# How the World got into the Computer

## The Emergence of Digital Reality

David Gugerli

# How the WORLD Got into the Computer

## The Emergence of Digital Reality

David Gugerli

# HOW THE WORLD GOT INTO THE COMPUTER: THE EMERGENCE OF DIGITAL REALITY

Translated by Giselle Weiss

CHRONOS


German Edition:
Wie die Welt in den Computer kam.
Zur Entstehung digitaler Wirklichkeit.
S. Fischer, Frankfurt am Main 2018.
ISBN 978-3-10-397226-9

Cover image: see p. 51
Cover design: Thea Sautter, Zurich

ISBN 978-3-0340-1671-1
E-Book (PDF): DOI 10.33057/chronos.1671

# Contents

| | | |
|---|---|---|
| 03 02 09 | CC | Hey, Jim, you do have your computer ON, don't you? |
| 03 02 13 | C | Negative. I don't have it ON. Do you want it ON at this time? |
| […] | | |
| 03 02 43 | C | Computer light is on. We're ready. |
| 03 02 45 | CC | Say again, Jim. |
| 03 02 47 | C | I say, my computer light is on. We're ready. |

Voice tape transcription between the ground station (CC) and astronaut (C) during the Gemini 4 Mission, NASA 1965, p. 26.

# 1 Switching on

This is the story of how the world came to inhabit the computer. It is the tale of a major relocation that began seven decades ago, around 1950. For a variety of reasons, efforts to shape a computer-based reality have been ongoing since then – amounting to millions of "man-years," in industry parlance.[1] Today, people refer to the near-total computerization of the world as if it were nothing. Yet the effort to "put the world into computers," as technology historian Michael S. Mahoney wrote, was a long, labor-intensive, and sometimes frustrating undertaking.[2]

How did the world get into the computer? The answer is not to be found in the mix of brilliant pioneering work, entrepreneurial risk taking, coherent genealogies, and exponential growth curves typically trotted out in computer histories. In fact, given the hard work, ambitious project planning, often naïve designs, the anxious wait for program updates, decades of anticipating new hires and new software, as well as the huge effort of developing computerized routines, the story of how digital reality emerged simply cannot begin with the haphazard nature of technological progress or even the machines themselves. Instead of lamenting the victims of computers and blaming the machines for causing "reading and attention disorders, anxiety and dullness, sleep disorders and depression, obesity, propensity to violence, and social decline,"[3] we should focus on what computer developers were thinking and the hopes and dreams of users.

Chance and victimization are poor guides for a good computer history. I would like to take a different approach, which is to explore how people perceived computer-related issues at the time and how they dealt with those issues. I will trace the expectations, mindsets,

and motives of individuals who worked on, directed, or supported this massive shift as technicians, managers, users, entrepreneurs, and civil servants. All were betting on the expanded design possibilities, the potential for analysis, and the acceleration of things in digital space and thus were willing to accept the bumps in the road for themselves and others. But not all of them did it in the same way. Thus, I will tell how the computer has been harnessed for different purposes. What prompted opening up the new space of action, and what challenges did it pose? How did the shift occur from the old registries to the unfamiliar databases, from broadcasting to the World Wide Web, from the floor of the stock exchange to computerized stock trading, and from the roulette tables of casinos to the sophisticated profitability of online games?

The question of how the world got into the computer compels you to think. With a little luck and critical perseverance, you might even come up with an answer. In any case, the sources for this history – the hundreds of thousands of lectures, discussion papers, and articles produced on the subject in the first half-century of computer history – are easily accessible.[4] Time and again these materials sparked new ideas and deliberation on promising courses of action. Essays, announcements, and progress reports furnish information on how the new digital space was to be configured and which rules were developed, tested, and ultimately discarded or implemented. What could reasonably be expected had to be communicated through lectures and articles, strategy papers, announcements, and debates. The record of this work is my inspiration. It tells of successful and failed exchanges in the dynamic project culture that has been part and parcel of the computer world. Contemporaries read these sources as a travel guide. Today, too, they help to navigate the digital space of that time.

The history of computing thus observes observations, and in doing so constitutes a synthesized, concentrated account of a

large number of contemporary accounts. Neither circuit board processors nor characters on long-dead screens, neither data sets nor programs, neither users nor operators can be historically comprehended other than through a critical reading of their commentaries preserved in archives or on the Web. Only rarely have I consulted memoirs or interviews with the dramatis personae of computer history.[5] They are generally more interested in explaining their own farsighted decisions than in tracing the course of history. They acknowledge a past with limited horizons (not theirs, of course) and compare it with an ungrateful or ignorant present. In doing so, they forget that the reduction of uncertainty is an inexact science and that the path to wisdom is rarely straight.

This is basically another way of saying what my history of the computer takes into account and what I wish to disregard. I rely heavily on the extensive holdings of the Association for Computing Machinery, because they provide very detailed information about motivations for shifting things into computers.[6]

The process of communication comprised both discreet and overt announcements, long and short explanations, big promises and little promises, both from within and outside the self-proclaimed group of experts. In the first promotional film in digital history around 1951, for instance, computer manufacturer Remington Rand held out the sort of promise that could easily find resonance today.[7] Like any advert, this one broadcast an upbeat message and, at its most superficial level, conveyed unbridled optimism at the progress made by civilization and technology. The recently founded company created a suitably impressive backdrop for the cinematic debut of its all-purpose computer. From the pyramids to urban skyscrapers, from the triumphs of scientific research to the mass production enabled by automated industrial plants, to the services of modern forms of government – the ensemble of spoken words and images served to evoke

the foundations and achievements, and the history and future of humankind. The arrival of UNIVAC overshadowed prior progress, and put it on a new footing: from now on, the entire world theater would benefit from the computing skills of the machine. UNIVAC, the first commercial digital computer ever, had freed itself of the main tasks of previous computing machines, which had consisted of calculating ballistic curves, cryptography, and the development of nuclear weapons of mass destruction.[8]

The Remington Rand promotional film presented the computer as the crowning achievement of civilization's development and, at the same time, as its instrument. The film explained in detail the various components, procedures, and possible uses of the computer, including coding stations, punched card readers, magnetic tapes, monitoring consoles, processors, memory "tanks" and printers – all surrounded by a scattering of humans. Mention was made of the amazingly fast solving of complex systems of equations in nuclear physics, but the focus was on the bureaucratic mass processing of data on the digital assembly line.

Particular emphasis was placed on the meticulous programmers and skilled operators who managed the machine. The computer was an automated, industrial, well-controlled calculating monster in the service of humankind. It came across as a smoothly functioning manufacturing system, fed with raw data that went through a whole series of processing steps and emerged as fully calculated and neatly printed results. These results might be thousands of payroll checks for a large company with deductions for taxes, social security, and union dues as well as individual adjustments for overtime, vacations, and night shifts all figured in. "In less than four hours per week, and with only a small operating staff, UNIVAC can complete the computation for this payroll of 15,000 employees. A saving in time and money that is tremendous."[9]

The capability of the system beggared credulity. It could handle "any task where data have to be processed or problems solved," making it clear that "tomorrow's office production will attain the high levels of speed and efficiency which mark industrial production facilities today."[10] The printer, for example, could print out a full three pages of a metropolitan telephone directory with names, addresses, and telephone numbers in less than one minute. Moreover (and this was the real kicker of the marketing film), "UNIVAC still has nearly 90% of its working week free to perform many other valuable computing assignments."[11]

With that, the film upended the relationship between the world and the computer, and also communicated something else with far-reaching consequences. Suddenly, the key message was no longer the grand unveiling of the new machine and its potential spread throughout the world. Rather, the message was that the computer offered uncharted territory to exploit.

The "all-purpose" computer had space – lots of space, actually. UNIVAC's space was so immense that it could accommodate all kinds of projects. Indeed, "the whole world" (or, at least, everything relevant to the world) would one day come to occupy this technologically generated, recently conquered, but still minimally structured digital space. A poster from that time depicts a Remington Rand computer commissioned by US Steel encircling the world and drawing it into its state-of-the-art computer room.

Inviting the world into the computer room had the effect of populating the computer with data, for example, from the most recent US census. From a computational point of view, the analysis was not a difficult task, but it was an extremely laborious one. The results of the previous census in 1940 were still being tallied when UNIVAC made its appearance. Yet the new machine handled the processing of the even more comprehensive 1950 census with ease and delivered the first results within a few weeks.[12]


Figure 1: The world of US Steel moves to UNIVAC in 1956.

Thus, it made perfect sense to enter the census data into UNIVAC. There even was an IT precedent for doing so. In 1890, Hermann Hollerith had supplied the Census Bureau with his electromechanical punched card machines, making it possible


Figure 2: UNIVAC sorting Democrats, Republicans, and undecideds during the 1956 presidential election.

to calculate the results of the 1890 census in under a year. In contrast, analysis of the 1880 census had required an additional eight years.[13]

The UNIVAC was also entrusted with tallying the 1952 US presidential election. During a CBS news broadcast, the computer projected victory for Second World War hero and Republican candidate General Dwight D. Eisenhower. What made the event so spectacular was not only the remarkably fast computing power of the machine but also that it foresaw the defeat of Democrat Adlai Stevenson, who had been favored to win.[14]

The world (or much of it) was also gravitating toward the computer for making faster weather predictions. Data from guided weather rockets and weather stations – "all of this can be fed into the computer through these magnetic tapes," explained the host of another Remington Rand[15] commercial, gesturing to banks of tapes lined up like trusty servants.[16]

The arrival of the computer was thus accompanied by a grand narrative. This narrative had to be told over and over by everyone involved both to help grasp events as they unfolded and to help endure the arduous work ahead. My account of this effort also relies on storytelling. Not because there are no analytical concepts to explain. I am compelled to tell stories because it is stories told in the past that moved the world (into the computer).

This narrative history is organized by the basic activities that shaped digital space and made it a reality. These activities include computing, programming, and formatting (Chapter 2). They make a good starting point because they figured particularly largely in the 1950s, and continued to remain important afterward. In the early 1960s, people began to formulate rules for sharing scarce resources and thus also rules for operating in digital space, that is, time-sharing and operating systems (Chapter 3). Around the same time, the issue of synchronizing the world with digital space became acute, as the example of the Houston Space Center (Chapter 4) shows. Indeed, the delicate matching of IT supply and demand runs like a thread through the history of the computer. Bringing the two together required endless negotiation. While manufacturers worked on the machines and programs of tomorrow, customers sought to clarify what it was that they actually wanted to do in digital space and how to go about it. Projects of varying scope provided a means of adapting customer expectations to the possibilities of digital space (Chapter 5). By the 1990s, networks of computers, differentiation of users, and routine storage of data had served to structure digital space in such a way as to produce a globally recognized digital order. Ever since then, the world's communications and transactions have found firm anchor – albeit in rapidly changing configurations – in digital reality (Chapter 6).

The form of this story is an essay. What happens when the usual perspective of computer history is reversed? Do new in-

sights emerge? What can simply be left out of the conventional narratives and what should receive greater emphasis? For me, there is a clear advantage to this exercise. By beginning with contemporary problems whose solutions had to be negotiated and whose implementation invariably led to new, unintended consequences, it is possible to recount computer history in such a way that its outcome need not be regarded as inevitable. And that matters if we are to understand why the world – even in the computer – is always subject to reinterpretation.

# 2 Computing, programming, and formatting

When Remington Rand introduced UNIVAC in 1951, nobody really knew what to expect from a computer. People generally assumed that it could calculate quickly. But experience with the inner workings of mechanical calculating machines had shown that even slow machine computing using only the four basic arithmetical operations could be a complicated affair. When well maintained, the confusing jumble of wheels, cams, rods, levers, and springs hidden away under the cover of a Brunsviga or Adler calculator could handle tens carries with ease. But the engineering could not easily be reconstructed with electronic circuits.[17] So how was such a machine supposed to go faster? No doubt there were specialists in mechanical calculators and in computational tasks who had some inkling of what a computer might one day be able to do. Surely a few had heard of the Turing machine or read John von Neumann.[18] But even they most likely had never seen a computer, let alone operated one. What they could do was to eagerly read, with wonder or skepticism, accounts by colleagues who knew something of the new world of computers. These accounts, for all their genuine optimism, made one thing abundantly clear: even the people who built computers could not agree on the essential features of the machines.[19]

Under such circumstances, what could be expected of an electronic calculating machine that supposedly not only computed automatically and accurately but that also served any purpose? Even practical engineers and managers fell under the spell of the UNIVAC spot.[20] Moreover, as the film moved from frame to frame, and from one explanation to the next, the viewer's amazement

increased. UNIVAC was said to process text and numerical data at "incredible speeds." The masses of data this "miracle of [modern] electronics" could process were simply unimaginable. What would have taken years to do in the past could now be done in minutes.

It is the business of advertising to exaggerate. Its purpose, after all, is to cater to dreams, awaken desires, stoke longing, and bring unanticipated possibilities within reach if only for a brief moment. Advertising achieves these feats by surprise, especially when an inspirational commercial spotlights an aspirational computer. In the case of UNIVAC, the surprise was that there was hardly any talk of computing. The film obviously made clear what it means to be able to process payroll checks for ten thousand employees. But it was precisely this tedious, boring process, repeated at monthly or biweekly intervals, that UNIVAC could apparently do in no time at all. So quickly, in fact, that it could not even be demonstrated. Indeed, the process of computation was never shown but only implied in mention of the result – "adjustments and rates calculated" (note the past participle). A thrilling vision of a printer spitting out paychecks flashed by on the screen. Computation itself could not be visualized. It lurked unseen in the black box of the machine. The presenter merely spoke tautologically of "computing systems" that performed and delivered "computations."

Although calculations were described as a result, not a process, and as a miraculous transformation, not a detailed process, they did occupy a specific place. Indeed, the camera allowed viewers a brief glimpse inside UNIVAC's central cabinet, stuffed with electron tubes, and then of its monster memory.

Even the points in the data processing sequence where calculation was involved were described rather than shown. The claim made at the beginning of the commercial was that the machine

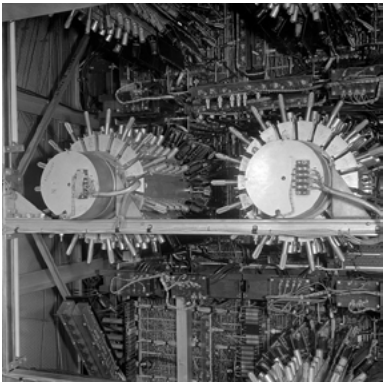

Figure 3: UNIVAC's run-time memory around 1955.

could be used for any task that required "sorting, classifying, *computing*, and decision-making."[21] Thus, in procedural terms, computing ranked third. The computer was initially an apparatus for sorting and classifying data, that is, an *ordinateur* – and not only in France (more about that later).

Moreover, although pretty much relegated to sorting, computing became nearly ubiquitous in the processing sequence. This was because each of the four sub-procedures enumerated in the UNIVAC spot (sorting, classifying, computing, and decision-making) comprised aspects of each other. *Sorting*, for instance, involved making computable, criteria-based decisions. *Classifying* similarly entailed decisions based on size, characteristics, output, and type. *Computing* involved overwriting data in the well-ordered space of numbers in such a way as to enable a correct result to be determined. Ultimately, however, *decision-making* could only proceed once the sorted and classified data had been computed.

UNIVAC's programming capacity depended on two operational functions. First, the sequence for sorting, classifying, computing,

and decision-making had to be established by a verifiable set of instructions. In other words, the sequence had to be *programmed*. Second, everything that was to be sorted, classified, computed, and decided on had to be prepared in a suitable *format* beforehand. Only then could it be read by a machine, and only then could the machine print the final result of the combination of instructions of data or pass it to the next procedure. Formatting was the most important prerequisite for successful program-based computing.

The bracketing of the processing sequence by programming and formatting could be interpreted as the growing importance of these activities, the mechanical work of sorting having given rise to the possibility of conscious decision-making. But, even more radically, the sequence of processes also represented the scope of everything a computer could do.

## Computing

If the most powerful of computers could make the process of computing disappear, you might ask where it was before it disappeared. How was computing done in the 1950, before UNIVAC? Where was computing commonplace, and how was its arsenal equipped? The scope of computing in commerce, insurance, machine shops, artillery, and surveying was extremely varied, as a few examples show very well.

In the retail trade, salespeople, accountants, managers, and warehouse supervisors computed every day. But they had very different expectations with regard to accuracy. Computing to estimate – a definite no-go in accounting – was the rule in sales. Depending on the situation, percentages, margins, production runs, and volume discounts were sometimes computed with astonishing precision, sometimes jotted down on the back of an

envelope, sometimes recorded with notes for double-checking, and sometimes thoroughly detailed on paper. In any case, situation-specific, usually approximate calculations were a characteristic feature of transactions. There was no other way to do business. Precise, verifiable computation occurred later when invoicing and in the books. Consequently, in retail, computational rigor could be postponed to some extent until after delivery and delegated to specialists in the accounts office. Banks issued exact calculations of interest at the end of the year, using bulky interest rate ledgers. Stock traders would head for the local bank at the end of their daily trading session to compute their position. Thus, calculation problems were temporized wherever possible.

Even in the computation-intensive insurance business, the focus was on delaying computation. The risky calculation of distributed risk was spread over a company's entire organization. In the process, there was always something that had to be calculated in advance or recalculated at various points. Insurance agents had to perform calculations under time pressure during customer visits; in the branch offices, it was more a matter of compiling, sorting, and customizing offers. The head office of an insurance company, where the accounting, billing, and claims departments were located, made their own calculations and at different speeds than the actuaries and statisticians, investment advisors, or claims managers. Insurance agents, for example, were able to read off the key values of their quotations from a precalculated rate manual and thus to adjust even complicated, particularly high-risk policies to a specific insurance situation on the spot with little computational effort.[22]

Machine shops, in contrast, relied only in part on temporization and distribution, and avoided calculation whenever possible. Here, probably contrary to the assertions of business economists, exact computation did not necessarily go hand in hand with more

efficient production or a better product. The hasty application of accounting catechism was countered by a skeptical attitude toward overprecision based on experience. Certainly the cost of lacquering a piece of furniture could be fairly precisely calculated in the carpenter's shop. But with a little practice, it could also easily be estimated and then determined after the work was done by counting the paint cans used. The arithmetic taught in vocational schools did have its niceties. Nonetheless, some of the textbook assignments may have been more about building confidence for the final apprenticeship exam than about encouraging computation. Such assignments included, for example, determining the cost of one meter of putty rabbet in the manufacture of windows. In a chapter on the "purchase and consumption of auxiliary materials," this assignment was among the most difficult. After solving it, students would likely have sworn off calculating forever in favor of the industry-standard rule of thumb. That would require simply setting the cost per meter of putty rabbet (like the cost of paste, pencils, and glass paper per square meter of wood) a little higher than the instructor had done in his time.[23]

Determining the adjustment angle of a lathe in a metalworking shop was a different story. Estimation and experience alone were insufficient to produce a cone with the desired pitch. Consequently, mechanics were expected to calculate more and even better than carpenters. A mechanic's training in vocational school included detailed explanations of how to calculate the tool setting using a combination of geometric sketch, taper formula, and tangent of the lead angle. Nonetheless, obviously aware of his student's moderate zeal for arithmetic and for safety's sake, Alfred Stahel appended to his 1950 textbook *Rechnen für Mechaniker* (Computation for mechanics) a table showing the cone angle (already halved) in degrees and the percentage pitch. As if doubting the effectiveness of his arithmetic lessons in general, he also

supplied a table of "squares, square roots, and circumferences and areas of circles."[24]

Time and relative speed played a very important role in any calculation, whether on the shop floor or the stock exchange or in times of war. But the time factor did not have the same urgency in every situation that called for calculation, nor was it always handled the same way. For example, in setting up artillery, precious time was saved by use of firing and correction tables prepared by weapons control, which transmitted to the gunners over a loudspeaker the target coordinates based on corrected tables for the fire plan.[25] Even faster, though mechanically very demanding, was the use of command devices in air defense.[26] Here, complex and costly calculations were replaced by mechanically and telescopically supported precision tracking of an aircraft's trajectory. The values to be set on the guns before firing could commence had to be mechanically translated by entering observable conditions onto their dials. The "calculation" was mechanically built into the command device. This saved computing time and enabled quick decisions to be made at the guns for the trajectory of the shells.[27]

Calculations could be done mechanically, they could be avoided by using rules of thumb, they could be kept in tabular or graphical form, or they could simply be postponed until later. There was considerable potential for optimization if, under the right circumstances, precision calculations were adjusted and implemented with the benefit of intuition, experience, and trust in the authority of tables. When mental arithmetic and paper calculations no longer sufficed, the arithmetic space could be mechanically expanded by automatic calculating machines, recording machines, and other analog computers. In most cases, however, a person could make do with a slide rule equipped with appropriate scales.[28] Among the arsenal of mechanical, tabular, and graphical tools that aided arithmetic around 1950, the slide rule was hard

to beat as a fast, portable, yet remarkably precise device for demanding calculations. Even in the mid-1960s, in the technically sophisticated setting of NASA's Mission Control Center where the Gemini and Apollo space flights were monitored, technicians seated at the consoles were still using slide rules to quickly check how much fuel was left in the rocket or the position of the spacecraft.[29] Day in and day out, merchants, insurers, craftsmen, and artillerymen calculated away, comparing graphical aids against measuring instruments, formulas against table values, and estimates against their own computations. There were, around 1950, an infinite number of situations in which it was possible to do lots of complex calculations accurately, quickly, and safely, or at least efficiently, and with little difficulty. A large battery of tools was available for carrying out extensive or time-critical calculations and for doing calculations in advance.

In other words, there was no obvious demand for a "miracle of electronics" that could compute at top speed. The demand first had to be created, defined, and recognized. The fact that it actually materialized over the course of the following decades was more projection than certainty around 1950. A machine that could be used to perform routine computing operations at great profit was indeed a miraculous machine.

Some of the mind-boggling mathematics that scientists had come up with to that point was now done by electromechanical computers. Such systems had been built, for example, in the 1920s and 1930s at the Massachusetts Institute of Technology (MIT) under the direction of Vannevar Bush and between 1939 and 1944 at Harvard University under the direction of Howard Aiken.[30] These systems had been used, among other things, to calculate the critical mass of an atomic bomb in a short amount of time, that is, before the end of the Second World War. The Electronic Numerical Integrator And Computer (ENIAC), built by John Mauchly and

Figure 4: Calculation could not be avoided completely – even for a flight navigator inside a Swissair plane (1959).

Presper Eckert on behalf of the US Army at the Moore School of Electrical Engineering at the University of Pennsylvania, was also oriented to the problems of scientific and military computing.[31] These machines were designed to numerically solve differential equations, for instance, that is, to mechanize the extensive calculation work of such tasks. Or, like the ENIAC, they were used to solve problems of optimization or stochastics in addition to their tasks in the military and nuclear realm. Electromechanical machines, which were used for administrative purposes and could reliably master even large sets of positive numbers with three to four basic arithmetic operations, were a different order of machine.[32] The possibility space envisioned by scientific computing dealt with atomic bombs, not setting angles, premium invoicing, accounts receivable, or the cost of putty rabbet.

Interest in computation among academics took many different directions. For example, in addition to his theoretical work on the computability of the decision problem and his secret work on computational decryption, Alan Turing attempted to computationally determine the processes of evolution and biochemical structure formation.[33] John von Neumann, writing in 1945, fancied rather an automatic computing system that could carry out computations of high complexity. The arithmetical problems he had in mind went far beyond what Bush and his colleagues at MIT could solve mechanically – off-the-wall things such "solv[ing] a non-linear partial differential equation in 2 or 3 independent variables numerically."[34] Konrad Zuse, for his part, had believed in 1942 that his computer should be used to calculate matrices. He then entertained the chilling prospect of applying this abstract notion from the field of linear algebra to research on race and genetics, popular topics in his milieu. With adequate support from the National Socialist state, Zuse figured that his machine could be used to determine the degree of kinship between any two individuals – a computational tool fit for the Nuremberg Laws. Not surprisingly, by 1948 Zuse was envisioning a much broader range of possible applications for his computer. These ranged from mathematical systems and checking of Boolean logic to designing high-performance numerical calculation devices to atomic physics and computational chemistry, and from bookkeeping, job costing, and other types of commercial computing to applications for design problems, calculator-aided design tools, civil engineering, circuitry, chess, and even problems in linguistics.[35]

These computer applications were future oriented, but they did have one thing in common with UNIVAC's own past in the history of the technology, independent of political context. Zuse's applications were intended to enable a selective shift of scientific

problems to the computer and, in return, to mechanize computing in such a way as to avoid its tribulations, if not computing itself. Automating and speeding up arithmetical operations would thus go a long way toward achieving this goal.

Separating computing staff from computing was one thing. Separating the debate over computers from computing and its electronic implementation was something else again. The latter, in particular, seems to have prompted Presper Eckert, James Weiner, Frazer Welsch, and Herbert Mitchell, all engineers at the Eckert-Mauchly Computer Division of Remington Rand, to give a detailed talk on UNIVAC in New York in December 1951. The speakers did not ignore the subject, as the commercial did, but rather went right to the point, showing how the machine could be used to compute. Perhaps they hoped that they could then go on to talk about genuinely interesting matters. The occasion was the joint conference of the American Institute of Electrical Engineers and the Institute of Radio Engineers. The Remington Rand delegation served as ambassadors for a complex development project that had involved hundreds of engineers.

Eckert and his colleagues had chosen an unusual approach to their four-part presentation.[36] For starters, they said, their purpose was not to talk about performance or objectives, but rather how the UNIVAC system was organized. Functions independent of the main computer were outsourced to auxiliary components with their own power supply. That included, for example, keyboard to magnetic tape and punched card to magnetic tape operations. Data entering or leaving the computer were synchronized via special input–output circuits. The computer was equipped with a high-speed bus amplifier through which all data passed at a fixed rate, like packages on an assembly line, "during transfer between any arithmetic register and the main memory or between the main memory and the input–output registers."

Following this brief but critical structural point, the presenters turned to a complex block diagram of UNIVAC's interior. Doing so required them to rely on the momentum they had gained to explain digital computing one last time, as such explanations were to disappear almost entirely from the computer literature in the years that followed. Addition and subtraction processes came first (and simultaneously), followed by multiplication and, finally, division. All three sections were concerned with comparing quantities in one register with those in another register, making changes based on the comparison and desired operation, and moving the quantities to another register. The audience likely understood little of this during the lecture, and reading the description of these basics also requires great concentration. But one thing was clear: computing in the digital computer could also be described as a mechanical process. What was new about it was that quantitative values were not changed by precise turns of a sprocket wheel, as in mechanical calculating machines, but rather by comparing and writing them. Computing, as claimed in the UNIVAC ad spot, actually went hand in hand with sorting, classifying, and decision-making.

As it turned out, the explanation of basic arithmetic operations under digital conditions was all the presenters had to say on the subject of arithmetic. No mention was made even of how to deal with logarithms or trigonometric functions. Instead, the speakers held forth on counting and checking. In the computer, cycles and program steps were counted, and each letter or number was checked to see whether its six-digit coding had an even or odd number of ones, and consequently a one or a zero in the seventh position (and control point) of the code. Even or odd – that was the question confronting every character in digital space. Once the fine details of the computing process and data control in the innermost part of the machine had been discussed, the lecture re-

shut the door to the inside of the computer and explained from the more comfortable outside perspective how this black box was to be supplied with power and cool air.

The second part of the talk was dedicated to the applications of UNIVAC. UNIVAC lived up to its name, the speakers said, because it really was a "Universal Automatic Computer."[37] Although they were referring to the still very vague future in the "ever-widening field" of electronic digital computers, Eckert and his colleagues did not want to stop at sweeping assertions. After all, they were also talking about the very real future of Remington Rand. As with computing, their rhetorical strategy was to explain the applications of their all-purpose automatic computer in concrete terms. To this end, the applications were sorted and classified, and a distinction made between scientific, statistical, commercial, and logistical uses.

The order of the applications was not random. It produced a sequence that led from scientific insight to entrepreneurial action. In doing so, it assumed (as sorting, classifying, computing, and decision-making) both the epistemic and practical relatedness of the fields of application. First up were scientific problems, which had more to do with logistics and business than many scientists were comfortable with. Howard Aiken, for example, had no objection to using conventional elements of mechanical accounting machines when working with IBM. But at the same time he had always insisted there was a fundamental difference between accounting and scientific computing. His Automatic Sequence Controlled Calculator, with its wide range of mathematical functions, was designed exclusively for scientific computing.[38] The presentation of UNIVAC, however, took a totally different tack. The computer was primarily intended for the commercial market. That meant biting the bullet and showing that UNIVAC could also hold its own scientifically if need be. The presenters enthusiastically

explained that their machine could compute large, general-purpose matrix algebra routines in a reasonable amount of time. In less than half an hour, they said, they had found six solutions to a system of 385 simultaneous equations based on a "second-degree nonlinear differential equation" for the flow of gas in a turbine.[39] UNIVAC's claim to the field of science and engineering had been neatly secured with just three examples.

No less was expected of a computer that wished to play in the top league. And it permitted a segue into the much more bureaucratic field of population statistics, where electromechanical Hollerith machines had previously held sole sway. Thirty tables listing age, sex, race, ancestry, education, occupation, employment, and income had been compiled for each county, city, and borough in the 1950 US Census. The focus of this application was not computation, but sorting, classifying, and aggregating tens of thousands of punched cards. Once the information on the cards had been transferred to magnetic tape, there was no need for manual labor except to change the rolls and collect the printed results. UNIVAC was obviously capable of processing data that were already undergoing rationalization at large companies with punched card machines.

UNIVAC also provided automation for the third area of application, which was commercial problems. Here, the issue was dealing with the roughly 250,000 changes made each month to 1.5 million premium, dividend, and commission statements for a life insurance company. Although the computer required 135 hours to accomplish this task, each policy took only half a second to calculate and nine seconds to print.[40]

The fourth, and final, field of application was logistical problems. This application ensured UNIVAC's all-purpose future. The aim of these kinds of computations was to quantitatively assess whether a desired production process or mobilization plan could

be supported logistically and how it could be optimized. As an example, Eckert and his colleagues described computations performed to determine the critical raw material requirements for constructing a given number of machines of different types, broken down by quarters over a production period of two years. A further problem, the complete stock control for a large supply office, had yet to be run.[41]

With the expanding range of fields introduced by the speakers, UNIVAC was coming to resemble less an instrument for science and the military than a device for helping to sort, classify, compute, and decide questions concerning population development, business administration, and civilian forms of operations research.

These and similar machines would increasingly be occupied with problems of sorting, classifying, computing, and decision-making in digital space. Programming made it possible to migrate to computers the typical calculations done in science, statistics, economics, and logistics – speeding them up and hiding them, if not avoiding them altogether. At the same time, and this was the big promise, the first realistic possibilities for practical use opened up a world of quasi-industrial production, processing, and organization of characters.

With UNIVAC, a field of work had emerged around 1950 whose calculation principles had already ceased to be the focus of attention. Of course, UNIVAC was ultimately also a "computer." In terms of IT and programming, however, the "C" at the end of UNIVAC's name signified only the control bit of a promising field of activity for which techniques of organization, programming, formatting, and rules had yet to be developed.

This fact had also caught the attention of French classical philologist Jacques Perret. In the spring of 1955, IBM France asked Perret to think about a name for a new, completely different com-

puter. IBM's aim was not to produce a machine that could simply decode secrete messages by brute arithmetic force, calculate ballistic curves, or compute the critical mass of a plutonium bomb. Rather, the company was developing a machine to do much more general information-processing tasks. Perret answered immediately: "Que diriez-vous d'ordinateur?" (What about "organizer"?).[42] The word was already in Littré's French dictionary, though only in adjectival form.[43] Still, it was grand enough to apply to God (as organizer of the world). Language-wise, *ordinateur* could easily be converted to a verb – *ordiner*, "to organize" – or the act, *ordination*. As a Latin specialist, Perret had a penchant for the feminine form of the noun – *ordinatrice* – which would spare IBM any religious or ritualistic connotations.

The company had a different idea and did in fact choose *ordinateur* for its first commercial computer in France, giving it a decidedly masculine and slightly sanctified air. In English, it was called the IBM 650 Magnetic Drum Data Processing Machine. Both languages preserved UNIVAC's procedural aspirations. Moreover, like UNIVAC, the IBM machine was obviously not just a computer. Rather, it was also intended to point the way and, as the Remington Rand group said of UNIVAC, be "true to its name."

The IBM machine did not become famous because of its name, which ostensibly had nothing to do with computing. The machine became famous for its ability to detect and automatically correct errors in the program, and thus keep the process of sorting, classifying, computing, and decision-making going.

## Programming

Now that calculating took place in the innermost part of a black box, without the fascination of shafts, gears, cams, or counters, it increasingly resembled a machine-aided sorting task. It became unspeakably boring. The overwhelming significance, the unprecedented speed, and the sheer breadth of the *ordinateur*'s fields of application did absolutely nothing to increase interest in computing. Rather, they served to postpone to some future time the critical functions that the machines and their operators would be performing.[44] Because electromechanical relays had been replaced with noiseless vacuum tubes, it stood to reason that, in this brave new order, personnel could soon simply issue commands and make decisions while the machine labored silently away. But in the early 1950s, the difficulty was getting the computer to work (reliably). Every job for the machine had to be parsed into appropriate command sequences, which in turn had to be meticulously written so the computer could interpret them properly. Demands on personnel increased.

Eduard Stiefel, a professor of applied mathematics at ETH in Zurich, took note. Stiefel and his group ran the "Z4" computer that Konrad Zuse had transported from Berlin to the Aerodynamics Research Institute in Göttingen in April 1945 and shipped from there to a flour warehouse in southern Germany. Zuse wished to save himself and his computer from Allied bombs and from requisitioning by the Red Army. In 1948, he adapted his computer to suit the new conditions in bombed-out, occupied Germany. At the same time, he was seeking a base with a stable power supply, a lively interest in applied computing, and an intact industrial clientele. In 1949, he found what he was looking for at ETH's Institute for Applied Mathematics.[45]

In 1954, Stiefel reported on his experiences with the Z4 and his own device, the Electronic Calculating Machine at ETH (ERMETH).[46] The general tone of the report is remarkable. The first sentence alone dismissively refers to "digital calculating machines," which "like desk calculators are devices capable only of performing the four basic arithmetic operations on digital representations of numbers." The only difference from conventional desktop calculators was that the "sequence of the individual arithmetic operations [was] automatically managed by a control unit that read the arithmetic program." This control unit contained the individual commands to be executed by the machine, "which have been prepared by a mathematician and are recorded, for example, on a punched tape." Everything else, of course, was known and, Stiefel implied, not worth mentioning.[47]

Stiefel obviously wished to emphasize the "totally replicative character" of his systems. Thus he also stressed "preparation of the calculation program by the mathematician." This required "mostly much more time and mental effort than the one-time execution of the calculation by hand would take, and often involves an additional burden for the person whom we will henceforth call the *programmer* owing to the crude organization of tasks for the computer."[48]

If the work of programmers was to be worthwhile, they should limit themselves to preparing tasks for the machine that would need be to be calculated again and again in the future. Examples included routine tasks such as solving linear equations or calculating flight paths. Stiefel found it useful to create an entire library of standard programs that could make use of the high electronic computing speed of the machine with little preparation. But if the computers had to deal with a different problem for each job, or if some calculations had to be performed only once, or only on a trial basis, then programming would consume "too much of

Figure 5: Electronic "calculators" were labor-intensive machines. The ERMETH workforce around 1953.

a talented employee's time and stamina." Programmers would be constantly "badgered" by the machine.[49] "The calculating automata," Stiefel lamented, "have taken numerical computing away from us, but have brought us the even more tedious work of programming in its place."[50]

Computing and programming proved to be highly interdependent challenges, not only due to their intrinsic tedium but also because of the discipline and organization they required. No sooner had the last wires been soldered and the last electron tubes inserted into their sockets than the huge task of organizing the computer began. For that reason, noted John W. Carr of the MIT in 1952, programming would soon prove to be a decisive bottleneck in digital space.[51] The US Air Force had made a similar observation that same year: "The inert machinery solves no problems and satisfies no needs."[52] The scene around the computer

urgently needed to be enlivened with personnel who could give the machine a hand. And that, in turn, meant that mastering the machine was also a matter of training, selecting, and managing programmers – directing them, as it were.

The processes in the machines had to be structured in such a way as to allow commands to be translated into machine code and to enable access to stable program elements or subroutines. Computers had to learn to report faulty code to the programmer. Processes involving personnel, that is, the division of labor among mathematicians, customers, programmers, operators, and electrical engineers, had to be specified in such a way as to guarantee a high degree of reliability in processing any and all orders.

In this respect Stiefel was very inventive. Were it up to him, the programmer working the front end would have as little to do with applied mathematics as possible. It was important not to unnecessarily jeopardize the already precarious academic reputation of the discipline. So Stiefel tried to keep programming away from ETH's mathematicians and pass it on expeditiously to his customers in industry. Customers who wished to reserve computing capacity to calculate, for example, the stresses in a dam, the vibrations experienced by a four-axle locomotive, or the critical speed of a turbine, were required to take a programming course with one of Stiefel's assistants.[53] Employees of the institute only checked customer programs. Engineers were entrusted with the electromechanical work on the machine, and the institute's own programming tasks were delegated to specialized support personnel.

The history of digital space is replete with cases where developments in a particular area immediately led to major disruptions and bottlenecks in other areas. Due to the increased speed of computing, a bottleneck formed in programming, which – as Stiefel's experience with the Z4 and ERMETH showed – could be

eliminated either by supportive organizational measures or by creating a specialist role in the form of the programmer. However, it proved difficult to train a large number of programmers adept at controlling the machine to meet demands without constantly introducing new errors in the logic of complex programs.

What to do? You might treat programmers as a distinct development problem – putting them in a well-shielded office (as in the UNIVAC film), and seeing to it that armed with their stencil, a pack of cigarettes, and a limited number of commands, they were neither badgered by the computer nor seduced into feelings of superiority by the task at hand.[54] Alternatively, you could try to get the computer to develop parts of programs. Tedious programming in particular could be performed by the computer itself. This had long been advocated by Heinz Rutishauser, one of Stiefel's associates: "Thanks to its versatility as a plan assembly device," the program-controlled calculating machine could be used for calculating the "totality of commands comprising individual steps of the calculation."[55] That same year, John W. Carr, who was working on the development of the Whirlwind computer at MIT, reported on his work on automatic programming procedures.[56]

The "automated performance of calculating tasks" articulated by Zuse before the war had progressed to computer-assisted production of programs that told the computer how to compute. Questions of mutual learning and clearly defined responsibilities for both machines and personnel were addressed.[57] In the process, programmers and computers taught each other what they should do, could do, and possibly would do if they managed to interact successfully. Programmers sometimes expertly intervened directly at the machine level when something went wrong, and computers learned to reject unreasonable jobs. "You're trying to divide by zero," read one such bit of feedback. The computer's astuteness even made it onto an advertising poster for UNIVAC: the


Figure 6: Reciprocal learning between humans and machines.

fine print claimed that the computer could locate human errors, execute commands in ordinary business English, and produce its own programs.

