The German TR 440 Computer: Software and Its Development

Hans-Juergen Siegert

The TR 440, a large-scale computer developed and manufactured by the German company AEG-Telefunken, was one of the first commercial time-sharing systems. This article focuses on the TR 440's software and software development between 1965 to 1974, describing the technical highlights and showing how the software development was managed, who was responsible for various parts, and how and why certain decisions were made.

The problems of software development in the late 1960s had more than one dimension and included programming problems, project management, and system development. To provide an understanding of the complexities and resulting problems of software system development in the late 1960s, this article describes the software and its development of one of West Germany's most important large-scale commercial computing systems, the TR 440, which was developed and manufactured by the German company AEG-Telefunken (see the related sidebar). (The development of the TR 440 hardware and software was partially funded by the German Ministry of Research and Technology.[1,2])

The TR 440's impact was felt far and wide within West Germany as the lessons learned during its development were applied to the nascent computer science and computer engineering community there. This article shows exciting and forward-looking developments undertaken by Telefunken's development team, which were unknown in the West at the time. More specifically, this article (one of a set of four in this issue on the TR 440) focuses on the TR 440's software and software development between 1965 to 1974.

AEG-Telefunken, especially the large-scale software development team, had the policy of using only German technical terms. Because there was much pioneering work, new terms often had to be coined. For the sake of authenticity, I use those German terms in this article, while explaining them or giving corresponding English terms. I have also translated the original quotations into English, accepting a loss of authenticity to give readers a better understanding. Unless otherwise indicated, the quotations in this article originated from my personal correspondence with the speakers.

Introducing the TR 4

In 1956, the Telefunken board of directors decided to start a prototype for an electronic switching system for telecommunications.[2] A small team began to develop not just a switching system but a large, fully transistorized mainframe computer for universal use, the TR 4. The team worked in Backnang, a small town near Stuttgart.

The first TR 4 was installed at the University of Hamburg in 1962. Most TR 4 computers were used at universities, but there were also other installations, such as the *Bundesanstalt für Flugsicherung* (German air-traffic control) and the Ministry of Finance in Nordrhein-Westfalen (North Rhine-Westphalia). These scientific and commercial-administrative customers had an important influence on the software of the TR 4 and its successor the TR 440.

In the 1960s, batch processing was the typical operating mode of large computers. A stream of batch jobs was sequentially processed. The user normally had no access to the computer and no way of influencing the operation of his job after delivery because of the characteristics of the computer software at that time. Most programs were written in assembly language, and the user expected only simple job control statements, an assembler, and a library of a few mathematical subroutines from the computer manufacturer. A special luxury was a set of

 Published by the IEEE Computer Society 1058-6180/10/$26.00

History of Telefunken

The TR 4 and TR 440 large-scale computers were initially developed by the German company Telefunken, founded in 1903, and a subsidiary of the German company AEG since 1941.[1–3] AEG and Telefunken merged in 1967 to form AEG-Telefunken, where the development and manufacturing of the TR 4 and TR 440 was continued. On 1 January 1972, the large-scale computer division of AEG-Telefunken became the separate company Telefunken Computer (TC), owned by Nixdorf Computer and AEG. TC was taken over by Siemens on 18 July 1974, and the new company was named *Computer Gesellschaft Konstanz* (CGK).[4]

References

1. P. Strunk, ''Die AEG—Aufstieg und Niedergang einer Industrielegende'' [AEG: Rise and Fall of an Industry Legend], Nicolaische Verlagsbuchhandlung Berlin (Sonderausgabe), 2002.
2. E. Thiele, ed., *Telefunken nach 100 Jahren—Das Erbe einer deutschen Weltmarke* [Telefunken After 100 Years: The Heritage of a German World Brand], 2nd ed., Nicolaische Verlagsbuchhandlung Berlin, 2003.
3. E. Jessen et al., ''The AEG-Telefunken TR 440 Computer: Company and Large-Scale Computer Strategy,'' *IEEE Annals of the History of Computing,* vol. 32, no. 3, 2010, pp. 20–29.
4. H. Janisch, ''30 Jahre Siemens-Datenverarbeitung—Geschichte des Bereichs Datenverarbeitung 1954-1984'' [30 Years Siemens Data Processing: History of the Data Processing Division], Siemens AG, 1988.

subroutines to access devices. Manufacturers therefore provided only limited software. At that time, nearly all users programmed and implemented their own application programs. Generally, the importance of and the possibilities achievable by software were underestimated by the user and manufacturer.

