment when the theory of peaceful coexistence was articulated and enacted. The theory provided a functional framework for the rise of Cold War scientific internationalism, best known for the Atoms for Peace meetings and the spectacular launch of Sputnik during the International Geophysical Year. The early Western-Soviet contacts in computing fit the same larger scheme: spanning activity from participation at international professional conferences and workshops held by the new International Federation for Information Processing (IFIP) to bilateral exchanges. While similar to other strategic technologies, information collection was one of the driving forces behind the exchanges in computing; the dynamic of the first Soviet-American exchanges of computer specialists demonstrates that intellectual coordination and preoccupations with the nature of human-machine interactions were the key elements.

Soviet integration into the international Algol project—a result of visits by American scientists in the late 1950s—provides the best example of mutual efforts triggered by shared beliefs in the power of mathematics for transcending all barriers. By 1960, the project acquired a large set of European participants and an anti-IBM ideology. Unlike the already popular Fortran devised to fit the characteristics of IBM machines, Algol was thought to be a truly universal, machine-independent, and mathematically sound
language, empowered not by corporate capital but by scientific internationalism embodied in collective work on its definition and standardization. Considered a practical failure in the US—on the basis of the number of compilers and not its larger influence—and a moderate success in Europe, Algol became the most widely used computer language in the Soviet Union and Eastern Bloc countries during the 1960s.

The puzzle behind Algol’s popularity in the East still involves many unknown elements, but the core element on the level of ideas was the conceptualization of the program as a mathematical object, an approach familiar in Soviet circles thanks to the Soviet version of mathematical cybernetics. By the late 1960s, there were at least half a dozen Algol compilers for the m-20 computers in the Soviet Union, along with others for the newer and more powerful Soviet machines, such as the besm-6. Competing research groups in Moscow, Leningrad, and Novosibirsk made important efforts at distributing and publicizing their work. The particularly rich published accounts and documentary sources produced by the Novosibirsk group—which benefited from the showcase status of the scientific center Akademgorodok, located in Novosibirsk—demonstrate the changing conditions of programmers’ work and the emergence of new organizational challenges. These challenges surmounted by Akademgorodok computer pioneers would find an echo in the post-Soviet Siberian initiatives to capitalize upon the region’s reputation as an IT hub dating from the 1960s (Indukaev, this volume).

To produce large-scale software systems such as compilers it was not enough to add together individual skills and a labor force. Published by the project leader Andrey Ershov in the local newspaper in January 1965, “The Alpha-Birth” recounted the challenges of producing an automatic programming system competitive in quality to manual programming. The unexpected technical troubles, the missed deadlines, and the doubling of the code volume from the expected twenty thousand to forty thousand lines, were all typical problems that demanded solutions bridging the technical and the social. The coordination of effort was paramount for the ultimate success of the Novosibirsk group and is clearly still a major issue for today’s companies, such as in the case of Yandex’s emphasis on a shared set of code-writing skills (Fedorova on Yandex, this volume). In Akademgorodok, collective coding became a personally fulfilling experience. “We will keep the gained experience, deep satisfaction with the completed work and the priceless camaraderie,” wrote Ershov, “that was born and matured during the years of work on Alpha-system.” In other words, the “birth” of a compiler was predicated on the creation of a collective with a family-like cohesion.
The work of Ershov’s group is of particular importance not only because of the technical features of the system, but because the creation of a compiling system and the formation of a programming collective came together with the coming of age of a new leader in the field of programming. Riding on the success of his Alpha system, Ershov claimed a professional and disciplinary identity separate from Soviet cybernetics. A talented and ambitious student who participated in Lyapunov’s famous class on the principles of programming at MGU, and in 1957 a PhD student and group manager at the new Akademgorodok Computer Center, by the late 1960s Ershov had grown into a pundit and spokesperson for programming on both the national and international level (Kraineva and Cheremnykh 2011).