Nonetheless, the extent to which standard procedures could relieve human programmers was clearly limited. As Stiefel put it, each program stored "in the automaton" ate up memory. In addition, the automaton mainly carried out "logical rather than arithmetic operations…It interprets, modifies, and iterates instructions, switches programs on and off, and tries to find a way through the nested loops of computational structures. But it rarely computes." Here, Stiefel was being deliberately provocative. If technical equipment were replaced with a hierarchy of programs, the result would probably be a "device consisting of an amorphous pile of vacuum tubes or other electronic components, in which even the addition of single-digit numbers would have to be programmed" – what any child could already do in their head. All you'd get would be a not very efficient calculating monster, and an idle one at that. Stiefel may have been among the first to take this line of reasoning. He was by no means the last.

Despite the considerable skepticism, the instruction bottlenecks caused by the increased speed of computation and inadequately trained personnel were at least partly mitigated by technology. At a conference in Toronto in 1952, Richard Ridgway of Eckert and Mauchly reported on the state of the art. Most of the time was lost in programming, when already written parts of programs had to be searched for, adapted, and rewritten. The process was totally inefficient, he said, and it was an inexhaustible source of new programming errors. For that reason, Eckert and Mauchly's Computational Analysis Laboratory had developed the "compiler" method. The technique was currently being tested on mathematical problems, but it would soon be applied to commercial ones. A compiler independently searches for subroutines, adapts them, and assembles them into a complete program. Ridgway showed that the programming time required to do that could be dramatically reduced with just a fraction of additional computing time.[58]

Thus, it was not only computational and sorting tasks that shifted to the computer in the early 1950s. Production of the parts of programs that did those jobs also shifted to the computer. Compiler development showed that programming was a crucial feature of computer organization. But the shift also highlighted the uncomfortable position of programmers, caught as they were between the machine and mathematics, between engineering and creativity, and between code and text.

Reconceptualizing mathematics in more formalistic or even mechanistic terms might well have proceeded along the line of modern mathematics. Or coding might have been afforded more design freedom.[59] Both were conceivable, but in the mid-20th century, neither appears to have been on the radar. Despite compilers, programmers still had to be massively trained and instructed in a new discipline and were expected to keep their initiative in check. Their superiors required them to learn specialized languages, write reliable code, be extremely patient, and – sympathetically and with the aid of the compiler – force the wayward machines to read and execute each program correctly. The training and taming of programmers was the most critical factor in getting the computers to obey. And vice versa.

Once the soldering was done, the work of organizing the computer to make it functional was exciting. Programmers were forever having to break new ground in issuing sequences of instructions to computers and getting them to comply. And despite its formal rigidity, programming remained an ever-changing task. New languages, new dialects, and other things were always popping up.[60] Programmers were considered exotic, elusive figures, operational and functional necessities, a deficient form of human capital, a source of project delays, and reluctant participants in retraining courses. Yet all the while, the machine-side prerequisites for programming continued to evolve. Consequently, beginning

in the 1950s, it was impossible to say for sure what a computer was, what a programmer did, or how the machine and human related to each other.

Understandably, commentaries on programming and computers are plentiful. But it is hard to spot any consistency in their relationship to one another, or even a trend. Any commentary on programming necessarily implies computers, whether explicitly stated or not. In any event, by the late 1950s, Fortran, Algol, and Cobol were widely used, sufficiently monotony-proof programming languages that moderately creative programmers could handle. The principle of the compiler had also become commonplace and a prerequisite for developing programming languages. Most important, by the end of the decade no one was surprised that moving sorting and calculating tasks to the digital computer always involved a fair amount of programming. The programming bottleneck that had caused headaches at MIT in 1952 was now plainly obvious.

As computers manufactured by IBM and the "seven dwarfs" – Burroughs, UNIVAC, NCR, Control Data, Honeywell, General Electric, and Radio Corporation of America (RCA) – became available to handle an increasing number of applications in the early 1960s, demand for all forms of programming shot up. Not just the military but also government, airlines, banks, insurance companies, and retailers advertised for programmers to meet the needs of their customers by automating and rationalizing their operations, spurred by the economic boom. When necessary, these programmers were trained to use the Beginner's All-purpose Symbolic Instruction Code (BASIC), a starter language developed in 1964 at Dartmouth College in New Hampshire. BASIC eased their way into more powerful programming languages.[61]

Some programmers who had been trained by manufacturers stayed with the customers. But programming was critical to the

development of computers, and recruitment of programmers was therefore an ongoing issue, especially in Europe.[62] Research teams tried to work out precise needs, appropriate qualification systems, recruitment incentives, forms of training, and evaluation criteria for programmers. They strove to distinguish between systems analysts, junior systems analysts, programmers, and coders – only to have their carefully constructed categories collapse.[63] Not even aptitude or talent tests helped. The attempt to create an ideal classification of programming personnel failed from the outset because of widely divergent ideas about what a programmer should be able to do, other than write lean, error-free code.

It wasn't even clear whether a "strong thinker with a capacity for abstraction and logic" who was a "fairly good worker" was preferred or an "intuitive, imaginative type." The "ideal programmer" would combine both profiles. He was "obviously a rare phenomenon" as an elite Dutch study group noted in attempting to refine the profile of an ideal programmer.[64] More important than specific requirements for programmers, however, were specific programming languages and documenting them in manuals, teaching programming courses, and reckoning with an extremely diverse pool of prospective recruits. Participants in programming courses were typically teachers and accountants, PhDs in physics or mathematics, and electrical engineers who liked computers. The fact that career guidance literature also advertised programming as a profession did little to improve the dry labor market. Even an instructor's recommendation that programming courses employ "teaching machine technology" was inadequate to solve the problem of the enormous demand for programmers.[65] To produce programmers who could program computers, you needed computers that could program programmers. The interdependence of computers and personnel was enormous.

## Formatting

In theory, once computing itself had disappeared into the black box of the computer, and programming had been delegated to well-trained specialists and to the machine, the tasks to be executed in the computer could actually begin. The promise that "digital computer techniques" would simplify, accelerate, and lower the cost of big information processing jobs was welcomed with enthusiasm by many. Instead of having to slog through a large number of arithmetic operations, people now anticipated being able to process large amounts of data.

George Brown and Louis Ridenour of the International Telemeter Corporation in Los Angeles were among those who remained skeptical. In 1953 they judged programmable punched card machines to be so advanced that large amounts of data could hardly be processed faster by digital computers. For example, IBM's card-programmed electronic calculator had already solved practically every desired calculation problem. A digital computer wasn't really needed, because the problem was not with calculating, but with the format of the data. There were simply no input or output devices powerful enough to "couple" the "world of the digital computer" to the "world of men." A solution would have to be devised.[66]

In 1951, in introducing its card-programmed electronic calculator, IBM had chosen an illustrative example that fell in the gray area between scientific computing and technology development and that involved huge amounts of data. The issue was missile control during a 100-mile test flight. Along the route, the customer had set up whole arrays of cameras and phototheodolites that shot a hundred images of the passing rocket per second. Previously, these images were handed over to a group of female "computers" who labored away for two weeks at analyzing thousands

of images. The new, card-programmed electronic computer could do the task in eight hours.[67]

The successful exercise did not lead the unnamed customer to conclude that card-programmed machines would essentially solve the problem. In 1954, Jerome J. Dover of Edwards Air Force Base in California reported with a note of despair that data collected by people in uniform were not automatically uniform. Test flights could really only collect "raw uncorrected data," he said, and that applied to all Air Force experiments on the high-speed track and the rocket engine test station.[68] Some classified projects also involved large heterogeneous batches of data. Like Brown and Ridenour, Dover noted the need for fast and reliable input techniques, and maintained that faster computing machines were unnecessary. It seemed much more important to him to get the records of the measuring instruments into a form in which they could be processed, for example, by an IBM punched card computer or a digital calculator.[69] The IBM punched card computers would have reduced the computing time considerably. At the same time, reading and processing the data had become a real bottleneck. For Dover, the key thing was to develop a centralized, automatic data processing system and in the future to capture data in such a form that the processing machine could read it without the help of a human computer. For analysis, it would be possible to avoid having to hire and train so many recruits to read automatically produced films, photographs and curves depicting flight parameters. Moreover, a smaller workforce meant not only lower personnel costs but also less "contamination" of test results. Reading errors occur constantly when transferring raw data into a standardized data format. Only an automated data preparation system for capturing raw data in a standardized, machine-readable format would enable the Air Force Flight Test Center to survive the oncoming flood of data.[70]

Figure 7: Preparing data for analysis at the Erie Railroad Company (1958).

Data processing thus necessitated data processing. The central computer destined to prevent the end of the world by replacing the raw data from the measuring instruments and the dubious performance of the human analysts was still a work in progress. Therefore, the Air Force had asked the Ralph M. Parsons Company in Pasadena to solve the raw data problem. According to many reports, the company came back with an idea for an "all inclusive and extremely comprehensive black box." The black box would take both analog and frequency-modulated, digitally encoded signals as input, and output them as digital type, digital punched cards, analog signals, or magnetic tape. The output would then in turn be available as input for analog or digital computers.[71] The gates to the world of computers had to be equipped with a large translation machine. This

machine would automatically convert analog records into discrete values so that they could be encoded and processed by computers. Whether the processing was analog or digital was secondary to the solution of the formatting problem.

There is little point in trying to decide whether the Ralph M. Parsons Company actually solved the data formatting problem or not. What is of interest here is the strategy employed in solving the problem. Obviously, the idea was to handle data similarily to (digital) computing and (digital) programming. Which basically meant creating a central processing unit that behaved like a black box and ensuring that the result was sufficiently versatile. Data formatting was thus the sine qua non for adapting highly diverse fields of activity to the capabilities of the computer. Experience with program-controlled expansion of computing capacity showed that processing always depended on appropriate input format. In the case of the Air Force, that meant reducing the amount of data in an intelligent but machine-like way, and devising suitable formats to do it. It was to be hoped, wrote Dover in concluding his report, that in the future the limiting factor in evaluating test flights would once again be the processing capacity of the lead engineer and his staff, and not "obsolete data processing methods."[72] For that reason, he preferred to reduce data heterogeneity by automatically formatting it rather than to rely on increased computing capacity.

Adapting formats affected users, peripheral devices, programs, as well as computers and the people who operated them. Everything was subject to formatting procedures so that program-supported transactions could be set in motion in the still unfamiliar space of the computer. The formal rules of programming language, the format of input and output data, the equipping and training of programmers, and the layout of the equipment and connections – all had to conform to fixed formats so the

computers could process information.[73] The resulting changes in speed did not always lead to congestion at the same point. While for the aircraft test heterogeneous raw data was a major problem, in the case of the census the problem was mainly limited capacity of input and output devices. James L. McPherson of the Census Bureau reported in 1953 that the available input–output facilities could not keep up with the information processing capacity of UNIVAC.[74] Census takers could expect fairly uniform data formats, because they had standardized their forms over many decades. But the entries from the surveys were recorded on paper and first had to be brought into the computer.

McPherson vividly demonstrated the complicated work of shifting from analog to digital space. A census taker who was participating in a survey recorded the information received on a form. In a second step, this information was translated into numerical codes by a clerk at the Census Bureau. Another bureau employee in turn transferred the numerical codes to punched cards using a manually operated punch machine. Only at this point did an automatically operated machine come into play. A card-to-tape machine transferred the "intelligence" from the punched cards to magnetic tape. Now, finally, the input was ready for processing by the computer.

According to McPherson, it would be ideal to have a machine at hand that could do away with the entire translation sequence, that is, the encoding, the punching, the change of media from card to tape, and perhaps even the tape itself. Together with the National Bureau of Standards, the Census Bureau was in the process of developing a machine that could at least transfer the census takers' notes directly from the form to a magnetic tape. But this would require changes to the recording process, that is, a completely restructured form which would no longer have words and numbers written in fields but rather marks at certain positions.

To bring the world into the computer, the speed of information processing in the world's homes and factories would have to increase. And for that to happen, new ways of formatting the world would have to be found.[75]

As McPherson showed, the input process remained a trouble spot. It represented the transition from the world in which people operated (and time was measured) in months, days, and hours to the electronic world of data processing, in which time is measured in seconds, milliseconds, and microseconds. Better mechanical components might help to solve the problem. But input devices would probably never operate at speeds comparable to those achieved inside the "information processing equipment." Means would simply have to be found to minimize the inefficiencies caused by slow input.[76]

Although the census takers may well have wished for faster input methods, at UNIVAC, in view of the impending transfer of the census into the computer, great importance was already being attached to reading speed. In contrast to the options available using IBM's ubiquitous punched card computers, Eckert and Mauchly used Uniservo magnetic tape machines and some of their computer's programmable computing capacity for the input and output processes. Input and output would be run automatically, and consequently all magnetic tape activities were triggered by "programmed instructions in the computer memory."[77] In other words, already in the early 1950s the principle of the self-controlling computer was also applied to reading and writing data. Consequently, part of the input process was handled by the computer, just like parts of the programming routines were.[78]

Taming the computer – that is, exploiting the new digital sphere of operation – was enabled by a culture of systematic instruction implemented by programmers. Grace Hopper, a mathematician from Yale University who had done computing at


Figure 8: Processing data for data processing: a Census Bureau employee translates a questionnaire into punched-card format around 1950.

Harvard in the service of the navy during the war and who had been instrumental in developing UNIVAC, took up this subject in a 1952 article on educating the computer. A female role model in the thoroughly male-dominated world of early computers, Hopper was convinced that developing subroutines would free programmers of the burden of formulas and tables of functions. Nor did they need to know the particular instruction code used by the computer. Anyone who could read a catalog would know what to tell a computer to do to solve a problem. "The programmer may return to being a mathematician," wrote Hopper optimistically.[79]

The experiences of the census takers and test pilots, however, showed that successful taming of the computer through program-

ming and transferring known routines into the machine's digital space depended on solving the formatting problem. The fact that Hopper, a mathematician in uniform, believed data formatting to be trivial and, consequently, underestimated it in no way mitigates this insight. The computer calculations Hopper had carried out during the war under the command of Howard Aiken were not data-intensive but rather computationally intensive tasks. What was striking about a computer solving a differential equation was not the amount of numerical test material fed into the machine to calculate a particular variable in an equation during a trial run. In fact, most of the values in a system of equations could be produced in the computer itself, using a simple counter that stopped counting at a given highest or lowest value and then told the machine to stop calculating in turn. The striking thing about such tasks was that they required the same arithmetic operations over and over again, in overwhelming numbers.

The initial underestimation of the formatting problem was not only due to the relatively low amounts of data used in scientific computing in the early 1950s. As computers became more frequently used for commercial, data-intensive tasks, it was assumed that any difficulties in sorting volumes of data had long since been resolved. Since the Taylorist bureaucratization of companies in the first decades of the 20th century, Hollerith machines had become established in large firms.[80] Insurance companies had whole truckloads of punched cards on which customer addresses and premium billings were encoded. Banks would never have been able to cope with massive amounts of transactions in foreign exchange or payments without punched card machines. Large administrative tasks involved collections of punched control cards.[81] It seemed an easy thing not only to have the data from information-intensive companies re-sorted by Hollerith machines but also to have it read into the more universal dig-

ital computers. Yet doing so was a mistake, as experience with raw data from Air Force missile and aircraft tests and the laborious transfer of census taker survey sheets to UNIVAC's magnetic tapes showed. The assembly line on which sorting took place had indeed already been set up and the punched cards selected. But a slew of translation challenges remained to be mastered to make fast, program-based computerized data processing a reality. It was not only a question of how data were formatted, how processes were programmed, and how the computer did its calculations. It was also a question of organization and operation.

# 3 Sharing and operating

Traffic within the digital space was plagued by bottlenecks. Speeding up computing caused programming glitches. Any attempt to make the programming more efficient strained computing capacity or increased the data formatting requirements. The long path that led from encoding, formatting, and inputting data to processing it into a readable form was susceptible to peak-time traffic jams and hold-ups that slowed operations of meticulously prepared flowcharts to a trickle at certain spots. So-called batch processing was the most important form of traffic control. It required all data with identical processing steps to be processed in batches without further user intervention. This approach was akin to temporizing the capacity problem. Only when everything was lined up correctly *and* there were many uniform tasks did using the computer make sense.

It was hardly surprising that many administrative computing jobs had to be carefully prepared. Data that often required only minimal updating annoyingly still had to be rearranged, recopied, and sorted all over again so they could be properly processed and assembled into reports. But even when the data were impeccably formatted, the computer would sometimes stall, waiting for instructions while a new program needed for a specific analysis was prepared. Both cases ran counter to the analogy of the assembly line, in which small, repetitive operations on standardized units followed in rapid, perfect succession to deliver uniform products. It was time to take a more aggressive approach to organizing the entire computer system.[82]

Yet the question of organization went to the very heart of digital space, from basic responsibilities to procedural rules and the

interaction of subsystems to the architecture of entire installations.[83] Clarity was needed regarding how autonomous automated processes should be, how priorities would be decided, and which interactions should be allowed between machine components, data, users, and programs.

A shaky concept of automation invites reflection about whether the means are adequate to the goals. Alternative organizational principles must be discussed. Given the high cost of a computer system, it made little sense to leave it idling. For universities and parts of the army that relied on computing and cared little about market forces, a certain nonchalance toward efficiency may have been understandable.[84] But companies that wished to migrate their administrations into the computer to speed up their business processes had to face up to the problem. If tomorrow's administrations were to be automated like factories, and if they were to employ a veritable fleet of computers for this purpose, then it was only logical to judge them by the standards of factory equipment. The efficient organization of the computer therefore inevitably also became a question of economics.[85]

Based on experience, the cost–benefit ratio of a piece of equipment – a common metric in the industrial sector – could be improved if the machine did not have to be dedicated to a specific job. The same principle applied to computers. While waiting, they could, for example, sort cards, copy other files, and then – in the fast lane, as it were, and with the help of magnetic tape – continue processing the jobs with adjusted programs. However, this flexible approach to traffic required abandoning the idea of literally translating conventional forms of data processing for digital space. In other words, organization in digital space would have to follow its own rules. Familiar processes could then be redesigned thanks to the computer's increased internal complexity.

## Time-sharing

Only in the second half of the 1950s do computer engineers seem to have found the courage to stop thinking of commercial data processing systems as assembly lines.[86] By and large, observed IBM development engineers Murray L. Lesser and John H. Haanstra trenchantly in December 1956, the computerized handling of business data was not so very different from historical, "in-line" data processing. Lesser and Haanstra employed this bit of rhetorical downplay to bide their time before making a bigger point: what companies were doing with digital, programmable computers was nothing more than a "file-maintenance operation." Most of the computer's time was spent not on computing or producing new reports, but on arranging the data.[87] If the condition that information had to be stored in a predetermined serial order were removed, that is, if information could be retrieved separately from any memory location for processing, then the principle of batch processing could be dispensed with and any job could be done exactly when it needed to be done.

This kind of memory system was called random access. In order to correct, supplement, and reorder entries, it should be possible to access any point in the memory at any time. Data would no longer have to be fed to the machine as a batch. With random access to any place in the memory, it would be possible to read out individual elements from the reference files in the memory, and in the order required by the new input or the current processing procedure. "In theory," wrote Lesser and Haanstra, "a machine can be built to take an input transaction record and carry it all the way to the final output document." En route, the corresponding records in the machine's memory could also be updated.[88] Because now, according to Lesser and Haanstra, processing a transaction could always take place at the moment it occurred. In this sys-

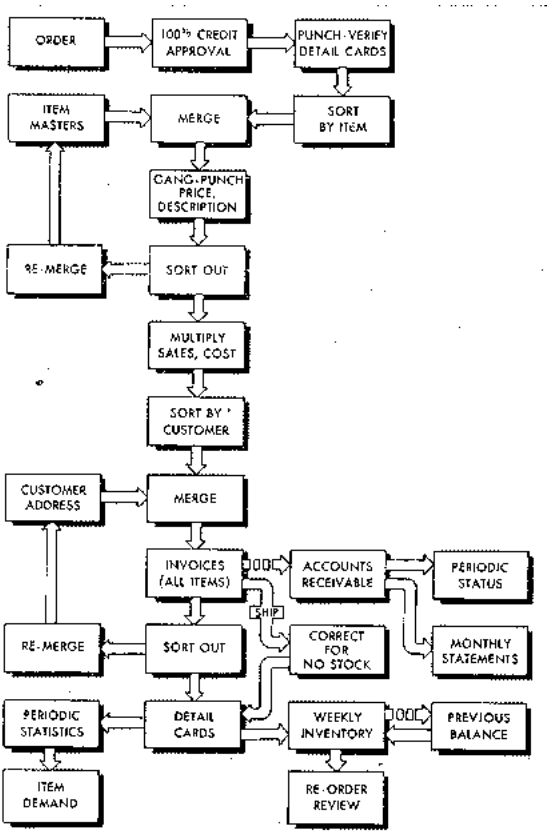


Figure 9a: Random access: the "floor plan" of a conventional data processing system.

tem, the time needed for processing was independent of whether or not further transactions of the same type followed. In other words, creating homogeneous batches no longer offered any efficiency gains. A random-access machine thus kept a supply of data ready in the computer for ad hoc access on a rotating, magnetically coated drum.

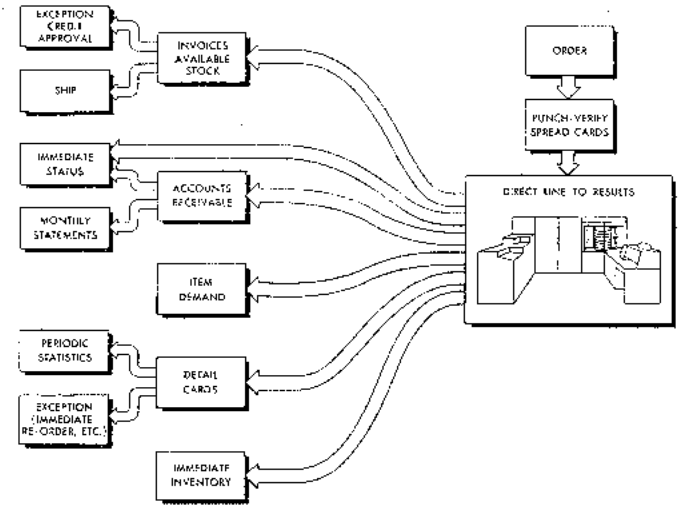


Figure 9b: Random access: one input, many forms of output.

This change in concept was also reflected in terms of representation. The diagram for the conventional data processing system that Lesser and Haanstra took issue with looked like the floor plan of a manufacturing plant with an assembly line running through it. In the diagram of a system that worked with random access, on the other hand, the assembly line was hardly recognizable. That is, if you really wanted to keep things flowing, you had to abandon the assembly line concept and increase the internal complexity of the system. From now on, there would be only one input station outside the machine, where punched cards with additional data were produced. The data would then be processed immediately without any further sorting or arranging steps. The results could be output in many different forms.[89]

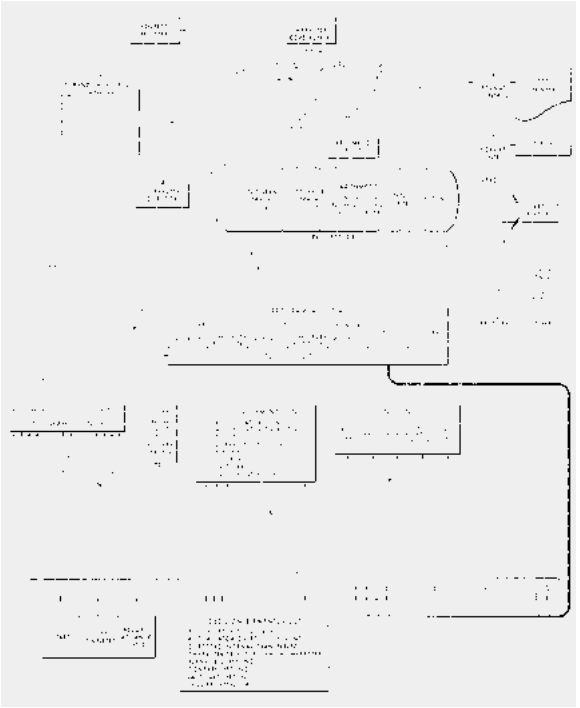

Figure 9c: Random access: additional internal complexity.

A third schematic depicting the system's organization showed how complex it actually was. A view of the computer's insides showed the numerous points at which the machine had to decide what should happen next for a given job. Organizationally speaking, the machine was the authority that decided processing steps and procedures for reading, modifying, and collating information, and that distributed updating and output tasks inside the machine.[90]

As presented by IBM, the random-access concept was promising. The bottleneck at the gateway to digital space would be re-

## Solving a queuing problem

**University of Michigan students line up for a go at the 7090. Soon, they'll have time-sharing**

At the University of Michigan during the winter trimester, 2,084 students enrolled in 133 different courses used the University Computer Center to complete course assignments. As a result, the line-up at the computer window last month was longer than the queue at the ticket window of the local movie.

The Computer Center staff at Michigan thinks the line-ups will disappear late next year when its 7090/1410 computers are replaced by a System/360 time-sharing computer (page 8). Michigan plans to expand its time-sharing computer capacity in steps until it will be able to handle 200 remote terminals simultaneously. With the new computer, turn-around time for a typical student problem will be a matter of seconds. Now, at busy periods, it's an overnight wait.

Figure 10: Line-up at the window of the University of Michigan's computer center in 1965 – time-sharing promised to reduce visible waiting time.

lieved by making procedures more flexible and by distributing data within the computer. Later, the principle of segmentation was introduced to get around a second bottleneck related to programming. Despite efforts to use programs as many times as possible, "universal" computing actually required constant (re)programming. Management, the personnel department, and warehouse staff constantly needed new reports. And each of these reports required the data to be linked in a particular way,

tracked different products, or had to adjust payroll accounting to reflect changes in rules. Arithmetically, these tasks involved trivial calculations, but the programming code still had to be written and tested. Programming involved a time-consuming search for errors. Every infringement of programming language syntax, no matter how small, had to be found and corrected, and each time the program had to be tested again. During this painstaking work, a computer was only ever occupied for a very short time – if at all – and did nothing but wait patiently for further program versions from the programmer. Other programmers also waited, in their case, for the computer's input console to free up.

For precisely this reason, the second development related to programming aimed to free the computer world of its fixation on the idea of the assembly line. This shift in thinking became noticeable at the end of the 1950s, when system developers like John McCarthy proposed time-sharing to reduce waiting times for programmers and computers.[91] If computer users actually spent only a small fraction of their time at the console computing, perhaps the number of possible users should be increased by granting them access to the computer only when computing time was really needed. Thus, while a programmer obsessed over errors ("bugs") or thought up new (equally buggy) instructions, the computer might as well be left to another user. No programmer should have to wait just because someone else was catching a breath and preparing to compute. In a 1959 memorandum on time-sharing, McCarthy referred to computer history and, in so doing, brought a new rhetorical flourish into play. Computers had originally been developed with the idea that they would be used to solve general classes of problems and that computing time would be spent running these standard programs with new sets of data. "This view completely underestimated the variety of uses

to which computers would be put."[92] The current situation was much more like the other extreme, where each user wrote their own program and, once all the bugs were finally worked out, ran it only once. That meant that the time needed to solve a problem was mostly spent on debugging. Better programming languages such as Fortran, LISP, and COMIT would help to reduce programming time. However, further reduction could only be achieved by shortening the computer's response time.[93]

In July 1961 the *Science News-Letter* announced that individual computers would soon be able to serve many companies. McCarthy and others at MIT were developing methods for using high-performance electronic computing to work simultaneously on many problems for many users at once. According to the *Science News-Letter*, this would be a big step toward building data centers that could receive data over telephone lines and transmit weather, economic, or other forecasts to their customers.[94]

The article was a bit out of date. McCarthy had in the meanwhile left MIT to share his own time with Stanford University. Nevertheless, his former colleagues at MIT continued to pursue time-sharing. The rate of interaction between programmer and computer had to be drastically increased without increasing costs. Moreover, according to a 1962 paper on time-sharing by Fernando Corbató and colleagues at the MIT Computer Center, individual interactions had to be made "more meaningful" through "extensive and complex system programming to assist in the man-computer communication."[95]

In May 1963, MIT science reporter John Fitch filmed an interview with Corbató titled "Timesharing: A Solution to Computer Bottlenecks."[96] From the point of view of narrative, the film was hopeless. The unambiguous promise of the title was compromised by the film itself. Fitch had clearly done his homework. But he got few direct answers to his questions from Corbató, who stood at

a blackboard drawing squares and circles with a fiddly piece of chalk, not stopping to label the symbols with numbers. Even the slightest query from Fitch elicited a lecture from Corbató. "To really explain that, I have to backtrack a bit," he said several times, because you could only understand (whatever the question was) if you knew what a computer was and how it worked.[97] However, what a computer was and how it worked was precisely the problem from the vantage of the time-sharing concept. Thus, Fitch repeatedly had to bring Corbató back to the original question or volunteer the answer himself. And, like Lesser and Haanstra's attempt to explain random access, Corbató was in the unenviable position of having to talk his way through a computer model. To understand the advantage of time-sharing, you had to understand the old model, and that was not the point.[98]

Nevertheless, Corbató's efforts gave rise to two striking analogies and one clever rule. The first analogy stressed neither the system architecture nor the process bottlenecks, but the perspective of the users.[99] For Corbató, they were the people who were stuck waiting for the machine. To solve the interaction problem, he wanted to make the computer available to many users at the same time, drawing an analogy to the relationship between a telephone line and a switchboard. Each user would be able to use his or her own machine – his or her own console – and not have to worry about the activities of other users. The switchboard would be responsible for them.[100]

The digital space managed by computers was thus reallocated, the distribution of resources made more flexible, and their overall utilization improved. In other words, capacity was increased by expanding the number of buffer spaces while at the same time consolidating the flow of traffic inside the computer. That required a suitable mechanism for registering the peak computing demand of each user and for temporarily pausing a currently running program

of another user. At MIT, the mechanism was a (virtual) supervisor that regulated and monitored internal computer traffic, allocating computing time to some users and putting others on hold. To illustrate the process, Corbató briefly adopted the perspective of the computer. Imagine, he said, a chess master playing several weaker opponents. The computer quickly makes a move, and while the first opponent is thinking things over, the computer has already gone on to the next opponent. Opponents who take a particularly long time thinking are skipped over and only come back into the game when they have finally made their move.

At this point, Corbató returned to the perspective of the individual user. Say that the user is sitting in the computer center at a ball-head typewriter connected to the computer and reads a result, an error message, or a question from the computer that has been printed out onto fanfold paper. As soon as the user knows what the next command is, he bangs it out on the keyboard and hits ENTER. Now it is the computer's turn to respond at the next opportunity.

In this way, Corbató wished to explain that users in time-sharing mode had to wait less long because the computer only had to attend to them once they had pressed ENTER and their thinking had come to a temporary end. Only those who presented the computer with a particularly complex or difficult task had to reckon with a slight delay in the answer. This had to do with a clever rule to facilitate time-sharing that they had come up with at MIT and were particularly proud of. Priority was given not to the most demanding computational tasks, but rather to small, routine tasks. That made it possible to reduce the number of people waiting much more rapidly than would have been the case if priority had been given to large and cumbersome tasks.[101]

Time-sharing was about much more than simply overcoming the computing capacity bottleneck. Long before sharing became

a socioeconomically attractive feat of computational action and interaction, the MIT project showed that digital space resources could be multiplied almost miraculously through appropriate organization. The impressively stolid computer could be turned into a flexible tool. But success relied on satisfying one condition. Just as random access required reorganizing the computer so that data could be processed in any order, time-sharing required organizational measures that would enable users to interact with the computer in any order.

Apart from the coordinating role of the supervisor, which was indispensable for time-sharing and whose widespread application I will come back to in a moment, time-sharing gave rise to two other big and far-reaching questions that engendered much debate.[102]

The proliferation of the consoles, or terminals, attached to computers highlighted the issue of interactivity and made it the focal point of organizational development for decades. It was no accident that Corbató compared his system to a telephone exchange. The sociotechnical problem of communication with or via the computer became particularly acute not only from the moment when the number of consoles hitched to a computer increased but also when they began to be distanced from the computer. Some kind of cable had to lead from the user's workstation to the machine in order to transmit requests and responses. The question was whether this cable was still effectively part of the internal wiring or whether it already constituted a telecommunications link between computers.[103]

The second, no less far-reaching question followed directly from the first. Would electronic data processing done by universal computers sooner or later be understood as a "utility," that is, as infrastructure? And how centralized should its systems and organization be?[104] Once random access and time-sharing came

into the picture, the question of how a computer was to be organized and, as a consequence, how it would be operated also became a question of the form of operation and the operating system.[105]

## Operation

The supervisor was crucial to the experimental operation of the time-sharing system at MIT. The supervisor regulated the traffic in the computer, allocated computing time and storage space, and determined how much latitude to allow individual users. The increasing number of inputs and outputs in commercial computing, with its many but small computational operations, as well as the interactivity that time-sharing programming entailed, meant that computer operations required an entirely new type of control. The computer had to be able to make condition-based decisions, and these decisions had to be matched with flexible but secure procedures. Metaphorically speaking, an administration was created in digital space. This administration weighed competing use requirements, access rights, patterns of use, and accountability against each other and regulated the system-compliant interaction of hardware components, data, application programs, peripheral devices, and users.

Since the early 1960s, computer scientists had dealt with these issues through abstraction and by developing operating systems.[106] The idea behind these systems was to provide a simple view of all parts of a complex system. The need for operating systems was justified by the need to optimize the allocation of storage space and computing capacity due to economic constraints. Operating systems thus constituted a response to the problem of relative scarcity.[107]

Although organizing the computers made economic sense, what operating systems actually had to do was political.[108] Operating systems were the policy that applied inside the computer. They determined what was allowed and what was not allowed, and they monitored compliance with the rules they set. To appreciate the idea of operating systems as government, you have to study them in *statu nascendi*, that is, during development, before they became commonplace. The description of the operating system for the Atlas[109] computer at the University of Manchester, published by Macmillan under the dubious title *Computers: Key to Total Systems Control*, provides an example.

Writing about computers in the early 1960s meant writing about two simultaneous types of control. First, the computer system obviously had to be organized in such a way that whatever the analog world expected of it at any given moment could actually be done in digital space. Second, the organizational structure of the computer could be thought of as key to understanding the organizations that might conceivably be relocated to the computer. Tackling these fundamental issues was no trivial exercise, which may explain why the word "total" prefaced "systems control" right in the title. Whether the control of the system was thus meant to be full and complete or whether, in the authors' view, it might shift more in the direction of authoritarianism, if not totalitarianism, is a matter of speculation.[110] It is equally unclear who is addressed as subject and who as sovereign may declare a state of emergency – the system that controls everything or those who control the system? In any case, the multilayered formulation was strategic and aimed at defining responsibilities.

Like Fernando Corbató at MIT, the authors of the report on the Manchester computer – Tom Kilburn, R. Bruce Payne, and David J. Howarth – introduced an entity that they initially named supervisor, only to have it morph into the broader notion of operating sys-

Figure 11: Tom Kilburn and his colleagues reflect eager anticipation of the Atlas supervisor's monitoring system in 1961.

tem. According to Kilburn, Payne, and Howarth, all the activities of the system were controlled by this supervisor, which became active frequently and for a whole range of reasons and which ran on the same computer as the application programs.[111] However, there was mutual protection between the computer's object programs and the supervisor program, which consisted of many, normally dormant branches. The supervisor would have to be shaken awake, for example, if data transfer between different memories was pending, if something had gone wrong in calculating an exponent, if memory space became overcrowded, or if computing time was insufficient. Any object program could call up the supervisor and claim one of its 250 subroutines from the core memory. However, the relationship between the system and the programs was not limited to one-way emergency calls from the programs. The system was also able to easily interrupt programs (interrupt control) and gain access to

the "private" memory space currently reserved for a user if system stability required it or if peripheral equipment was being used.

The relationships between different memory types, instances, and control commands were dealt with in a separate chapter on the coordination of routines. Here, the focus was on the logical structure, effective time management, and rules-based interactions of the system and its parts. The authors described a carefully thought-out ensemble of rules that ensured the system's interacting components the degrees of freedom they needed, but at the same time protected them from unauthorized encroachment. At such instants, the policy also policed.

All of this was quite naturally subsumed under the term "operating system" only a few years later. It is therefore striking that in the description of the Atlas computer system, the chapter on structure and coordination was followed by whole chapters on store (memory) organization, magnetic tape routines, and peripheral equipment attended by operators, and then a separate chapter on the "operating system." An operating system obviously needed a long lead time, even rhetorically: in 1962, the operating system was not yet a stable, sufficiently independent, and comprehensive concept. It could also be thought of in terms of the supervisor. That is why the supervisor figured in the title of the essay, and not the operating system.

This was to change in the near future, for very good reasons. *Operating system* would become a generic term, because it denoted a comprehensive regime or a set of abstract rules distributed over the entire system. The idea of a set of reliable, intelligent, and well-balanced rules that gave everybody sufficient freedom and kept a lid on expectations was already apparent in the text by Kilburn and colleagues. For as soon as the operating system was actually mentioned, albeit somewhat late, its significance for the entire system was immediately and directly obvious. It could

be used to deal with a wide range of problems, from small tasks which required no data beyond the program itself, to large jobs that relied on several data sources and that might have to be input from different devices and data carriers.

The setup of the Atlas operating system was strongly oriented toward allocation security and processing efficiency. When available computing capacity matched the requested capacity, queues were shorter and working memory could be emptied after the job was done. This "regulatory effect" of the operating system was tantamount to an independent computer administration. After all, in everyday life, control is inevitably administrative by nature. The operating system produced information that it could use for control. How often was the program changed? How many instructions was the system processing? When did it need to access which store? How long did it take to read data and print results? Did magnetic tapes need to be accessed, read out, and overwritten? All this, and more, was logged, and the log was then used to assess machine charges. The log entries were printed out so that operators could check the computing time used and charge it to the users. The precarious economics of the computer led to the systematic monitoring of its operations by an operating system whose records served as *contre rôles* for determining individual operating costs.