Here is a typical example: In February 1963, the *Recheninstitut der Technischen Hochschule Stuttgart* (Stuttgart University's mathematical institute, which also operated the university's computer center) informed prospective users about hardware details of the new TR 4 computer, planned to be installed in June 1964. Concerning the software, only this was said:

> 5.) Program Library
> ALGOL60 compiler (already proven, and successfully used)
> about 30 programs for elementary mathematical functions
> about 30 programs for matrix calculus
> about 15 programs of various kinds
> about 20 programs for organization and testing.

In another example from 1962, a 12-page Telefunken flyer (*Großrechenanlage TR 4 Kurzbeschreibung*, short description of the large-scale computer TR 4) described the software with only a third of a page in a section headed ''Programming Support:''

> Besides a detailed manual . . . Telefunken provides a comprehensive system of effective, harmonized utilities, programs of standard type for using magnetic tapes, and Superprogramme [Special system programs]. The utilities include programs for testing the main computer, external devices, and programs, along with organizational utilities for input/output of 80-column punched cards and 5- to 8-channel paper tape, as well as a program to communicate with the operator's typewriter. The programs of standard type for work with magnetic tapes include automated input and output of data, sort and merge programs as well as a symbolic addressing system as coordinating program. Superprogramme are the Verteilerprogramm for automated organization of parallel execution of multiple application programs, and the generating formula translator for ALGOL60. For science and research, in addition to ALGOL60 we [Telefunken] also provide programs for mathematical methods, which begin with elementary functions and extend over matrix calculus, polynomial calculations, first-order differential equations, and Fourier analysis/synthesis to statistical autocorrelation; all programs in numerous variations to support programming.[3]

Upon reading this, Chris Earnest wrote in September 2007:

> I had nearly forgotten how unimportant software was still thought to be in the fifties and early sixties. Still, a lot of progress had been made since the early days, as illustrated by this story that John Cocke told me, which really happened: In about 1949, the engineers at MIT's Lincoln Labs finally finished assembling the Whirlwind computer, the first

> real-time, general-purpose digital computer, which later evolved into the AN/FSQ-7 computer used in the SAGE air defense system. The computer used vacuum tubes, so was huge, and it had taken some three years to design and build. Component tests had been successful, and once assembly was complete, the engineers tried to run a small test program for the entire computer. It wouldn't run. They rechecked all connections, reran the component tests, etc., found and fixed a few hardware bugs, then tried to run the test program again. Still no luck! They repeated this sequence several times, and although finally there seemed to be no more hardware bugs, the test program still wouldn't run. Finally an insight—everyone knew how simple software was, and the test program was short, but could it have a bug?!? Sure enough! Once it was fixed, the test was successful. John postulated that this was the first software bug ever identified as such.

In keeping with the importance attached to software at that time, the software team in Backnang was small. The team was headed by Gudrun Beyer and Mr. Url. The focal point of the development was the coordination between parallel CPU and I/O device tasks by the *Verteilerprogramm*,[4] which was a dedicated small part of an operating system. (I use the abbreviation BS [*Betriebssystem*] for an operating system from now on.) The *Verteilerprogramm* was hardwired and allowed no more than eight tasks (processes) with fixed priorities. The software team already envisioned the possibility of an extension to a simple monitor (a small BS). Manfred Evers remembered:

> I started to work with Telefunken in 1963 and took over the Verteilerprogramm from Dr. Url and the program for the operator's typewriter from Mrs. Dr. Beyer. At first, I recall it exactly, I understood next to nothing. Then came the many nights on the hardware testing floor to make the Verteilerprogramm ready for hardwiring, and a red face if an error still remained and the girls in manufacturing had to rewire the program. Then very soon I got my first dressing-down: I was summoned to Mr. Gierse [who managed the computer center of the Ministry of Finance of North Rhine-Westphalia] in Düsseldorf and was obliged to watch about 20 magnetic tape drives engaged in the computation of the automobile tax for North Rhine-Westphalia, until everything came to a sudden stop! ''Now Mr. Evers it is your turn,'' said Mr. Gierse. This was literally burned into my own ''hardwired memory.'' My precious Verteilerprogramm had dribbled away a single interrupt, which I then had to generate artificially/artistically while all the others went to lunch. There are things you never forget!

In 1959, Telefunken founded a new division in its plant in Constance: *Informationstechnik* (Information Technology).[2] The TR 4's development and production was transferred to Constance in 1963.

The TR 4 software team was headed by Wolfgang Frielinghaus until 1966. He joined Telefunken in Backnang on 1 April 1962 and collaborated on the software development for the TR 4. In December 1963, he became head of the TR 4 software group with about 15 members. In 1966, he collaborated on the specification of the TR 440 system software, and on 1 January 1967, he took over the team responsible for the programming system and the compilers for the TR 440.[5] The job reference for Frielinghaus emphasized that he conceived and implemented the TR 4 Fortran compiler within one year. In 2007, Frielinghaus noted:

> A further big flap arose because a TR 4 FORTRAN compiler was promised to the University of Darmstadt. The implementation was commissioned to a university. One year before the delivery date I asked the assistant, responsible for it, to present his project. All that existed were two memos with vague ideas. His doctoral thesis got in the way. I immediately canceled the contract, wrote a FORTRAN compiler within four months, and then tested it. My staff implemented the I/O, the mathematical library, and the source dump. The internal list management was completely dynamic and the report of syntax errors was user-friendly.

The TR 4 software development group was headed by Günther Schlenstedt from 1966 to 1974, when maintenance and further enhancements were discontinued. The group was responsible for system software, a library of mathematical subroutines, and some application software such as utilities for graphics and plotting, sort and merge, critical path methods and network planning, and digitally programmed machine tools (EXAPT).

TR 4 basic software

The TR 4 software developed by Telefunken was not always successful. The assembler lost its importance because most application programs could be written in Algol since a compiler for Algol 60 was provided early on by the Munich University of

Technology. Using the TR 4 *Verteilerprogramm* as a base, a BS for the TR 4 was built at the Munich University of Technology in 1963 and 1964 by Hans-Rüdiger Wiehle, Gerhard Seegmüller, Ferdinand Peischl, and Wolfram Urich,[6] in close cooperation with the Leibniz Computer Center of the Bavarian Academy of Sciences and Humanities. The system philosophy came largely from Wiehle, Peischl implemented the assembler, Wiehle implemented a program to modify the system, and Seegmüller designed the operating system's implementation-oriented structure and implemented it in 1964. (The German term for an operating system, *Betriebssystem*, was coined in the early 1960s, perhaps even within this group.) In a 2007 email, Seegmüller wrote, ''Writing these lines it became clear to me, that we used the term *Betriebssystem* as early as the end of 1961 or the beginning of 1962 and not initially in 1963 or 1964 ...''

The TR 4 BS was technically ahead of its time but was hardly noticed by the international community (a similar fate befell many other European achievements). The BS was a batch-processing system with one main thread of tasks. Besides this main thread, other programs could run in parallel, especially utilities for the operator. Thus, the BS supported multiprogramming. The hardware, however, provided no support for access protection between programs (including the BS) in main memory. Some other concepts of the BS also surpassed the hardware capabilities of that time, particularly the following.

An *operator* (OP) was a program running under a BS. This term was derived from mathematics and denotes a procedure to perform a mathematical operation (obeying mathematical laws). Therefore, in the context of Telefunken software, OP always denoted a program obeying certain rules. OPs could run conceptually in parallel and were allowed to change only those devices or memory allocated to them. The software people compared OPs with machine instructions and inferred that therefore OPs needed a sophisticated super automaton (a BS) to manage their sequencing.