Ershov was named head of the state commission formed by the GKNIT to monitor the development of mathematical supply in the Soviet Union and became the main author of its report submitted in summer 1968 (Ershov 1968). A snapshot of the Soviet programming landscape, the report estimated the number of Soviet system programmers at about one thousand, almost equally distributed between the academic computing centers, the key hardware production facilities, and a series of military-industrial organizations. But the report’s most interesting aspect was its language and the policy recommendations that squarely placed the Soviet programming community within the international milieu. In it, Ershov argued for rapid growth and the professionalization of system programming in order to achieve the Soviet computerization goals: the creation of “system programmer” as an established engineering profession; the separation of service and research functions in academic computer centers; and an orientation on borrowing software libraries for the new Soviet family of computers, later known as the United System (ES).

The date of the report, July 1968, is crucial for understanding the full meaning of its content. On the one hand, it captures a moment in time when Soviet experts were still debating the costs and benefits of orientating the new Soviet series on the IBM architecture and the best mechanism for doing so. On the other, the language of and arguments within the report reflected and prefigured the largest concern of the international community at the time, which became encapsulated in the notion of “software crisis.” The NATO-sponsored conference on software engineering held in Garmish (West Germany) on October 7–11, 1968, became the epicenter for reflection on software risk and reliability as well as a forum for a very pragmatic preoccupation with the costs of software production, the status of programmers’ labor, and the solution to the “software crisis” associated with the
creation of “software engineering.” To sum up, Ershov’s deep integration into the international community, traceable to the Algol group’s formal and informal networks, shaped his solutions to peculiar Soviet problems. At the same time, as a Soviet professional, Ershov took on as his responsibility the state’s interests in international prestige, the computerization of research and production, and the education of new professionals.

Recognizing that to become an accepted profession programming needed its own mythology, he used the available cultural resources to articulate his vision of an ideal professional for both domestic and foreign audiences. Invited to deliver a prestigious keynote speech at the main American professional conference for computer experts in 1972, Ershov described a universal ideal programmer by creatively combining Soviet rhetorical structure with Western references: “In his work, the programmer is challenged to combine, with the ability of a first class mathematician to deal in logical abstraction, a more practical, a more Edisonian talent, enabling him to build useful engines out of zeros and ones, alone [sic]” (Ershov 1972, 502). To emphasize the transcendent quality of the new profession, Ershov did not shy away from borrowing biblical language and imagery, where a programmer “feels himself to be the father-creator of the program, the son-brother of the machine on which it runs, and the carrier of the spirit which infuses life into the program/machine combination” (504). The highest aspiration of such an ideal practitioner according to Ershov was to spread the gospel of programming to all humanity in a recognizable logic of both Christian and Marxist worldviews. “Is it not however the highest aesthetic ideal of our profession,” concluded the Soviet expert, “to make the art of programming public property, and thereby to submerge our exclusiveness within a mature mankind?” (505).

The reception of the speech, which immediately met with enthusiasm from its Western audience, and its present-day relevance visible in recent citation patterns point to the ongoing elaboration of professional identity in the field of programming. All grandiloquence aside, Ershov nonetheless reflected upon the mundane and concrete aspects of a programmer’s labor as shared across the Atlantic and behind the Iron Curtain: their interaction with the machine, between themselves, and with society at large as complicated by the pace of hardware evolution, the status of scientific knowledge, and institutional struggles. At the same time, such shared concerns did not preclude Ershov’s and his Western colleagues’ awareness of the different political and economic structures that were in place. They considered it their duty to serve the needs of their respective countries. For Ershov, this duty
found its utmost expression in his involvement in the 1985 reform introducing computer education into the Soviet school system.

CONTESTED VISIONS OF LATE SOVIET COMPUTERIZATION

While it is beyond the scope of this overview to provide a full account of Soviet programming developments during the 1970s, 1980s, and early 1990s, the following question is nevertheless unavoidable: Is failure the only way to describe late Soviet computer developments? By the early 1970s, there was not only a Soviet computer industry but also a new Soviet profession: the programmer. Although not nearly as numerous as their American colleagues, programmers had specialized journals, professional meetings, and even a public relations coup thanks to the national and international victories of the chess program, Kaissa. Yet when the Cold War ended with the Soviet collapse, Western specialists were quick to observe that the Soviet Union entered the 1990s without computers. Although such observations chiefly referred to the absence of personal computers in Soviet households, they implied a more general Soviet failure to experience what is generally coined the “Computer Revolution” and to enter the “Information Age.” The 1968 state decision to make the new Soviet computer series compatible with IBM 360 architecture was considered a major turning point away from original Soviet research and development to systematic illicit borrowing.