The only way to prove the intrinsic worth of the system lay in generalizing instructions, that is, developing overall routines and rules of operation, particularly for time-sharing. For starters, that meant developing large operating systems. Here, however, the situation quickly became very confusing. The development of OS/360 at IBM is said to have cost around 5,000 man-years.[112] The development of a "Multiplexed Information and Computing Service" under the auspices of MIT also involved a long-term commitment by the companies involved. With the help of MULTICS,

Boston would claim the first future-oriented regional computer center with general-purpose time-sharing.[113] Both OS/360 and MULTICS featured extremely dynamic requirements; both would have to develop a highly connectable system that was capable of anticipating future applications, would require several years of development time, and yet could assure the stability of its procedures in a rapidly changing context. While IBM wanted to develop a single operating system for all of its own product lines, MULTICS pursued the goal of supporting the widest possible range of users and their human-machine interactions around the clock. As Corbató and Vyssotsky noted in a 1965 conference paper, the demands ranged "from multiple man-machine interaction[s] to the sequential processing of absentee-user jobs…to programming of the system itself," and from operation of centralized bulk card and tape readers to "remotely located terminals."[114]

The Atlas, MULTICS, and OS/360 operating systems each were designed with different goals in mind. Atlas primarily controlled a variety of programs, MULTICS a variety of users, and OS/360 a variety of machines. However, all three strategies had in common that they managed the interaction of components, distributed rights, allocated computing time and different memories, and logged this activity of machine operations.

The work at the interface between users and machine and the increased sophistication in memory access and capacity management should not distract from the fact that moving the world into the computer required tremendous restructuring and reinterpretation, which shook the world to its core. One of the most prominent sociologists of the late 20th century addressed this problem in his habilitation thesis, published in 1966, and drew interesting conclusions from it in terms of his theory of social systems. In the preface to his "Automation of Public Administration," Niklas Luhmann singled out the high investment costs for the large-

scale computers of the 1960s as the main reason why the computer posed such a conceptual and organizational challenge for bureaucracies. According to Luhmann, "a refreshing compulsion to think emanates from the fortunate circumstance that the machines are so expensive. Their price forces one to rationalize the organization of data processing, even outside the actual system, to an extent that would have remained impracticable without this impetus."[115] There was certainly much thinking – deep thinking – to be done to understand administration as a system and then to translate the processes typical of administrations into a computer-compatible format.

In so doing, two paradoxical issues arose. First, very different administrative units had to cooperate across departments. For instance, the state department of motor vehicles could hardly be expected to finance computer administration of license plates on its own, so the computer would have to be used simultaneously by the revenue department, the local university, and the human resources department.[116] From the perspective of its users, the expensive computer was rigidly universal: it remained consistently general purpose and rules bound, had no care for administrative traditions, and was especially impervious to administrative quirks and specializations. Administrations only became computer-capable once they abandoned specialized formats, processes, and forms in favor of fast electronic data processing, that is, when they deviated from the path that had hitherto been considered the high road in differentiating bureaucracy. This cleared the way for what promised to be an eventful journey on the part of state and corporate administrations into the computer.[117]

Second, the complexity of administration actually had to be increased so that a computer-friendly simplification of automatable transactions could take place. Because administration does not automatically become simpler by simplifying decision-mak-

ing processes (assuming performance remains the same), and because automation presupposed simplification on the machine side, complexity had to be shifted, in Luhmann's words, "from decision-making behavior into the system structure." That in turn would "[bring] organizational problems with previously unknown requirements to the fore." To be sure, this was nothing new for theorists dealing with bureaucracies. For them, "simplifying individual decision steps" was always "achieved at the cost of complicating the system structure and, by extension, system planning." The daily activities of administrators could be relieved by system complication. Electronic automation of administration, according to Luhmann, "only took this old insight to its extreme" and "therefore made it apparent, because in this case the individual decision steps to be assigned to the computer must be simplified quite radically."[118]

Programming the computer presupposed an in-depth analysis of the decision process. That is why it was also an almost inexhaustible source of problem-oriented thinking about the formatting of the social, which made possible its ability to compute.

This insight could also be found across the Atlantic, most prominently in the work of Herbert A. Simon. In terms of intellectual history, Simon's writings on decision-making processes, administrative behavior, organization, and automation could not have been written without considering the possibilities and circumstances of structure formation in the digital world. In Simon's case, this thinking began surprisingly early. He wished to formalize organizational conditions mathematically (very much in the tradition of the operations-research literature and cybernetics). Luhmann never made such an attempt. He was much more influenced by the world of computers in his style of thinking. Nevertheless, Luhmann and Simon were clearly inspired by the abstract interplay of rules that computers required to op-

erate, and likewise by concepts and procedures used in the field of computer organization. This can be seen, for example, in "programs"[119] or the "belief that abstract principles of structure may be discerned in organizations of great variety [like computers], and that ultimately it may be possible to state principles of general organization."[120] This possibility was of crucial importance for what computers had to offer businesses, organizations, and bureaucracies. Both the organization of computers by means of operating systems and the operation of organizations by means of computers depended on the interplay of abstract rules.

It became particularly interesting when, once this structural uniformity had been established, people refocused on differences, for example, between what tasks could be moved into computers and what tasks in the analog world were structured differently as a result of this move. In considering the special problem of the relationship between an administrative lawyer and a computer, Luhmann raised the following, quite tricky questions: "Does [the lawyer] decide less deliberately, less prudently, less rationally than the machine? And if he follows the same principles, can he transfer all or part of his deliberative steps to the machine? Is the rationality of the machine different from that of the lawyer?" Luhmann wondered whether a computer and a lawyer have the same function in an administrative system and if so whether in principle one could be substituted for the other. "Or are their functions and their contributions to decision-making different? If so, are they contradictory, such that the lawyer must distrust the machine, and vice versa?"[121]

Such questions about the nature of transactions in digital and social space had arisen since the early 1960s, not only in general administrative and legal terms but also as other areas moved online, including currency trading,[122] airline reservation systems,[123] library cataloging,[124] and supply-chain management in the retail

trade.[125] Almost inevitably, the change added sociotechnical analysis to the skills set of project managers. That would, at least, seem to be the case given that even modest electronic automation projects had a way of turning into major reorganizations with some regularity. The fundamental characteristic of technology and technological change is as true here as elsewhere: technologies are tools used by social actors (user groups, organizations) to advance their own interests, consolidate existing positions, or gain additional influence over the course of events. The preoccupation in the late 1950s and early 1960s with regulating traffic in digital space had consequences not only for the formation of theory. It also had a lasting impact on "operational education." The cross-departmental use of computers, which Niklas Luhmann found so stimulating, allowed an insight into administrations and companies that had previously been difficult to gain. If the computer could become an unrivaled information system of management, then the position of management probably changed with it. The sharply drawn line between what was considered engineering knowledge and concepts and observational routines that had been inherent in management since the early 20th century began to shift. The distinction between the figure of the engineer and that of the manager became uncertain and fraught. As American historian of technology Thomas Haigh has shown, some observers even assumed that transferring substantial administrative procedures to the great archipelago of computers would give rise to a new class of knowledge bearers. These "systems men" would be sufficiently expert in both IT and business management to take on positions of power in companies and other authorities.[126]

Most of the expectations that these systems men held for the computer, for their own skills and competencies, and for the world's sociotechnical future in general proved to be exaggerated and ultimately failed to materialize. But they did increase the

semantic market value of what came to be called a management information system. Everything that was most important to corporate management and operations was to be stored and analyzed through this system. A management information system represented a container for all the knowledge possessed by managers with a knack for abstraction, experience in business administration, and close contact with a computer. Established practitioners of analog business administration were having none of it.[127] Furthermore, computer experts were uninterested in sharing their career paths with IT-savvy managers, and corporate executives ignored the naive attempts of systems men to seize power.

Nonetheless, at the beginning of the 1960s it was perfectly in line with the expectations of companies to be able to provide management with a basis for decision-making by outsourcing management-relevant knowledge to the computer.[128] Nor did it prevent anyone from starting work on management information systems, designing them, discarding them, renaming them, and re-promoting them – regardless of what was meant by the term *management information system* in each case.[129] Later, sophisticated management tools and enterprise resource planning systems would emerge from this breeding ground. That would require stable rules for computer operating systems and computer-based operations of organizations as well as building data centers that would serve as obligatory transit points for procedures and data. For now, however, universal computing systems required flexible traffic rules, traffic flows, and a range of operating procedures. Powerful operating systems were a sine qua non. The guiding principle for the organization of such computers was not divide and rule, but divide and operate.

# 4 Synchronizing

Around the mid-1960s, digital space developed a particularly intimate relationship with the parts of the world that were organized within it. It is important to note that computer-based organizations gained power not through digital representation, but through digitally accelerated and extended interaction. To prevent this interaction from degenerating into the computers' preoccupation with themselves, the world in the computer had to be synchronized with the conditions in the organizations whose business had to be transacted. Organizations typically have a keen sense for how to control implementation of their plans.[130] Digital space offered them controlled simulation of planning, real-time monitoring, and logging by the computer that would allow them to check results later. Nothing provides a better illustration than NASA's Mission Control Center, which was established in Houston, Texas, in 1962 and equipped to the teeth with IBM technology.[131]

"Houston" was an extreme model of the computerized monitoring and control culture of the 1960s.[132] The Mission Control Center was set up to oversee everything for a series of spectacular space missions from planning and simulation to coordinated monitoring of the long excursion to the moon and processing reports of expeditions to the responsible authorities. It was a huge effort. Take, for example, the installation of five transistorized IBM cutting-edge computers, the worldwide network of radar and radio stations, the army of highly qualified employees, and the center's enormous expenditure on media technology.[133] Even in the glare of the global television spotlight, the sociotechnical interactions appeared highly credible and procedurally effective.[134]

To this end, the center had developed a complex internal organization and infrastructure that set completely new standards. Houston emerged as a place where synchronization of the world with the computer monitoring and control room played out in an exemplary and forward-looking manner.[135]

NASA's computerized setup was paradigmatic, even epoch-making, because it could claim to operate in real time through skillful structuring of the computer system and its environment. Spaceflight problems are extremely time critical. Unprecedented velocities and forces require very short reaction times owing to particularly serious consequences. All events that deviate even slightly from the plan must be handled without appreciable delay, that is, almost synchronously with their occurrence. Operationally, this means responding in real time and minimizing the time difference from the generation of data to processing and display by the computers. That's why the computer setup in Houston's control center was also designed with a real-time operating system. However, as in all computationally intensive activities, the ultimate goal of operational real time relies on selectively limiting precision and varying speed. Approaching the desired immediacy was done by sorting tasks and classifying them by urgency. To this end, the center provided a complex structure, consisting of a host of technical and procedural components. This structure enabled the problems of spaceflight to be distributed and delegated, or timed and shifted. In this way, future problems were slated for simulation and training, current problems for monitoring, and past problems for reporting.

The tendering process for the new center's computers had already made clear that the demand for computing capacity would dwarf anything previously available. According to estimates, between five and nine large computers would have to be installed to ensure sufficient and sufficiently reliable computing capacity.

Nevertheless, the need for equipment, software, and for human resources was still massively underestimated by both NASA and IBM, which was awarded the contract. Instead of the project's planned 161 computer engineers, more than 600 ended up working on it simultaneously, two-thirds of whom were assigned to programming the operating system and the myriad of applications alone.[136]

The program landscape being created in Houston, called Executive, consisted of three main subprograms. At the center was the Mission Operations Program System, which performed flight path calculations, processed measurement data, monitored the spacecraft environment, acted as backup for the on-board computer, and calculated rendezvous maneuvers between different spacecraft. This key program was supported by a program called NETCHECK that monitored data flow and by the ANALYZER program, which evaluated logged flight data after a mission.[137]

On the machine side, one computer was dedicated to the ongoing mission, the second served as a dynamic backup computer, and the third was used for simulations. Later, a fourth computer was added for simulating ground systems and a fifth for software development. An IBM 1401, which had proven a solid data processing machine, was used solely for input and output. Around 1965, this formidable infrastructure allowed Houston to "handle 10 times the data flow, have 100 times the computing power, and 275 times the capacity to present display information" at the start of the Gemini spaceflights than had been the case at the Goddard Space Center in Langley, where the Mercury spaceflights had been monitored from 1959 to 1963.[138]

For real-time operations, NASA and IBM relied on the redundancy of the monitoring and control system for the computers, provided sufficient backup, purchased additional memory, and accelerated both input and output with expanded capacity.[139] By

the time the center was commissioned in 1965, the first three IBM 7094s had already been upgraded to twice the computational memory and five times the auxiliary memory. Real time gained time by expanding machine memory.[140] If the facility had not been fixated on such a specific aim, it might have been described as an all-purpose facility with great future potential. In Houston the same kinds of complex technoscientific problems were dealt with, and huge amounts of data from operations were sorted and classified. The output provided a thoroughly calculated basis for decision-making, as would be expected of any powerful computer in the service of purely terrestrial large-scale enterprises.

It would take a very large computational effort to digitally handle space travel. That much was evident. What was surprising, however, was the fact that the operations and communications going on around the computers also had to increase enormously. Countless installations assured the proper temporary storage, rearrangement, channeling, and distribution of information, so as not to burden the computer system except in selected cases (or for an emergency). Viewed in this light, the control center was a massive, highly intelligent sorting facility in the service of the computer center. What proved too complex or too voluminous for the computers to process immediately floated through the back-end services of the flight control room's analog world until the task was reduced in complexity and a computer was free to work on it. It was the conventional part of the facility that made good the strategic claim to real-time computing by keeping complex information away from the computer.

Architecturally, the control room favored a showy, even theatrical layout. The consoles were grouped by function and staggered according to a fixed hierarchy. Controllers who dealt with engines and fuel, that is, with the thermodynamics of space travel, sat at the front. Right behind them sat half a dozen electrical engi-

neers who took care of power supply, data, and communications. The third row was responsible for organizational matters and command. Finally, in the fourth and top row sat senior officials responsible for liaising with dignitaries and the public: the director of flight operations, the two representatives from NASA headquarters and the Department of Defense, and the mission's press officer. The room was closed off by a pane of glass, behind which was limited cinema-style seating for 74 invited guests.[141]

The strict spatial arrangement of the control room did little to ease navigation of its technical complexity, even for those who worked in it. Accordingly, NASA produced and repeatedly updated its *Familiarization Manual*, already a good 150 pages long in 1967, which described the entire infrastructure of the Mission Control Center for the "orientation/indoctrination" of the employees, as stated in the preface.[142] The manual was the pinnacle of a whole family of manuals on problems of simulation, operation, and maintenance of technical systems that secured communications between astronauts, ground personnel, and flight control; provided links to rescue ships, aircraft, television channels, and the government; and supported the interaction of computers, printers, teleprinters, radar stations, and projectors.

To handle temporary storage, as well as the classifying and distributing of tasks, a complex technical and personnel organization was created near the mainframes. Only a small selection of these problems found their way into the computer system; everything else was examined and processed by ground personnel at consoles in the control room. Each flight controller worked his way through the detailed flight plan prepared for his area of responsibility. At his back was a series of manuals in ring binders that could be consulted, supplemented, and corrected at any time; in front of him were log entry forms, an ashtray, and a slide rule. His headset enabled him to connect to a team of engineers

sitting outside the control room for support with tasks that were too large and too complex to be solved single-handedly at the console. When unexpected events occurred, the flight controllers relied on the knowledge and experience of the back-room staffers. The manual describes endless halls of offices and meeting rooms where the invisible control center staff pored over such assignments.[143]

The individual flight controller's sorting system at the console also helped to restructure the rigorous work schedule as needed, obviating the use of computing power. The system specifications and supplementary tables on emergency procedures stored in the ring binders would be calculated and checked long before a mission. Preparatory work could be called up at any time for special situations or to make changes to the flight plan. But the potentially continuous stream of messages, questions, and answers on voice radio also facilitated emergency responses. This air-to-ground audio loop followed a highly structured pattern. The steady communicative interactions were distributed over different channels into which each flight controller could listen with his headset, knowing that everything spoken during the spaceflight was recorded on tape and would also be transcribed later. The console of a flight controller was not unlike a mixing console. Here, a specialized program adapted to the task and situation at hand was assembled and combined with documented routines.

A selection of relevant data, freshly processed by the data center, was transmitted to the flight controller on his own screen. Here, too, various time scales, formats, and speeds could be identified. The small TV screen on the console continuously displayed images captured by video camera from the mainframe computer's cathode ray tube (CRT) screen. The process was very analog. The data themselves might be numbers or even a single moving point. For the controller to make sense of the unadorned informa-

tion, a physical slide that contained formatting information such as axis labels or column and row headings was projected over the video camera, and the result was sent to the controller's console screen as a mixed image.[144] If the flight controller wished to consult a colleague, he could send the mixed image to the colleague's console screen. If the controller needed more time, he could request a printout on paper by pressing a button. The printout was then sent to his workstation from the computer center via the pneumatic tube system. Now he could study the table in relative peace, compare it with older copies, or if he had questions, forward the table to the team of engineers outside the control room. When needed, the team sent back additional documents, also by pneumatic tube, to be discussed with the flight controller over internal telephone lines.[145] As soon as the situation was clear again, an instruction could be forwarded to the spacecraft via the flight director on duty and the capsule communicator responsible for communication with the astronauts. What they received were terse instructions for handling a problem.

Tasks could thus be distributed, procedures sequenced, and tasks could be prepared or postponed to the future. Most of the buttons on the consoles in the control room were wired to pneumatic tube and call addresses precisely for the work of postponement. At the same time, sufficient synchronization had to be ensured. A variety of provisions saw to it that outsourced, postponed, and distributed work could be seamlessly brought back into the flight plan and be ready for real-time intervention on time.

A central system clock provided the temporal framework necessary to synchronize all procedures and subsystems. Real-time monitoring and control depends critically on a stable system time. In Houston's *Familiarization Manual*, few words occurred as frequently as the word *time*. The reference for system time was a signal disseminated by radio from the National Bureau of Stand-

ards as Greenwich Mean Time. In Houston itself, this nationally assured reference to universal time was locally anchored and distributed to all other functional units by an elaborate timing subsystem. The timing system consisted of "master instrumentation timing equipment, [a] countdown processor, two sets of relative time accumulators, six sets of dual stop clock equipment, two serial decimal time converters, three timing signal distributors, time display and control modules, a timing interface unit, and wall clock equipment."[146] Synchronization of monitoring processes, when needed, was referenced to the control center's highly stable system time, which by no means excluded the supplementary use of wristwatches and handheld stopwatches.

A second synchronization was done via voice radio between the capsule and the ground station. In the early Gemini missions, responsibility for this task was passed from one terrestrial receiving station to the next and involved constant adjustment of frequencies and repeated link checks. Only stations that theoretically had visual contact with the spacecraft also had voice communication with the astronauts. The result was alternating bursts of communication and radio silence, which affected other channels. When, in the second half of the 1960s, all radio communications began to be handled the same way as data transmission via fixed data lines – using time-division multiplexing – interruptions diminished and little of this alternating pattern remained.[147] Moreover, transcripts of various space missions indicate that radiotelephony was becoming increasingly more continuous. It was more standardized and provided flight controllers with a secure auditory structure to which further radiotelephony channels and tape recordings could be added.[148]

The third synchronization thread was visual. All flight controllers, as highly expert viewers, kept an eye on a common three-part display at the front of the control room. They understood that

the real-time coordinates of the spacecraft referred to its current position in space, in time, and with reference to the flight plan. This position and the further flight trajectory were continuously recalculated in the computer and "plotted only a few seconds behind" via an "Eidophor" video projector onto the center screen before the rows of consoles. This allowed the flight controller to follow it at all times.[149]

This image, like those on the screens at the consoles, was also created by overlay. Slides containing formatting reference information, and video images of current flight data recorded from the screen as naked points or numbers, were optically combined into a readable video image. This image appeared on the middle and largest section of the three-part screen. To the left and right were two smaller projection surfaces angled inward, like altar wings. A frieze above this triptych comprising nine illuminated displays showed the most important numerical parameters of the current mission. Whereas the left screen of the triptych typically displayed the current page of the flight plan via an overhead projector, the right screen displayed video images. Here the controllers were shown live images of the launch of the carrier rocket, for example, photos taken from inside a spacecraft, or the "live broadcast" on 20 July 1969 of Neil Armstrong's long-awaited exit from the lunar module.

Mission Control Center's picture screen combined many, very different sources and media. This created a multimedia imaging system that synchronized the monitoring and control setup both internally and externally in such a way as to conflate beginning and end, and inside and outside worlds. At first sight, the Houston facility recalled the closed worlds of the subterranean command centers designed in the 1950s for nuclear first or retaliatory strikes that had long become a fixture in film and television by the 1960s.[150] But Houston's imaging system was essentially much more com-

plex. A good illustration is the most spectacular live TV broadcast in history. On Christmas Eve 1968, a global TV audience watched on their own window to the world (and flight controllers on the right-hand screen in the control room) as the crew of Apollo 8 snapped a photo of Earth with a video camera and read aloud a *Reader's Digest* version of the story of creation from Genesis.[151]

The procedure was tricky, because the astronauts' camera had no viewfinder. Earth had to be captured in (delayed) radio contact with Houston, where the image could be seen but the camera could not be moved. Following some degree of success, astronaut Bill Anders said that he hoped everyone was enjoying the picture "that we are taking of themselves." As if doubting anyone could really grasp the crazy coincidence of extreme distance and simultaneity, the artistic commingling of inside and outside, he added: "You are looking at yourselves at 180,000 miles out in space."[152] No one may have been waving,[153] but this satellite view, produced jointly by Mission Control and the spacecraft, enabled an extraordinary, collective mirror experience, the ultimate in synchronization.[154] At the Mission Control Center, a select group of viewers looked through the glass panel in front of them at the flight controllers, who themselves were looking over the edge of their screened consoles at the right side of the giant display. Like millions of TV viewers, they saw what the Apollo 8 astronauts could see and record of the world from inside the spacecraft – so long as all the signals were transmitted correctly, the spacecraft was properly positioned, and the camera was held correctly with the help of the controllers.

This was real-time operation at its best. The computational real time was overlaid and complemented by combined media and simultaneous communications. The image of Earth appeared on the right side of the giant triptych, coordinated with the flight plan, which was displayed on the left side. It provided evidence of

Figure 12: "Real time" in Houston: the Mission Control Center's view of the television image of the Earth rising behind the moon, which Apollo 8 transmitted to Earth.

the real-time positional information on the center screen and on the small data displays atop the triptych.

As a result, both parts of Mission Control in Houston were redundant – the data center with its battery of mainframes and the control room with its multiple displays and perspectives. Most important, the parts were complementary. For while the data center was expanding its computing capacity, memory, and data flow, the control room focused on preparation, distribution, communications, and recording. This division of labor was constantly changing during the 1960s, for example, by shifting computing capacity (and intelligence) into the spacecraft and by bringing coordination of communications into the computer. Houston had begun to move the computational aspect of space flight as well as monitoring and control into the computer, and with Apollo 8 it

had brought the image of the blue planet as "Spaceship Earth" into the control room.[155] The space center became a model, even an obsession, for people who liked to monitor, control, and command. Most significant in terms of the development of digital space was the concept, demonstrated here in the extreme, of closely synchronized electronic simulation, monitoring, and reporting, because it managed to bring together the world and the computers, space and the spaceship, and the astronauts and the global television audience in Houston's multimedia control room, all at the same time.[156]

# 5 Production and setting up

The political and cultural upheaval of the late 1960s appears to have had little impact on digital space.[157] Programming languages, operating systems, application programs, and data centers turned out to be useful and survived numerous endurance tests, and not only at Houston's space control center. The universal machine, which could be further expanded for special tasks or run with new programs, also proved its worth. The cost–benefit ratio of the computers improved significantly, and time-sharing had been demonstrated with many terminals, users, and programs. Even real-time processing was becoming established, at least as an expectation. Much of what IBM had envisioned at the beginning of the decade had actually been achieved.[158]

Data center programmers and operators now wore their hair somewhat longer and left their ties at home, but that didn't matter much.[159] Inhabitants of digital space were able to negotiate old rules as well as invent new ones. The growth of the industry was almost unstoppable. In 1968, the number of computers in the United States alone is said to have risen to over 70,000. This expansion pleased those selling computers, who still wore ties.[160] But the future also looked inviting for bearded techies in T-shirts. It afforded room for ever more, possibly even radical, proposals. The question of how to further arrange digital space could be approached with imagination. The far-reaching technical upgrade underway in companies, universities, and the public sector of industrialized countries permitted broad scope for creative proposals and audacious solutions in a novel mix for practical implementation. It was of course sometimes difficult even for computer professionals to maintain perspective. Nothing was easier

than coming up with new suggestions and new forecasts. These were then presented with élan at the next computer conference, in the *Harvard Business Review,* or a weekly magazine, thus contributing to the future design of digital space.[161] It was a time of unbridled confidence. Somehow, it would be possible to generate workable, generalizable rules for production and installation from the various proposals. The development of the computer industry up to that point gave no reason to think otherwise. In particular, it was easy to extrapolate or to extend growth with a reference to the future, namely, that a new "generation" of computers, programs, and peripherals would soon be coming to the market that would furnish digital space with even more sophistication, lavishness, and comfort.[162]

The tension between exaggerated expectations and apparent success was admittedly thrilling. But it wasn't necessary to strictly differentiate success and failure. Both could be integrated into the rapidly expanding space of computer activities and the preparatory work required. When conventional procedures were moved lock, stock, and barrel (as they often were) to computers, some things did not work. Nevertheless, the move could usually be interpreted positively. Each project's losses and gains went hand in hand. Because one draft followed another, and each plan was quickly followed by another, criteria for evaluating them were in flux. Thus, people were generally correct in talking about the future of digital space, even if not always at the same time and not always for the same reasons. The fact that this ended neither in crisis nor in unemployment was due to the fact that the large and small questions raised by the prospect of computer technology were continuously met with creative rethinking.

## Production

In 1961, IBM drew up a comprehensive plan for data processing products. Quite naturally, this strategy paper took processors as its starting point.[163] They were to form "a single compatible family" regardless of performance class.[164] From the smallest to the largest, the plan was to manufacture all machines according to uniform principles. Together with the expansion options available for each machine, IBM's plan would result in a seamless range of computing power of different sizes. IBM made this computing power describable with the so-called "processor point." Each processor point corresponded to one dollar of monthly leasing income. At the end of 1961, 29 million of these processor points had been installed at customers' sites; by 1965, 79 million points were expected, and by 1970, more than 162 million points.[165]

The section of IBM's strategy paper devoted to software was somewhat less clear-cut. Here, too, the idea was to apply the principle of compatibility and create clarity. The task group envisaged a maximum of three programming environments, a single system for input and output control, and a greatly reduced set of application programs. The entire software offering would be divided into (only) three "configuration classes" and associated with a family identity over all processor types.[166] Consequently, the processor also influenced decisions about software. Insofar as possible, the work of local setup was interpreted as a "natural consequence" of a customer's choice of processor.[167]

In 1964, the decision to prioritize the processor received massive support from IBM's System/360. Intended for all IBM processors, the System/360 operating system constituted a proprietary computational compatibility constraint. It was even hinted at in the advertisements for System/360, which showed processor cabinets and peripherals arrayed in a circle like menhirs at Stone-

henge, waiting to be used. IBM wanted to be well positioned in all directions. "You choose what you need now. You add new components when you need them," the brochure read. The company's promise was that System/360 would solve today's problems and could be extended to solve tomorrow's.[168] The most sophisticated of all operating systems to date made the company's processor-centric strategy operationally future-proof.

Support for the decision to place the processor at the center of the company strategy also came from outside. In 1965, Gordon E. Moore, a chemist and physicist working at Fairchild Semiconductor, had been thinking about the economics of integrated circuits. He noted that the number of functions that could be integrated on a silicon chip had doubled every year since 1959. Moore reasoned that by 1970 the manufacturer's cost for the least expensive chips would be only 10 percent of the 1965 cost, a prediction that became known as Moore's law. While it could not foresee the effect of ever-increasing processing power on the development of computers as a whole,[169] Moore's law – like IBM – nonetheless tied the future to the processor. The "law" radically reduced the complexity of computer development. To be guided by it meant that home computers, digitally monitored automobiles, portable telephones and – provided a suitable display could be found – electronic watches were no longer the mere dreams they were for Moore.[170]

Making the processor the cornerstone of the company's strategy was a far-reaching policy decision on the part of IBM. Thanks to compatible processor lines, any issues encountered by the sales staff and the technicians who had to get the machines up and running could be solved in the same way for all IBM computers. With a single set of rules, computers could be adapted and upgraded to meet specific needs. On the other hand, IBM accepted that the competition could more easily predict what IBM had in mind, where its weaknesses lay, and where, if necessary, the com-

Figure 13: The computer as an all-round solution for a variety of contemporary problems.

petitor's own products could be positioned advantageously.[171] The strategic focus on the processor also meant that much that did not directly concern the processor's architecture and performance was sidelined in terms of attention economics and dealt with by specialists with narrow, if not marginal, interests. Implications were evident in the processors, the networking technology, and the software.

First, when it came to *processors*, competitors could only operate at the lower end of IBM's range. For a long stretch of the 1960s, the performance of a top-class IBM processor could not be surpassed. Competing companies had no choice but to design more basic computers. IBM had regarded this as a less attractive

option because it had been assumed since the early 1950s that computer performance increased as the square of the computer price, and so large, expensive computers provided disproportionately high computing power.[172] Nevertheless, the Digital Equipment Corporation (DEC) had already assembled a first minicomputer from transistorized circuits in 1960 and equipped it with a punched tape reader, light pen, teleprinter, and CRT display. The Programmed Data Processor (PDP) combined logic circuits, which DEC had previously used to make digital controllers, into a computer that offered much lower performance than any IBM machine. The low acquisition cost, short response times, radically simple collection of commands, and direct operability made the DEC devices, which initially were not supposed to be called computers, an extremely successful niche product. Users had to program anyway. To program on an affordable machine without constantly being reminded by a computer center or an IBM operating system of the difference between what was allowed and what was not allowed was quite attractive. In particular, DEC's PDP-8 minicomputer, available since 1965, added a veritable archipelago of tens of thousands of small, independently programmable islands to digital space.[173] Such a development had not been seriously anticipated by IBM strategists in 1961.

Second, IBM's strategy also completely ignored *connections* between computers. In the world of mainframes, terminals were indeed the end of the line. In the IBM computing world, networking was limited to connecting the periphery (memory, magnetic tape stations, punched card readers, printers, monitors, and keyboards) to the center (processor).[174] For large transfers of data or programs, the physical parcel post was used for transporting tapes and punched cards conventionally to the nearest data center. In 1966, an American magnetic tape standard was developed to facilitate the exchange of information, even between computers

from different manufacturers.[175] If, on the other hand, data had to be transmitted electronically, ordinary telephone lines should suffice, and the respective telecommunications providers were expected to see to it themselves.[176]

Third, as it turned out, IBM's uniform *operating system* strategy very quickly ran into difficulties. No sooner had the strategy been introduced than the "uniformity" was lost, because the system had to be constantly expanded and adapted to special customer needs. By the end of the 1960s, there was a rash of announced but not yet delivered versions and delivered but not yet fully functional add-on modules of what had once been touted as an unproblematic all-round solution.[177]

Basic strategic assumptions and other apparent fundamental things previously taken for granted began to unravel at IBM, and among competitors and customers. The situation became complicated, and important new issues arose for hardware and operating systems. A crisis appeared to be looming.[178] But computer professionals had no time for crises. They might rail against the nonconformism of programmers,[179] lament the fatal consequences of unreliable software,[180] and warn that the appetite of operating systems for computing capacity would not end well.[181] This was not a crisis. The processors worked, and with sufficient patience, software could be adapted to the needs of the users. Of all 575 presently known articles published by the Association for Computing Machinery between 1967 and 1973 in which the term *software* appears in the abstract, only 15 also feature the word *crisis* in their abstract. Just one of them refers explicitly to a "software crisis," but puts the term in qualifying quotes. The title of the article, on the other hand – "Machine-Independent Software" – forthrightly announced the topic of particularly intensive research at the end of the 1960s. Machine-independent software was about designing programs in such

a way that they could be transferred from one computer to the next.[182]

For IBM, machine-independent software was a matter of course, but only when it came to IBM machines. The company's processor family was held together by operating systems that all supported IBM programs for that very reason of "independence." The point of the compatibility was to ensure that IBM customers remained IBM customers if they wanted to purchase a more powerful machine. Everything they had produced for their previous machine in terms of code could be transferred to the new machine. At the same time, machine-independent software was a nightmare for IBM. In the 1960s, Watts S. Humphrey was a member of the company's management staff responsible for software development and systems engineering. Humphrey vividly described the uproar at IBM when a competitor positioned its own product line not at the bottom of the processor range, but in the middle of IBM's hardware portfolio. In 1965, the Radio Corporation of America (RCA) announced its RCA Spectra 70. The electronics giant, which had always focused exclusively on equipment, wanted to build a machine on which IBM software could be easily installed.[183]

This was a frontal attack on IBM's hardware franchise and was bound to have serious consequences for software production. At IBM, people only wrote on blackboards, which were erased after each session. Nothing, absolutely nothing, was allowed to leak out. If software and data compatibility were to apply not only to IBM machines, but to all computer models, IBM's entire business model would unravel. For the first time, a competitor had adapted to IBM's fleet of processors not by circumnavigating, but by sailing alongside. RCA's new system would support its 3301, 301, and 501 computer programs "and other systems," according to an RCA promotional brochure. Instructions, formats, and character codes

would be identical to those of IBM's System/360. And as if that didn't tell the whole story, RCA corporate strategists reiterated for the record that RCA customers would soon be able to "put [RCA systems] to work side-by-side with your other systems…You can conserve your heavy programming investment."[184]

Thus, RCA also celebrated the primacy of the processor, but so radically that IBM software became mobile in a most unwelcome way. Even though RCA had made its presence felt in the past with grand promises that remained to some extent unfulfilled, IBM had to reckon with RCA's success.[185] Should application programs be more tightly coupled to machines in the future, perhaps even by cryptographic means, or would they have to be sold separately?[186] The attack on IBM's business model thus transpired via hardware. A spirited debate ensued about what software actually was, and whether it could be leased, licensed, or sold as a separate product. Whether software could be properly evaluated at all was just as unclear as what it should cost. Could future owners of an RCA Spectra 70 machine be asked to pay for the use of IBM software without knowing what loyal IBM customers had paid for it and what they could do with the programs? Somehow, the well-packaged bundle of equipment, code, and service had to be untied, and its contents repackaged as commercially distinct products. In June 1969 IBM announced that it would no longer market software and hardware together.[187]

This "unbundling" decision is considered the birth of the independent software industry. Hundreds of new companies emerged whose sole purpose was to adapt software to fit the specific needs of customers who had already purchased hardware. New priorities and the dynamics of rapid change reconfigured digital space. Ongoing deployments had revealed more diverse requirements than had been foreseen. Furthermore, it was clear that translating everyday interactions into application programs was easier than

trying to implement them using specialized hardware. Unbundling was a veritable gold mine for all who had previously worked with software at a hardware manufacturer and were now seeking their fortune in their own dedicated software company.

Software enjoyed not only a production boom in the early 1970s but also a conceptual change that went much deeper than the question of sales formats. If software was to become more versatile and independent of hardware, then (and primarily) the difference between software and hardware had to be clear. In other words, people had to be able to understand which problems were more likely to be solved by circuits, cables, and (in the case of printers) spinning chains, and which were arguably better solved using flowcharts, command sequences, and programs. The second requirement was to be able to assess both hardware and software performance. This, in turn, required software development to become generalizable and more abstract, perhaps even formally specified. This task was most likely to be accomplished by groups concerned with developing programming languages, such as Working Group 2.1 of the International Federation for Information Processing (IFIP), founded in 1962. IFIP had set itself the task of systematically revising the possible applications of Algol 60, a widely used programming language. The working group was now in the process of developing a precisely defined programming language that could be rigorously evaluated and expanded as required.

To develop a programming language that could be used on different computers, the IFIP working group concentrated on syntax (the form of the code) and tried to reduce the semantics (what the code means) to a concise core language. This approach required many abstract preliminary decisions, whose fitness for purpose was debatable. The range of opinions within the working group under the leadership of Adriaan van Wijngaarden grew larger

and larger, and a final report with concrete results of the body's work became more and more distant. Fundamental contributions from individual members could hardly be reconciled with each other. One proposal followed the next. It was no longer possible to keep track of the working group's output. The revised report on Algol 60 (1962) was succeeded by proposals for a new language (1964) as well as a proposal for an introduction on the goals of the working group (1965) by Peter Naur. Also in 1965, Gerhard Seegmüller presented a proposal for a basis for a report, Niklaus Wirth immediately provided a proposal for the entire report, and van Wijngaarden commented more generally on questions of design and the description of a formal language. Barry J. Mailloux wrote a new design proposal in October 1966, John E. L. Peck a second in May 1967, followed by a third later that year. A penultimate and a final draft report appeared in 1968. But even before the Algol 68 group could adopt the final report, some of the most radical members resigned in protest and published a minority report.[188]

Computer historians have incorporated these events into the discourse on the "software crisis"[189] and overlooked the fact that the Algol 68 proponents were hardly pursuing different goals than those who opposed it. Both camps were concerned with formalization, assessment, and the creation of rules. Such abstract goals produce many and far-reaching conditions. The mutual accusations between the working group and its dissidents ranged from excessive formalism to negligent pragmatism, from inefficient obsession with detail to logical inconsistency and little relevance to practice. The official report nonetheless thanked the leavers somewhat understatedly for "whole-hearted cooperation, support, interest," and even "violent objections." The minority report, on the other hand, complained of obvious failure, mannerisms, inaccessibility, "grave" deficiencies, and the uselessness of Algol 68 as a programming tool.[190] The goal of the former, to develop a

tool that could be used to write complex programs in an elegant way, was confusingly similar to the goal of the latter, to develop a programming language that was as clean as possible and easy to evaluate.[191] In both cases, the question was how to free programming both from its artisanal origins and from electrotechnical specifications. Indeed, the controversy over Algol 68 could be seen as a hardly surprising sign of growing pains as "computer science" began to emerge from the fraught nexus of applied mathematics, electrical engineering, and physics.[192]

In 1972, the minority report's spokesman, Edsger Dijkstra, gave a Turing Award speech desingenously titled "The Humble Programmer."[193] He began by recounting a conversation he had as a young student in Amsterdam in 1955 with Adriaan van Wijngaarden, the later *spiritus rector* of the Algol 68 Working Group. According to Dijkstra, the conversation was about whether programming was a respectable profession and where the respectable knowledge was to be found that could support the intellectually demanding work of programmers. After listening for a long time, van Wijngaarden asked Dijkstra whether he himself would not like to be among those called to make programming a respectable discipline.[194] Anecdotes are no less adaptable than software. Both can be applied to suit the situation. In the case of Dijkstra, the Algol 68 rebel, the anecdote served as a hagiographic validation of what self-confident computer scientists in the early 1970s had just begun to promote using the redundant term *structured programming* and the oxymoron *software engineering*.[195]

## Setup

The development of digital space in the 1970s and ideas about how this development could be generalized in a controlled manner broke with the primacy of the processor, which IBM had opted for in 1961. The range of hardware and software became more differentiated, while the rules became more abstract and the procedures more generic. That made it possible to promise ever more with regard to machines and programs and to "anticipate" demand. However, it also became apparent that substantial setup was required to bring such a proposal to the point where it even came close to meeting customers' needs and where it made sense for them to buy a computer, although they'd still have to wait for the next software release or for an additional device.