An important idea was that housekeeping tasks of the BS should also be implemented as OPs and not as part of a monitor. This train of thought leads immediately to the current concept of system processes. ''Unhooking all control tasks from the system—except the few which are for the sequencer—and structuring them into operative units of standardized shape allows a much higher level of universality and flexibility than with traditional monitor systems.''[6]

The super automaton's basic tasks included adding and removing OPs. There were sequences of OPs, and data had to be transmitted from an OP to its successor OP. The currently planned sequence was defined by a control list, which could be changed by any OP. Therefore, an OP could easily define any desired OP as its successor depending on the result of its work. For instance, this feature was used when an error was detected during a run; then the OP inserted an error-handling OP, such as a post-mortem dump, as its successor. An OP could also behave like a shell of today. The OP had only to insert itself in the control list as the successor of the OP it was going to start (its immediate successor in the list). Because OPs could change the control list, this planned sequence could also be changed subsequently. If the control list became empty, a special OP was run that read the next job from an input device.

There was a system call to define a port for an OP to handle errors. ''Every modern computing automaton allows a program to handle its errors by itself.''[6]

OPs could request hardware resources as needed. The BS kept track of allocated and free resources and monitored the use of allocated components.

The paper cited concludes: ''A system of the kind described is currently being implemented for the Telefunken TR 4 computer by the computer center of the Bavarian Academy of Sciences and Humanities.''[5] The references contain papers about the Atlas-BS, the IBM 7090/7094 monitor, dynamic memory allocation, and concurrent programs.[6]

As I mentioned earlier, this BS became the BS for the TR 4 and was transferred to Telefunken. Maintenance and further enhancements were made in Constance. Gisela Hoffmann (together with two Telefunken programmers) and Gerd Sapper (together with two experienced staff members of Hamburg University) implemented the extension to a disc operating system from mid-1966 to mid-1967.[7] Further enhancements also carried out under Schlenstedt's leadership include long-term data storage, remote batch, and coupling of two TR 4 systems. Besides the TR 4 BS, Seegmüller also designed and implemented the Algol 60 compiler for the TR 4. In connection with this work, he created and implemented innovative testing

Table 1. Estimation of manpower required to develop the TR 440 (in man-years).[15]

System development	1965	1966	1967	1968	1969	1970	Total
Hardware development	25	25	20	15	6	4	95
Hardware I/O	10	15	15	10	5	2	57
General	5	10	10	5	2	1	33
Programming	20	25	25	10	7	3	90
Total	60	75	70	40	20	10	275

aids and a source-code-oriented post-mortem dump. Seegmüller remembered:

> The idea of such a function had its origin in H.R. Wiehle expressing displeasure in the spring or summer of 1962 based upon experiences with the ALGOL services in the computer center running the PERM (a computer built by Munich University of Technology). Afterwards F. Peischl and I went to a blackboard on the second floor and within a day designed an (obvious) realization of the mechanism required: ''nibbling from the stack.''[8]

From TR 4 to TR 440

Work on the TR 4's successor began in approximately autumn of 1962. The name changed several times—TR 14, TR 44, TR *, TR 400—but finally TR 440 was chosen. In a 1963 memo, Fritz Rudolf Güntsch,[2,9] head of the AEG-Telefunken computer division, reported on a conversation—presumably with Fritz Ludwig (F.L.) Bauer (professor of mathematics and informatics at Munich University of Technology) and Seegmüller—about the TR 4 successor in a meeting in Munich during the acceptance process for the TR 4:

- The instructions of the TR 4 are seen as unbalanced. The successor should have a much smaller and more systematic (compiler-friendly) instruction set.
- The times where the TR 440 had to be program compatible with the TR 4 are definitely gone, since programs needed to be compiled. It is therefore not worthwhile to ignore new insights about computer architecture and instruction sets.