In fact, the practice of reverse engineering shaped the materiality of late Soviet computing from ES large-frame computers to minicomputers to Apple and PCs and even to pocket programmable calculators. While there is no single systematic study of the late Soviet computer industry, the common Western perception holds that the planned economy could not handle the sophistication of microelectronics production, and that the party's monopoly on information could not allow for the diffusion of personal computers, a technology that allegedly enabled freedom of expression. To reconstruct the major changes in Soviet programming in this material environment defined by imitation we need to break with the circularity of such explanations and account for the ongoing contestation of Soviet computerization schemas from within.

Several observations are relevant for understanding the professional challenges and aspirations of Soviet programmers during the late Soviet period. On the one hand, changes due to the importation of Western technologies were mitigated by important institutional and social continuities.
For instance, the pioneer computer organization focusing on military computing, the skb-245, was integrated into the Scientific Research Center for Electronic Computational Technology (NITsEVT), the lead organization supervising the developments of es computers in the Soviet Union. By the same logic, the NITsEVT software department relied on established academic experts in programming to oversee its projects. On the other hand, major changes appeared due to large-scale diffusion of hardware: by the late 1970s about 70 percent of all computers in the country were es machines. Although plagued by many delays and reliability issues, the mass production of es machines (estimated at about sixteen thousand units for the entire period of production) led to the spread of computers across the country, entailing the demand for many more exploitation engineers, system programmers, and operators. New economies of scale led to major changes in the Soviet computing landscape. Yet social continuities also translated these changes into particular hybrid practices on the ground; hybridization not only explains the operation of post-Soviet it but in a more uncanny way appears to foreshadow the unpredictable outcomes stemming from the Putin-era initiatives to transplant Western forms of innovation onto the Russian soil (Simonova, this volume).

A sketch of the activities of Ershov’s department at the Akademgorodok Computer Center elucidates the issue. The work on the original computer research and development never completely stopped. In the early 1970s, Ershov was able to create a production spin-off of his department, a software construction bureau institutionalized as the Novosibirsk branch of the IT-MiVT and charged with developing software for the supercomputer that continued the besm-6 line, called Elbrus. At the same time, Ershov’s group was able to obtain funds under the es umbrella to work on an extremely ambitious experimental project that was never delivered to its customer—a multilingual translating system called beta for Algol 68, Simula, and PL/1.

Understanding how various actors adapted to the practice of borrowing on the local level helps us to appreciate the paradoxical characteristics of more general patterns that appear on a national scale: for example, the widespread custom of modifying hardware and programming in machine codes at the institutional level was disrupted, but tinkering with electronics became a common hobbyist practice with the availability of discarded pieces and the mass production of affordable programmable calculators; experts trained in the tradition of Soviet mathematical cybernetics turned into traveling lecturers on the es operating system, while in the 1970s PL/1 took the place of Algol 60 as the most popular language in civilian computer centers;
Western home computers were sold on the black market and at the closed foreign currency shops, and the network of afterschool education centers carrying programming classes for children led to a wide popularity of Basic by the mid-1980s.

This usage of Western programming languages and software packages and the adoption of corresponding programming practices had its advantages, and its costs. As became apparent to the early critics of the Soviet policy of giving priority to technology transfer over investment in the indigenous projects, the oft-cited argument about the benefits of copying hardware architecture to obtain access to millions of lines of software did not account for two crucial factors: First, there was the inherent difficulty of understanding foreign, often illegally obtained, programs. Without appropriate documentation and human interaction, the economy of labor was questionable. Second, there was the concern for nurturing a critical mass of qualified specialists working on cutting-edge developments in systems’ software. The expertise of system programmers was not solely evaluated by a degree or diploma but represented a personal profile built over years of experience and corresponded to a particular mindset. Borrowing software disrupted such dynamics of professional growth. To sum up, the orientation on copying hardware to save on software development led to a radical growth in the number of programmers, but it simultaneously aggravated questions surrounding the status of their work and training.