The following sections examine this work of setting up in the context of three relocation projects. All three projects originated in German-speaking countries of the early 1970s. It would be easy to find a similar triad in, for example, Spanish- or French-speaking countries. More important is to mention the project goals and scope of action: one project pursued a mundane goal in tourism, the second an ambitious one in global banking, and the third a controversial one in national policing. To structure a new field of action in digital space, all three projects had to translate a local situation into writable software and installable hardware, or adapt available software and hardware to a specific situation, and vice versa. Consultants, programmers, managers, engineers, insiders, and experts were all required for the massive preparatory effort.

The first project had a very clear starting point. In the fall of 1968, the city and spa administration of Bad Wörishofen, Bavaria, commissioned the Institute for Management Consulting and Development Planning and the Geographical Institute of the Technical University of Munich to prepare a municipal development

plan. The project envisaged reorganizing the "management style" in tourism management and, for this purpose, setting up a "spa information system" for "60,000 spa guests and 1.2 million overnight stays in 260 spa establishments" every year.[196]

The corporate planning experts were not concerned with the question of how the spa's water treatment (developed by naturopath Sebastian Kneipp) should be applied, nor with the question of how people – whether healthy or sick – should live.[197] Rather, as reported by Bernd Bienek and Volker Kreibich in the *IBM-Nachrichten* of 1970, the "planning, implementation and control of a long-term development concept" for the Kneipp spa made it "imperative that a comprehensive information system be established." For, under the conditions of increasingly differentiated tourism markets, "conventional management" of operations through "decisions based primarily on experience and sentiment" was failing.[198]

To respond to a rapidly growing and increasingly diverse clientele, an information system was needed to analyze their needs, an expanded product line to treat them, and a new way of making decisions to guide operations.[199] According to the experts, however, sufficiently specific management information could only be provided with the help of electronic data processing, and an information system would simultaneously help to rationalize various administrative processes. The project was not described in precisely these terms. It was simply based on the consensual assumption that the clients' requirements, travel motives, and patterns of consumption had changed considerably, and that people therefore expected a more customized offering.[200]

The rationale for moving the spa's traditional administration into the electronic space of actionable information for management and marketing was thus established. But how was the move to be accomplished? In the summer of 1969, an "analysis of the

previous guest registration system" and the "spa statistics and spa tax accounting based on it" was conducted. Data from the official registration form that guests filled out on arriving at the hotel and information from each service was forwarded to the spa administration. Analytically, procedurally, and argumentatively, this was a perfect starting point for operationalizing a new information system. Forms also reduce complexity, clarify relationships, and structure procedures. They were the substrate of the spa administration's previous analog information system and easy to translate into digital procedures.

The management consultants proceeded step by step. By tracing the path of the completed registration form through the administration's processing mills, they simultaneously generated a list of weak links, a catalog of possibilities for expansion, and a structured translation program. The description of the "traditional information system" became an assessment tool for the old program and a reference for the new one. Anything in the previous reporting system that indicated inefficiency or poor performance was catalogued. That included the frequent need to copy the content of the forms. A municipal official messenger's need to carry forms through the small town and guest departures indicated by "scratching out" in the annual gathering of information for the state statistics were noted with particular relish. Bienek and Kreibich were likewise taken with the crude distinction made in the guest statistics between "domestic guests" and "foreign guests."[201]

Moving the Bad Wörishofen spa administration into digital space was not a revolutionary act. "The most important basis for this system is still the registration form [*Meldeschein*]," stated the 1970 report, adding that it had been made more meaningful by the addition of a short questionnaire and could be used at the same time as a "punch form [*Ablochbeleg*]." The decisive factor was that the registration slip was now prepared for transfer to a "guest

master punched card" and that all data could be transferred to punched cards by the administration with the help of the IBM 29 card puncher.[202]

Only now did things start to simplify, namely, with regard to spa tax accounting and spa card administration. Hoteliers probably had mixed feelings about the progress made. The delivery of spa cards and "the daily collection of the spa tax by the municipal messenger" were definitely things of the past. In addition, the new procedure had the invaluable advantage that the spa administration could set up a monthly collection of fees "by direct debit from the tourist accommodation's account." The old registration form from Bad Wörishofen had found its way into the Bavarian banking system after being translated into an extended guest master card – and to a computer in the state capital. Once a month, a System/360 machine in the IBM computer center in Munich generated all the lists needed to calculate the spa tax. These lists were sent to the tourist accommodations and read as itemized accounts.[203] At the same time, the computer center provided monthly operating statistics on hotel guests, sorted by gender, age, birthplace, group structure, frequency of visits, means of transportation, reason for stay, and occupation. Even the costs for calculating the service statistics were shown "and debited when due" from the service's account. Spa guests hardly needed to do anything different than they did in the past. But the spa's registration system was turned upside-down organizationally: collection of the spa tax was automated through cashless payment transactions; guest data were recorded on punched cards, monthly operating statistics were compiled, and the costs of the computing time incurred were distributed among the services. The debit slips and bank lists, like the invoices and calculations, were generated by the computer in Munich.

In this project, the original aims of development and rationalization were to some extent pushed to the background. The

main advantage of the computer solution, as Bienek and Kreibich wrote in the *IBM-Nachrichten*, was "not in financial savings, but in information gain." After all, the system was expandable, especially since there was now an IBM System/360-20 – IBM's answer to DEC's successful minicomputers – in neighboring Kaufbeuren. By way of reassurance, the experts informed their clients that they would soon be drawing a random sample from the guest punched card index and conducting detailed interviews with 2,000 guests. For this purpose, the computer supplied "a list of target persons, punched cards with names and addresses," and punched cards "containing all the data from guest cards along with short questionnaires for monitoring the interviewers." The spa guest survey had become part of the overall information system of the tourist community. The system could now be continuously expanded, for example, to include an electronic accommodation service or to develop a guest address card index. One even dared speculate how tourism might develop over the long term. Which brought the project back to the original objectives after all.[204]

The second example of a large-scale project setup in digital space had goals similar in the abstract to those of the project in Bad Wörishofen, yet was infinitely more ambitious. In 1969 the Union Bank of Switzerland (SBG) decided to put its information system on a completely new footing. This seemed the obvious thing to do for a bank that covered all facets of the financial business, that was growing rapidly, and that wished to support an increasingly diverse clientele and an ever broader range of businesses. SBG had been gaining experience with computers for almost 10 years. Its coffers were full to bursting, and the financial sector was booming. Loans, stock exchange transactions, mortgage administration, foreign exchange trading, and international payment transactions for companies and private customers of all sizes were now to be handled in digital space. That meant signifi-

cant amounts of money and personnel, many customers, and an overwhelming number of transactions.

In the world of banking, as in the world of space travel, much of what goes on is time-critical.[205] Union Bank Information System Concept (Ubisco) was the name given to the project that would move the bank away from time-delayed batch processing to a real-time online system. Using fully integrated data processing in real-time mode to be implemented in two project stages, the bank wished to record all forms of its business. The technologies to be employed also happened to be among the finest available for spaceflight, the military, and sophisticated research.[206]

Initial trials with selected transactions on brand-new UNIVAC 494 machines had shown promising results in 1969 and 1970. Taking a look at what the competition was doing also provided SBG management with strong support for its project.[207] SBG knew how computers could be used for banking and was confident that it could share this knowledge with the top-class computer manufacturer Control Data Corporation (CDC) in a joint project.[208] CDC specialized in scientific computing and was working for a major bank for the first time. SBG tried to minimize the obvious risks through contractual provisions, but the project failed nevertheless. Between 1974 and 1980, Ubisco existed exclusively in law firms and the courts.[209]

Exactly what caused Ubisco to fail is not easy to determine. The dispute between the project partners ended in an out-of-court settlement. One problem was the transaction-oriented operating system (TOOS). TOOS was supposed to operate in connection with a database. In addition to the usual operating system tasks, it also contained a communication handler, a database manager, a transaction manager, and a journaling system to log all operations. CDC began to build this edifice on the latest in-house operating system.[210] However, in the middle of the development work, CDC

decided to fall back on an older, proven operating system that was also suitable for real-time computing. TOOS now had to be rewritten, which not only meant a huge amount of work but also resulted in structural limitations. CDC had obviously erred in its choice of operating system.[211]

Another probable contributor to Ubisco's failure was the huge work of translation into the application programs. These were developed by the bank itself owing to the sophisticated nature of bank-specific procedures. However, the bank quickly fell behind schedule. And no wonder. Building and operating a database that could integrate all applications and keep a huge number of simultaneous transactions up to date with an extensive terminal network in real-time mode was unprecedented in its complexity. There was a clear list of priorities, but there was no overview. The database was undisputedly in first place. But right after that came the application programs. Their task was to translate the different categories of transactions carried out by the bank (such as customer payments, customer authorizations, securities management, and teller transactions) into digital form. Third on the list of priorities were the much more specific operations, such as customer accounts, accounting/controlling, security transactions, stock exchange transactions, management information, and a catch-all category of "new projects."[212] But in the end, everything was interrelated, and significant delays in computerized transactions were not permitted.

As a byproduct of their approach and their far-reaching claims and promises, the Ubisco project partners ended up with a massive time problem and a degree of complexity that were their undoing. As in other ambitious computer projects, it proved impossible to scale up tasks that had been successfully completed on a small scale.[213] SBG tried to solve the problem by adding personnel, but as the level of detail increased, additional staff became

more uncomfortable with the problem. CDC, on the other hand, seemed to be taking advantage of IBM's experience in developing System/360, played for time, and even shifted personnel to other, more promising projects.[214] The bank accused their project partner of breach of contract and was able to demonstrate clear functional deficiencies in CDC's software interface bridging the hardware and the operating system. For the time being, SBG was able to keep delays in its own application program development a secret. Sooner or later, however, the obfuscation was bound to stymie productive cooperation. By 1974 it was no longer possible even to agree on a neutral procedure to assess the situation.[215]

Like many other large projects, Ubisco showed that there were no shortcuts to digital space. The project outcome might have been different had the effort been properly assessed from the start. But the experience of other banks in moving their transactions into digital space points in a different direction: comprehensive rethinking of the banking business from the ground up was required in the 1970s to move it into digital space. To do this, the computer had to be seen as more than a tool for speeding up the processing of (highly complex) information. The Ubisco project centralized the problem of information processing in a radical way and wished to turn the branches of SBG into organizational terminals. In contrast, the Schweizerische Kreditanstalt – forerunner to Credit Suisse and SBG's biggest competitor – was quick to recognize that restructuring the organization of the bank was necessary to conduct its business in digital space and vice versa.[216] At the Swiss Bank Corporation (SBC), too, SBG's particularly nimble competitor bank in Basel, preparations for a real-time banking system assumed from the outset that moving the bank into digital space would require a new organizational structure. Even before debating which IT equipment to work with, SBC's general management was discussing new functional models for its various

work areas. Together with the heads of the major branches, the general management defined a long-term development concept covering computer selection, cost–effectiveness, staff training, security, and project organization.[217]

These three examples of the complexities of setting up in a digital space conclude with the relocation of the Federal Criminal Police Office (Bundeskriminalamt, BKA) in Wiesbaden to digital space. As with Bad Wörishofen, the BKA was concerned with translating registration forms into digital format; and as with Ubisco, the stakes were at the limit of what was possible in terms of IT. The basic issue was inevitably how to respond appropriately to rapid growth. The goal of computerized policing, however, required such a complex learning process for all involved that it was nearly impossible to speak simply of success or failure. As it developed, the project changed from an adequate response to contemporary policing problems into an inexhaustible source of political controversy. By the end of the 1970s, the relocation of policing into the computer became a hallmark of West Germany's self-image, epitomized by the term *dragnet investigation*.[218]

At the end of October 1969, newly elected chancellor Willy Brandt promised to launch an immediate program to intensify and modernize the fight against crime. It was no longer possible to ignore rising crime rates, insecurity evident in public opinion surveys, and endless discussion of a crisis in policing. The impression of poor coordination, a chronic shortage of security personnel, and inadequate equipment had taken hold of public opinion over the course of the 1960s. For this reason alone, Brandt's emergency program found broad acceptance. Moreover, Eduard Zimmermann's TV program *Aktenzeichen XY ungelöst* (Case number XY unsolved), which aired from 1967 onward, illustrated the appetite of the nation's viewing public for crime. Feature film sequences, witness testimonies, crime scene evidence and expert opinions,

and feedback from the audience collected by telecommunication made for successful television as well as successful government. While broadcaster ZDF's crime entertainment show stimulated the audience's brains with terrifying scenes on dark forest roads and networked them into one big search engine, the technocratically oriented federal government designated the BKA as a house of horror – namely, a place where overcrowded filing cabinets and millions of index cards could hardly be searched successfully. The BKA in Wiesbaden became the epitome of the organizational inefficiency and technological backwardness of the entire German police force. Indeed, several weeks of rummaging through files and index cards of the BKA were frequently required before police could secure access. The effort to automate, centralize, and process standardized information was intended to speed the work and transform the dusty letterbox authority into a leading producer of internal security. For this reason, the Federal Ministry of the Interior wished to ensure that the BKA would "have its own data processing system by the end of 1972."[219]

The project goal was set and the necessary funds were released. Nevertheless, the relocation work was slow to get off the ground. The BKA had concentrated on the complex criminal intelligence service. This meant all the subtle ramifications of collaboration between the federal and state governments had to be accommodated. To speed up the process, the Ministry of the Interior now handed management of the project over to external experts, namely, Eduard Zimmermann, Horst Herold, and the Kienbaum company (Germany's first consultancy). As the popular host of *Aktenzeichen XY ungelöst*, Zimmerman in particular (whose character was known as Ganoven-Ede) was expected to facilitate the process through his authoritative and conservative image. However, he did not have an especially good grasp of either computer technology or the organizational intricacies of policing. The technocratic

faction of the reform commission was supported more competently and effectively by Nuremberg police chief Horst Herold and by Kienbaum. Herold had been expounding for years and in numerous publications on the advantages of computerizing the police, how he wished to control the deployment of police forces using "criminal geographic analyses," and the preventive effect of a police presence that could anticipate criminal activities.[220] When the consultants presented their report on the future BKA project strategy in February 1972, the plans for a computer-based regime could be approached anew. Especially since Horst Herold had in the meantime been appointed the new president of the BKA and received much advance praise from the press.[221]

Herold and his team concentrated on the most problematic part of the investigation system and restructured the approach to investigating persons and property. Forensic applications would be translated into digital form only in a second step. The effort put into transforming the BKA was unprecedented. The reporting system was upgraded to interactive terminal connections, and the agency's personnel and equipment were expanded. Siemens installed two 4004/150 mainframe computers (an RCA-inspired answer to IBM's System/360) in Wiesbaden in just under 10 months and distributed 35 terminals to security-critical points in West Germany. From then on, data could be entered and retrieved at these stations. In October 1972, a spectacular presentation of the capabilities of the new "Inpol" search system was made to the press at Frankfurt Airport. The old-style manhunt had clearly had its day.

Only six months later, an amendment to the BKA law expanded the powers of the criminal investigators in Wiesbaden. The BKA became the central office for the data network between the federal and state governments and was now in charge of international cooperation and organized crime. Within a very

short time, the number of terminals for Inpol queries and entries connected via the telex network of the Bundespost (federal post office) was increased to about 600.[222] In early 1975, 160,000 persons were registered in the search system. The centralization of data with decentralized access, which Herold had long championed, and even described as a "fundamental democratization" of the police, had been successfully implemented. After members of the Red Army Faction (RAF) were arrested in Stammheim, the BKA jubilantly proclaimed the restoration of internal security in West Germany. The digital search system was not used in the arrest of Andreas Baader, Gudrun Ensslin, Jan-Carl Raspe, and Ulrike Meinhof. Nevertheless, the relocation of police investigations to the computer was treated as a success, fulfilling the promise talked up by the German news magazine *Der Spiegel* in 1972 when Herold was appointed president of the BKA.[223]

The BKA continued to expand its digital search power. In addition to shifting forensic procedures into digital space, it began to focus very specifically on the problem of the conspiratorial behavior of terrorist organizations. As early as 1974, the BKA had begun systematically registering prisoners' visitors as part of its prisoner surveillance program. At the same time, a file was set up for an "observational manhunt," which was used "as a precautionary measure" to record the movements of persons with prisoner contacts. This qualitative shift to "preventive manhunts" was linked to preparations for the major trial of the RAF. To combat terrorism, tips from the population were amassed in a computerized database on "persons, institutions, objects, and things" (PIOS). This was in line with Herold's conviction that factual evidence was the only remedy against subjectively colored, unreliable witness statements. With PIOS, even extremely complex contexts could be searched in a "multi-dimensional" way. This, in turn, generated information relevant to a manhunt that would have been

impossible relying on the conventional files for vehicle or person searches. The BKA was fighting the threat of a "data gap" between the uniform Inpol database and the untapped content of criminal intelligence files. "Crime-relevant shards of information" remained hidden in a "dark field." That, Herold later explained, was why "the PIOS register of found objects" was created, "which, with five file columns, was akin to a rudimentary index."[224]

A few years later, the modernization and intensification program for crime fighting throughout West Germany was viewed differently, mainly due to the growing importance of terrorist activities. At the end of 1974, Günter von Drenkmann was shot during a kidnapping attempt by the RAF; in February 1975, Peter Lorenz, the top candidate for the Christian Democratic Union in Berlin, was kidnapped; in April, the German embassy in Stockholm was occupied; in May, federal prosecutor general Siegfried Buback was shot; and in July, banker Jürgen Ponto was shot. In the fall of 1977, Hanns Martin Schleyer, the president of the employers' association, was kidnapped and murdered by the RAF.

The BKA applied its computerized investigative power to all of these cases, increasing its effort but decreasing the success rate. Inpol queries provided the names of the occupiers in Stockholm, but the authorities knew only too well where to find them. The vehicles used in the Peter Lorenz kidnapping were quickly identified, but proved to be decoys that did not advance the investigation, and the "people's prison" where Schleyer had been held for weeks was not located until after his murder owing to a communications problem. "Sloppiness, missing documents, a tangle of jurisdictions" had characterized the Schleyer manhunt, *Der Spiegel* commented after an initial official analysis of BKA procedures.[225] The project to intensify and modernize the fight against crime had gone awry at its most prominent location, the BKA, which lacked little in terms of computer technology.

Arguments against the use of computers in the BKA were strangely restrained for the time being. Criticism was directed primarily at the agency's top brass and the incompetence of its staff. Writing in *Das Kursbuch* and *Der Spiegel*, noted author and editor Hans Enzensberger lambasted what he called "Dr. Herold's Empire of the Sun,"[226] but even he did not criticize the failure of the computers, only the incompetence of the team operating them. Computers obviously enjoyed the privileges of a neutral procedural authority. They were considered a somewhat civilized form of police power, especially compared with the similarly unsuccessful large-scale raids against the RAF.

And yet the use of computers was extremely problematic, both legally and technically. PIOS in particular marked the conspicuous step from computer-based information *processing* of criminal investigations to computer-based information *production*. Whereas processing was based merely on data collected and maintained on the basis of prior suspicion, the production of information for a criminal investigation required data without known relevance to be generated. For that, additional data acquired without solid initial suspicion were required. Initially, the legal justification for negative dragnet searches was very subtle. There was no way of legitimizing the qualitative transition from processing information to producing it.[227] Any police force that stored data on unsuspected persons as a means of filtering information, and which also used other official data sources for this purpose, was on shaky ground. This was the case even when requests for administrative assistance were made correctly and the production of information was primarily done by filtering and deleting data.[228]

The massive project of shifting federal police work into the digital realm therefore fell into disrepute, though only after some time. By 1979, 4.7 million personal names and several thousand organizations had been recorded in the BKA's computing center.

This information was increasingly interpreted as mortgage due on the surveillance state.[229] The technocratic optimism of the Brandt government in 1969, which had the broad support of the public, had evaporated. It had given way to the pessimistic assessment that electronically produced security can also threaten the democratic order. "Feeding the computer" not only required a huge bureaucracy of civil servants but also generated an enormous wave of distrust that engulfed the nation.[230]

The three projects described here for relocating to the computer a municipal spa administration, a globally active major bank, and a German police authority are naturally very different. Their similarities, however, are no less interesting in terms of history of technology. Handling growth and the need to be able to respond to varied demands or tasks were important motives for moving previous administrative, business, and investigative procedures into digital space. All three projects aimed at setting up an information system and were sooner or later confronted with the fact that such systems must not only process information but also produce it. The projects also had in common the radical reduction of complexity to the smallest possible, flexible, and at the same time robust information units that can be handled by computers. In Bad Wörishofen, these were the "overnight stay" or "guest," for SBG they were the "transaction" or "business," and for the BKA they were the "clue" or "track." Each project required an enormous amount of setup work to adapt the computer to the project goals and the organization to the command structure of the computer. Successful adaptation could often only be achieved through increased abstraction, rethinking the task, or unconventional solutions. Structurally, the projects brought about changes that were to become the new normal in the future: they changed operations and their speed, and they created fast zones and new waiting rooms. Everyone involved or affected had to learn to de-

sign new structures as well as to apply them, that is, to make them operational. In the digital world, as in sports, after the game is always before the game.

# 6 Connecting, differentiating, and storing

Computers are an ensemble of signs and electronic building blocks that constantly interact during operation. They manage the connections between their components and, in time-sharing mode, they strengthen the culture of connectivity because their users (while sitting at terminals) are constantly communicating with the computing center and its interconnected devices.[231]

As computer centers expanded their processing capacity and their program libraries, they became attractive to a growing community of users with a very different set of issues. It became increasingly rare for the desks of such users to be located in close proximity to the computer. As distance between the terminals and the computer centers increased, the connection problem had to be solved with the help of the telephone companies. At the same time, computer developers and operators were contemplating how to expand available programs and capacities by accessing other computer installations. They even tried to make use of minicomputers – typically located at the system periphery with users – for smaller tasks or for the preparation of large jobs. This, too, could sometimes only be accomplished with connections that were not controlled by the data centers themselves. Moreover, the connection problem became even more complex when interconnecting external computers and external networks.

Even at the level of the elementary components of computers, there was an issue not only of connection, but also disconnection.[232] Switches, relays, and transistors establish connections but can also interrupt them. Indeed, the time-sharing operating systems of the 1960s controlled the simultaneous access of several users to scarce computing time by interrupts. The interrupt

regulated internal traffic in digital space: processes could only be routed past and separated from each other if one process was temporarily interrupted and the computing capacity thus freed was made available to a second process. Ten years later, it had become necessary to develop procedures not only for sharing computing time, but also for sharing scarce transmission capacities. This was done to contain the relative increase in transmission costs associated with "long-distance data transmission." Consequently, computer-based telecommunications technologies began to regulate long-distance traffic in digital space. They relied on packet switching to send clearly defined, uniformly delimited and addressed data packets over lines that were needed by several users at the same time.

By the late 1960s, developer attention had been shifting away from processors and toward software. In the 1970s, it shifted to the user and to connectivity. Anything that supported individual or group autonomy was attractive. Since the 1980s, emblematic of this trend, the "personal computer" restructured digital space in a nuanced way. It offered smart local computing capacity, which greatly enhanced users' level of freedom.

The personal computer was a computer for personnel and for the individual. People who worked "at the PC" were bringing their own world into the computer. The PC highlighted questions of relative user autonomy regarding questions of connectivity and defining digital space. The same applied to problems of data organization and storage in digital space, which intensified dramatically over the course of the 1970s and 1980s.

## Connections

In December 1969, the Association for Computing Machinery invited its members to a major meeting in New York on "Computers and Crisis" to explore the question of how computers would change the future. One panel was devoted to anticipated challenges of data transmission. The panelists discussed the question of how communication – which is the basis of society and holds it together – could be relocated in the computer.[233]

John M. Richardson of the US Department of Commerce kicked off the discussion with a lecture on "computation, communication, and content," enthusiastically describing this trinity as a "pregnant union" for the future. Computers, Richardson said, played an important role in monitoring networks, connecting telephone exchanges, and billing. Telecommunications, however, offered additional computing resources, which was something new. What had hitherto appeared as a rather loose coupling of two technologies had already given rise to "new data processing markets" and would, Richardson predicted, lead to a visibly close connection in the foreseeable future.[234] In this context, Richardson referred to an "information technology" resulting from the technological "pregnancy" of computing and transmission.[235] While transmission was already providing for additional users of computers in time-sharing mode, he predicted, the closer linkage of computer and transmission technology would not only expand imputed capacity but also generate a greater variety of content and information.[236]

To General Lee M. Paschall of the US Air Force it was also a foregone conclusion that the future evolution of digital space was toward time-sharing and data exchange. As a communications specialist, Paschall found it easy to envision a network linking many mini-computers and served by a database. But those approach-

ing the connectivity issue from the perspective of the computer might prefer to think of a very large computer with many terminals. Paschall's take-home message was that while the connectivity problem might be conceptualized either in terms of computers or their interactions, it could not be eliminated; that much was certain. Probably, over time, networks would emerge that would link centralized and distributed structures. This would, however, require merging very heterogeneous subsystems or replacing them incrementally, both of which posed major challenges.[237]

While the Air Force operated 1,200 computers, only the air surveillance machines were interconnected, Paschall said.[238] Not much progress had yet been made in this regard, although "command, control, management, surveillance, reconnaissance, and communications" were actually closely intertwined. But there was no direct "cross-fertilization" between the management of computers and that of telecommunications networks, said Paschall, extending the reproduction metaphor. For that, he noted skeptically, the systems had been developed with little attention to system design principles. In other words, there was a lack of viable standards for coupling telecommunications and computer technology.[239]

The ACM panel consensus was that computer-based companies and administrations could only grow if they expanded both their computing and telecommunications infrastructures. Dodging the increasing pressure to connect was not an option. Moreover, organizations were, in any case, using their central computing facilities to expand the controlling power of the head office. This could only work if ever larger portions of the organization were integrated into the system. The links between the branches and the computer center at headquarters had to be strengthened. Computers were moving almost inevitably closer to communications technology. They had to translate local computing into com-

puting at a distance if the periphery of an organization was to be linked to its computing centers.

Around 1969, however, the existing hardware stood in the way of a quasi-natural transition from mutual usefulness between computer and communications technologies to their symbiotic interaction.[240] Computers of different years of manufacture or from different manufacturers were already compatible only to a limited extent on-site. How, then, were they to be interconnected in any combination and over long distances in order to create something one might call an "information technology" system?[241] Since uniform standards were inadequate, the task was to somehow upgrade them. This could only succeed in the near future by approaching the problem in such a way that new rules and standards did not necessarily conflict with old ones. For the 1969 ACM conference participants, that meant shifting their attention away from machines and programs and toward users and their connectivity.

"Everyone talks about the computer user, but virtually no one has studied him in a systematic, scientific manner," Harold Sackman of the System Development Corporation in Santa Monica had noted a year earlier.[242] It was impossible to transform users into predictable and understandable – indeed, unified and well-integrated – components of a computerized information system. Computer engineers produced an unexpectedly diverse collection of species, even as they moved the "user" as far away as possible from the machine, by designating an "end user." At the 1969 ACM conference in New York, more than two dozen speakers sat on panels addressing the end user. Another dozen papers were cobbled together for sessions on "professions," which (unsurprisingly) discussed professional users. The conference served to produce a sizable catalog of the newly discovered creature called "user" in all its varied present and future manifestations.[243]

In view of this diversity of species, the "(end)user" could only be dealt with by transferring him into a new abstraction. How this was to be done was laid out in an article published under the title "The Computer as a Communication Device" by J. C. R. Licklider and Robert W. Taylor, with the editorial help of Evan Herbert.[244] Licklider and Taylor belonged to a small community of engineers and scientists who were paid by the US Department of Defense to conceive new uses for computing.[245] It was part of their core business to promote the future of computers again and again and in very different contexts, including in experiments, at conferences, and in anthologies, journals, and memorandums. And if – as with the article by Licklider, Taylor, and Herbert – some of the ideas got waylaid in a marginal publication, they could certainly be used again in the next paper. Or they flowed into projects that the community initiated, supervised, and carried out or supported as cheerleaders. Licklider had shown the same inclinations as program manager at the Advanced Research Projects Agency (ARPA), which was founded after the Sputnik shock.[246] For ARPA, the future use of computers encompassed all kinds of things. It was hardly ever about nuclear war, and never about the Cold War. What was of interest was interaction, exchange, communication, cooperation, and symbiosis. In 1968, ARPA was focused on computer-based collaborative projects and particularly on users.[247] In each project, the participants were expected to develop a shared idea of whatever it was they were working on. Therefore, project work amounted to the reconciliation of individual ideas. It was largely a reconciliation of differences between different mental models.[248] Computers could play a supporting role in this process. The members of a spatially widely dispersed working group, for example, who had different styles of thinking and heterogeneous expertise, could – so the idea went – make their own mental models comprehensible to each other by supporting their formulations at all

times with data, programs, documents, and simulations. As soon as the basis for everyone's mental model was clear to all participants, essential commonalities within the working group could be articulated. The simplified gestalt conceptualization regarded computers as aids to collaboration and cooperation, regardless of where the computer was located.[249]

This meant, first, that the heterogeneity of the actual machinery was hidden. Second, the phenomenological diversity of users was reduced to the ideal "user," who was thought of as a communications-oriented creature closely connected to the computer. Finally, connectivity, which had received scant attention from computer engineers, remained unsolved. In well-equipped computing centers, cables were usually invisibly tucked away. Whereas in the early days computers were still programmed by plugging cable connections into the control panel, now everything that served to connect components was shoved either to the back of the computer or under a false floor. At IBM, a rumor circulated that the thick "boa" cables lurking in cable ducts (made in the United States by the Anaconda Copper Company) could not exceed 200 feet in length because otherwise they got dangerous.[250] Cables leading out of the data centers were hardly noticed. Robert M. Fano, who headed a large time-sharing project at MIT, stated succinctly in his 1967 report titled "The Computer Utility and the Community" that terminals could interact with computers "through existing communication facilities." He neither knew nor said anything more specific on the subject.[251]

Computer engineers had marginalized the transmission problem because they considered wires and cables to be a trivial matter and because in any event they did not wish to deal with the particularly old-fashioned field of low-current technology known as telephony.[252] Electrical engineers who had found their way into the development departments of telephone companies

were pitied because they had to deal with electromechanical relays or, at best, with directional radio antennas.[253] Unless, that is, they were looking for ways to transfer conventional connectivity and transmission problems to the computer in which case they might be considered computer engineers in the realm of telephone engineers.[254] It was here that computerized transmission and switching got exciting. In the late 1960s, pulse-code modulation, time-division multiplexing, and packet switching became high-value terms among these computer-savvy transmission and switching specialists. At IBM, UNIVAC, CDC, and Honeywell, however, no serious thought was given at the time to problems of interconnectivity, and the telecommunications companies had not yet recognized data transmission as a promising field of business.[255]

It was not until the 1970s that a decisive change in patterns of perception occurred. Telecommunications companies were no longer exclusively concerned with the possible shift of switching telephone calls to computers. Rather, they began to think about using their telephone lines to connect computers and to develop inexpensive forms of data transmission. At the same time, computer engineers concluded that not only computers and programs but also connectivity could become a critical factor in the industry's growth.[256]

As a result, centralized computers were surrounded by a confusing patchwork of user communities supported by telecommunications.[257] Companies built exclusive connections between their computers at headquarters and users at branch offices, while information services sought a way to offer temporary connections to their databases despite prohibitively high telephone charges. The business models varied widely and could even be combined, as the history of CompuServe shows. When the company was founded in Columbus, Ohio, in 1969, the initial purpose was to

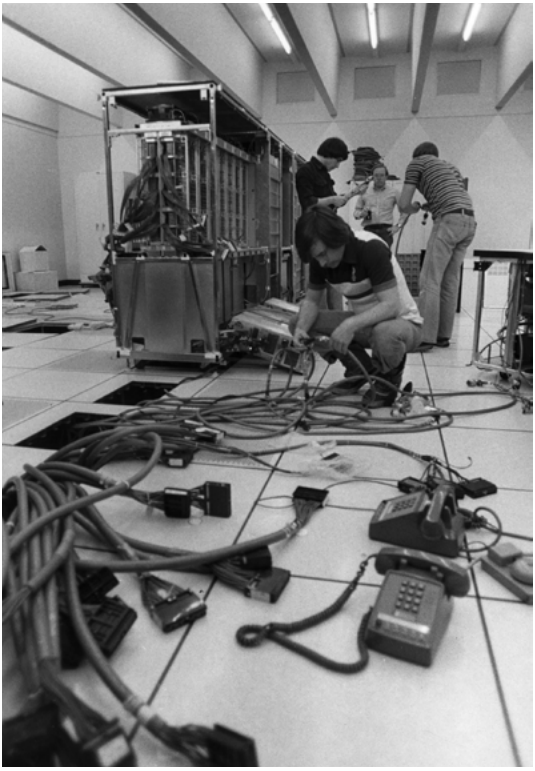

Figure 14: When computers were moved and reinstalled, as at the University of Michigan in 1980, the perpetual battle against cables became apparent.

provide time-sharing services for a life insurance company. The company also aimed to rent spare computing capacity to third parties who wished to analyze their own data using either applications they had written themselves or off-the-shelf software.[258] These supplemental customers relied on the public telephone network with its high long-distance rates to connect computers. To lower connection costs to reasonable levels, CompuServe de-

veloped its own switching services in various cities that ran over minicomputers (PDPs). Users could now connect their computer or terminal via their telephone line at local rates to a CompuServe switching computer, which passed the connection request on to another switching computer in Columbus, via a very busy dedicated CompuServe line used by several connections. This computer in turn put the request through to the computer that was actually supposed to handle it.

The example of CompuServe shows that in the 1970s, business models and types of use for connecting to and between computers were combined in different ways. Both communicative and computational tasks were delegated to computers at CompuServe; computers were used to operate a company network *and* a customer network, and to organize data processing transactions *and* to provide software. The result was an extremely complex network that served diverse demands and combined very different connection technologies. A similar trend was evident at General Electric Time Sharing Systems, whose supercenters expanded globally as early as the mid-1970s, making the company the largest provider of computing power in the world.[259]

In addition to service companies in the data processing business, start-up companies sought to make their fortunes by offering specialized information, for example, literature searches for scientists and engineers (Dialog) or case law collections for lawyers (Lexis). In addition, companies such as Telenet and Tymnet offered services exclusively for the purpose of transmitting information between computers. Electronic Data Interchange (EDI), in turn, offered a collection of standards and a series of networks through which different companies within an industry could exchange data worldwide, such as ORDERNET for the pharmaceutical industry or IVANS for insurance companies. In the early 1980s, ODETTE for automobile manufacturers, RINET for reinsur-

ance companies, SHIPNET for transport companies, and EDICON for the construction industry were added.[260] The SITA network of European airlines, which operated nine data centers with 1,200 employees as early as 1970 and moved the formerly telegraphic message service to the computer, was seen as a significant development.[261] Finally, the Society for Worldwide Interbank Financial Telecommunication (SWIFT), founded in Belgium in 1973, built a service for its members that transmitted both messages and monetary transactions over a packet-switching network.[262]

The huge network boom could not obscure the fact that expanding networks were constantly reaching their limits. It was already extraordinarily difficult to develop a common control language between two machines. When installations with different rules of control met, it could easily happen that one machine did not recognize the request of the other to connect simply because it did not understand the language. But it could also happen that the controlled machine knew how to handle the commands of the others so well that it allowed itself to become completely dependent, putting its administrative or jurisdictional control at risk.[263]

Computer engineers interpreted this problem as a matter of diplomacy and called the rules that formalized delicate communicative tasks "protocols." Whereas committees "glued their protocols to the front of the files of a procedure"[264] (a reference to the etymological sense of the word) to indicate how the files should be read, diplomatic protocols guided the conduct of meetings between rulers. Protocols not only facilitated exchanging data between computers and programs, they also determined what was allowed to happen while connected. They did this with all the "solemnity and complexity that are associated with such communication."[265] The question of the difference between a computer center and a center of power was left unresolved. In fact, the analogy really applied only to the fact that protocols must be *agreed*

*upon*, whether they are intended to facilitate communications between the powers that be or between computers.[266] Relatively little effort was required to connect similar computers in a data center. It became somewhat more difficult when they came from different manufacturers. But things got really bad when not only computers but entire networks of computers that were not under common supervision had to be connected.[267]

As always, when confusion reigned, computer engineers responded with radical abstraction in search of a generalizable solution. Such had been the case with the development of operating systems and of programming languages, as well as with every translation of administrative processes into programmable procedures. For the connectivity issue, that meant anyone who wanted to master the confusing world of protocols had to create protocols for specific types of machine connections. At the same time, these protocols, which formed an entire world unto themselves, had to be coordinated in such a way that they could interact. To do that, protocols had to be organized according to functionality. Only in this way could heterogeneous computer systems be made compatible, at least in terms of their connections.

In the mid-1970s, there were essentially two ways of achieving this abstraction, not just on paper but with the prospect of operational effectiveness. Either a major computer manufacturer had to succeed in developing a worldwide de facto standard and enforcing it on the customer side, or the national telecommunications providers had to settle the matter among themselves. Both options were pushed forward with great energy and considerable success. IBM published a first version of its Systems Network Architecture (SNA) as early as 1974. This extensive collection of protocols for connecting computers consisted in turn of several software packages. These included, for example, the Virtual Telecommunications Access Method (VTAM). Beginning in 1975, DEC,

which had a special interest in computer connectivity because of its minicomputers, also attempted to develop an industry standard with its Digital Network Architecture (DNA).[268] The national telecommunications providers were no less active. In 1976, the Comité Consultatif International Téléphonique et Télégraphique (CCITT) voted on a packet switching protocol called X.25 and declared it an international standard. X.25 was immediately adopted by the national telecommunications companies of Canada, Great Britain, France, and Japan, as well as by the US computer network operator Telenet.[269]

Large users and small computer manufacturers could therefore either follow one of the two industry standards – IBM and DEC – or they could rely on the international organizational power of the national telecommunications companies with X.25. The choice was between two differently protected monopolies. In 1977, a third way emerged in the great protocol war between these two fronts. The initiative came from small (European) computer manufacturers along with multinational corporations such as Kodak and Unilever; British, Canadian, American, French, and Japanese academics; and the British Department of Trade and Industry. The idea was to work out an alternative, "open" collection of protocols.[270] Under the auspices of the International Standards Organization (ISO), rash decisions on standards were avoided. Instead, the group first worked on a general framework for connectivity in digital space. In other words, thought was first given to the conditions that telecommunications standards had to meet. The project was called Open Systems Interconnection (OSI), and its very name made it clear that the intent was to develop an open and systematic approach to interconnectivity. The goal was not the gradual harmonization of existing standards but the development of a theoretically stable framework for which specific collections of protocols could successively be developed.