New hardware and software architectures emerged at this time, stemming especially from initial ideas for the transition from single-user batch systems to multiuser time-sharing systems:

- 1961: IBM defined the System/360 principles of operation and announced the first computers of this family in 1964.[10]
- 1961: The Massachusetts Institute of Technology (MIT) carried out the first experiments with the Compatible Time-Sharing System (CTSS) for the DEC PDP-1 and the IBM 709.[11,12]
- 1962: The first time-sharing-system in the world was implemented for the DEC PDP-1.
- 1962: British Ferranti realized paging for their Atlas computer.[13]
- 1964: The Darthmouth time-sharing system is presented, developed by John Kemeny and Tom Kurz for a GE-265.
- 1965: Corbato presented a paper on Multics at the 1965 ACM Fall Joint Computer Conference.[14]

The AEG-Telefunken board of directors was indecisive about the TR 440. For example, a development stop was ordered on 19 October 1964. On 4 November 1964, Eike Jessen, head of the large-scale computer development department in Constance since 1964, and Heinz Voigt, chief architect of the TR 4 and TR 440 CPUs and project manager of hardware development, sent a letter to the board of directors in which they pointed out that Voigt had developed a new concept for the TR 440.[15] He planned to build it using integrated circuits. It seemed possible to finish a first computer in time for the 1967 Hannover trade fair and even to deliver one or two computers in 1967 to customers. Jessen and Voigt estimated that eight computers per year could be built from 1968 on.

It was characteristic of that time that the letter did not mention system software. Only in the estimate of the manpower needed is there a row with a total of 90 man-years (MYs) for programming (see Table 1). It is highly probable that ''programming'' included software for debugging and testing the hardware, perhaps even microprogramming and tools for computer-aided design. It is further mentioned that eight MYs were needed in addition to realize the program library, for which the sales department was responsible.

Jessen continued the letter as follows:

> The valid and frequently raised objection that programming probably cannot be finished within the necessary time seems to me less threatening since we have extended the project time to 2.5 years up to the prototype, and we are going to end our activities for the TR 4 and TR 10 promptly. My opinion is, that the estimate of the programming effort is viable, and I am sure that it will allow the realization of a good software package. The whole programming effort for the TR 440 is 90 MY compared with 110 MY for all TR 4 programming up to now and envisaged for the future.[15]

It became apparent later that—for whatever reason—the manpower needed for software development was at least 10 times as much and the period of time to delivery more than twice as long. The estimated manpower curve also did not account for the fact that software testing could start only when the hardware was available, so the hardware and the software testing had to be sequential. Therefore, any delay in hardware would cause about the same delay in software.

The final decision to build the TR 440 came only when AEG-Telefunken was awarded a contract with the German Research Foundation (DFG) for installing a large computer at the German Computer Center in Darmstadt. Now a successful realization of the TR 440 was a matter of prestige for AEG-Telefunken and the DFG.

Operating system BS1

In 1964, the TR 4 operating system designer Wiehle joined AEG-Telefunken and became head of the TR 440 BS development group. The head of the group for the programming system and compilers was Frielinghaus.

Wiehle had far-reaching, fundamental visions about a BS that would allow an extremely rich and, insofar as possible, unrestricted interaction with hundreds of users simultaneously. Of course, all the basic components of today's multiprocess BS had to be designed and invented first, such as resource management, resource allocation planning and tasks scheduling, data management for long-term data storage, and access protection mechanisms. No paradigm existed for this kind of operation of a computer system, however, and there were at most some ideas and some limited experimental implementations of parts. This meant that potential customers, even at universities, had no experience with dialog systems nor were they familiar with the technical concepts and notations. For example, AEG-Telefunken staff members, especially Wiehle, were repeatedly asked to explain the notion of a process. Clearly, conflicts were inevitable given the extreme visions about a BS coupled with hardware that was not suitable and too expensive for these visions. For technical or economic reasons, contemporary large-scale computers had a main memory below 1 Mbyte and disc storage of 10 to 100 Mbytes on a single drive. The TR 440 CPU had a clock rate of 16 MHz and achieved approximately 800,000 operations per second.