These implications are best demonstrated following Ershov’s and his group’s trajectory. While Ershov ascended in the Soviet scientific hierarchies, and as his research on the mathematical nature of compilation became recognized on the national level with the prestigious Krylov Prize, his vision of the professionalization of programming soared. Masses of programmers were trained on the job by the computer industry and dispersed throughout the computer centers of the production ministries. While there were many specialized national and international conferences on the topics of computer science and theoretical programming held in the Soviet Union in the 1970s and 1980s, no meetings followed the two Union-wide professional conferences for programmers held in Kiev (1968) and Novosibirsk (1970). Despite the requests for more specialists, the universities, including the Novosibirsk State University where Ershov taught, resisted classes in computing.

In the context where Soviet scientific hierarchies prioritized the authority of pure science, applications, including programming, became niches for what was considered a less reliable labor force—women—and safe havens for “suspicious” social elements: Jews. As a result, Soviet computer centers
were prone to high rates of turnover; there was widespread headhunting for system programmers and women often did not return or had to be retrained after taking maternity leave. In addition, programming was one of the areas directly impacted by the Soviet decision to allow Jewish emigration in the early 1970s (Fedorova on Israel, this volume; West, this volume).

The challenges of the Soviet professionalization of programmers’ labor described above provide crucial context for understanding the trajectory of the late Soviet project to universalize programming skills, crystallized in the 1981 slogan “Programming, the second literacy” (Ershov 1981). During the 1970s, Ershov reaffirmed his belief that reliable software is a direct reflection of the personal virtues of the programmer: logic, patience, and discipline. Addressing the young readers of the specialized mathematics journal for youth, Kvant, Ershov claimed that “good programmers are people of a special kind” and invited schoolchildren to learn programming to become members of tomorrow’s computer centers (Ershov and Zvenigorodskiy 1979). In the following years, Ershov articulated a fully developed pedagogical agenda of computer education whereby the human qualities shared by “good programmers” could be nurtured early on, from childhood, by developing a set of mental habits: “algorithmic thinking.” Such an agenda was backed by results of the collective work of his computer education group, which developed experimental pedagogical software and ran successful summer camps and distance-learning experiments throughout the 1970s and 1980s.

Although the educational activities of the Akademgorodok group were not unique per se, they led to very distinct results: Ershov’s status as a pundit contributed to the international diffusion of his philosophy of computer education, and his authority within the Soviet hierarchies helped him to integrate Moscow’s political networks. Ershov’s alliance with the ambitious physicist Yevgeniy Velikhov, known for his close connection with the rising party leader, Mikhail Gorbachev, underlie the transformation of his vision into state policy. Among the first state orders signed by Gorbachev when he entered office in winter 1985 was the computerization of education, including the introduction of compulsory classes in “The Basics of Informatics.” Although grounded in the ideas and experiments of the 1970s, within the changing political context of the mid-1980s the reform and its pedagogical agenda became rearticulated in coordination with perestroika and its ideals of democratization.

At the same time, the implementation of the reform was anything but a simple affair: teachers were to be trained, textbooks printed and delivered, computers and software produced and distributed. Characterized by numer-
ous setbacks at every level, the reform became associated with the weaknesses of the Soviet state. The delays with the delivery of computers to schools led to descriptions of the reform as a late Soviet absurdity: learning to program without computers was akin to learning to swim without water. The relative ease of obtaining the Japanese computers in Vladivostok, as documented by Aleksandra Masalskaya and Zinaida Vasilyeva in this volume, was more of an exception to the general scarcity. But the main attack driving numerous discussions in the Soviet press concerned the meaning of “computer literacy” itself. According to many Soviet and international specialists, the proficient use of applications such as word processing and spreadsheet calculations, not programming, should be the goal of computer literacy and should define computer education in school curricula.