OSI developed a layered architecture that distinguished seven connectivity functions. Without worrying about issues of technical implementation, the project clarified the functionality for the machine, data, network, transmission, connectivity, display, and application. Each of these layers relieved the layer below and the layer above of tasks and was in turn served by those layers. As layers, machines, data, connections, and so forth were interconnected and yet at the same time clearly distinguishable from each other. Through the collaboration of the project participants, OSI established a model that structured thinking about the interconnections of and with computers by specifying clear, functional boundaries.

When the OSI project was developed in the mid-1980s, expectations were enormous. In Europe, it was seen as a way to strengthen the European single market for computers. In the United States, in 1983, the government organized OSI courses for computer manufacturers and recommended the OSI model to communications technology specialists in the military. For French technology policy makers OSI represented a means of preventing IBM from dominating the global interconnectivity market.[271]

Nevertheless, progress in developing and implementing OSI protocols was slow. The documentation that appeared over the years was extraordinarily detailed and, at times, confusing.[272] The model had grown into a veritable paper tiger, an offer that could be refused but not understood.[273] At the same time, however, neither IBM's nor DEC's network architectures, equipped with workable "protocol suites," nor the X.25 protocols of the national telecom providers were in a position to assume sole dominance over connectivity in digital space.

The simplest solution was ultimately provided by a rather modest set of protocols – in contrast to the number of networks in operation at the time – which concentrated on interlinking in-

stead of standardizing all of them. Since the 1970s, work in this area had been carried out within the framework of ARPA and in international, academically oriented projects.[274] The solution was presented in 1983 by Vinton G. Cerf and Edward Cain as the US Department of Defense Internet Architecture Model.[275] Cerf and Cain went to some pains to encourage military interest in the model. Therefore, to sell "their" network architecture to the military, they readily underplayed the international and academic character of the working group in which American, British, and French scientists had been developing techniques since 1972 for interconnecting networks.[276]

The requirements placed on the Internet model could not have been more extensive nor more diverse. It was supposed to link heterogeneous networks of the most diverse provenance, enable interoperability, demonstrate high reliability under adverse or even hostile conditions, transmit files and make them readable at a distance, permit the targeted distribution of messages, and accept all possible types of terminals. As if that were not enough, users also wanted to be able to send text, fax, and graphic and audio messages, preferably over any channel that had ever been set up anywhere in the world.[277] It goes without saying that this promise of connectivity appealed to the military. But scientists and engineers with a penchant for analytical and abstract thinking also felt irresistibly attracted by the wealth of opportunities.

It was, therefore, the combination of operational capability and a radically simple approach that led network engineers to focus on the problem of interconnecting networks. In 1977, three completely different networks were experimentally linked, and in 1978, the protocol used for this purpose was separated into two parts. The Transmission Control Protocol (TCP) was responsible only for exchanging data between two computers. The Internetwork Protocol (IP), located on a newly introduced abstraction

layer, regulated traffic on the intermediate network independent of the transmission medium.[278]

Toward the end of the 1980s, the Internet protocols began to spread, to the great surprise of the previous leaders in the field. Neither IBM, DEC, X.25, nor OSI could resist the practical, robust connectivity that TCP/IP had to offer. A collection of protocols that was supposed to work globally had to be able to deal with heterogeneity and worldwide organization. ARPA had always assumed a diversity of users and systems. What they offered was not a perfectly constructed overall system, but network connectivity that was as simple as possible and therefore particularly robust. Users only had to worry about the standards "on their side of the hall," so to speak. The transition from the intermediate network to a local installation was the task of those in charge on-site. In fact, some network operators in the US armed forces also dreamed of a single, all-connecting network. What they probably meant was an all-military connecting network, one that would have enabled them to bring the heterogeneity of *their* networks under *their* control – just as other network operators, operating in highly diversified markets, were primarily interested in reducing heterogeneity in *their* networks.

The TCP/IP developers did not need to separately consider the military and civilian dreams of homogeneity. That was the developers' great advantage. There was a (growing) demand for interconnectivity, and it was manifold in nature. Whoever could handle the problem of linking connections had the upper hand, because connected networks increased the possibility of moving the transactions, messages, and debates of the world into the computer. The Internet development community demonstrated this potential particularly well. In 1967, they had met in front of screens in Douglas Englebart's lab at the Stanford Research Institute and imagined the new users whose communication processes would be sup-

ported by interconnected computers. Two decades later, it was the generic connection between computers that enabled communication to be shifted into the computer – regardless of which computer or network was used to access the digital world.[279]

## Differentiation

In the spring of 1975, an electronics company called MITS that was based in Albuquerque, New Mexico, placed a full-page ad in several electronics magazines. The ad presented the MITS Altair 8800 computer kit, and there was a lot of explaining to do. The ad's (necessarily) lengthy text was illustrated by a photograph of carefully arranged objects. Circuit boards, cables and a transformer, capacitors, and other electronic components lay scattered decoratively about a workstation. They were apparently waiting to be installed in the empty aluminum case that could be seen in the background of the picture. Soldering irons, screwdrivers, and assembly instructions were at the ready. All that was missing was a skilled electronics hobbyist to put things in order.[280]

"Building your own computer won't be a piece of cake," announced the large type next to the photo. In fact, the ad continued in smaller type, you'd need more than an hour or two to assemble the device. Indeed, the Altair 8800 was not a toy. It was a fast, powerful, flexible device – a full-fledged computer. Enthusiasts who had already assembled a simple electronic calculator from the same manufacturer would have a particularly satisfying experience with the Altair 8800. For 439 dollars, you got everything you needed, plus the prospect of being busy for a long time even after the soldering was done. The three manuals that came with the kit promised credibly that it would be a real challenge, even for demanding hobbyists.[281]

Publicity for the kit was so completely focused on the experience of assembly and expansion that typical questions of what to do with the computer did not arise. The modest 256 byte memory wasn't suitable for programming an autopilot for an airplane, as *Popular Electronics* magazine had boldly promised a few weeks earlier. Nevertheless, connecting the computer in its aluminum case to an electric typewriter and the television in the living room, and bringing in the sticky cassette recorder from the kitchen to use as a storage device for programs, did fire up people's imaginations.[282]

No matter what code you entered into the computer using the toggle switches, when you got to programming using the circuit-filled, aluminum-cased device, you felt as though you were on board the starship *Enterprise* traveling to glimmering Altair. And if you paid attention, there were some nice surprises along the way. Steve Dompier from Berkeley had spent a good 30 hours soldering and another 6 hours searching for a connection error on the circuit board. Then he went to test the basic functions of his new computer on the front panel because there were no usable input and output devices. One of the things he did was to load an elementary sorting program in the microprocessor. While doing so, he noticed that his small transistor radio picked up the switching noise of the Altair. He recognized that the computer was producing interference that drowned out the weather report broadcast on the radio. The same thing happened with all the programs he ran on the computer, which became audible on the radio. After another eight hours, he found he could link particular programming steps to produce specific tones on the radio. Subsequently, at the next meeting of the Homebrew Computer Club, Dompier had something very special to offer the members: Paul McCartney's *Fool on the Hill* arranged for Altair and transistor radio.[283]

The performance was a success. The audience clamored for an encore, a first indication of the program's significance for com-

# MITS

## BUILDING YOUR OWN COMPUTER WON'T BE A PIECE OF CAKE.

**(But, we'll make it a rewarding experience.)**

Chances are you won't be able to assemble the *Altair 8800 Computer* in an hour or two. But, that's only because the *Altair* is a real, full-blown computer. It's not a demonstration kit.

The *Altair Computer* is fast, powerful, and flexible. Its basic instruction cycle time is 2 microseconds. It can directly address 256 input and 256 output devices **and** up to 65,000 words of memory.

**Thanks to buss orientation** and wide selection of interface cards the *Altair 8800* requires almost no design changes to connect with most external devices. Up to 15 additional cards can be added inside the main case.

The *Altair Computer* kit is about as difficult to assemble as a desktop calculator. If you can handle a soldering iron and follow simple instructions, you can build a computer.

You see, at *MITS*, we want your experience with our kits to be rewarding. That's why we take such pains to write an accurate, straight-forward assembly manual. One that you follow step-by-step. (We leave nothing to the imagination.)

**Some electronic kit companies are experts** at cutting the corners. They promise you the sky and deliver a box full of surplus parts and a few pages of faded instructions run off on their copying machine.

We're experts at **not** cutting the corners. Our *Altair Computer* has been designed for both the hobby and the industrial market. It has to be constructed of the finest, quality parts. And it is.

That's why we give you double-sided boards, gold-plated connectors, a 10 Amp power supply (enough to power 15 additional cards), toggle switches and an all aluminum case complete with sub-panel and detachable dress panel.

That's why we give you three manuals (Assembly, Operator's and Trouble-shooting) in a hard-cover, 3 ring binder plus an Assembly Hints manual.

Buy our computer and we'll automatically make you a member of the *Altair User's Group*. You'll have access to a whole range of custom software designed exclusively for the *Altair 8800*.

**We're quite serious about making computer power available to you at a price you can afford.**

### BASIC ALTAIR AND OPTIONS

The basic *Altair 8800 Computer* includes the CPU, front panel control board, front panel lights and switches, power supply and expander board (with room for 3 extra cards) all enclosed in a handsome, aluminum case.

Options now available include 4K dynamic memory cards, 1K static memory cards, parallel I/O cards, three serial I/O cards (TTL, RS232, and TTY), octal to binary computer terminal, 32 character alpha-numeric display terminal, ASCII keyboard, audio tape interface, floppy disc system, and expander cards.

**PRICES:** Altair Computer Kit with complete assembly instructions **$439.00**
Assembled Altair Computer **$621.00**
1,000 word static memory cards **$176.00** kit
**& $209.00** assembled.
4,000 word dynamic memory card **$264.00** kit
**& $338.00** assembled.

NOTE: Altair Computers come with complete documentation and operating instructions. Altair customers receive software and general computer information through free membership to the Altair User's Club. Software now available includes a resident assembler, system monitor, text editor, and Basic compiler.
Prices and specifications subject to change without notice. Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units.

**MITS/6328 Linn N.E., Albuquerque, N.M., 87108, 505/265-7553**

MAIL THIS COUPON TODAY!

☐ Enclosed is a Check for $_____
☐ or Bank Americard # _____
☐ or Master Charge # _____
Credit Card Expiration Date _____
☐ ALTAIR 8800 ☐ Kit ☐ Assembled
Include $8.00 for Postage and Handling
☐ Please send free Altair System Catalogue
NAME _____
ADDRESS _____
City _____ State & Zip _____
MITS/6328 Linn, N.E., Albuquerque, New Mexico 87108
505/265-7553

MAY 1975

*Circle 16 on reader service card*

25

Figure 15: The Altair 8800 – a challenge for adventurous hobbyists.

puter history. Quite unexpectedly, Dompier later reported, he got the computer to break into an "apparently genetically inherited" version of *Daisy Bell*, a popular song from the 1890s. The song had been programmed in 1961 by John Kelly, Carol Lockbaum, and

Max Mathews for an IBM 7094 machine at Bell Labs. It was also hummed by the "dying" HAL 9000 computer in Stanley Kubrick's 1968 film *2001: A Space Odyssey*. The fictional HAL 9000 was not made by International Business Machines, though its name can be derived by shifting each character of IBM to the preceding letter. HAL's name referred to its function, serving as a Heuristically programmed ALgorithmic computer. When HAL showed unexpected emotions during a trip to Jupiter, it had to be shut down module by module.[284]

Dompier had not only constructed and programmed an Altair 8800 but also created an iconic story. *People's Computer Company* magazine published it with an illustration of the Altair sitting weirdly on a hill – just like *The Fool on the Hill*. The machine held a transistor radio in its hands that channeled the programmed version of McCartney's song, transfixing all the local wildlife. As the program for the first song ran its course, the Altair then produced a love song to a distant Daisy that harkened to the era of the IBM mainframe. In a sense, the echo seemed to be programmed in its core. Despite the demise of Kubrick's HAL 9000, hope for *Daisy Bell* resonated in Dompier's Altair narrative, even gaining new power, and fascinating readers. At another level, with its "spontaneity" the story acquired new meaning seeded by Bell Labs and expressed in the discourse of flower power: "There is a flower within my heart, … planted by Daisy Bell." In the popular song, the lover wanted not just to share his bicycle with the beautiful Daisy, but also to throw in his lot and share his life with her. Of course, he was a fool, as in McCartney's tune. And crazy, too, as the Bell Labs' IBM 7094 and Stanley Kubrick's HAL 9000 computers reiterated in their renditions of the popular song.[285]

The flowery story aside, the situation was very difficult to reckon with, even for the most seasoned of amateurs. Jef Raskin, the busy editor of a hobbyist journal, sought to steer readers to-

ward "a bit of wheat amongst the chaff." His experience with kits, spare parts, additional components, and instructions showed that not everything was as exemplary and viable as the Bytesaver memory card distributed by Cromemco for the Altair. In *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia*, Raskin tried to spare potential kit purchasers the worst, and offered "brotherhood, sympathy & emotional release" to those who had already bought kits.[286]

The computer for the everyman had not yet arrived. Raskin's belief that computers should work for ordinary people sounded heretical unless, that is, you reinterpreted the devotion of amateur electronics engineers to the microprocessor as a tinkerer's aimless, indulgent pleasure and accepted it as a cultural practice whose very personal, rather ascetic path was the goal. If the willful combination of cultural techniques (soldering, reading, and programming) did not necessarily lead to anything more useful than *Kill the Bit* (a game written for the Altair 8800) with a glorious hour of purposeless play, then the service relationship between human and machine was actually balanced.[287] The interest of hobbyists in programs that could calculate prime numbers was not driven by a pressing need for prime numbers. Hobbyists continued to develop the programs purely because the easily verifiable results confirmed their ability to read manuals, the functional efficiency of the printed circuit boards they had assembled themselves, and the suitability of their own programs even under precarious technical conditions. That was all it took. Sorting programs, interference, and outlandish narratives helped to make the personally constructed computer usable by all and for any conceivable purpose. Games and music could not be excluded. Their significance actually crystallized through use and should be construed neither as "misuse of military equipment" nor as misappropriation of research and administrative computers.[288] On the other hand, in

Silicon Valley, which extended all the way to Albuquerque, people were hard at work moving the hobby, and with it the garage, into the computer.

With the Altair, everything was still written in the stars. The hobby computer represented an adventurous combination of knowledge and imagination, intrepidness, and advance billing on the part of many players.[289] Nevertheless, it sold like hotcakes. The kit's price rose from ad to ad, delivery times spiraled out of control, and production of promised peripherals was postponed again and again.[290] These developments encouraged many imitators, suppliers, and programmers to jump into the gap between supply and demand themselves. Soon there were two dozen kit products enabling persistent hobbyists to equip a microprocessor with a programmable environment.[291]

ACM members didn't have to read Altair advertisements to know about the completely unexpected movement at the lowest, barely serious edge of hardware and software supply. At the 1976 National Computer Conference in New York, the professionals even addressed (albeit fleetingly) the problem of how to build a personal computer around microprocessors or with discarded equipment parts. However, the larger question of what an amateur might do when he got tired of playing on his "machine" was not answered in this "overview for computer engineers."[292]

That task was taken on by the editor of *Dr. Dobb's Journal* at the 1977 National Computer Conference in Dallas. Jim Warren was a traveling salesman between worlds. He had just founded the *Silicon Gulch Gazette* and organized the first West Coast Computer Fair. As a math teacher, he knew his way around the world of college students. As a doctoral student in computer science at Stanford, he'd had a connection to computer science. That he believed in improving the world technologically made him an above-board source for both established computer engineers and countercul-

ture-inspired amateurs. Warren knew how to translate the story of Dompier's creative use of hardware, rudimentary commands, and a quirky Beatles song. Moreover, he knew how to explain the history as well as the current state and trajectory of the development that transformed the universal digital computer into a consumer product. Warren gained credibility with both camps by blithely enumerating the differences between "personal computing" and "professional computing," and by further delineating the two fields.

Personal computers, Warren said, were characterized by complex hardware and a lack of software for serious applications. Low cost was crucial, while speed, capacity, and reliability remained of secondary importance because users' time and effort cost nothing. Their main concern was entertainment in the broadest sense of the word. Consequently, the computers did not have to be attractively designed, and you could see the widespread use of second-hand equipment among amateurs. On the other hand, customer support did matter.[293]

Thus, when it came to personal computing, there were many different factors and components, and the hardware was as "independent" as the users. That meant that each machine placed its own demands on the software. Cost–effective software development was simply impossible, in part because of the widespread sharing culture of amateurs.[294] Attendees at the ACM may well have wondered what could actually be achieved with such computers. Warren remained grounded in reality. He had recounted the journalistic and organizational development of computers with enthusiasm and expertly described the wealth of variation in the hardware landscape. But his answer to the question of usefulness was unabashedly sobering. Conventional business and industrial applications were simply out of the question, said Warren. The most important application was games, and in the future perhaps

group games as well. Games also had an educational value. In addition, music and radio applications might gain in popularity. There might be some interest in word processing. But without good printers, you could not get very far. Perhaps the future would produce a digital library, in which case, at least marginal notes made while reading could be stored locally.[295] In other words, the personal computer was a product of the hobbyist sector. As such, everything could be moved to the device so long as it was a game. Sounds could obviously be handled, and perhaps it would even eventually be possible to write things down. But these capabilities had yet to be proven.

A year later, the ACM's newly formed Special Interest Group on Personal Computing (SIGPC) published a position paper on how to move from hobby computing to personal computing. Alan Kay, Adele Goldberg, and Larry Tesler of the Xerox Research Center in Palo Alto would have known that position papers are a static thing, especially when they concern the future. "Neither hardware nor software is right; most everything is yet to be done," read the paper's opening sentence. Nevertheless, the trio didn't hesitate to take the plunge into a very indeterminate future right from the second sentence: "Some day, personal computing will be exciting and useful to a large range of people of all ages."[296]

In that regard, it was difficult to harness the personal computer. No sooner had it been boxed into an aluminum container as a rudimentary programmable microprocessor, resulting in the Altair, than it broke free of its hardware and made its presence felt in the transistor radio. When Jim Warren assigned it to a conceptual corner, where it might have enjoyed an existence distinct from the business computer, people were impatient to use it for word processing. And even as the team at the Xerox Palo Alto Research Center denied the current usefulness of the personal computer, in the same breath there was talk of the personal com-

puter imminently becoming exciting and useful on a large scale. Notwithstanding expectations in the second half of the 1970s, the personal computer remained a small but surprisingly unwieldy thing, whose characteristics were hard to pin down. "The personal computer defies exact definition," wrote Portia Isaacson, Adam Osborne, Robert Gammill, Larry Tesler, Richard Heiser, and Jim Warren in an essay.[297] In the spring of 1978, these experts had dealt with the expected personal computing problems of the 1980s and, after a conference in Portland, had written their "Oregon Report" on personal computing, which was published a short time later in *Computer*, a trade journal published by the Institute of Electrical and Electronics Engineers (IEEE).[298]

The fact that the authors were at pains to name the key features of the personal computer right at the beginning of their report speaks volumes. Artists could use it to create new art forms. Financial market analysts could study stock market prices. A secretary could enter and edit manuscripts. In short, personal computer applications were as diverse as the individuals who used them.[299] To deal with this diversity, the all-purpose computer was once again the order of the day. At the beginning of the 1950s, with UNIVAC, the Rand Corporation had universalized *procedures* such as sorting, classifying, computing, and decision-making; in the mid-1960s IBM had attempted to unify different *classes of processor performance* with its System/360; and the figure of the "end user" had served to generalize the different *application areas* of the 1970s. With the personal computer of the 1980s, users were the source of complexity; any claim to universality would therefore have to start with their machines. In its most limited, individual form, the computer was to be turned into a personal computer that anyone could use to do practically everything. To achieve this, the machine had to be versatile and provide several highly divergent and even contradictory services. It could process texts,

track private expenses, or compute melodies and play them electronically. Regardless of whether the computer was imagined as a home computer, a consumer computer, or a hobby computer, or as a personal computer for business, education, or science, it was intended to be a general-purpose, intelligent machine that could fulfill specific purposes for particular categories of users.

What is striking is how quickly this claim to universality broke down into individual fields of development at the end of the 1970s. The "Oregon Report" is essentially a time-lapse account of computer history, a technological journey focusing on the microprocessor. On the hardware side, the focus was on processor performance, memory, mass storage, monitors, printers, and input devices, and on the software side on programming languages, operating systems, application software, and, as a critical long-term problem, interconnectivity.[300] The "Oregon Report" therefore reads like a comprehensive development plan for a personal computer from a major manufacturer. All that was missing was the large consumer market to be able to mass produce computers in the low-end microprocessor range. If hobby and home were too small for that, it was possibly worthwhile to take another, closer look at the office.[301]

The challenge in moving office work to the computer was that almost every activity was different. For the most part, office work is weakly structured.[302] This lack of structure had become even more marked since the 1950s, when the exceptions to the rule had been moved to the computer, namely, particularly well-structured, mass-produced tasks. Anything that was not sufficiently monotonous to be moved to the computer was simply left at the desk and consisted primarily of odds and ends with high text content. However, three decades after UNIVAC, it appeared that a personal computer for workers might happen after all and that the large amount of residual desk work could also be shifted to

Figure 16: Nativity scene: the IBM personal computer and skeptical but impressed users (1981).

the computer. Doing that would require rethinking services not from the vantage of an organizational and technical center, but from the periphery. In other words, instead of adding further time-sharing system terminals to specialized workstations and personalizing computers that way,[303] screens and keyboards with minimal microprocessor intelligence could be provided right where they would be used for small jobs. They could then replace (electric) typewriters, desk calculators, index cards, notepads, and more. Such a multifaceted machine could lead to mass business, providing manufacturers with the economies of scale that were considered a prerequisite for shifting highly individual forms of computing, even in the home and garage.[304]

Possibly, with luck, maybe, and soon. Meanwhile, sales projections for the Apple, Texas Instruments, and Commodore microcomputers sold in thousands of Radio Shack stores had to be

revised downward, and at the meeting of the ACM's provisional special task force, the chairman confessed that still nobody could actually say what a personal computer was.[305] Was it its external appearance, portability, location, lack of connectivity, software offerings, or form of use that characterized it? The *New York Times* noted in 1979 that PC manufacturers introduced their devices that year not at the Consumer Electronics Show in Chicago, but at the National Computer Conference in New York. Lewis F. Kornfeld, president of Radio Shack, had even asked reporters at the launch of the TRS 80 Model II line of computers not to call them "home computers." The machine, he said, was intended primarily for the "small-business man." In the opinion of the *Times*, no one really wanted to pay between $500 and $3,000 just to watch their monthly budget erode, educate their children, and turn on the sprinklers in the garden. But even if the home computer market never actually emerged, demand from small businesses was real. Accountants, lawyers, and dry-cleaning firms, for example, would be able to manage their mailing lists, their submissions and pleadings, or simply their billings and estimates on the computer.[306]

Nonetheless, the IBM personal computer unveiled in the summer of 1981 was met with great skepticism. According to a shaken *New York Times*, "big IBM's little computer" at least showed that the market leader apparently had learned from its experience with the DEC minicomputer. With the personal computer, IBM seemed to want to make inroads into the consumer electronics market dominated by Apple and Radio Shack. The "newspaper of record" had its suspicions confirmed by computer dealer Michael McConnell, who was quoted as saying that personal computers were no flash in the pan.[307]

IBM's offering was intentionally ambivalent, as was obvious in almost every line of the product description. On the hardware

side, the IBM personal computer was expandable, unlike other microprocessor computers. In addition, it offered interesting and well-tuned peripherals, and buyers were supported by a professional service network. On the software side, IBM always pointed out professional and private, that is, sophisticated and modest, application possibilities at the same time. There was a built-in speaker for music and an operating system, a matrix printer, and a Pascal compiler for programmers. Word processing was made possible with EasyWriter. Users who preferred doing financial forecasts and simulations could use VisiCalc. In addition, there were fantasy games from Microsoft and access to *Dow Jones News*.[308] Following IBM's only moderately successful attempt in the late 1970s to produce small or very locally distributed computers for the office, the focus was now on blurring the boundaries between hobby, home, and office, with a decided emphasis on the latter in terms of price.[309]

Critics of the personal computer pointed to the lack of a technical breakthrough; defenders cited a considerable increase in computing power if a coprocessor were added to the PC.[310] Both overlooked the inherent focus on adaptability, emphasizing the variety of options available to users. Indeed, the IBM personal computer was brilliantly tailored to the Reagan era: a cultural model of options and choice took shape inside the PC.[311]

Everything was packaged into a single device, neatly delineated yet configurable as an all-purpose machine. It offered computational intelligence in the form of local computing capacity, and enabled individual programming, writing, simulating, playing, learning, and calculating in every conceivable way. The world that had moved into the personal computer was the world of one's own work, play, spreadsheets, individual simulations, if necessary the world of one's own programs, but certainly the world of one's own texts, notes, reminders, and memories.

In 1981 there was hardly any viable alternative to this universal orientation of the IBM personal computer with its diverse configuration options. It was not until a good two years after the delivery of the IBM PC that such an alternative began to emerge. It was particularly radical in its attempt to universalize the personal computer for anyone who did not have access to a "real" computer. Such a device – a computer "for the rest of us" – was unveiled by Apple in early 1984 with a vast promotional campaign.[312] Since computers were smart, Apple's marketers claimed, it would make sense to teach them about people instead of trying to teach people about computers.[313] According to one of its ads, Apple set about "teaching tiny silicon chips all about people. How they make mistakes and change their minds. How they label their file folders and save old phone numbers." The computer of the future was supposed to know everything about its users – "[h]ow they labor for their livelihoods. And doodle in their spare time." That's what it took to produce a computer that was so genuinely personal it could practically shake your hand, and so easy to use that "most people already know how."[314] When a computer was shown how people felt, thought, and acted, the computer and the user changed, and so did their relationship. The ad ended by introducing the computer by its name: Macintosh. "Hello," the computer said.

The microprocessor computer embodied empathy for both learned and for somewhat limited users, providing essentially a black box, encasing components that were safely sealed away and out of reach of soldering irons and screwdrivers. The graphical user interface not only kept the user away from the components but also from the operating system and its command lines. The very personal Mac was diverse in its applications, completely oriented toward users and their idiosyncracies, but insofar as possible separated off from everything else.


Figure 17: The Macintosh in 1984 – relatively compact, well-bred, and decidedly user-friendly.

## Storage

In the early 1950s, a Rand Corporation commercial invited people to use the largely idle computing capacity of the first commercial computer in history to solve problems of sorting, classifying, computing, and decision-making. UNIVAC was portrayed as a well-organized, powerful machine that could be efficiently fed with data, worked unimaginably fast, and generated output in a very short time. Under ordinary circumstances, whole armies of office workers would have had to struggle for days with the tasks that this electronic miracle machine completed in an instant.[315] Even for skeptics, there was nothing extraordinary about the media previously used and still now used with the machine. Paper, punched cards, and magnetic tapes were all familiar and had been used

for many years, decades, or centuries for recording, transmitting, and storing administrative information.[316] The computer could take what it was given (*datum*) from such media; run it through task-specific programs; and print, punch, or record it again as a processed result (*factum*) on equally familiar media.

After processing, the computer was empty and clean. Input and output were stacked up as if they had never had anything to do with each other, as piles of cards or paper, and nothing had stuck in the computer's "memory." No traces or residues, zero contamination. This depended neither on how clean the entered data were nor on the validity of the results obtained. It depended instead on the design of the memory. In UNIVAC, the memory consisted of four-meter-long tubes filled with mercury. If the computer had to remember something, an oscillating crystal sent a sound wave through one of the tubes. The time it took for the wave to pass through the mercury and hit a second quartz crystal at the other end, where its pressure caused a voltage rise and generated another electrical signal – that time constituted the elementary functional performance of the computer's memory, which was also called delay line memory. In UNIVAC, 100 such delay line memories were installed, which together formed a thick bundle of mercury tubes. Ten alphanumeric "words" encoded as oscillation patterns ran in each individual tube. If such a word had to be remembered longer than it took the sound wave, slowed by the mercury, to travel, it had to be sent electrically back to the beginning and through the tube again when it arrived. The process was repeated until the word was used in the program and the oscillation could be stopped.[317]

If the computer was loaded with particularly complex operations, the costly delay line memory, which had low capacity, was inadequate. The only alternative was to write down intermediate results. What was recorded on this "notepad" in the form of a

magnetic tape could be retrieved and put back into the computer as needed. The designers of UNIVAC called this auxiliary memory "storage." It represented a small but revealing semantic shift from the terminology commonly used for calculating machines since Babbage. Conventionally, any of the numbers in the machine could be said to be in the "store," whether they were inputs, intermediate results, or final results.[318] With UNIVAC, there was obviously a need to functionally differentiate the conceptual landscape: while the input and output were stored outside the machine, the operands were located in the registers. The running program and the data that could not yet be moved to a register were in the memory. And intermediate results that could not be placed elsewhere were recorded in storage. Strangely enough, UNIVAC's designers spoke of "temporary storage," as if the expression required qualification.[319] The curious aspect of this is that memories are only meant for storing information that will be used later. Memories bridge time and as such are necessarily "temporary." "Temporary memory" is redundant. Might plain talk about "memory" have been embarrassing because it pointed to the limited capacity of the computer and betrayed the inelegant solution to an operational bottleneck? In the prestigious high-tech field of computers, this approach had a dubious charm – UNIVAC apparently was an electronic marvel that could no longer hold everything in the intermediate steps of its computing work.

In any event, any reluctance to use the term is not of historical interest. Rather, it is the fact that over the next years and decades the "notepad" developed into a multifunctional long-term memory of previously unimaginable size.[320] For the question of how the world got into the computer, the rapidly expanding auxiliary space of temporary storage is hugely significant. In addition to process junk, the space could accommodate critical data files and frequently used programs. Here, as in the warehouses of large

maritime ports, further transactions could be prepared, and data could be regrouped according to customer-specific wishes and combined into other order units. The world was moored in storage, though it was thought to be on a long line.

The history of storage is conventionally told with reference to the effect of falling storage *costs* or rising storage *density*. But it may be better interpreted in terms of storage *management*, that is, as the evolution of a system flaw to a general feature of the system. This narrative focuses on the organizational interaction of the various memory and storage units and the linking of data.[321] After the assembly line concept was dropped toward the end of the 1950s in favor of the more flexible random-access model, management of the digital data repository was transformed in four major waves of development: (1) In the 1960s, the sophisticated logistics of procedural databases made it possible to keep data available for quick access and to link them in a stable and resource-efficient way. (2) The mid-1970s saw the rise of relational database technology, which had to do with tables and the ability to freely combine data. (3) About the same time, a major change occurred when interactions between different types of storage were managed systematically. (4) Finally, in the late 1980s, the hypertext structure of the emerging World Wide Web relied on the possibility of linking even very heterogeneous data sets on different machines. On the basis of these four, historically overlapping developments, the following section explores the question of how dealing with memory and data issues contributed to anchoring the world in the computer.

The first of the fundamental changes in storage and data management in digital space might well be called procedural database management with chains and programmers. In the early 1960s, there was talk of developing a general programming system for random access memories. The focus was data logistics and asso-

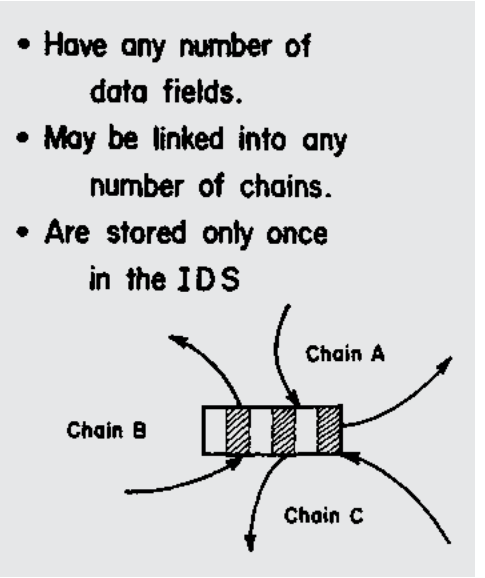


Figure 18: Records had any number of data fields and could be linked to any number of chains.

ciated procedures. At the Joint Computer Conference in Minneapolis in the fall of 1964, Charles Bachman and Stan Williams of General Electric presented the Integrated Data Store (IDS), a general programming system for massive *and* flexible access to data storage. IDS was a sophisticated database management system that could be used for very large and very diverse applications.

The system was based on so-called records, which were composed of "data fields" and "chain fields." Chains were task- and query-specific links between the individual records. They ensured the integration and association of the records, and procedures for storing and accessing the data were adapted for that.[322]

To plan and control complex business processes, the flow of orders and materials had to be managed; information had to be simultaneously stored, retrieved, communicated, and processed. And for that, reliable techniques for organizing data were needed. Furthermore, a database had to be suitable for use by different applications, so that data did not have to be formatted separately or updated for each procedure.[323]

With the IDS, General Electric had developed an actual data logistics system for data traffic. An input/output controller monitored the transport of the data between the hard disk (disk memory) and core memory. For this purpose, it used data blocks on which the records currently used in the core memory were combined. The blocks had a fixed size but could still contain records of different lengths; data blocks functioned like pallets in the transport of goods by rail.[324] The blocks could be used to move records comprising (interlinked) data fields between the large but slow disk memory and the small but fast core memory. The core memory, however, contained a constantly updated inventory of all data blocks. To speed up the search and transfer procedures, blocks that were not currently needed were released from core memory, while those that were used repeatedly were allowed to remain. Data blocks that were changed after processing were written back to the disk memory.

The IDS featured a large variety of linking possibilities and was nevertheless space saving in its memory management.[325] Linking the records via chains and transferring them on data blocks ensured the mobility of the data while economically handling the different forms of storage in the computer. Only programmers could create new chains, redeploy data blocks, and submit new procedures. Procedural databases were subject to the authority of programmers, and their data were labeled in such a way that they allowed specific queries but others only with additional program-

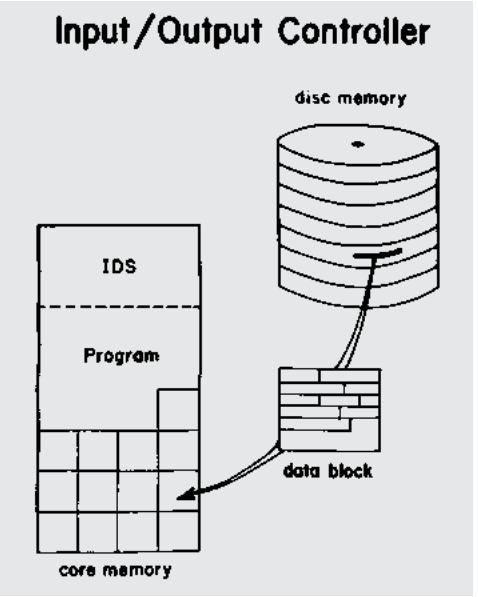


Figure 19: The data block as a means of transporting records between hard disk and core memory.

ming effort. This strict regime could only be broken by developing a completely different, much more elaborate, and even more radical database concept.

Relational databases of the 1970s aimed to simplify ways of linking data. Their main focus was on the table and ways of configuring data. In 1970, Edgar F. Codd, of the IBM Research Laboratory in San José, presented his thoughts on a radically new database architecture.[326] In the future, he said, operation of large data banks would produce new groups of users who had to be protected from having to know how the data that interested them were organized (internally represented) in the machine. Users should be released from the care of the programmers. But the

chains that held records together in procedural database systems also needed to be broken, freeing data from addresses, shipping labels, chain codes, and other labels.[327] Codd promised that a relational database structure would serve a greatly expanded circle of future users who were unskilled in data processing but able to query databases.[328] By no longer having to worry about how data were stored, users would be better able to focus on ever more flexible querying.

The database of the future was to be set up in such a way that its content could also be confronted with questions that had not even occurred to its creators when the database was planned. Codd therefore set himself the task, first, of developing a simple and general organization of the data in tables that could be linked to one another via a primary key. Second, a management tool had to be built for the consistent modification of entries and expansion of the database. Finally, a query language had to be created that satisfied mathematical requirements and yet was as close as possible to the natural language of users who had no knowledge of programming.

The more precise the description of the relational model, the greater the uncertainty among database specialists who were familiar with conventional database models down to the last detail.[329] Programmers who wished to hold on to their privileged role as professional navigators through the waters of complex databases were especially confused and worried.[330] They had every reason to be. Their conviction that databases had to be operated hierarchically and procedurally and had to contain hard-coded links for fixed applications was repeatedly under attack by the growing community of "relationalists."[331] This group argued that the relational model represented data exclusively in their "natural structures"; it had nothing do with the details of storage and access. In a word, no representational garbage.[332]

By the mid-1970s, the central concepts for Codd's relational model had been determined.[333] All data in a relational database system had to be able to be represented by a set of named tables, called relations. Each relation contained named columns. The order of the rows did not matter, but each row was distinct and described one instance of the entity described by the relation. In addition, each relation had a column called the primary key.[334] What data independence theoretically meant had become clear in the discussions of the past years, especially in the debate with the supporters of procedural database architecture. Exactly what it would take to realize the goal of increased user independence, however, had to await the actual development work.[335]

Between 1974 and 1979, the IBM Research Laboratory in San Jose, California, was at work on a project that later became known as "System R."[336] In a first phase, between 1974 and 1975, a Structured English Query Language (SEQUEL, later SQL) was developed that would enable a future user to formulate his queries at an interactive terminal.[337] Great importance was attached to the human factor, and various experimental studies were carried out on the learnability and usability of the system. The second phase of the project (1976 and 1977) was concerned with rebuilding the system for multiple, concurrent users and adapting the existing SQL so that it could be used on different systems. Users familiar with the PL/I and Cobol programming languages were to have the same capabilities and to use the same syntax as "ad hoc query users."[338] In 1978 and 1979, in-service tests were conducted at IBM itself and at three customers, and users' experiences were evaluated.

IBM clearly intended System R to be a major experiment in user orientation. The construction of user-friendly interfaces[339] and the system's use of modules were the main strategies employed to achieve the project goals.[340] However, the work on data and user independence caused the developers many headaches.

While the second version of the query language seemed to work quite well, the developers were still struggling with how to provide a functional replacement for the additional information on structuring data used in procedural database systems without burdening users with representational garbage. The problem of performance could hardly be discounted either, because the prerequisite for a relational database was immense storage capacity.

The progress made from the mid-1970s onward on the design and organization of very large memories was a response to the relentless demands of the relationalists. It also provided a substantial boost to the data warehouse without having to worry about the data organization itself. The strategy of combining memories with different speeds, capacities, and costs into a single system was decisive for future memory development, that is, consistently organizing individual memories according to their function in the memory store of the computer. For example, the IBM 3850 mass storage system introduced in 1975 consisted of a hierarchically arranged, three-tier storage architecture.