The time-sharing BS developed by Wiehle and his team (especially Herbert Meißner and Mr. Rübin) was called the BS1. Detailed descriptions of the BS1 are available elsewhere,[16–19] so I only outline some important features in this article. Two concepts central to BS1 were the notion of an *Auftrag* (order) and the *Nutzenfunktion* (benefit or utility function) used for scheduling jobs. The BS1 had a modular construction with a process architecture (see Figure 1).

The BS1 managed numerous programs, or more exactly running OPs. A running OP was called *Operatorlauf* (OL). An OL could be suspended for a long time, for example, because it was waiting for input from a user who had temporarily left the terminal. The BS1 resource planning and scheduling process (*Kontrollfunktion*, KFK) used *Nutzenfunktionen* (NF) internally. An NF was a time-dependent function that specified in a simple way the benefits (*Nutzen*) to a user if his job was successfully finished at a certain time (see Figure 2). The KFK's goal was to maximize the total benefit. In a talk given for the sales department, Wiehle stated:

> According to a widespread view it is totally clear that the goal of operation is maximal utilization of all computer components. But for a computer of the kind of a TR 440 it is a totally erroneous conception that this could be a primary goal of operation.[17]

NFs provided a basis for scheduling tasks. An NF was computed by the BS from settings of the computer center and from the user's external job specifications, such as the priority requested and resources required. Recomputation was necessary when the computer operation's external or internal situation changed. NFs are far from simple, and questions arise when one looks at them more

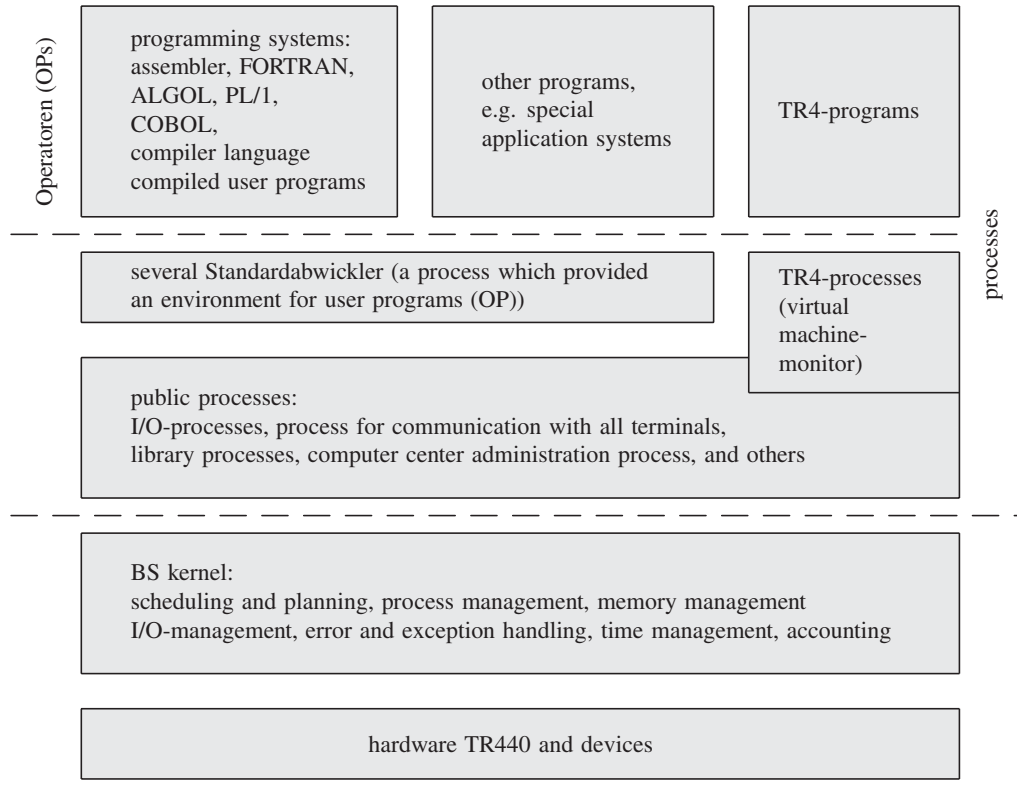


Figure 1. Block diagram of the time-sharing BS1 operating system.