For the proponents of the reform’s philosophy of universal programming skills there was little contradiction between its goals and the instrumental approach: learning to program does not prevent mastering typing or editing. Agreeing that every child would not grow up to become a professional programmer, Ershov (1988) insisted that interacting with the computer should be an empowering human experience: “The discipline of action is equally necessary for a human, like the discipline of mind and the discipline of speech. Exercising in the control over a computer, a human being elaborates an ability to control himself.” Drawing on the cultural baggage of the revolutionary project of making a new man and the early Soviet campaign for eradicating illiteracy, the slogan of “Programming, the second literacy” allowed proponents of the reform to envisage a socialist “information society” where programming fostered personal virtues and social harmony.

Because he died in 1988, Ershov did not witness the fall of the Berlin Wall and the dissolution of the Soviet Union. Even if many computers were eventually delivered to schools and the class on “The Basics of Informatics” remained on the curriculum, the reform’s goal of developing special thinking habits—algorithmic thinking via programming—was discredited. While the limited and short-lived 1985 reform does not lend itself to causal claims about the numbers or quality of Russian IT specialists abroad, beyond the specific cases of several model schools in capitals and Novosibirsk, the intense debates it provoked demonstrate the high level of interest in IT technologies in late Soviet society and help situate the radical transformative visions among its post-Soviet heirs from Kazan to Estonia (Kontareva, this volume; Savchenko, this volume). Furthermore, the philosophy and pedagogical experiences that guided the reform did not disappear altogether; they can still be found at Akademgorodok—with its ongoing tradition of annual
summer camps for young programmers—and in the present-day work of prominent computer education specialists who started their careers within the reform framework of the 1980s.

CONCLUSIONS

From their place in secret military installations to their use as tools of political legitimacy, computers were at the very core of post–World War II Soviet history. Soviet programming did not appear as an afterthought but was part and parcel of Soviet computing efforts. Its specific features were a product of national structures as well as interactions with the West. Following the radical reappraisal of cybernetics in the Soviet Union in 1955, the early public discourse and professional identification of programmers took place under the umbrella of a new metascience. The technological challenges of software portability and international coordination of expertise shaped the definition of a separate professional identity in the 1960s. The rationale behind the much-contested orientation of the late Soviet computer industry on reproducing Western hardware architecture ranged from scientists’ interest in integrating the Western community, to problems of coordination between the military and civil sectors, to the state’s goals of mass computerization. However, the numerical growth of Soviet programmers did not lead to a stronger corporate identity as the distance between the aspirations and realities of academic professionals and typical computer centers mounted.

The Soviet expert in automated programming, Andrey Ershov, became the main spokesperson for the profession at home and abroad. According to his vision, the essence of programming was the realization of human intelligence predicated on an intimate connection: the brotherhood of man and machine. Fusing the experiments of American and Soviet educators in the 1970s and Soviet revolutionary aspirations to create a “new man,” Ershov developed an original philosophy of computer education that guided the Soviet educational reforms of the 1980s and the attempt to create the socialist “information society.” According to the slogan “Programming, the second literacy,” programming should become a universal skill and a guarantor of social and political cohesion. Although the reform’s goal of human engineering was contested and discredited, the end of the Soviet state did not entirely bring to an end the technocratic visions in post-Soviet societies.

While accounting for the radical novelty and the rapid evolution of digital computer technology associated with the notion of computer generations,
the concepts of “Computer Revolution” and more recently “big data” do not answer all our questions; it is equally important to trace the continuity of practices predicated on the much slower change of human generations. The Soviet programming cultures and visions of the socialist “Information Age” reveal overlapping social networks, values, and rhetoric stretching from the early Soviet to the post-Soviet periods. From Lenin’s famous declarations that without literacy there is no politics and that communism equals Soviet power and electrification, to Gorbachev’s perestroika and education computerization, the communist project was predicated on the re-engineering of nature, humans, and machines. However, unlike pilots and cosmonauts—the iconic Soviet heroes embodying the power of flight gained by a man-machine interaction—Soviet programmers were and remain less visible and members of a potentially disposable occupation caught between mathematical designs to automatize programming and the realities of the laborious and error-prone practice of reading and writing code.

REFERENCES