In the case of large but slow mass storage (tapes or disks), access time played a minor role. In a sense, it functioned like the high-bay warehouse in the wholesale trade. In contrast, the smaller direct access storage device (DASD) was much faster and more flexible. It was akin to a picking room where orders from a retail store are assembled into a transport unit. The DASD was where the required data were prepared, new data were input, and the programs currently in use were loaded. Above all this, however, hovered the fast main storage, which regulated the data traffic to the processor and supported it.[341] The previously embarrassing temporary storage had become a well-organized storage structure. Not only did it serve all existing and planned database architectures, but in the lowest, slow range it could also accommodate a large data and program archive. The need to link local

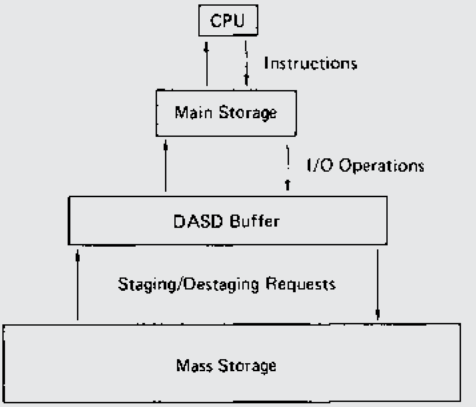


Figure 20: a well-organized storage landscape (1975).

storage environments (similar to computers) over growing distances was obvious. Thus, in the 1980s, in addition to telecommunications links between computers, a patchwork of storage technologies emerged that ensured the proper infrastructure for linking data. Fast and slow, large and small, fixed and mobile, and hard and soft memories were linked in countless combinations, whether through a stable division of labor or only in a temporary so-called master–slave relationship.[342] Programs, data, and reports were stored worldwide. Accumulated heaps of data were subject to a continual loss of information content, producing as they did so an impenetrable, non-compostable layer of data garbage that rarely ever again rose to a level suitable for data processing.

The fourth fundamental development in storage was the advent of hypertext and the heterogeneity of the stored data. The concept of hypertext had a long history. Technically, it had been understood since the 1960s as an ensemble of texts held

together with nodes and links, comparable, for example, to the conventional linking services of an encyclopedia or a product catalog with cross-references to individual "articles."[343] Hypertexts seemed to be of practical use wherever extensive documentation with thousands of pages was to be made accessible, for example, in space travel, in the maintenance of an aircraft carrier, or in the development of a large operating system. In this context, hypertext structures could translate the conventional linking capabilities of analog encyclopedias, catalogs, and manuals into a computer-based structure. In terms of storage technology, hypertexts had the advantage of low redundancy; in terms of cognitive science, they resembled associative learning. In short: hypertexts had something seductive or even anti-authoritarian about them not only for their early advocates, like Vannevar Bush, Douglas Engelbart, and Theodore Nelson. Perhaps they could be used to discover new and unexpected associations. Or perhaps they were a tool for eliciting the information that "the system" locked away in procedural databases and did not intend ever to release. Ted Nelson had dreamed such dreams in the 1967 Summer of Love.[344] Certainly with hypertexts there were no official and easily guarded entrances and exits. They were structurally open, undermined authority, and encouraged interpretive freedom and (at the extreme) critical thinking on the part of students.

Trying to predict the theoretical and material consequences that hypertext structures would have on a large scale was more difficult. Even California computer guru Jef Raskin warned against the "hype in hypertext." "If the details are kept sufficiently vague," Raskin said, hypertext would be a "wonderful, universally applicable, powerful, natural, human-oriented model for organizing and accessing knowledge."[345] Considerable discussion, however, was required to specify the details.

This discussion was conducted with increasing urgency beginning in 1987. Hypertext was now indeed hyped and a booming area. Microsoft reissued Ted Nelson's cult book *Computer Lib/Dream Machines*.[346] Apple was working on HyperCard, and the ACM had started a whole series of hypertext conferences.[347] Jeff Conklin of General Electric had just written an extensive technical report for the IEEE readership. In it, he asserted that hypertext systems could manage machine-based connections in and between texts in a way that made the computer (yet again) a new tool for communication and thought. From the perspective of a computer scientist, he said, hypertext was a *database method* that allowed direct access to data, entirely without "traditional" queries. Hypertext was also a *form of representation* that combined informal textual material with data from formal evaluation methods. Finally, hypertext offered an *interface* that allowed users to attach "control buttons" to material of interest at will. For Conklin, however, access, presentation, and interface were not three separate applications of hypertext. They were a fully integrated functionality.[348]

In his report, Conklin listed and compared no fewer than 20 existing hypertext systems. Their functionalities were impressive.[349] Raskin remained censorious, firing back a list of reservations. Would the storage capacities be sufficient? Was there enough bandwidth in the networks and sufficiently powerful software for extensive hypertext applications? Who would pay for the costs of "the central system" (sic!)? How would compatibility between systems be ensured, and how would authors be paid? Apparently, Raskin had learned to distinguish between development and marketing during his work on the Macintosh. He certainly no longer counted himself among the computer enthusiasts who saw technical problems as something that would be solved "in the natural course of things."[350]

The debate was not going to end with the hypertext skeptics drowning in the rising flood of data occasioned by the linking zeal of the hypertext supporters. But framing the problem differently might change the tone of the debate. After all, no one could overlook the increasing amount of data on organizations' hard drives. But the problem of information management could be dealt with regardless of the amount of data stored if it was described as a general problem of *information loss* in organizations. That is exactly what physicist and software specialist Tim Berners-Lee, working at CERN (the European laboratory for particle physics, near Geneva) tried to do in a proposal he wrote in 1989.[351] In it Berners-Lee proposed a distributed hypertext system. The impulse behind the proposal was the constant loss of information "about complex evolving systems" at CERN. This distributed hypertext was not intended to establish a new data management system for new data. Rather, it was about ensuring access to aging data sets in a constantly changing organization based on information exchange.[352]

Long stretches of Berners-Lee's proposal read like a refresher course for CERN physicists. For the scientists, a clearer, hierarchically structured data storage system on the CERN computers would naturally be essential to scientific success. Documented knowledge was just as indispensable. At first, the idea of an unruly linking technique to halt the informational entropy among forgotten or barely retrievable tables, data blocks, and lists sounded a bit abstract. But Berners-Lee demonstrated great communicative skill by showing that there was nothing in the specific IT requirements at CERN to contradict his organizational goal. For example, because CERN was a distributed organization, access from remote machines was essential. No one questioned the heterogeneity of CERN's computer landscape, which consisted of very different types of systems, for example, "VM/CMS, Macintosh, VAX/VMS[, and] Unix." Authoritarian centralization and access control to

what was stored was therefore unthinkable because knowledge from nuclear physics experiments started very small, could grow very quickly, and had to be able to be combined with knowledge from other experiments and research groups. Data that already existed could not simply be deleted, as the data might take on a different meaning in light of new measurement results. In addition, it had to be possible to annotate data and documents with private and public comments, a customary practice in day-to-day nuclear physics research. Copyright and data security, on the other hand, were of secondary importance because the researchers considered information sharing more important than secrecy, which in any case did not correspond to the thinking style of the scientists associated with CERN.[353]

Berners-Lee found a hearing among the scientists because he reinterpreted the data flow problem as a data loss problem and because he brought the IT requirements in line with the organizational requirements of CERN. In addition, he separated the storage of data from its presentation and explicitly ignored the operating logic of the individual machines. In this way, he accommodated the actual heterogeneity of the data, the experiments, and the personnel at CERN.[354] Berners-Lee's proposal was quickly taken up and implemented elsewhere. For example, in IEEE's *Computer*, John Noll and Walt Scacchi of the University of Southern California pointed out that a distributed hypertext architecture would enable access to heterogeneous information repositories. Hypertext combined a user interaction facet, a data representation facet, and a data storage facet. The approach preserved the autonomy of repositories and users and also accommodated the heterogeneity of data and machines. As such, the distributed hypertext architecture constituted a powerful organizational tool.[355]

It is one of the peculiarities of history that the concept of a distributed hypertext system for linking heterogeneous stored

content became attractive at the very moment when, in the competition between different network architectures, protocols that were able to deal particularly well with the heterogeneity of other networks prevailed. Once a hypertext transfer protocol (http) was added to the Internet protocol collection, networking, storage, and linking techniques for data, machines, and organizations began to mutually support one another in an unprecedented way.[356]

"The problem of information loss may be particularly acute at CERN," Berners-Lee had noted in his original proposal. But CERN was nothing if not "a model in miniature of the rest of the world in a few years time."[357] The global and local interconnection of computers, their user-oriented differentiability, and flexible access to stored data had in the 1990s woven a fabric of machines, data, organizations, and actors that structured the order and mode of communication of the World Wide Web as computer-supported "distributed hypertext." Four decades after UNIVAC issued its invitation, the world had arrived in the computer.

# 7 Switching off

The story of how digital reality has emerged over the course of four long decades holds some surprises. In concluding, I would like to note a few of them. What I find particularly striking is the disappearance of computation in the computer. In the early 1950s, computing had been stowed away in a black box, which was only to be opened in case of emergency. Computing was made invisible and downgraded in importance as sorting, classifying, and decision-making were given higher priority. This turn of events created a demand for people who understood something about programs and machines. It is all the more surprising how these specialists of a new cultural technique were denigrated as "code monkeys" and at the same time kept as far away as possible from truly interesting questions. For mathematicians, managers, users, engineers, system developers, and project leaders, it was enough for programmers to attend to the disciplined implementation of instructions that were supposed to tame the computer and determine what users could do. The only annoyance was that programs had to be rewritten again and again and adapted to the respective tasks and machines. Standardized program libraries would have been very nice. But this desire proved elusive even after the hardware and software markets were separated. At the same time, computer engineers and administrative cadres underestimated the problem that arose on both sides of the computer in formatting input and handling output. Early applications of computers made it surprisingly clear that preparation in particular – that is, acquiring, formatting, and organizing data – involved enormous amounts of work. It took less time to process data than it did to make the world machine-readable. But it soon became clear that

even the output was less obviously useful than the payroll checks printed by the very first computers.

The politics and economics of digital space are arguably nowhere more visible than in the development of operating systems. This book links the efforts to develop a form of computer use called time-sharing in the 1960s with the development of rules that separated the permitted from the prohibited, monitored users' behavior, and protected their rights. Costly computing time was divided among different users in such a way that the computer was always available to them when they needed it. According to dictum, users should have to wait as little as possible and the machine should never be idle. However, the "system" had to enforce the operating rules and ensure that users, data, and programs did not interfere with each other's work. Operating systems structured digital space with short-term access privileges, well-conceived interventions, and rule-based interruptions. The systems were able to distribute computer use over time and make the most of expensive computing capacity.

If the world was to be moved into the computer, then digital and analog realities had to be synchronized. Synchronization was not always as demanding as at NASA's Mission Control Center. However, the example impressively shows that computers have to be freed up if they are to interact with the analog world in real time. With its clearly delineated staff responsibilities, complex organizational routines, specialized technical high-speed zones, and well-equipped waiting rooms, the system in Houston was tightly coupled with a globally operating and locally focused media operation. Houston demonstrated computerized control of spaceflight based on personnel- and equipment-intensive monitoring of computers, with the Mission Control Center bringing not only the Moon but also Earth to flight controller consoles equipped with voice radio, pneumatic mail, television, and telephones as well as to living room screens.

Since the early 1950s, a key question had been how to create sufficient processing capacity on the one hand and how to deal with the expected overcapacity of computers on the other. Only when there was more "space" in computers could further processes be shifted to them, and only when this space was fully utilized would the business be worthwhile. Manufacturers had no choice but to make bold assumptions about the future pull of digital space and to formulate corporate strategies to match. Around 1960, undisputed market leader IBM had made the processor the starting point for all strategic decisions. This example shows that strategies can be convincing and successful, even if they cannot fulfill their promises for good reasons. The situation was not much better when it came to demand-side estimates of future computer use. The customer's wish list – what computers should be able to do and how digital space should be set up – was constantly expanding and constantly unfulfilled. At the same time, something unexpected was gained, or one could at least assume that it would be possible to continue work on a next project. In digital reality, the project became the means of productively mediating between promises and expectations.

The surprise that comes from observing the interconnectivity of computers that began in the 1970s has little to do with disappointed expectations and much to do with unintended consequences. Within a decade, a patchwork of networks had emerged. Though some of these networks were notable for their clever and carefully conceived design, none could prevail over the others to achieve all-inclusive networking in digital space. The protocols conceived in the early 1970s in experimental networks belonging to the US Department of Defense that ended up winning could not have been anticipated as late as the late 1980s. Their ultimate success was due neither to their special performance nor to a particularly military orientation. Rather, it was due to the fact that

they were based on local solutions and operated as a network of networks, that is, as the Internet.

It was easy to hold California counterculture responsible for the differentiation that arose in computing, which personalized the computer and thereby brought about the personal computer. My reading leads to a different conclusion. Neither the displacement of leisure activities (hobbies) nor of domestic concerns (home) into the computer were the decisive factors behind the goals for differentiation. Rather, it was the shift of small, everyday office work into machines equipped with microprocessors that turned the computer into a personal computer in the 1980s.

Finally, the development of the storage capacity of digital space calls for reinterpretation. It started with auxiliary memory, which was used as a notepad for intermediate results, so to speak. It continued through the concatenation of data in database management of the 1960s and its liberation in the relational database table of the 1970s, to the organization of interacting storage systems and the growing problem of data garbage. Links set by users were understood to be a means of mitigating loss of information in organizations.

My story of how the world came into the computer just happens to end at a moment when some people were proclaiming the end of history and others the beginning of a new world order. But it is not such announcements that determine the end of this story. Rather, it is that the question of how digital reality arose and became indispensable is simply less compelling. Newcomers to digital space toward the end of the 20th century encountered a strongly structured arrangement of space, time, and objects. Now, when data met databases, protocols interacted with protocol families, programs encountered operating systems, and users interacted with each other, the concern was no longer how to make the transfer into the computer, but the self-evidence and the independence of the digital world.

In the 1990s, the focus on the "emergence of a digital reality" was replaced by a focus on the "autonomy of digital space." The issue was initially approached as a conventional case of technology out of control, for example, in the aftermath of the 1987 stock market crash, caused in part by program trading gone awry.[358] Soon, however, talk of autonomy shifted to fundamental considerations, sometimes viewed positively, sometimes with apprehension. Forms of individual and collective autonomy, which could be developed and cultivated in digital space, were not alone in booming with the blossoming of the World Wide Web. Computers and computer connections also became autonomous. Autonomous search engines plowed through hyperlinked data garbage and indexed everything they found. Their lists of finds were sorted by algorithms that also appeared to have a high degree of autonomy. Ultimately, the "Y2K problem" sparked widespread fear about the autonomy of digital reality. No one could say with certainty what computers would do on 1 January 2000, when their current year number took a step backward from 99 to 00 for the first time in history, even before the end of the millennium, and thus experienced a veritable historical break. Shortly thereafter, a policy paper from IBM heightened the uncertainty of dealing with computer autonomy by referring to a looming "software complexity crisis." In the near future, systems would become so complex that their interactive components could not be installed, configured, optimized, maintained, or assembled even by highly specialized systems managers. The only way out of the dilemma would be systems that managed themselves.[359]

The fact that it was precisely the autonomy of digital reality that now gave people pause is, in my view, a historically significant change that cannot be reconciled with the narrative of the great move of the world into the computer and therefore marks its end.

| | | |
|---|---|---|
| 75 47 25 | CC | Well, what mode are you in on your computer now? |
| 75 47 27 | C | I just went from Prelaunch to Catch-Up. That turned the Comp light OFF. Let's see if it comes back on now. Okay. It's back on. Now we'll turn the whole computer OFF and see what happens. |
| 75 47 53 | C | I still get the Computer Running Light when I've got the switch OFF. |
| 75 48 00 | CC | You still have the Computer Running Light with the switch OFF? |
| 75 48 02 | C | That's affirmative. |

Voice tape transcription between the ground station (CC) and astronaut (C) during the Gemini 4 mission, NASA 1965, p. 305.

# Acknowledgments

In the fall of 1997, I gave my first lecture on the history of computing. Because the hype around the net was just beginning to take hold, I tried to understand the history of the Internet and stumbled into a complex of topics that were new to me. In computer history, then as now, it was possible to discover new things all the time. Since then, in seminars and at conferences, in scattered journal articles and many conversations, I have tried to unpack issues in and around the history of computing until a stable picture temporarily emerged for me. One day, Daniela Zetti, who enjoyed taking part in my attempts at interpretation, abruptly switched from my subjunctive to her imperative: "People should know how the world got into the computer. Write it down!" Since then, Daniela has accompanied the project with endless patience and good humor, generously overlooking fruitless starts to chapters in the confidence that subsequent sections would simply make the bumpy beginnings superfluous. Where new connections appeared, she urged me follow them up and sometimes even make them the focus. I am deeply grateful for her guidance.

I also owe a great debt of gratitude to Gisela Hürlimann. She persistently reminded me over the long months that writing is a deeply interesting endeavor and can sometimes even be enjoyable, but that I must above all spare some thought for my imagined readers and not abuse their time. She insisted on clarity and demanded proof where I preferred to rely on memory. The files that I sent her for preliminary reading came back at all hours in such a way that, color-blind as I am, there was no doubt what I needed to do, for example, where the text needed further clarification and where it was best keep things simple. For particularly

dense passages where efforts at decluttering would only have left black holes, I could count on the support of Maya Wohlgemuth. Her fantastic research skills ensured that the technical bits were either solid or could at least be elegantly sidestepped.

I also had the opportunity to float some of the ideas in more recent lectures, for example, at the Digital Culture Research Lab in Lüneburg, the Collegium Helveticum in Zurich, the Center for Contemporary History Research in Potsdam, on Monte Verità in Ascona, and at the annual conference of the Society for the History and Philosophy of Computers in Brno. I received valuable suggestions from Lutz Wingert, Hansjörg Siegenthaler, Wilfred van Gunsteren, Michael Hampe, Lea Haller, Hannes Mangold, Lea Pfäffli, Luca Frölicher, Nick Schwery, Mirjam Mayer, Luca Thanei, Claus Pias, Martin Warnke, Liesbeth de Mol, Frank Bösch, Monika Dommann, Renate Schubert, Ricky Wichum, and Thomas Hengartner. Forgive me for being unable to follow up each pointer in detail and from time to time having to fall back on unsupported opinion. Erich Projer read the chapters critically and, after careful reflection, picked up the phone time and again to tell me that such-and-such was the case or that there were structural weaknesses that required further work or cuts. He would perhaps now say that this was not often necessary. Jakob Tanner, too, read the whole book in *statu nascendi* and, also after careful reflection, pointed out a great many possibilities for elaboration, many of which I managed to avoid. He might now say there are others.

To Simone Roggenbuck I owe infinitely more than I can say. She saw me in every possible state of (dis)aggregation, thought with me, let me know when I was wrong, and never lost her sense of humor. Her conviction that "the little book" could happen was decisive. Simone, this is for you.

# Postscript to the English edition

I like this translation of my attempt to recount the emergence of digital reality. Not only because the editing process allowed me to eliminate at least a few errors, but also because Giselle Weiss has done an admirable job of translating. For me, as author, the new text represents a more elegant version of the thoughts that consumed so much of my energy to write that at times I could not muster the courage needed for final deletions.

Naturally, any reference to new work that has appeared since the publication of the German edition is missing. The book still provides no critical review of the literature. In "Computer History – the Pitfalls of Past Futures," Daniela Zetti and I consider idiosyncracies in the history of computers since the late 1950s. There we offer an answer to the question of when historical-critical studies of computers and computer programs really make a difference, and we have at least thought carefully about various ways of relating computer history.

Zurich and Locarno, in the second year of the pandemic

# Photo credits

1: Alamy Stock Photo
2: CBS Photo Archive/Getty Images
3: Orlando/Hulton Archive/Getty Images
4: ETH-Bibliothek Zurich, Bildarchiv/Stiftung Luftbild Schweiz/Photographer: Swissair
5: ETH-Bibliothek Zurich, Bildarchiv/Photographer: unknown
6: www.digibarn.com/collections/ads/UNIVAC-50s/divide-by-zero/zero.jpg © Digibarn Computer Museum
7: Underwood Archive Photos/Getty Images
8: www.census.gov/library/photos/1950_08010.html
9: Lesser and Haanstra, pp. 140, 141, 144.
10: Courtesy of the Computer History Museum/Courtesy of IBM Archives
11: School of Computer Science, University of Manchester
12: Bettmann/Getty Images
13: Courtesy of IBM
14: Information Technology Division (University of Michigan) records, Bentley Historical Library
15: *Popular Electronics,* Mai 1975, p. 25
16: The LIFE Images Collection/Getty Images
17: Courtesy of the Computer History Museum IBM/Apple
18: Bachman and Williams 1964, p. 415
19: Bachman and Williams 1964, p. 414
20: Johnson 1975, p. 510

# Notes

1 The 5,000 "man-years" required between 1961 and 1965 to develop the OS 360 are the stuff of legend. Frederick P. Brooks's discussion of the "mythical man-month" puts the "man-year" into perspective. Brooks 1955.
2 I thank Daniela Zetti for the reference to Michael S. Mahoney's question. See Mahoney 2005, p. 128, and Mahoney 2011.
3 Spitzer 2012. Jacket copy at www.droemer-knaur.de/buch/manfred-spitzer-digitale-demenz-9783426300565 (accessed 25 August 2021).
4 See the ACM Digital Library at http://dl.acm.org/ (accessed 25 August 2021).
5 Oral History Collection at www.computerhistory.org/collections/oralhistories/ (accessed 25 August 2021).
6 Misa 2017.
7 See www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 2021), Remington-Rand Presents the UNIVAC. See also Eckert et al. 1951b.
8 For a conventional overview of computer history with a strong focus on entrepreneurial issues, see Campbell-Kelly et al. 2014. On the Manhattan Project, see Gosling 1994. On the role of electronic computers in the Manhattan Project, see Pepaez 1999. For an early description of ENIAC, see Goldstine and Goldstine 1996 (1946). On the importance of electronic computers in cryptology, see Sale et al. 2000. On the demilitarization and commercialization of computers in the transition from ENIAC to UNIVAC, see Stern 1981.
9 See www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 21), Remington-RAND Presents the UNIVAC. On the logistics of data, see Gugerli 2009a.
10 www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 21), Remington-RAND Presents the UNIVAC. Emphasis D.G.
11 www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 21), Remington-RAND Presents the UNIVAC.
12 McPherson and Alexander 1951; Gray 2001.
13 Heide 2009.
14 In 1956, Walter Cronkite was able to switch quite naturally to the "UNIVAC corner" for the latest figures on the election. Again, UNIVAC predicted a win for Eisenhower, and again the computer was right. www.youtube.com/watch?v=v7K8MW8wQWs (accessed 25 August 21).
15 www.youtube.com/watch?v=FMXT4f8C63A (accessed 25 August 21).
16 "The world in a machine," in the words of Paul Edwards. Edwards 2000.
17 See the 1947 demonstration of the "Mechanische Rechenmaschine Brunsviga" (Brunsviga mechanical calculating machine) at www.youtube.com/watch?v=o66EWIZn Zaok (accessed 25 August 21).
18 For the history of the Turing machine, see Herken 1988; on John von Neumann, see Glimm et al. 1990.
19 The disputes between IBM and Howard Aiken surrounding the building of the Mark I at Harvard University continue to inspire books to this day. See Cohen and Aspray 2000.
20 www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 21).

21 "To meet this need for high-speed data processing, the scientist and technicians of the Eckert-Mauchly division of Remington Rand have created a miracle of electronic development: UNIVAC, *a complete electronic system for sorting, classifying, computing, and decision-making*. Acting upon alphabetical as well as numerical data at incredible speeds and with complete accuracy" (emphasis D.G.), www.youtube.com/watch?v=j2fURxbdlZs (accessed 25 August 21).
22 See the ratings manuals of the Vita life insurance company in the archives of Zürich Versicherungen: Z-Archiv, Q 129 204 30343:1, VITA: Tarifbücher div. Versicherungsformen 1952–1991. My thanks to Luca Frölicher for pointing these out to me. On "actuarial practice" in the insurance business, see Stadlin 2010.
23 The problem involved double glazing 10 windows in a gymnasium. The windows had a total light size of 3.20 × 1.60 m and "32 B 4/4 panes measuring 37.5 × 77 cm" per window. The glazing required 12 kilograms of putty, and a 25 kilogram vat of putty cost 12 francs. The problem cannot be assumed to be taken directly from daily workshop calculations. Hirzel and Käfer 1943, pp. 25–29.
24 Stahel 1950, pp. 91–104.
25 Reed 1942.
26 Mindell 2002, pp. 87–91.
27 This kind of geometry-based "computing" was known since the mid-19th century from the polar planimeter, which was used to determine the area of an irregular map section by tracing its edges. Amsler 1856; Amsler and Erismann 1993; Bruderer 2015.
28 Mayer 1908.
29 Failure Is Not an Option. A Flight Control History of NASA, 2014, www.youtube.com/watch?v=7f51Jzm7M4w28:08 (accessed 25 August 21).
30 Crank 1947, Owens 1986, Aiken 1975 (1937), Cohen and Aspray 2000.
31 Eckert et al. 1945; Goldstine and Goldstine 1996 (1946); Van der Spiegel et al. 2000.
32 "The features to be incorporated in calculating machinery specially designed for rapid work on scientific problems, and not to be found in calculating machines as manufactured for accounting purposes, are the following." Aiken 1975 (1937).
33 Turing 1952.
34 Neumann 1945, p. 355.
35 For applications to racial research and heredity, see Füssl 2010, p. 109. For the broader fields of application, see Zuse 1948.
36 Eckert et al. 1951a.
37 "True to its name, Universal Automatic Computer, the UNIVAC system is capable of handling data processing or calculation in virtually all fields of human endeavor." Eckert et al. 1951a, p. 11.
38 Aiken 1975 (1937). See also Bashe 1999, p. 71.
39 Eckert et al. 1951a, p. 12.
40 Eckert et al. 1951a, p. 12.
41 Eckert et al. 1951a, p. 12.
42 Perret, Jacques to Christian de Waldner, general manager of IBM France, 16.4.1955. See https://journals.openedition.org/bibnum/534 (accessed 25 August 21).
43 Perret apparently had not checked the first edition of the *Dictionnaire de la langue française*, which was published between 1863 and 1872.
44 Zetti 2008; Zetti 2009.
45 Zuse 1980. See also Bruderer 2010; Tobler 2001.

46 Furger and Heintz 1997.
47 Rutishauser et al. 1951.
48 Stiefel 1954, p. 29. Emphasis in the original.
49 Stiefel 1954, p. 30.
50 Stiefel 1954, p. 32.
51 "The job of planning and programming problems may well become the bottleneck in operation." Carr 1952, p. 238.
52 Johnson 1952, p. 78.
53 The 1953 project proposal to build the Ermeth, cited in Bruderer 2015, pp. 566–568.
54 Gugerli 2009a.
55 Rutishauser 1952; Rutishauser 1956, p. 2.
56 Carr 1952. See also Bemer 1957b.
57 Zuse 1936.
58 Ridgway 1952.
59 On Fordist culture in mathematics in the early 20th century, see Heintz 1993.
60 Wexelblat 1981.
61 Kemeny and Kurtz 1964.
62 Studienzentrum für Administrative Automatisierung 1966.
63 See Studienzentrum für Administrative Automatisierung 1966; Palormo 1967; Jensen 1967 and Willoughby 1971.
64 Studienzentrum für Administrative Automatisierung 1966, p. 70.
65 Levine 1961.
66 Brown and Ridenour 1953.
67 Sheldon and Tatum 1951, p. 30.
68 Dover 1954, p. 172.
69 "Before the data could be reduced, that is, reduced on IBM machines or desk calculators, it had to be processed; that is, put into a form where it could be handled readily and easily by the IBM card programmed calculators." Dover 1954, p. 172. On IBM's electronic punched-card computer, see Sheldon and Tatum 1951.
70 Gugerli 2012.
71 Dover 1954, p. 173.
72 Dover 1954, p. 178.
73 Regarding the rigorous formatting standards for automated input in data-processing systems in business, see Eldredge et al. 1957.
74 McPherson 1953, p. 52.
75 McPherson 1953, pp. 52–53.
76 McPherson 1953, p. 52.
77 Welsh and Lukoff 1952, p. 47.
78 Welsh and Lukoff 1952, p. 47.
79 Hopper 1952, p. 244.
80 Aspray 1994.
81 Austrian 1982; Campbell-Kelly 1998; Yates et al. 2001; Yates 2005.
82 Lesser and Haanstra 1957, p. 140.
83 Simon 1962.
84 According to an IBM strategy group meeting held in 1961, it would be impossible to develop computers that could simultaneously serve the military and commercial sectors at competitive prices. Haanstra et al. 1983 (1961), p. 18.

85 On the difficulties of quantifying computer costs, see Haanstra et al. 1983 (1961), p. 23.
86 See Heintz 1993 on how deeply the idea of the assembly line was anchored in the minds of computer engineers.
87 Lesser and Haanstra 1957, p. 140.
88 Lesser and Haanstra 1957, p. 140.
89 Lesser and Haanstra 1957, pp. 140 and 141.
90 Lesser and Haanstra 1957, p. 144.
91 McCarthy 1959; Teager and McCarthy 1959.
92 McCarthy 1959.
93 McCarthy 1959.
94 McCarthy 1959; Teager and McCarthy 1959; *Science News-Letter* 1961.
95 Corbató et al. 1962, p. 335.
96 www.youtube.com/watch?v=Q07PhW5sCEk.
97 Bemer 1957a showed that this point was arguable.
98 www.youtube.com/watch?v=Q07PhW5sCEk.
99 On the programmer's perspective, see Bauer 1958.
100 See also Corbató et al. 1962, p. 335.
101 On the priority problem in time-sharing, see Greenberger 1966.
102 Corbató 1964. Anonymous 1964. On the supervisor, see also Vyssotsky et al. 1965.
103 Dennis 1968. See also the chapter on connecting, setting boundaries, and storing.
104 Fano 1967.
105 Fano and Corbató 1966.
106 Tanenbaum 2014, pp. 6–7. In the sections that follow on time-sharing and operating systems I have drawn on material from Gugerli and Mangold 2016 and adapted it to the present context.
107 McCarthy (1983), Reminiscences on the history of time sharing, www-formal.stanford.edu/jmc/history/timesharing/timesharing.html (accessed 25 August 21).
108 Klossner 1980.
109 Sumner et al. 2000.
110 Kilburn et al. 1962.
111 Kilburn et al. 1962.
112 Brooks 1995.
113 Corbató et al. 1972; Corbató and Vyssotsky 1965; Vyssotsky and Corbató 1965.
114 Corbató and Vyssotsky 1965.
115 Luhmann 1966, p. 9. See also earlier work by Luhmann 2007 (1964).
116 Hausammann 2008.
117 On the effect of forms, files, and filing systems on design structure and form in bureaucratic systems, see Vismann 2001.
118 Luhmann 1966, pp. 9–10.
119 March et al. 1958, pp. 142–150. See especially Simon's book on the new science of management decision. Simon 1977 (1960) and Simon 1960 as well as Luhmann 1968.
120 According to Chester Barnard, management theorist and later president of the Rockefeller Foundation and chairman of the National Science Foundation, in a foreword to Herbert A. Simon's book on administrative behavior. Simon 1976 (1946), p. xlvi.
121 Luhmann 1966, p. 18.
122 Gugerli 2010.

123 Campbell-Kelly 2003; Egger 2013.
124 ETH-Bibliothek 1968.
125 Girschik 2010. The mail order business began considering the use of computers very early on. Martin 1954.
126 Haigh 2001. Haigh's argument was anticipated by Daniel Bell; see Bell 1967 and Bell 1973.
127 Dearden 1964; Dearden 1965; Dearden 1972.
128 Girschik 2010.
129 Haigh 2001.
130 Not even chance was left to chance, as research on computer programming to produce random numbers shows. Certaine 1958; Greenberger 1959; Green et al. 1959; Coveyou 1960; Greenberger 1961.
131 Kranz 2001; Mindell 2008. Meanwhile, "Houston" has also become a site of national historical interest: Launius 2009.
132 On the culture of electronic surveillance and control see Gugerli and Mangold 2016.
133 Gates and Pickering 1965.
134 See longtime flight director Gene Kranz's mantra "Failure is not an option" (probably ex post). Kranz 2001.
135 Johnstone 1969.
136 Tomayko 1988, pp. 249–250.
137 Tomayko 1988, p. 252.
138 Hamlin 1964, p. A2.2-1; Donegan et al. 1964. The Goddard Space Center also claimed to be using real-time computing. See, for example, Gass 1961 and Adams and Federico 1964.
139 James 1981, p. 422.
140 Tomayko 1988, p. 251.
141 Hutchinson 2012.
142 Philco 1967. "The manual is primarily an orientation/indoctrination guide and, in addition, furnishes a reference source for information pertinent to the MCC-H systems, subsystems, and major components." Philco 1967, pp. v/vi.
143 Philco 1967, passim.
144 Philco 1967, para. 3-3-1-4, p. 3.12.
145 On pneumatic tube mail at MCC Houston see Philco 1967 para 2-3-7, p. 3.9. On the display of TV images see Hutchinson 2012, "The flight controller's console."
146 See Philco, para 2-2-2, p. 2.6. On time management in the computer center under real-time computing conditions see also Johnstone 1969, p. 29.
147 Bhola et al. 1968.
148 Philco 1967, para 2-2-2, pp. 3.3–3.4. See also Hutchinson 2012, p. 3.
149 Philco 1967, para 1–4. On Eidophor, see Meyer 2008.
150 On the meaning of "closed worlds" in computer history, and in particular the electronic SAGE project, see Edwards 1996. Stanley Kubrick's *Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb* (1964) dramatically satirizes events in the war room.
151 Apollo Flight Journal, Apollo 8, Day 3, 055:02:46 to 055:28:34, https://history.nasa.gov/afj/ap08fj/09day3_green.html (accessed 25 October 2016). On the broadcasting range of space television, see Rosenfelder 2003.

152 Apollo Flight Journal, Apollo 8, Day 3, 055:02:46 to 055:28:34, https://history.nasa.gov/afj/ap08fj/09day3_green.html (accessed 25 October 2016).

153 To Lovell's comment that a traveler at that height would not be able to tell whether Earth is inhabited or not, Mike Collins responded (from the CapCom console in Houston): "Don't see anybody waving; is that what you are saying?", Apollo Flight Journal, Apollo 8, Day 3, 055:18:22 to 055:18:31, https://history.nasa.gov/afj/ap08fj/09day3_green.html (accessed 25 October 2016).

154 On the role of the mirror stage in the development of ego function, see Lacan 1986 (1948). At the end of the 1960s, apparatus theory asserted that the attention of the audience resulted from a blind spot of the media installation (see Rosen 1986). The TV broadcast from Apollo 8 in the control room at Houston and on TV sets around the entire world undermined this cinematic apparatus, insofar as it demonstrated self-reflection from outer space. On the satellite view, see Sachs 1994.

155 Boulding 1966; Fuller 1969.

156 The fact that this is a communicative performance by the control center does not diminish the contribution of its real-time computers to the production of local synchronicity between different, asynchronous systems. On the relationship between simultaneity and synchronization, see Luhmann 1993, p. 119. NASA's accountability made possible ex post other forms of (visual) simultaneity, such as the graphical synopsis of both the pre-calculated soft and elegant landing of the Apollo 11 mission on the Sea of Tranquility and Neil Armstrong's rodeo-style "manual" approach to landing under the threat of depleting fuel. Mindell 2008, p. 227.

157 Frei 2008; Kraushaar 2000.

158 See IBM's strategic development goals as formulated by the SPREAD Task Group in late 1961. Haanstra et al. 1983 (1961), p. 6.

159 Unless long hair was understood to be part of a syndrome of excessive independence and mild paranoia, as Dick H. Brandon maintained at the 1968 ACM national conference. Brandon described programmers during an ACM policy debate on managing the economics of computer programming as egocentric, slightly neurotic, bordering on mildly schizophrenic. The evidence, he said, was the simultaneous appearance of beards and sandals and other symptoms of rugged individualism and nonconformism, as found in programmers. Brandon 1968, p. 333.

160 At least that was the claim made inside the ACM with reference to *Moody's Computer Industry Survey Fall 1967*. Brandon 1968, p. 332.

161 And giving rise to a whole series of discussions. Morton and McCosh 1968; Miller 1969; Boulden and Buffa 1970; Jones 1970.

162 According to Denning 1971, p. 175, the term *generation* became commonplace after 1964, the "year in which third generation machines were announced." Originally, the term had applied only to hardware.

163 Haanstra et al. 1983 (1961), p. 6.

164 IBM even constructed the family tree of processors: www.computerhistory.org/revolution/mainframe-computers/7/162/573 (accessed 25 August 21).

165 Haanstra et al. 1983 (1961), p. 8.

166 Haanstra et al. 1983 (1961), p. 14.

167 Haanstra et al. 1983 (1961), p. 9.

168 www.computerhistory.org/revolution/mainframe-computers/7/162/575?position=0 (accessed 25 August 21).

169 Even in 1965 it was no secret that computer performance was also defined by operating systems, memory access, applications software, and peripherals.
170 See Moore 1965. On Moore's law as a possible instance of technological determinism, see Ceruzzi 2005.
171 Haanstra et al. 1983 (1961), p. 18.
172 Grosch 1953, p. 310. Grosch's rather intuitively formulated hypothesis of economies of scale was confirmed by Knight (1966) after considerable experimental effort. See also Solomon 1966. It was not until the late 1970s that people began to second-guess Grosch. Cale et al. 1979 showed that the price–performance ratio could be predicted only for mainframes, not for small computers. The original validity of Grosch's claim may have stemmed from the fact that IBM's pricing was guided by his observation, such that in 1966 Knight could only measure what IBM was compelled to specify according to "Grosch's law." With the IBM 360/20, which was not originally planned, IBM launched its own competing minicomputer product in the mid-1960s, manufactured by IBM Germany. Pugh et al. 1991, pp. 445 and 639.
173 Voelcker 1988.
174 The "IBM 1060 data communications system" announced in 1963 served this purpose. It was basically a printer with a keyboard. The IBM 1050 was connected to a telephone line via a modem and could either receive or transmit in half-duplex mode. See Campbell-Kelly 1988, p. 224.
175 Gorn 1966.
176 "Ordinary telephones" was a term intended to downplay user–computer interaction at a distance. See, for example, Merrill et al. 1963, p. 622. In Haanstra et al. 1983 (1961), the term *communication* occurs primarily in connection with the term *console*, while *transmission* occurs in the sense of "transmission checking" in "input/output control systems." "Teleprocessing" applications are said to require integration into a centralized data processing center.
177 The confusion was apparently so great that around 1970, efforts were made to classify operating systems with the help of a technical description. See Katzan 1970. CDC attempted to help with a tree diagram and an annotated table. CDC 1976.
178 Poole and Waite 1969, p. 22.
179 Brandon 1968.
180 Dijkstra 1968.
181 Anonymous 1968.
182 Poole and Waite 1969.
183 The alternative response to IBM's System/360 was to exploit the compatibility gap created by IBM. Honeywell introduced an offering compatible with IBM computers that were no longer supported by System/360. Following the introduction of IBM's System/370, Honeywell was able to continue its independent development, whereas RCA stopped making computers. For further details, see Gandy 2014.
184 RCA 1965. www.computerhistory.org/brochures/full_record.php?iid=doc-4372956eb9810 (accessed 25 August 21).
185 Zetti 2014, pp. 17–53 on the ultimate RCA videorecorder.
186 Humphrey 2002, p. 59.
187 Humphrey 2002, p. 61.