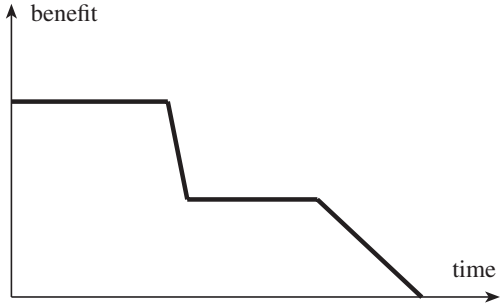


Figure 2. A simple function of benefit (*Nutzenfunktion,* NF).

closely. For example, how should an NF be defined for a conversational job? Does an NF have to be dynamically modified to achieve certain operational goals such as a balance between batch and conversational jobs?

When maximizing the benefit, it was explicitly allowed and sensible that a BS could reject running a job. Wiehle explained in 1967:

> With this I just want to tell you: If we look at the NF and some consider it as terribly complicated, then on the other side I can tell you only, this is extremely simple compared to the assessment one actually needs. I believe this is something we will inevitably encounter in the near future in a high degree.[17]

This kind of computer operations control did not become accepted, and scheduling today is still primitive compared with the ideas devised for the BS1.

BS1 deployed virtual addressing, but no demand paging. (''Although virtual storage is appealing in concept, it has yet to be proven entirely satisfactory in practice.''[20]) Instead, *Gebiete* (data regions) were transferred between main memory and secondary storage. These regions were contiguously stored in the process address space and in secondary storage. Therefore, only a single hardware transport instruction was necessary to transfer a whole region. For example, a region could contain the program code, the permanent data, or the dynamic variables and stack. This concept probably made sense for optimizing transport times, but it did not solve other problems, such as handling programs too big for main memory. The extent to which those other considerations played a role in the decision is unknown.

Multiaccess and multiterminal services were mechanisms that gave many users direct access to the computer from suitable dialog stations. At any time, users could interact with their conversational programs running in the TR 440. It was planned to allow the start of a (nested) conversation at any time, without restriction. The traditional conversation with alternating program output and user input, where the program would have the initiative, was considered too restrictive.

This short exposition of the BS1 clearly reveals the ambitious goals, which were definitely far beyond the state of the art at that time, and which in hindsight posed an extreme risk for the company. Therefore, we can understand today that the potential customers were skeptical and there was a deep gap between development and sales personnel. The situation was aggravated by a serious manpower shortage in the development group. Wiehle explained in 1967 at the beginning of a talk to the sales department:

> The heavy burden on the program development group prohibits a special preparation of these lectures, which therefore have to be regarded as lectures of the shop floor. It also can not be the obligation of development, to engage in a user-oriented presentation, even if it might be desirable from a marketing view. The sales-oriented perception must and can only be expressed by descriptions written by sales people. Nevertheless we hope our lectures are as supplements to our internal documents a useful input for the sales people and can contribute to overburdened developers no longer being requested to support sales activities.[17]

Figure 3 shows that software development was part of the TR 440 development department in 1968. It became apparent later that this organization did not meet the requirements, importance, and necessary environments for the software. Therefore, software development was soon separated from hardware development into a new department (denoted by GR/EP or GR/P) on the same organizational level as hardware development (then denoted by GR/ER).

GR	large scale computers	F.R. Güntsch
GR/ E	development	E. Jessen
GR/E1	system, devices	H. Voigt
GR/E7	programming TR440	H.R. Wiehle
	GR/E71 planning, coordination	H.-J. Siegert
	GR/E72 operating system	Rübin
	GR/E73 I/O management	Rösner
	GR/E74 data management	Leeb
	GR/E75 terminal communication, TR4 virtual machine	M. Evers
	GR/E76 Abwickler	Pohlmann
GR/E8	programming systems	W. Friehlinghaus
	GR/E81 assembler	R. Durchholz
	GR/E82 FORTRAN	W. Froehlich
	GR/E83 ALGOL	H. Zima
	GR/E84 COBOL	Brandmarker
	GR/E85 I/O-procdures, sort	Mühlbach
	GR/E86 program library	Schäfer
	GR/E87 command language, programming utilities	P. Namneck
GR/E9	programming TR4	G. Schlenstedt

Figure 3. The AEG-Telefunken TR 440 organization chart shows that software development was still not a separate department in June 1968.