188 Wijngaarden et al. 1969, pp. 79–80. "Minority report" in *Algol Bulletin* 31, March 1970, http://archive.computerhistory.org/resources/text/algol/algol_bulletin/A31/P111.HTM (accessed 25 August 21).
189 Campbell-Kelly et al. 2014, pp. 178–182. For a contrasting view, see Haigh 2010.
190 Wijngaarden et al. 1969, p. 79. "Minority report" in *Algol Bulletin* 31, March 1970, http://archive.computerhistory.org/resources/text/algol/algol_bulletin/A31/P111.HTM (accessed 25 August 21).
191 Haigh 2010, p. 5.
192 See Gugerli 2005.
193 Dijkstra 1972.
194 Dijkstra 1972, pp. 859–860.
195 Dijkstra 1970. Structured programming relied on the division of programs into individual, easily verifiable sequences with clear starting and terminating points. This facilitated assessment and debugging of programs.
196 Bienek and Kreibich 1970, p. 416.
197 Kneipp 1880; Kneipp 1889.
198 Bienek and Kreibich 1970, p. 417.
199 On the growth of the hotel and spa business in Bad Wörishofen, see Burghardt et al. 1967, pp. 141–144.
200 Bienek and Kreibich 1970, p. 417.
201 Bienek and Kreibich 1970, p. 417.
202 Bienek and Kreibich 1970, p. 417. Card punches were standard equipment for small and medium-sized companies. Aikele 1962; Haake 1965; Stubenrecht 1960; Zentralinstitut für Information und Dokumentation 1967. The manual for the IBM Card Punch 29 machine dated June 1970 marked its seventh edition. IBM 1970.
203 Bienek and Kreibich 1970, p. 418.
204 Bienek and Kreibich 1970, p. 421.
205 Everything known about Ubisco is based on an insider report published 35 years after the end of the project in Neukom 2009. Hans Neukom was given access to files in the archives of today's UBS for this purpose, but he was not permitted to quote document titles, names, or figures from them.
206 CDC 1968.
207 For example, the Schweizerische Kreditanstalt had just purchased the latest IBM System/370 machines and launched an information management system, reaffirming the bank's determination to move its transactions fully into digital space. The future SBG system could not be allowed to lag behind the competition. Gugerli 2010.
208 CDC, Control Data 3200 Computer System/Real Time Applications 1963. http://archive.computerhistory.org/resources/text/CDC/CDC.3200.1963.102646086.pdf (accessed 25 August 21). An example of CDC's early use of the buzzword *real time*.
209 Neukom 2009, p. 32.
210 CDC developed TOOS along the lines of the Zodiac operating system created for the US Air Force Worldwide Military Command and Control System. See Laccabue 2009 and CDC 1968. TOOS linked two powerful CDC 6000 computers via a large central memory and could process large volumes of data.
211 In any case, both systems were further developed into successful CDC products in the late 1970s. Neukom 2009, p. 33.
212 Ubisco brochure, SBG, Zurich 1973, cited in Neukom 2009, p. 37.

213 Roughly at the same time, a Swiss PTT project was struggling to transfer the entire telephone network to a computerized switching center model that had been successfully tested in 1972. The "integrated telecommunications system" project ended abruptly and unsuccessfully in 1983, having also fallen victim to scaling up. Gugerli 2002a.
214 Brooks 1995, pp. 274–275; Abdel-Hamid and Madnick 1991. On the reasons why software projects fail, see also Charette 2005.
215 Neukom 2009, p. 34.
216 Gugerli 2010.
217 Rogge 1997, pp. 268–271.
218 The following discussion draws on my own work in Gugerli 2007b and Gugerli 2009b, and especially on the research of Hannes Mangold 2017. See also Bergien 2017.
219 Bundesinnenminister 1970, p. 20, cited in Mangold 2017, p. 85.
220 Herold 1968a; Herold 1968b; Herold 1968c; Herold 1970.
221 For example, in *Der Spiegel*: 1971, p. 53.
222 Mangold 2017, p. 145.
223 *Der Spiegel* no. 44, 23 October 1972, pp. 65–68.
224 Anonymous 1975. Gugerli 2007b, p. 176.
225 *Der Spiegel* no. 11, 13 March 1978, pp. 22–27.
226 Enzensberger 1979.
227 Denninger 1990.
228 Herold 1985; Simon and Taeger 1981; Wanner 1985.
229 Bölsch 1979.
230 Siebrecht 1995.
231 On the increasing density of the computer's "internal traffic," see Batcher 1968; on the design possibilities for intensified "network traffic," see Kaplan 1968.
232 On the "disconnection of connection," see Sprenger 2015.
233 ACM 1971.
234 On this topic see also ARPA 1970, Roberts and Wessler 1970, and Kahn 1972.
235 Auwaerter 1970, p. 34.
236 Auwaerter 1970, p. 35.
237 Auwaerter 1970, p. 42.
238 Redmond and Smith 2000; Edwards 1996.
239 Auwaerter 1970, p. 42.
240 On the other hand, there was no shortage of "visions" that could be updated at will. J. C. R. Licklider's 1960 article on human–computer symbiosis served for decades, usually without explicit reference, as a source of legitimation and argumentation. Licklider 1960.
241 The expectations for information technology had already been articulated.
242 Sachman 1968, p. 93.
243 ACM 1971. The previous year, the association had published a record-breaking number of articles dealing with the future. ACM articles with the word *future* in the title or abstract had reached an interim peak in 1968: 1965, 12; 1966, 16; 1967, 17; 1968, 30; 1969, 22; 1970, 13.
244 At the time, they would likely not have caused much of a stir as the publication was obscure. *Science and Technology for the Technical Men in Management* appeared only in 1968 and 1969 and was a failed attempt to resuscitate *International Science and*

*Technology*, which had been discontinued in 1967. The essay by Licklider, Taylor, and Herbert was republished by Taylor as a digital reprint in 1990, a good two decades later (Licklider and Taylor 1990), and quickly became a hallowed artifact of media history. See Waldrop and Licklider 2002; Mayer 2003b; Norman 2005.

245 National Research Council 1999, p. 98; Hauben 2001.

246 Some of Licklider's essays are hard to distinguish by title. See Man–Computer Symbiosis (1960), On-line Man–Computer Communication (1962), Man–Computer Communication (1968), The Computer as a Communication Device (1968). Licklider 1960; Licklider 1968; Licklider and Clark 1962; Licklider et al. 1968. See also the 1967 publications listed in Licklider 1968: Dynamic Modeling (1967), Interactive Dynamic Modeling (1967), and Interactive Information Processing (1967). Licklider 1967a; Licklider 1967b; Licklider 1967c.

247 Abbate 1999, pp. 123–127 and pp. 133–140.

248 Licklider et al. 1968, p. 23.

249 Licklider et al. 1968, p. 22. See also Dennis 1968, p. 372. On Licklider's familiarity with Wolfgang Köhler's Gestalt psychology, see the reference in Pratschke 2011, p. 280.

250 Cowlishaw 1990, p. 8. This aesthetic and mechanical concept of operational reliability was violated when systems were cobbled or plugged together ad hoc. At IBM, this practice was called "cabling together" as a play on "cobbling together." Cowlishaw 1990, p. 12.

251 Fano 1967, p. 35. From the perspective of the operator of a data center with time-sharing needs, the use of existing telephone lines in particular involved a bit of educational effort vis-à-vis the telephone companies: Steadman and Sugar 1968, p. 23.

252 A major exception is the research into military technology conducted by Rand under the direction of Paul Baran. Baran and Boem not only investigated the possibility of packet switching but also simulated different network topologies in the computer. See Baran and Boem 1964.

253 Fontanellaz 1964; Neu and Kündig 1968. See also Gugerli 2002b.

254 Bächi 2002.

255 Georgii 1966; Campbell-Kelly 1988. At least they recognized the challenge. "Nearly all PTT administrations are dealing with the problem of data transmission through their networks," reported the *Technische Mitteilungen der Schweizerischen PTT* as early as 1964. Fontanellaz 1964, p. 429.

256 Davies and Barber 1973; Fraser 1972; Gold and Selwyn 1968.

257 The following discussion draws on Campbell-Kelly and Garcia-Swartz 2005 and Abbate 1999.

258 Campbell-Kelly and Garcia-Swartz 2005, p. 31.

259 Campbell-Kelly and Garcia-Swartz 2005, p. 14, cite companies such as Automatic Data Processing Inc., CSC, EDS, Key data, and University Computing.

260 Campbell-Kelly and Garcia-Swartz 2005, p. 18.

261 Cretien et al. 1973.

262 SWIFT used a packet switching system developed by Bold Beranek and Newman (BBN). On the development of SWIFT, see Scott et al. 2008.

263 Mendicino 1972, p. 95.

264 In terms of administration and information, the protocol is the sheet pasted to the papyrus scroll indicating the purpose or subject of the text (from *protocollum*).

265 According to Cowlishaw 1990, p. 43.

266 Bhushan and Stotz 1968, p. 95. The analogy between diplomatic and computer protocols has certainly been made for teaching purposes. See, for example, Miller 1981.

267 "Getting equipment from multiple vendors to communicate is the stuff bad dreams are made of, but TCP/IP provides one possible solution," wrote David Forsberg in an article in *Network World* dated 27 August 1990. The article was headlined "The Interconnectivity Nightmare." Forsberg 1990, p. 42.

268 Campbell-Kelly and Garcia-Swartz 2005, p. 22.

269 International Telegraph and Telephone Consultative Committee 1977. Campbell-Kelly and Garcia-Swartz 2005, p. 23.

270 Campbell-Kelly and Garcia-Swartz 2005, pp. 26–28. Abbate 1999, p. 168.

271 Abbate 1999, pp. 171–172. As late as 1990, Tillman and Yen were confident that the OSI model would force IBM to adapt their proprietary Systems Network Architecture (SNA) to OSI. Tillman and Yen 1990, p. 214. TCP/IP interested them only marginally and in terms of how to get "from there to here," that is, from the TCP/IP world to the SNA world. This had already been explained by Lew and Jong 1988.

272 CCITT 1984; Deasington 1985.

273 The second edition of Andrew S. Tanenbaum's textbook on computer networks (1989) was still based completely on the OSI model. The third edition therefore had to be completely revised, to wit, OSI had come too late; it had been poorly conceived and too strongly oriented to the IBM model. Tanenbaum 1997, pp. 55–60; see also Kerner and Bruckner 1989.

274 Roberts and Wessler 1970; Cerf and Kahn 1974; Metcalfe and Boggs 1976; Roberts 1978.

275 Cerf and Cain 1983.

276 Abbate 1999, p. 123, refers to the close collaboration between ARPA projects, the British National Physics Laboratory's network projects, and French research on the Cyclades network.

277 Cerf and Cain 1983, p. 311.

278 Abbate 1999, p. 130.

279 The best example is the virtual conferences used to develop numerous protocols in the Internet protocol family. The request for comments (RFC) was a highly condensed and rapid form of collaboration that served to bind the members of a project group together. King et al. 1997, pp. 8–13.

280 MITS 1975, p. 25.

281 In the 1920s, radio tinkering had been declared a respectable occupation because it kept more young people at home. See Boddy 2004, pp. 27–28.

282 Roberts and Yates 1975a; Roberts and Yates 1975b.

283 The story and the accompanying code were first published in May 1975 in Menlo Park-based *People's Computer Company* magazine, Vol. 3, pp. 8–9, and reprinted without illustrations in Dompier 1976 (1975).

284 For more, though undocumented, detail, see Levi 2010, Chapter 10.

285 Meanwhile, at the University of Pennsylvania, an office automation project by the group of David Ness also produced a DAISY (Decision Aiding Information System). Morgan 1976, p. 607.

286 Raskin 1976.

287 Dompier 1976 (1975) provides the code for *The Fool on the Hill* and transistor radio, first published in May 1975.

288 In order to critique Friedrich Kittler's sweeping assertion that consumer electronics had its origins in the "misuse of army equipment," Claus Pias had to generalize the idea of misappropriation. Pias 2015, pp. 39–41. One could also argue that the youthful hobbyists were not cultivating misappropriation but rather celebrating generalization.

289 See Hey and Pápay 2015, pp. 143–147, for the respective roles played by Paul Allan, Monte Davidoff, Bill Gates, John Kemeny, Thomas Kurtz, Ed Roberts, Steve Russell, Dartmouth College, Digital Equipment Corporation, Honeywell, Harvard University, the University of Washington, C-Cubed, Traf-O-Data, and various PDP-10 computers in the development of the Altair BASIC interpreter alone.

290 Green 1975.

291 Gray 1976, pp. 238–239.

292 Gray 1976, p. 239.

293 Warren 1977, p. 493.

294 Warren 1977, p. 495.

295 Warren 1977, pp. 496–497.

296 Kay et al. 1978, p. 29.

297 Isaacson et al. 1978, p. 46.

298 *The Oregon Report on Computing* came out of a conference held in Portland, Oregon, on 20–22 March 1978. Isaacson et al. 1978.

299 Isaacson 1978.

300 Isaacson et al. 1978, p. 50.

301 Landau 1979.

302 Morgan 1976, p. 605.

303 Weiderman 1979.

304 Isaacson et al. 1978, p. 46.

305 "What is a personal computer?" Holden 1979, p. 3.

306 Peter J. Schuten, "Home Computer: Demand Lags," *New York Times*, 7 June 1979, p. D2.

307 "Big IBM's Little Computer," *New York Times*, 13 August 1981.

308 Product fact sheet (1981). www-03.ibm.com/ibm/history/exhibits/pc25/pc25_fact.html (accessed 5 June 2017).

309 Goldstein and Goldstein 1982; Henk 1983; Trost and Dederichs 1983. A precursor to IBM's personal computer was the 1975 IBM 5100 Portable Computer. Weighing 50 pounds, it was of limited "portability." www-03.ibm.com/ibm/history/exhibits/pc/pc_2.html (accessed June 5, 2017). See also Katzan 1977. The 1979 IBM 5520, aimed at professional word processing, weighed even more, but could drive 12 printers and operate 18 screens. It was compatible with the IBM 6670 Information Distributor. See www-03.ibm.com/ibm/history/exhibits/pc/pc_5.html (accessed 5 June 2017).

310 Gotwals 1983, in response to Brannstrom 1982.

311 In line with Daniel T. Rodgers' assessment of the Reagan era as an "Age of Fracture" that, like Thatcherism, made individual choice and individual configuration (of the wealthy) into a political platform. Rodgers 2011.

312 For Ridley Scott's "Big Brother" commercial, see Scott 1991. The following description of the Macintosh ad, with slight adaptations, is drawn from Gugerli and Mangold 2016, pp. 157–158.

313 www.computerhistory.org/revolution/personal-computers/17/303/1201 (accessed 25 August 21).

314 www.computerhistory.org/revolution/personal-computers/17/303/1201 (accessed 25 August 21).
315 See www.youtube.com/watch?v=j2fURxbdIZs – Remington-Rand Presents the UNIVAC (accessed 5 June 2017). See also Eckert et al. 1951.
316 For a history of the card index, see Krajewski 2002; on card indexing in policing, see Kaleth 1961; in the retail trade, Dalheimer 1962. The path of the card index from the humanities to IBM and back is traced by Jones 2016.
317 Eckert et al. 1951, p. 11.
318 Babbage 1982 (1837).
319 Eckert et al. 1951, p. 12.
320 Temporary storage was moved into the computer – with constant compression and capacity expansion – and made respectable, sometimes even dubbed disk memory. Soon it was used to store much more than just interim results. As early as the mid-1950s, the huge mercury tube-driven delay line memories disappeared. Easily addressable magnetic core memories or drum memories now formed the main memory, while hard disk memories recorded things that had been temporarily stored on magnetic tape at UNIVAC. On the development of magnetic recording storage until the mid-1970s, see Hoagland 1976. Hoagland emphasizes (in a footnote) that there is no obvious difference in meaning between memory and storage, except for proximity to the processor. Hoagland 1976, p. 1283.
321 See Brown and Ridenour 1953; Ridenour 1955. A very conventional but well-informed history of memory can be found in Gerecke and Poschke 2010. It was not until 2005 that it occurred to people to formulate a type of Moore's law for storage media as well. What emerged was called Kryder's law. It had apparently held for decades but proved invalid almost as soon as it was formulated. Rosenthal et al. 2012.
322 Bachman and Williams 1965, p. 413; Haigh 2007.
323 Backman and Williams 1964, p. 411.
324 On the history of the pallet, see Dommann 2009.
325 Bachman and Williams 1964, p. 415.
326 Codd 1970. The account of this second fundamental shift in the history of computer database systems recycles and condenses a section from Gugerli 2007a.
327 Codd 1970, p. 377.
328 Codd 1970, p. 377.
329 Everest 1974.
330 Bachman 1973.
331 Codd and Date 1975.
332 Date and Codd 1975, p. 95.
333 Codd 1970; Codd 1971a; Codd 1971b; Astrahan and Chamberlin 1975, p. 580.
334 Astrahan and Chamberlin 1975, p. 580.
335 Everest 1974; Codd and Date 1975; Date and Codd 1975; Sibley 1975; Haigh 2006.
336 The account draws on Chamberlin et al. 1981.
337 Astrahan and Chamberlin 1975, pp. 580–588.
338 Chamberlin et al. 1981, p. 636.
339 Chamberlin et al. 1981, p. 636. On the IBM's user orientation see Cowlishaw 1990b, p. 57.
340 Chamberlin et al. 1981, p. 633.

341 On the IBM 3850, see Johnson 1975. On the design of the mass storage system, see Penny et al. 1970.
342 Hoagland 1976 organized his concise overview for that year around magnetically recorded memory technologies. Söll and Kirchner 1978 survey a storage landscape on the basis of access time and capacity. An overview of "floppy disk drives and diskettes, hard disk drives, removable disks, optical storage, and tape drives" is given by Daniels et al. 1987.
343 Walker 1987; Akscyn et al. 1987.
344 Nelson 1967.
345 Raskin 1987, p. 325.
346 Nelson 1987 [1967].
347 ACM 1987.
348 Conklin 1987, 33.
349 Conklin 1987, pp. 17 and 21.
350 Raskin 1987, p. 327.
351 Berners-Lee 1989/1990.
352 Berners-Lee 1989/1990, p. 5.
353 Berners-Lee 1989/1990, p. 11.
354 On the development of CERN from a history of technology perspective see Krige 1996.
355 Noll and Scacchi 1991.
356 Berners-Lee et al. 1996.
357 Berners-Lee 1989/1990, p. 4.
358 On software-supported program trading, see Katzenbach 1987; on the related stock market crash, see Carlson 2006.
359 See Kephart and Chess 2003, p. 41.

# Bibliography


Abbate, Janet 1999: *Inventing the Internet*, Cambridge MA.

Abdel-Hamid, Tarek K. and Stuart E. Madnick 1991: *Software Project Dynamics: An Integrated Approach*, Prentice-Hall Software Series, Englewood Cliffs NJ.

ACM (Ed.) 1971: *Computers and Crisis: How Computers Are Shaping our Future*, New York NY.

ACM 1987: Proceedings of the ACM Conference on Hypertext, Chapel Hill NC.

Adams, W. I. and P. R. Federico 1964: Cadfiss Test System: Computation and Data Flow Integrated Subsystem Tests, Proceedings of the 1964 19th ACM National Conference, pp. 12301–12307.

Aikele, Erwin 1962: *Betriebsabrechnung mit IBM-Lochkarten*, Darmstadt.

Aiken, Howard 1975 (1937): Proposed Automatic Calculating Machine, in: Randell, Brian (Ed.): *The Origins of Digital Computers: Selected Papers*, Berlin, Heidelberg, New York NY, pp. 195–201.

Akscyn, Robert et al. 1987: KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations, Proceedings of the ACM Conference on Hypertext, Chapel Hill NC, pp. 1–20.

Amsler, Jakob 1856: *Über die mechanische Bestimmung des Flächeninhaltes, der statischen Momente und der Trägheitsmomente ebener Figuren, insbesondere über einen neuen Planimeter*, Schaffhausen.

Amsler, Robert and Theodor H. Erismann 1993: Jakob Amsler-Laffon (1823–1912), Alfred Amsler (1857–1940). Pioniere der Prüfung und Präzision, in: *Schweizer Pioniere der Wirtschaft und Technik*, 58, Meilen.

Anonymous 1964: A Panel Discussion on Time-Sharing, in: *Datamation*, 10 (11), pp. 38–44.

Anonymous 1968: Thousands Wept: The End of OS, in: *Datamation*, 14 (4), p. 72.

Anonymous 1971: Kommissar Computer, in: *Der Spiegel* 27, p. 53.

Anonymous 1975: Das Informationssystem PIOS, in: *Inpolnachrichten* (12), pp. 1–3.

ARPA 1970: *Resource Sharing Computer Networks (Collection of Papers Presented at Spring Joint Computer Conference, Atlantic City NJ, May 1970)*, Washington DC.

Aspray, William 1994: The History of Computing within the History of Information Technology, in: *History and Technology*, 11, pp. 7–19.

Astrahan, Morton M. and Donald D. Chamberlin 1975: Implementation of a Structured English Query Language, in: *Communications of the ACM*, 18 (10), pp. 580–588.

Austrian, Geoffrey D. 1982: *Herman Hollerith: Forgotten Giant of Information Processing*, New York NY.

Auwaerter, John 1970: Challenges of the Seventies, Proceedings of the 1970 25th Annual Conference on Computers and Crisis: How Computers Are Shaping Our Future, New York NY, pp. 34–43.

Babbage, Charles 1982 (1837): On the Mathematical Powers of the Calculating Engine, in: Randell, Brian (Ed.): *The Origins of Digital Computers: Texts and Monographs in Computer Science*, Berlin, Heidelberg, pp. 19–54.

Bächi, Beat 2002: *Kommunikationstechnologischer und sozialer Wandel: "Der schweizerische Weg zur digitalen Kommunikation" (1960–1985)*, Zurich.

Bachman, Charles W. and S. B. Williams 1964: *A General Purpose Programming System for Random Access Memories*, Minneapolis MN.

Bachman, Charles W. 1973: The Programmer as Navigator, in: *Communications of the ACM*, 16 (11), pp. 653–658.

Baran, Paul and Sharla P. Boehm 1964: On Distributed Communications II. Digital Simulation of Hot-Potato Routing in a Broadband Distributed Communications Network, www.rand.org/pubs/research_memoranda/RM3103.html, Santa Monica CA.

Bashe, Charles J. 1999: Constructing the IBM ASCC (Harvard Mark I), in: Cohen, I. Bernard et al. (Eds.): *Maikin' Numbers: Howard Aiken and the Computer*, Cambridge MA, London, pp. 65–75.

Batcher, K. E. 1968: Sorting Networks and Their Applications, Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, Atlantic City NJ, pp. 307–314.

Bauer, W. F. 1958: Computer Design from the Programmer's Viewpoint, Proceedings Eastern Joint Computer Conference, Philadelphia PA, December 3–5, 1958, pp. 46–51.

Bell, Daniel 1967: Notes on the Post-Industrial Society I & II, in: *The Public Interest*, 6 and 7, pp. 24–35 and 102–118.

Bell, Daniel 1973: *The Coming of Post-Industrial Society: A Venture in Social Forecasting*, New York NY.

Bemer, Robert W. 1957a: How to Consider a Computer, in: *Automatic Control Magazine* (March), pp. 66–69.

Bemer, Robert W. 1957b: The Status of Automatic Programming for Scientific Computation, Proc. 4th Annual Computer Applications Symposium, Armour Research Foundation October 24–25, 1957, pp. 107–126.

Bergien, Rüdiger 2017: "Big Data" als Vision. Computereinführung und Organisationswandel in BKA und Staatssicherheit (1967–1989), in: *Zeithistorische Forschungen/Studies in Contemporary History, Online-Ausgabe*, 14 (2), pp. 258–285.

Berners-Lee, Tim 1989/1990: Information Management: A Proposal, www.w3.org/History/1989/proposal.html.

Berners-Lee, Tim et al. 1996: Hypertext Transfer Protocol – HTTP/1.0. Internet RFC 1945, May 1996, www.ietf.org/rfc/rfc1945.txt.

Bhola, S. K. et al. 1968: Electronic Time-Division Multiplexing, in: *Electronic Engineering*, 40 (484), pp. 298–299.

Bhushan, Abhay K. and Robert H. Stotz 1968: Procedures and Standards for Inter-Computer Communications, Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, Atlantic City NJ, pp. 95–104.

Bienek, Bernd and Volker Kreibich 1970: Planung und Aufbau eines Informationssystems im Kneippheilbad Wörishofen, in: *IBM Nachrichten*, 203, pp. 416–421.

Boddy, William 2004: *New Media and Popular Imagination: Launching Radio, Television and Digital Media in the United States*, Oxford.

Bölsche, Jochen 1979: *Der Weg in den Überwachungsstaat. Mit neuen Dokumenten und Stellungnahmen von Gerhart Baum*, Reinbeck b. Hamburg.

Boulden, James B. and Elwood S. Buffa 1970: Corporate Models: On-Line, Real-Time Systems, in: *Harvard Business Review*, 48 (4), pp. 65–83.

Boulding, Kenneth 1966: The Economics of the Coming Spaceship Earth, in: Jarrett, Henry (Ed.), *Environmental Quality in a Growing Economy*, Baltimore MD, pp. 3–14.

Brandon, Dick H. 1968: The Problem in Perspective, Proceedings of the 1968 23rd ACM National Conference, pp. 332–334.

Brannstrom, Arlin J. 1982: First Impressions of the IBM Personal Computer, in: *NCCI Staff Paper Series*, 4.

Brooks, Frederick P. 1995: *The Mythical Man-Month: Essays on Software Engineering*, Reading MA.

Brown, George W. and Louis N. Ridenour 1953: The Processing of Information-Containing Documents, Proceedings of the February 4–6, 1953, Western Computer Conference, Los Angeles CA, pp. 80–85.

Bruderer, Herbert 2010: *Konrad Zuse und die ETH Zürich. Zum 100. Geburtstag des Informatikpioniers Konrad Zuse (22. Juni 2010)*, Technischer Bericht/Departement Informatik. Professur für Informationstechnologie und Ausbildung, Zurich.

Bruderer, Herbert 2015 (2020): *Milestones in Analog and Digital Computer*, 3rd edn., trans. John McMinn, 2 vols., Cham.

Bundesinnenminister (Ed.) 1970: *Sofortprogramm zur Modernisierung und Intensivierung der Verbrechensbekämpfung*, Bonn.

Burghardt, Ludwig et al. 1967: *Wörishofen. Beiträge zur Geschichte des Ortes. Zusammengestellt aus Anlass der 900. Wiederkehr seiner ersten urkundlichen Erwähnung im Jahre 1067*, Bad Wörishofen.

Cale, E. G. et al. 1979: Price/Performance Patterns of U.S. Computer Systems, in: *Communications of the ACM*, 22 (4), pp. 225–233.

Campbell-Kelly, Martin 1988: Data Communications at the National Physical Laboratory 1965–1975, in: *Annals of the History of Computing*, 9, pp. 221–247.

Campbell-Kelly, Martin 1998: Data Processing and Technological Change: The Post Office Savings Bank 1861–1930, in: *Technology and Culture*, 39 (1), pp. 1–32.

Campbell-Kelly, Martin 2003: *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry*, Cambridge MA.

Campbell-Kelly, Martin et al. 2014: *Computer: A History of the Information Machine*, New York NY.

Campbell-Kelly, Martin and Daniel D. Garcia-Swartz 2005: The History of the Internet: The Missing Narratives (Draft, SSRN), https://ssrn.com/abstract=867087.

Carlson, Mark 2006: A Brief History of the 1986 Stock Market Crash with a Discussion of the Federal Reserve Response, Finance and Economics Discussion Series. Divisions of Research & Statistics and Monetary Affairs, Washington DC, pp. 1–24.

Carr, John W. III 1952: Progress of the Whirlwind Computer towards an Automatic Programming Procedure, Proceedings of the 1952 ACM National Meeting (Pittsburgh), New York NY, pp. 237–241.

CCITT 1984: *The X.25 Protocol and Seven Other Key CCITT Recommendations, X.1, X.2, X.3, X.21, X.21 bis, X.28, and X.29*, Belmont CA.

CDC, Control Data Corporation 1976: CDC Operating System History, https://archive.org/details/bitsavers_cdccyberCDtoryMar76_319856.

CDC, Control Data Corporation 1968: Control Data 6400/6600 Computing Systems' Configurator, www.computerhistory.org/collections/catalog/102646143.

Cerf, V. G. and E. Cain 1983: The DoD Internet Architecture Model, in: *Computer Networks*, 7, pp. 307–318.

Cerf, V. G. and R. E. Kahn 1974: A Protocol for Packet Network Interconnection, in: *IEEE Transactions on Communication Technology*, COM-22 (5), pp. 627–641.

Certaine, J. 1958: On Sequences of Pseudo-Random Numbers of Maximal Length, in: *Journal of the ACM*, 5 (4), pp. 353–356.

Ceruzzi, Paul E. 2005: Moore's Law and Technological Determinism: Reflections on the History of Technology, in: *Technology and Culture*, 46 (3), pp. 584–593.

Chamberlin, Donald D. et al. 1981: A History and Evaluation of System R, in: *Communications of the ACM*, 24 (10), pp. 632–646.

Charette, Robert N. 2005: Why Software Fails, in: *IEEE Spectrum*, 42 (9), pp. 42–49.

Chretien, G. J. et al. 1973: The SITA Network, NATO Advanced Study Institute on Computer Communication Networks, Sussex.

Codd, Edgar F. 1970: A Relational Model of Data for Large Shared Data Banks, in: *Communications of the ACM*, 13 (6), pp. 377–387.

Codd, Edgar F. 1971a: Relational Completeness of Data Base Sublanguages, in: Courant Computer Science Symposia (Ed.): *Data Base Systems*, Engelwood Cliffs NJ, pp. 65–98.

Codd, Edgar F. 1971b: A Data Base Sublanguage Founded on the Relational Calculus, 1971 ACM SIGFIDET Workshop, San Diego CA, pp. 35–68.

Codd, Edgar F. and Christopher J. Date 1975: Interactive Support for Non-Programmers: The Relational and Network Approaches, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control: Data Models: Data-Structure-Set versus Relational, Ann Arbor, Michigan, May 1–3, pp. 11–41.

Cohen, Bernard I. and William Aspray 2000: Howard Aiken and the Dawn of the Computer Age, in: Rojas, Raul and Ulf Hashagen (Eds.), *The First Computers: History and Architectures*, Cambridge MA, pp. 107–120.

Conklin, Jeff 1987: Hypertext: An Introduction and Survey, in: *IEEE Computer* (September), pp. 17–41.

Corbató, Fernando José 1964: Panel Discussion on Time Sharing, in: *Communications of the ACM*, 7 (7), p. 399.

Corbató, Fernando José et al. 1962: *An Experimental Time-Sharing System*, New York NY, pp. 335–344.

Corbató, Fernando José et al. 1972: Multics: The First Seven Years, in: *Spring Joint Computer Conference*, pp. 571–583.

Corbató, Fernando José and Victor A. Vyssotsky 1965: Introduction and Overview of the Multics System, Proceedings Fall Joint Computer Conference, pp. 185–196.

Coveyou, R. R. 1960: Serial Correlation in the Generation of Pseudo-Random Numbers, in: *ACM*, 7 (1), pp. 72–74.

Cowlishaw, Mike 1990: *IBM Jargon and General Computing Dictionary.* Tenth Edition, Winchester.

Crank, John 1947: *The Differential Analyser. With Diagrams and Photographs*, London.

Dalheimer, Karlheinz 1962: Fakturierung von Frischdienstlieferungen im Lebensmittelgroßhandel mit einer Lochkarten-Ziehkartei, in: *IBM Nachrichten*, 158, pp. 1867–1871.

Daniels, Siegfried et al. 1987: *Massenspeicher-Handbuch für Mikrocomputer alles über Floppy-Disk-Laufwerke u. Disketten, Festplatten-Laufwerke, opt. Speicher u. Bandlaufwerke*, Troisdorf.

Date, Christopher J. and Edgar. F. Codd 1975: The Relational and Network Approaches: Comparison of the Application Programming Interfaces, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control: Data Models: Data-Structure-Set versus Relational, Ann Arbor, Michigan, May 1–3, pp. 83–113.

Davies, Donald Watts and Derek L. A. Barber 1973: *Communication Networks for Computers*, London.

Dearden, John 1964: Can Management Information Be Automated?, in: *Harvard Business Review*, 42 (2), pp. 128–135.

Dearden, John 1965: How to Organize Information Systems, in: *Harvard Business Review*, 43 (2), pp. 65–73.

Dearden, John 1972: MIS Is a Mirage, in: *Harvard Business Review* (January–February), pp. 90–99.

Deasington, Richard J. 1985: *X.25 Explained: Protocols for Packet Switching Networks*, Ellis Horwood Series in Computer Communications and Networking, Chichester.

Denning, Peter J. 1971: Third Generation Computer Systems, in: *ACM Computing Surveys*, 3 (4).

Denninger, Erhard 1990: *Der gebändigte Leviathan*, Baden-Baden.

Dennis, Jack B. 1968: A Position Paper on Computing and Communications, in: *Communications of the ACM*, 11 (5), pp. 370–377.

Dijkstra, Edsger W. 1968: Letters to the Editor: Go to Statement Considered Harmful, in: *Communications of the ACM*, 11 (3), pp. 147–148.

Dijkstra, Edsger W. 1970: *Notes on Structured Programming*, Eindhoven.

Dijkstra, Edsger W. 1972: The Humble Programmer: 1972 ACM Turing Award Lecture, in: *Communications of the ACM*, 15 (10), pp. 859–866.

Dommann, Monika 2009: "Be Wise – Palletize." Die Transformationen eines Transportbretts zwischen den USA und Europa im Zeitalter der Logistik, in: *Traverse*, 16 (3), pp. 21–35.

Dompier, Steve 1976 (1975): Music of a Sort (reprint), in: *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia*, 1 (February 1976), p. 28.

Donegan, James J. et al. 1964: Experiences with the Goddard Computing System during Manned Spaceflight Missions, Proceedings of the 1964 19th ACM National Conference, pp. 12101–12108.

Dover, Jerome J. 1954: A Centralized Data Processing System, Proceedings of the February 11–12, 1954, Western Computer Conference. Trends in Computers – Automatic Control and Data Processing, Los Angeles CA, pp. 172–183.

Eckert, J. Presper et al. 1945: Description of the ENIAC and Comments on Electronic Digital Computing Machines, Moore School of Electrical Engineering, University of Pennsylvania.

Eckert, J. Presper et al. 1951: The UNIVAC System, AIEE-IRE '51 Papers and Discussions Presented at the Dec. 10–12, 1951, Joint AIEE-IRE Computer Conference. Review of Electronic Digital Computers, New York NY, pp. 6–16.

Edwards, Paul N. 1996: *The Closed World: Computers and the Politics of Discourse in Cold War America*, Cambridge MA, London.

Edwards, Paul N. 2000: The World in a Machine: Origins and Impacts of Early Computerized Global Systems Models, in: Hughes, Agatha C. and Thomas Parke Hughes (Eds.): *Systems, Experts, and Computers: The Systems Approach in Management and Engineering, World War II and after*, Cambridge MA, pp. 221–254.

Egger, Josef 2013: *"Ein Wunderwerk der Technik." Frühe Computernutzung in der Schweiz (1960–1980)*, Zurich.

Eldredge, K. R. et al. 1957: Automatic Input for Business Data-Processing Systems, Papers and Discussions Presented at the December 10–12, 1956, Eastern Joint Computer Conference. New Developments in Computers, New York NY, pp. 69–73.

Enzensberger, Hans Magnus 1979: Der Sonnenstaat des Doktor Herold, in: *Der Spiegel*, 25, pp. 68–78.

ETH-Bibliothek 1968: *Automatisierung der ETH-Bibliothek. Planungsunterlagen Oktober 1968*, Zurich.

Everest, G. C. 1974: The Futures of Database Management, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control, Ann Arbor, Michigan, May 1–3, pp. 445–462.

Fano, Robert M. 1967: The Computer Utility and the Community, in: *IEEE International Convention Record Part 12*, pp. 30–34.

Fano, Robert M. and Fernando José Corbató 1966: Time-Sharing on Computers, in: *Scientific American*, 215 (3), pp. 129–131.

Fontanellaz, Gustav 1964: Datenübertragung auf dem öffentlichen Fernmeldenetz, in: *Technische Mitteilungen PTT* 11, pp. 429–434.

Forsberg, David 1990: The Interconnectivity Nightmare, in: *Network World* (August 27, 1990), pp. 42, 46, 60, 61.

Fraser, A. G. 1972: On the Interface between Computers and Data Communications Systems, in: *Communications of the ACM*, 15 (7), pp. 566–573.

Frei, Norbert 2008: *1968. Jugendrevolte und globaler Protest*, Munich.

Fuller, R. Buckminster 1969: *Operating Manual for Spaceship Earth*, New York NY.

Furger, Franco and Bettina Heintz 1997: Technologische Paradigmen und lokaler Kontext. Das Beispiel der ERMETH, in: *Schweizerische Zeitschrift für Soziologie*, 23 (3), pp. 533–566.

Füssl, Wilhelm (Ed.) 2010: *100 Jahre Konrad Zuse. Einblicke in den Nachlass*, Munich.

Gandy, Anthony 2014: Product Strategy Choices. Honeywell and RCA Mainframe Computer Product Strategies 1963–71, in: *Business History*, 56 (3), pp. 414–433.

Gass, Saul I. 1961: The Role of Digital Computers in Project Mercury, Proceedings of the December 12–14, 1961, Eastern Joint Computer Conference. Computers – Key to Total Systems Control, Washington DC, pp. 33–46.

Gates, C. R. and W. H. Pickering 1965: The Role of Computers in Space Exploration, Proceedings of the November 30–December 1, 1965, Fall Joint Computer Conference, Part II. Computers – Their Impact on Society, Las Vegas NV, pp. 33–35.

Georgii, Eugen 1966: Neuzeitliche Vermittlungstechnik, in: *Technische Mitteilungen PTT*, 1966 (7), pp. 197–209.

Gerecke, Kurt and Klemens Poschke 2010: *IBM System Storage-Kompendium. Die IBM Speichergeschichte von 1952 bis 2010*, Ehningen.

Girschik, Katja 2010: *Als die Kassen lesen lernten. Eine Technik- und Unternehmensgeschichte des Schweizer Einzelhandels 1950 bis 1975*, Bd. 22, Munich.

Glimm, James et al. 1990: *The Legacy of John von Neumann*, Proceedings of Symposia in Pure Mathematics, Providence RI.

Gold, Michael M. and Lee L. Selwyn 1968: Real Time Computer Communications and the Public Interest, Proceedings of the December 9–11, 1968, Fall Joint Computer Conference, Part II, San Francisco CA, pp. 1473–1478.

Goldstein, Larry Joel and Martin Goldstein 1982: *IBM Personal Computer: An Introduction to Programming and Applications*, Bowie MD.

Goldstine, H. H. and Adele Goldstine 1996 (1946): Electronic Numerical Integrator and Computer (ENIAC), in: *IEEE Annals of the History of Computing*, 18 (15), pp. 10–16.

Gorn, S. 1966: Recorded Magnetic Tape for Information Interchange (800 CPI, NRZI), in: *Communications of the ACM*, 9 (4), pp. 285–292.

Gosling, Francis G. 1994: *The Manhattan Project: Making the Atomic Bomb*, Washington DC.

Gotwals, John K. 1983: Processing Power on the IBM Personal Computer, Proceedings of the 1983 ACM SIGSMALL Symposium on Personal and Small Computers, San Diego CA, pp. 132–142.

Gray, George 2001: UNIVAC I: The First Mass-Produced Computer, in: *Unisys History Newsletter*, 5 (1).

Gray, Stephen B. 1976: Building Your Own Computer, Proceedings of the June 7–10, 1976, National Computer Conference and Exposition, New York NY, pp. 235–239.

Green, Bert F. Jr. et al. 1959: Empirical Tests of an Additive Random Number Generator, in: *Journal of the ACM*, 6 (4), pp. 527–537.

Green, Wayne 1975: From the publisher … Are they real?, in: *BYTE*, 1 (2), pp. 61, 81, 87.

Greenberger, Martin 1959: Random Number Generators, Preprints of Papers Presented at the 14th National Meeting of the Association for Computing Machinery, Cambridge MA, pp. 1–3.

Greenberger, Martin 1966: The Priority Problem and Computer Time Sharing, in: *Management Science*, 12 (11), pp. 888–906.

Greenberger, Martin 1961: Notes on a New Pseudo-Random Number Generator, in: *Journal of the ACM*, 8 (2), pp. 163–167.

Grosch, Herbert 1953: High Speed Arithmetic: The Digital Computer as a Research Tool, in: *Journal of the Optical Society of America 43* (April), pp. 306–310.

Gugerli, David 2002a: Die Entwicklung der digitalen Telefonie (1960–1985). Die Kosten soziotechnischer Flexibilisierungen, in: Stadelmann, Kurt et al. (Eds.): *Telemagie. 150 Jahre Telekommunikation in der Schweiz*, Zurich, pp. 154–167.

Gugerli, David 2002b: Steiniger Weg ins digitale Zeitalter, in: *Neue Zürcher Zeitung*, January 5, p. 25.

Gugerli, David 2007a: Die Welt als Datenbank. Zur Relation von Softwareentwicklung, Abfragetechnik und Deutungsautonomie, in: *Nach Feierabend. Zürcher Jahrbuch für Wissensgeschichte*, 3, pp. 11–36.

Gugerli, David 2007b: Vom Befehl zur Steuerung, von der Datei zum Index. Horst Herold im Gespräch mit David Gugerli, in: *Nach Feierabend. Zürcher Jahrbuch für Wissensgeschichte*, 3, pp. 173–184.

Gugerli, David 2009a: Das Monster und die Schablone. Zur Logistik von Daten um 1950, in: *Traverse*, 16 (3), pp. 66–76.

Gugerli, David 2009b: Suchmaschinen. Die Welt als Datenbank, Frankfurt am Main.

Gugerli, David 2010: Data Banking: Computing and Flexibility in Swiss Banks 1960–90, in: Kyrtsis, Alexandros-Andreas (Ed.): *Financial Markets and Organizational Technologies. System Architectures, Practices and Risks in the Era of Deregulation*, Houndmills, pp. 117–136.

Gugerli, David 2012: Nach uns die Informationsflut. Zur Pathologisierung soziotechnischen Wandels, in: *Nach Feierabend. Zürcher Jahrbuch für Wissensgeschichte*, 8, Zurich, Berlin, pp. 141–147.

Gugerli, David 2005: Computerization Strategies, in: Gugerli, David et al. (Eds.): *Transforming the Future: ETH Zurich and the Construction of Modern Switzerland 1855–2005*, Zurich, pp. 301–314.

Gugerli, David and Hannes Mangold 2016: Diskussionsforum – Betriebssysteme und Computerfahndung. Zur Genese einer digitalen Überwachungskultur, in: *Geschichte und Gesellschaft*, 42 (1), pp. 144–174.

Gugerli, David and Daniela Zetti 2019a: Computergeschichte als Irritationsquelle, in: Martina Heßler and Heike Weber (Eds.): *Provokationen der Technikgeschichte. Zum Reflexionszwang historischer Forschung*, Paderborn, pp. 193–224.

Gugerli, David and Daniela Zetti 2019b: Computer History – The Pitfalls of Past Futures, in: *Preprints zur Kulturgeschichte der Technik*, 33, Zurich.

Haake, Rolf 1965: *Einführung in die Informations- und Dokumentationstechnik unter besonderer Berücksichtigung der Lochkarten*, ZIID-Schriftenreihe, Leipzig.

Haanstra, J. W. et al. 1983 (1961): Processor Products – Final Report of the SPREAD Task Group, December 28, 1961, in: *Annals of the History of Computing*, 5 (1), pp. 6–26.

Haigh, Thomas 2001: Inventing Information Systems: The Systems Men and the Computer 1950–1968, in: *Business History Review*, 75 (Spring), pp. 15–61.

Haigh, Thomas 2006: Charles W. Bachman Interview: September 25–26, 2004; Tucson, Arizona. Interview conducted for the Special Interest Group on the Management of Data (SIGMOD) of the Association for Computing Machinery (ACM). Transcript and original tapes donated to the Charles Babbage Institute, in: *ACM Oral History Interviews*, pp. 1–106.

Haigh, Thomas 2007: "A Veritable Bucket of Facts." Ursprünge des Datenbankmanagementsystems, in: *Nach Feierabend. Zürcher Jahrbuch für Wissensgeschichte*, 3, Zurich, Berlin, pp. 57–98.

Haigh, Thomas 2010: Dijkstra's Crisis: The End of Algol and Beginning of Software Engineering 1968–72, www.tomandmaria.com/Tom/Writing/DijkstrasCrisis_LeidenDRAFT.pdf.

Hamlin, J. E. 1964: A General Description of the National Aeronautics and Space Administration Real Time Computing Complex, Proceedings of the 1964 19th ACM National Conference, pp. 12.201–12.202.

Hauben, Ronda 2001: Die Entstehung des Internet und die Rolle der Regierung, in: Maresch, Rudolf and Florian Roetzer (Eds.): *Cyberhypes. Möglichkeiten und Grenzen des Internet*, Frankfurt am Main, pp. 27–52.

Hausammann, Luzius 2008: *Der Beginn der Informatisierung im Kanton Zürich. Von der Lochkartenanlage im Strassenverkehrsamt zur kantonalen EDV-Stelle (1957–1970)*, Zurich.

Heide, Lars 2009: *Punched-Card Systems and the Early Information Explosion 1880–1945*, Baltimore MD.

Heintz, Bettina 1993: *Die Herrschaft der Regel. Zur Grundlagengeschichte des Computers*, Frankfurt am Main.

Henk, Martin 1983: *Der IBM-Personal Computer (beantwortet alle Fragen über Aufbau, Einsatz und Programmierung, Software und Hardwareerweiterungen)*, Computer persönlich, Munich.

Herken, Rolf 1988: *The Universal Turing Machine: A Half-Century Survey*, Oxford.

Herold, Horst 1968a: Die elektronische Datenverarbeitung. Möglichkeiten ihres Einsatzes für die Kriminalstatistik, bei der Gefahrenabwehr und der Erforschung des Sachverhalts. in: Polizei-Institut Hiltrup (Ed.): *19. Arbeitstagung für Kriminalistik und Kriminologie,* Hiltrup.

Herold, Horst 1968b: Kriminalgeographie – Ermittlung und Untersuchung der Beziehungen zwischen Raum und Kriminalität, in: Schäfer, Herbert (Ed.): *Kriminalistische Akzente*, 4, Hamburg, pp. 1–47.

Herold, Horst 1968c: Organisatorische Grundzüge der elektronischen Datenverarbeitung im Bereich der Polizei. Versuch eines Zukunftsmodells, in: *Taschenbuch für Kriminalisten*, 18, pp. 240–254.

Herold, Horst 1970: Kybernetik und Polizei-Organisation, in: *Die Polizei. Zentralorgan für das Sicherheits- und Ordnungswesen, Polizei-Wissenschaft, -Recht, -Praxis*, 61 (2), pp. 33–37.

Herold, Horst 1985: Rasterfahndung. Eine computerunterstützte Fahndungsform der Polizei, in: *Recht und Politik. Vierteljahreshefte für Rechts- und Verwaltungspolitik*, pp. 84–97.

Hey, Anthony J. G. and Gyuri Pápay 2015: *The Computing Universe: A Journey Through a Revolution*, New York NY.

Hirzel, H. and K. Käfer 1943: *Einführung in das berufliche Rechnen für Schreiner*, Zurich.

Hoagland, Albert S. 1976: Magnet Recording Storage, in: *IEEE Transactions on Computer*, 25 (12), pp. 1283–1288.

Holden, Willard 1979: *What Is a Personal Computer?*, SIGPC '79 Editor's Message, 2, New York NY.

Hopper, Grace Murray 1952: The Education of a Computer, Proceedings of the 1952 ACM National Meeting (Pittsburgh), Pittsburgh PA, pp. 243–249.

Humphrey, W. S. 2002: Software Unbundling: A Personal Perspective, in: *IEEE Annals of the History of Computing*, 24 (1), pp. 59–63.

Hutchinson, Lee 2012: Going Boldly: Behind the Scenes at NASA's Hallowed Mission Control Center. Apollo vet Sy Liebergot Shows Ars How NASA Got Men Safely to the Moon and Back, Ars Technica.

IBM 1970: *Reference Manual IBM 29 Card Punch*, Poughkeepsie NY.

International Telegraph and Telephone Consultative Committee 1977: *Orange Book*, Geneva.

Isaacson, Portia 1978: Personal Computing Position Paper, in: *SIGPC Note*, 1 (2), pp. 5–9.

Isaacson, Portia et al. 1978: Personal Computing: Problems of the 80's, in: *SIGPC Note*, 1 (3), pp. 46–55.

James, S. E. 1981: Evolution of Real-Time Computer Systems for Manned Spaceflight, in: *IBM Journal of Research and Developpement*, 25 (5), pp. 417–428.

Jensen, John 1967: *How to Pass Computer Programmer Aptitude Tests*, New York NY.

Johnson, Clayton 1975: IBM 3850: Mass Storage System, Proceedings of the May 19–22, 1975, National Computer Conference and Exposition, Anaheim CA, pp. 509–514.

Johnson, L. R. 1952: Installation of a Large Electronic Computer, Proceedings of the 1952 ACM meeting (Toronto), New York NY, pp. 77–80.

Johnstone, J. L. 1969: RTOS: Extending OS/360 for Real Time Spaceflight Control, Proceedings of the May 14–16, 1969, Spring Joint Computer Conference, Boston MA, pp. 15–27.

Jones, Curtis H. 1970: At Last: Real Computer Power for Decision Makers, in: *Harvard Business Review*, 48 (5), pp. 75–89.

Jones, Steven E. 2016: *Roberto Busa, S. J. and the Emergence of Humanities Computing: The Priest and the Punched Cards*, New York NY.

Kahn, R. E. 1972: Resource-Sharing Communication Networks, in: *Proceedings IEEE*, 60 (11), pp. 1347–1407.

Kaleth, Hans 1961: *Die elektronische Datenverarbeitung. Ein Beitrag zur Automatisierung der kriminalpolizeilichen Karteiarbeit*, BKA-Schriftenreihe, Wiesbaden.

Kaplan, Sidney J. 1968: The Advancing Communication Technology and Computer Communication Systems, Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, Atlantic City NJ, pp. 119–133.

Katzan, Harry 1977: *The IBM 5100 Portable Computer: A Comprehensive Guide for Users and Programmers*, Computer Science Series, New York NY.

Katzan, Harry Jr. 1970: Operating Systems Architecture, Proceedings of the May 5–7, 1970, Spring Joint Computer Conference, Atlantic City NJ, pp. 109–118.

Katzenbach, Nicholas de Belleville 1987: *An Overview of Program Trading and its Impact on Current Market Practices*, New York NY.

Kay, Alan et al. 1978: Position Paper on How to Advance from Hobby Computing to Personal Computing, in: *SIGPC Note*, 1 (2), pp. 29–31.

Kemeny, John G. and Thomas E. Kurtz 1964: *BASIC: A Manual for BASIC: The Elementary Algebraic Language Designed for Use with the Dartmouth Time Sharing System*, Hanover NH.

Kephart, Jeffrey O. and David M. Chess 2003: The Vision of Autonomic Computing, in: *Computer*, 36 (1), pp. 41–50.

Kerner, Helmut and Georg Bruckner 1989: *Rechnernetze nach ISO-OSI, CCITT*, Wolfsgraben.

Kilburn, Tom et al. 1962: *The Atlas Supervisor,* www.chilton-computing.org.uk/acl/technology/atlas/p019.htm.

King, John Leslie et al. 1997: The Rise and Fall of Netville: The Saga of a Cyberspace Construction Boomtown in the Great Divide, in: *Electronic Markets*, 7, pp. 3–33.

Klossner, Andrew 1980: A Parallel Between Operating System and Human Government, in: *ACM SIGOPS Operating Systems Review*, 14 (2), pp. 28–31.

Kneipp, Sebastian 1880: *So sollt ihr leben! Winke und Ratschläge für Gesunde und Kranke zu einer einfachen, vernünftigen Lebensweise und einer naturgemässen Heilmethode*, Kempten.

Kneipp, Sebastian 1889: *Meine Wasser-Kur*, Kempten.

Knight, Kenneth E. 1966: Changes in Computer Performance, in: *Datamation*, 12 (9), pp. 40–54.

Krajewski, Markus 2002: *Zettelwirtschaft. Die Geburt der Kartei aus dem Geiste der Bibliothek*, Berlin.

Kranz, Gene 2001: *Failure Is Not an Option: Mission Control from Mercury to Apollo 13 and Beyond*, New York NY.

Kraushaar, Wolfgang 2000: *1968 als Mythos, Chiffre und Zäsur*, Hamburg.

Krige, John (Ed.) 1996: *History of CERN*, Amsterdam.

Lacan, Jacques 1986 (1948): Das Spiegelstadium als Bildner der Ichfunktion, wie sie uns in der psychoanalytischen Erfahrung erscheint in: Lacan, Jacques (Ed.): *Schriften*, 1, Weinheim, Berlin, pp. 61–70.

Laccabue, Fred 2009: Oral History Interview with Fred Laccabue. Retrieved from the University of Minnesota Digital Conservancy, http://hdl.handle.net/11299/107417.

Landau, Robert M. 1979: Productivity, Information Technology and the Office, Proceedings of the 2nd Annual International ACM SIGIR Conference on Information Storage and Retrieval: Information Implications into the Eighties, Dallas TX, pp. 59–63.

Launius, R. D. 2009: Abandoned in Place: Interpreting the US Material Culture of the Moon Race, in: *Public Historian*, 31 (3), pp. 9–38.

Lesser, M. L. and J. W. Haanstra 1957: The RAMAC Data-Processing Machine, Proceedings of the December 10–12, 1956, Eastern Joint Computer Conference: New Developments in Computers, New York NY, pp. 139–146.

Levine, Stanley L. 1961: The Problem of Heterogeneous Groups in Computer Programmer Training, Proceedings of the 1961 ACM National Meeting, New York, pp. 131.301–131.303.

Levy, Steven 2010: *Hackers*, Sebastopol CA.

Lew, K. H. and C. Jong 1988: Getting There from Here: Mapping from TCP/IP to OSI, in: *Data Communication* (August), pp. 161–175.

Licklider, J. C. R. 1960: Man-Computer Symbiosis, in: *IRE Transactions on Human Factors in Electronics*, 1 (March), pp. 4–11.

Licklider, J. C. R. 1967a: Dynamic Modeling, in: Wathen-Dunn, Weiant (Ed.): *Models for the Perception of Speech and Visual Form*, Cambridge, pp. 11–25.

Licklider, J. C. R. 1967b: Interactive Dynamic Modeling, in: Shapiro, George and Milton Rogers (Eds.): *Prospects for Simulation and Simulators of Dynamic Systems*, New York NY, pp. 281–289.

Licklider, J. C. R. 1967c: Interactive Information Processing, in: Tou, Julius T. (Ed.): *Computer and Information Sciences*, II, New York NY, pp. 1–13.

Licklider, J. C. R. 1968: Man-Computer Communication, in: *Annual Review of Information Science and Technology*, 3, pp. 201–240.

Licklider, J. C. R. and Welden E. Clark 1962: *On-Line Man-Computer Communication*, New York NY, p. 113.

Licklider, J. C. R. and R. W. Taylor 1990: In memoriam, J. C. R. Licklider, 1915–1990, in: http://catalog.hathitrust.org/api/volumes/oclc/22964205.html.

Licklider, J. C. R. et al. 1968: The Computer as a Communication Device, in: *Science and Technology for the Technical Men in Management*, 76 (April), pp. 21–31.

Luhmann, Niklas 1966: *Recht und Automation in der öffentlichen Verwaltung. Eine verwaltungswissenschaftliche Untersuchung*, 29, Berlin.

Luhmann, Niklas 1968: *Zweckbegriff und Systemrationalität über die Funktion von Zwecken in sozialen Systemen, Soziale Forschung und Praxis*, Tübingen.

Luhmann, Niklas 1993: Gleichzeitigkeit und Synchronisation, in: Luhmann, Niklas (Ed.): *Soziologische Aufklärung. Konstruktivistische Perspektiven*, 5, Opladen, pp. 95–130.

Luhmann, Niklas 2007 (1964): Lob der Routine, in: Luhmann, Niklas (Ed.): *Politische Planung. Aufsätze zur Soziologie von Politik und Verwaltung,* Wiesbaden, pp. 113–142.

Mahoney, Michael S. 2005: The Histories of Computing(s), in: *Interdisciplinary Science Reviews*, 30, pp. 119–135.

Mahoney, Michael S. 2011: *Histories of Computing*, Cambridge MA.

Mangold, Hannes 2017: *Fahndung nach dem Raster. Informationsverarbeitung bei der bundesdeutschen Kriminalpolizei*, 1965–1984, Interferenzen. Zur Kulturgeschichte der Technik, 23, Zurich.

March, James Gardner et al. 1958: *Organizations*, New York NY, London.

Marill, Thomas et al. 1963: DATA-DIAL: Two-Way Communication with Computers from Ordinary Dial Telephones, in: *Communications of the ACM*, 6, pp. 622–624.

Martin, William L. 1954: A Merchandise Control System, Proceedings of the February 11–12, 1954, Western Computer Conference. Trends in Computers. Automatic Control and Data Processing, Los Angeles CA, pp. 184–191.

Mayer, Joh. Eugen 1908: *Das Rechnen in der Technik und seine Hilfsmittel. Rechenschieber, Rechentafeln, Rechenmaschinen usw.*, Sammlung Göschen, Leipzig.

Mayer, Paul A. 2003: *Computer Media and Communication: A Reader*, Oxford.

McCarthy, John 1959: A Time Sharing Operator Program for Our Projected IBM 709, www-formal.stanford.edu/jmc/history/timesharing-memo/timesharing-memo.html.

McCarthy, John 1983: Reminiscences on the History of Time Sharing, www-formal.stanford.edu/jmc/history/timesharing/timesharing.html.

McPherson, J. L. and S. N. Alexander 1951: Performance of the Census UNIVAC System, Proceedings of the December 10–12, 1951, Joint AIEE-IRE Computer Conference: Review of Electronic Digital Computers, Philadelphia PA, pp. 16–22.

McPherson, James L. 1953: Commercial Applications: The Implication of Census Experience, Proceedings of the February 4–6, 1953, Western Computer Conference, Los Angeles CA, pp. 49–53.

Mendicino, Samuel F. 1972: Octopus: The Lawrence Radiation Laboratory Network, in: Rustin, Randall (Ed.): *Computer Networks*, Englewood Cliffs NJ, pp. 95–100.

Metcalfe, Robert M. and David R. Boggs 1976: Ethernet: Distributed Packet Switching for Local Computer Networks, in: *Communications of the ACM*, 19, pp. 395–404.

Meyer, Caroline 2008: *Eidophor. Ein Fernseh-Grossbildprojektionssystem zwischen Nutzungsvisionen und Anwendungsrealitäten 1939–1999*, Zurich.

Miller, Irvin M. 1969: Computer Graphics for Decision Making, in: *Harvard Business Review*, 47 (6), pp. 121–132.

Miller, Leslie Jill 1981: The ISO Reference Model of Open Systems Interconnection: A First Tutorial, Proceedings of the ACM 1981 Conference, pp. 283–288.

Mindell, David A. 2002: *Between Human and Machine: Feedback, Control, and Computing Before Cybernetics*, Johns Hopkins Studies in the History of Technology, Baltimore MD.

Mindell, David A. 2008: *Digital Apollo: Human and Machine in Spaceflight*, Cambridge MA.

Misa, Thomas J. 2017: *Communities of Computing: Computer Science and Society in the ACM*, San Rafael CA.

MITS 1975: Building Your Own Computer Won't Be a Piece of Cake, in: *Radio Electronics* (5), p. 25.

Moore, Gordon E. 1965: Cramming More Components onto Integrated Circuits, in: *Electronics*, 38 (8), pp. 114–117.

Morgan, Howard Lee 1976: Office Automation Project: A Research Perspective, Proceedings of the June 7–10, 1976, National Computer Conference and Exposition, New York NY, pp. 605–610.

Morton, Michael S. and Andrew M. McCosh 1968: Terminal Costing for Better Decisions, in: *Harvard Business Review*, 46 (3), pp. 147–156.

NASA 1965: Composite Air-to-Ground and Onboard Voice Tape Transcription of the GT-4 Mission, NASA Program Gemini, Working Papers No. 5035, Houston TX.

National Research Council 1999: *Funding a Revolution: Government Support for Computing Research*, Washington DC.

Nelson, Theodore Holm 1987: *Computer Lib/Dream Machines*, Redmond WA.

Nelson, Theodore Holm 1967: Getting It Out of Our System, in: Schecter, George (Ed.): *Information Retrieval: A Critical View*. Based on a Colloquium, Philadelphia PA, May 12–13, 1966, Washington DC, pp. 191–210.

Neu, Walter and Albert Kündig 1968: Project for a Digital Telephone Network, in: *IEEE Transactions on Communication Technology*, COM-16 (5) (October 1968), pp. 633–648.

Neukom, Hans 2009: Ubisco and CDC: Analysis of a Failure, in: *IEEE Annals of the History of Computing* (April-June), pp. 31–43.

Neumann, John von 1945: First Draft of a Report on the EDVAC, in: Randell, Brian (Ed.): *The Origins of Digital Computers: Selected Papers*, Berlin, Heidelberg, New York NY, pp. 355–364.

Noll, John and Walt Scacchi 1991: Integrating Diverse Information Repositories: A Distributed Hypertext Approach, in: *IEEE Computer* (December), pp. 38–45.

Norman, Jeremy M. 2005: *From Gutenberg to the Internet: A Sourcebook on the History of Information Technology*, Novato CA.

Owens, Larry 1986: Vannevar Bush and the Differential Analyzer: The Text and Context of an Early Computer, in: *Technology and Culture*, pp. 63–95.

Palormo, Jean M. 1967: The Computer Programmer Aptitude Battery: A Description and Discussion, Proceedings of the Fifth SIGCPR Conference on Computer Personnel Research, College Park MD, pp. 57–63.

Pelaez, E. 1999: The Stored-Program Computer: Two Conceptions, in: *Social Studies of Science*, 29, pp. 359–389.

Penny, Samuel J. et al. 1970: Design of a Very Large Storage System, Proceedings of the November 17–19, 1970, Fall Joint Computer Conference, Houston TX, pp. 45–51.

Philco 1967: *Familiarization Manual Mission Control Center Houston PHO-FAM001*. Replaces PHO-FAM001 published 22 November 1965, Houston TX.

Pias, Claus 2015: Friedrich Kittler und der "Missbrauch von Heeresgerät," in: *Merkur*, 69 (791), pp. 31–44.

Poole, P. C. and W. M. Waite 1969: Machine Independent Software, Proceedings of the Second Symposium on Operating Systems Principles, Princeton NJ, pp. 19–24.

Pratschke, Margarete 2011: Why History Matters: Visual Innovation and the Role of Image Theory in HCI, in: Marcus, Aaron (Ed.): *Design, User Experience, and Usability*, Heidelberg, pp. 277–284.

Pugh, Emerson W. et al. 1991: *IBM's 360 and Early 370 Systems*, Cambridge MA.

Raskin, Jef 1976: Personal Computers: A Bit of Wheat Amongst the Chaff, in: *Dr. Dobb's Journal of Computer Calisthenics & Orthodontia* (September 1976), pp. 15–17.

Raskin, Jef 1987: The Hype in Hypertext: A Critique, Proceedings of the ACM Conference on Hypertext, Chapel Hill NC, pp. 325–330.

RCA 1965: RCA Spectra 70, www.computerhistory.org/brochures/full_record.php?iid=hdoc-4372956eb9810.

Redmond, Kent C. and Thomas M. Smith 2000: *From Whirlwind to Mitre: The R&D Story of the SAGE Air Defense Computer*, Cambridge MA.

Reed, Harry L. 1952: Firing Table Computations on the ENIAC, Proceedings of the 1952 ACM National Meeting (Pittsburgh), New York NY, pp. 103–106.

Ridenour, Louis N. 1955: Storage and Retrieval of Information, Papers and Discussions presented at the November 7–9, 1955, Eastern Joint AIEE-IRE Computer Conference. Computers in Business and Industrial Systems, Boston MA, pp. 79–82.

Ridgway, Richard K. 1952: Compiling Routines, Proceedings of the 1952 ACM National Meeting (Toronto), New York NY, pp. 1–5.

Roberts, H. Edward and William Yates 1975a: ALTAIR 8800. The Most Powerful Minicomputer Project Ever Presented – Can be Built for under $400, in: *Popular Electronics*, 7 (1), pp. 33–38.

Roberts, H. Edward and William Yates 1975b: Build the Altair 8800 Minicomputer (Part Two), in: *Popular Electronics*, 7 (2), pp. 56–58.

Roberts, L. G. 1978: The Evolution of Packet Switching, in: *Proceedings IEEE*, 66 (11), p. 1307.

Roberts, Lawrence G. and Barry D. Wessler 1970: Computer Network Development to Achieve Resource Sharing, Proceeding AFIPS (Spring) Proceedings of the May 5–7, 1970, Spring Joint Computer Conference, pp. 543–549.

Rodgers, Daniel T. 2011: *Age of Fracture*, Cambridge MA.

Rogge, Peter G. 1997: *Die Dynamik des Wandels. Schweizerischer Bankverein 1862–1997. Das fünfte Vierteljahrhundert*, Basel.

Rosen, Philip (Ed.) 1986: *Narrative, Apparatus, Ideology: A Film Theory Reader*, New York NY.

Rosenfelder, Andreas 2003: Medien auf dem Mond. Zur Reichweite des Weltraumfernsehens, in: Schneider, Irmela et al. (Eds.): *Medienkultur der 60er Jahre. Diskursgeschichte der Medien nach 1945*, 2, Wiesbaden, pp. 17–33.

Rosenthal, David S. H. et al. 2012: The Economics of Long-Term Digital Storage. In: Duranti, Luciana and Elizabeth Shaffer (Eds.): *The Memory of the World in the Digital Age: Digitization and Preservation*, Vancouver, pp. 513–528.

Rutishauser, Heinz 1952: *Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen*, Basel.

Rutishauser, Heinz 1956: *Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen*, Basel.

Rutishauser, Heinz et al. 1951: *Programmgesteuerte digitale Rechengeräte (elektronische Rechenmaschinen)*, Mitteilungen aus dem Institut für angewandte Mathematik an der Eidgenössischen Technischen Hochschule in Zürich, Basel.

Sachs, Wolfgang 1994: Satellitenblick. Die Ikone vom blauen Planeten und ihre Folgen für die Wissenschaft, in: Braun, Ingo and Bernward Joerges (Eds.): *Technik ohne Grenzen*, Frankfurt am Main, pp. 305–346.

Sackman, Harold 1968: Man-Computer Communication. Experimental Investigation of User Effectiveness, Proceedings of the Sixth SIGCPR Conference on Computer Personnel Research, Cambridge MA, pp. 93–105.

Sale, Anthony E. et al. 2000: The Colossus of Bletchley Park: The German Cipher System, in: Rojas, Raul and Ulf Hashagen (Eds.): *The First Computers: History and Architectures*, Cambridge MA, pp. 351–364.

Science News-Letter 1961: Single Computers May Serve Many Companies, in: *The Science News-Letter*, 80 (4), p. 52.

Scott, Linda M. 1991: "For the Rest of Us": A Reader-Oriented Interpretation of Apple's "1984" Commercial, in: *Journal of Popular Culture*, 25 (1), pp. 67–81.

Scott, S. V. et al. 2008: *The Impact on Bank Performance of the Diffusion of a Financial Innovation: An Analysis of SWIFT Adoption*, Paris.

Sheldon, John W. and Liston Tatum 1951: The IBM Card-Programmed Electronic Calculator, Papers and Discussions Presented at the Dec. 10–12, 1951, Joint AIEE-IRE Computer Conference. Review of Electronic Digital Computers, Philadelphia PA, pp. 30–36.

Sibley, Edgar H. 1975: On the Equivalences of Data Based Systems, Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access and Control: Data Models: Data-Structure-Set versus Relational, Ann Arbor, Michigan, May 1–3, pp. 43–76.

Siebrecht, Michael 1995: *Rasterfahndung. Eine EDV-gestützte Masssenfahndungsmethode im Spannungsfeld zwischen einer effektiven Strafverfolgung und dem Recht auf informationelle Selbstbestimmung*, Berlin.

Simon, Herbert A. 1960: The Corporation: Will It Be Managed by Machines?, in: Anshen, M. and G. L. Bach (Eds.): *Management and Corporations*, New York NY, pp. 17–55.

Simon, Herbert A. 1962: The Architecture of Complexity in: *Proceedings of the American Philosophical Society*, 106 (6), pp. 467–482.

Simon, Herbert A. 1976 (1946): *Administrative Behavior: A Study of Decision-Making Processes in Administrative Organization*, New York NY.

Simon, Herbert A. 1977 (1960): *The New Science of Management Decision*, Englewood Cliffs NJ.

Simon, Jürgen and Jürgen Taeger 1981: *Rasterfahndung. Entwicklung, Inhalt und Grenzen einer kriminalpolizeilichen Untersuchungsmethode*, Baden-Baden.

Söll, Wolfgang and Jörg-Hagen Kirchner 1978: *Digitale Speicher. Informationsspeicher in der Technik und im Gedächtnis*, Kamprath-Reihe kurz und bündig Technik, Würzburg.

Solomon, Martin B. 1966: Economies of Scale and the IBM System/360, in: *Communications of the ACM*, 9 (6), pp. 435–440.

Spitzer, Manfred 2012: *Digitale Demenz. Wie wir uns und unsere Kinder um den Verstand bringen*, Munich.

Sprenger, Florian 2015: *The Politics of Micro-Decisions: Snowden, Net Neutrality, and Internet Architectures*, trans. Valentine A. Pakis, Digital Cultures Series, Lüneburg.

Stadlin, Christofer 2010: Actuarial Practice, Probabilistic Thinking and Actuarial Science in Private Casualty Insurance, in: Pearson, Robin (Ed.): *The Development of International Insurance*, London, pp. 37–62.

Stahel, Adolf 1950: *Rechnen für Mechaniker*, Zurich.

Steadman, Howard L. and George R. Sugar 1968: Some Ways of Providing Communication Facilities for Time-Shared Computing, Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference, Atlantic City NJ, pp. 23–29.

Stern, Nancy 1981: *From ENIAC to UNIVA: An Appraisal of the Eckert-Mauchly Computers*, Bedford MA.

Stiefel, Eduard 1954: Rechenautomaten im Dienste der Technik. Erfahrungen mit dem Zuse-Rechenautomaten Z4, Arbeitsgemeinschaft für Forschung des Landes Nordrhein-Westfalen, 45, Cologne, pp. 29–65.

Stubenrecht, Alfred 1960: *Lochkarten im Klein- und Mittelbetrieb*, Wie-Buchreihe, Düsseldorf.

Studienzentrum für Administrative Automatisierung 1966: *Neue Berufsbilder in der Elektronischen Datenverarbeitung*, 10, Munich, Vienna.

Sumner, Frank H. et al. 2000: The Atlas Computer, in: Rojas, Raul and Ulf Hashagen (Eds.): *The First Computers: History and Architectures*, Cambridge MA, pp. 387–396.

Tanenbaum, Andrew S. 1997: *Computer-Netzwerke*, Munich, London, Mexico.

Tanenbaum, Andrew S. 2014: *Modern Operating Systems*, Essex.

Teager, Herbert and John McCarthy 1959: Time-Shared Program Testing, Proceedings of the 14th National Meeting of the ACM, New York NY, pp. 1–2.

Tillman, Matthew A. and David C.-C. Yen 1990: SNA and OSI. Three Strategies for Interconnection, in: *Communications of the ACM*, 33 (2), pp. 214–224.

Tobler, Beatrice 2001: Z4 und ERMETH: Maschinen im Dienste des wissenschaftlichen Rechnens, in: Tobler, Beatrice and Sandra Sunier (Eds.): *Loading History. Computergeschichten aus der Schweiz*, 1, Zurich, pp. 12–21.

Tomayko, James E. 1988: Computers in Spaceflight: The NASA Experience, https://archive.org/details/nasa_techdoc_19880069935.

Tomeski, Edward Alexander 1970: *The Computer Revolution: The Executive and the New Information Technology*, New York NY.

Trost, Stanley R. and Wolfgang Dederichs 1983: *Programmsammlung zum IBM Personal Computer*, Der IBM Personal Computer, Düsseldorf.

Turing, Alan M. 1952: The Chemical Basis of Morphogenesis, in: *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences,* 237 (641), pp. 37–72.

Van der Spiegel, Jan et al. 2000: The ENIAC: History, Operation, and Reconstruction in VSLI, in: Rojas, Raul and Ulf Hashagen (Eds.): *The First Computers: History and Architectures*, Cambridge MA, pp. 121–178.

Vissmann, Cornelia 2001: *Akten. Medientechnik und Recht*, Frankfurt am Main.

Voelcker, John 1988: The PDP-8: The First "Personal" Computer for Engineers and Scientists Ushered in the Minicomputer Era, in: *IEEE Spectrum*, 25 (11), pp. 86–92.

Vyssotsky, Victor A. and Fernando José Corbató 1965: Structure of the Multics Supervisor, Proceedings Fall Joint Computer Conference, pp. 203–212.

Waldrop, M. Mitchell and J. C. R. Licklider 2002: *The Dream Machine: J. C. R. Licklider and the Revolution That Made Computing Personal*, New York NY.

Walker, Janet H. 1987: Document Examiner: Delivery Interface for Hypertext Documents, Proceedings of the ACM Conference on Hypertext, Chapel Hill NC, pp. 307–323.

Wanner, Stephan 1985: *Die negative Rasterfahndung. Eine moderne und umstrittene Methode der repressiven Verbrechensbekämpfung*, Rechtswissenschaftliche Forschung und Entwicklung, Munich.

Warren, Jim 1977: Personal Computing: An Overview for Computer Professionals, Proceedings of the June 13–16, 1977, National Computer Conference, Dallas TX, pp. 493–498.

Weiderman, Nelson H. 1979: Personalizing Large Computers, in: *SIGPC Note*, 1 (4), pp. 33–35.

Welsh, H. F. and H. Lukoff 1952: The Uniservo-Tape Reader and Recorder, Joint AIEE-IEE-ACM Computer Conference, New York NY, pp. 47–53.

Wexelblat, Richard L. 1981: *History of Programming Languages [Proceedings of] the ACM SIGPLAN History of Programming Languages Conference, [Los Angeles CA], June 1–3 1978*, ACM Monographs Series, New York NY.

Whisler, Thomas L. 1970: *Information Technology and Organizational Change*, Belmont CA.

Wijngaarden, A. et al. 1969: Report on the Algorithmic Language ALGOL 68, in: *Numerische Mathematik*, 14, pp. 79–218.

Willoughby, Theodore C. 1971: Computer Programmer Aptitude Battery: Validation Study, in: *SIGCPR Computing Personnel*, 2 (3), pp. 6–9.

Yates, JoAnne 2005: *Structuring the Information Age: Life Insurance and Technology in the Twentieth Century*, Baltimore MD.

Yates, JoAnne et al. 2001: *Information Technology and Organizational Transformation: History, Rhetoric and Practice*, Thousand Oaks CA.

Zentralinstitut für Information und Dokumentation 1967: *Erfahrungsberichte zur Anwendung von Lochkarten in Informationseinrichtungen*, ZIID-Schriftenreihe, Berlin.

Zetti, Daniela 2008: Personal und Computer. Die Automation des Postcheckdienstes mit Computern, ein Projekt der Schweizer PTT, in: *Preprints zur Kulturgeschichte der Technik*, 22, Zurich.

Zetti, Daniela 2009: Die Erschliessung der Rechenanlage. Computer im Postcheckdienst, 1964–1974, in: *Traverse. Zeitschrift für Geschichte* (3), pp. 88–102.

Zetti, Daniela 2014: *Das Programm der elektronischen Vielfalt. Fernsehen als Gemeinplatz in der BRD, 1950–1980*, Zurich.

Zuse, Konrad 1936: Verfahren zur selbsttätigen Durchführung von Rechnungen mit Hilfe von Rechenmaschinen, in: *ZuP*, pp. 1–7.

Zuse, Konrad 1948: Über Theorie und Anwendungen logistischer Rechengeräte, http://zuse.zib.de, pp. 1–38.
Zuse, Konrad 1980: Installation of the German Computer Z4 in Zurich in 1950, in: *IEEE Annals of the History of Computing*, 2 (3), pp. 239–241.