Operating system BS2

In 1967, Helmut Köhler joined AEG-Telefunken from IBM and took over the leadership of the sales division (GR/V) from Egbert Ulbrich. He was supposed to increase TR 440 sales, especially by developing the commercial market. To enter the commercial market, a smaller and therefore less expensive version of the TR 440 seemed necessary. Because main memory was expensive, minimal main memory needed for a useful operation was to be reduced. A goal of 32,000 words with 52 bits each, and even 16,000 words, was set. This was outside of the realm of the BS1 and could be reached only with a plain, more or less traditional batch BS.

In 1968, Köhler hired Jürgen Esch to design and implement such a BS for the commercial and administrative market: the BS2. Esch had just received his doctoral degree and was head of the computer center of the University of Technology Hannover, where his advisor had been Wolfgang Händler. Esch brought all members of his group (about 10) with him to Constance, including Albert Noltemeier, Manfred Römermann, and Herbert Stuhlmann. Lothar Krause brought with him his profound TR 4 knowledge, acquired at the Hamburg University computer center. Later, Günther Stiege from Siemens joined the group. The *group Esch*, as they were called, belonged organizationally to the software department (GR/P), but was separately housed and formed a close community. To a certain degree, it had its own way of life.

The TR 440 programming system and the compilers used with BS1 were also to run with minor modifications under BS2. The

first version to be delivered allowed batch processing with one foreground and one background program.[21,22] Batch jobs were run in background. System administration programs, such as spooling utilities, ran in the foreground. To operate the computer, the BS2 used a dedicated program that ran with the highest priority.

BS2 occupied a part of main memory, commencing at the beginning. It consisted of two regions: a fixed part used for the kernel and loadable modules and a variable part that was broken into fixed-sized blocks of 128 words and used for buffers. The sizes of both parts were fixed at compile time; thus, the BS2 could be adapted to the size of main memory.

The remaining main memory (p-region) was allocated to programs and was broken down into frames of 1,024 words. One foreground and one background program could run simultaneously. When started, an initial amount of frames were allocated. Background and foreground programs could dynamically request more frames or free allocated frames. A background program could use the whole p-region, but a foreground program was limited in size. Conflicts were avoided because the maximal demands on main memory were known and the foreground program did not have access to system files. Thus, there was virtual addressing but no demand paging or program swapping.

A simple file-management system was oriented toward the requirements of high-level languages, especially Fortran. A user program could access only eight files, all of which had to be specified at the command level before running the program. The BS2 opened the files before starting the program. On a logical data-management level, there were records, on a physical level fixed size blocks. Blocks were directly transported between the device and the user program's address space.

The first version of BS2 (multiprogramming with two programs) was finished in the spring of 1970.[1]

For the second version of BS2, batch processing with up to eight programs simultaneously and remote batch were projected. Besides this, design and implementation of substantial enhancements of features were planned, such as dynamic job scheduling with priorities and preemption, dynamic resource allocation and revocation during program operation, code sharing, dynamic lists with their size adjusted to demands during the BS2 run, and indexed-sequential file access. Jobs were to be listed in one of four queues: express, special, I/O intensive, and computationally intensive. Jobs in the express queue were to get absolute priority; all other jobs would be scheduled to achieve as high a utilization of all computer components as possible. By spring 1971, important elements of this concept had been implemented, but the entire system was not yet finished nor acceptance tested.

In the spring of 1971 at the latest, I questioned the continuation of the BS2 efforts. My reasons included the fact that entrance into the commercial market had not occurred, the enhancements of BS2 inevitably led in the direction of the time-sharing system BS3 and to a bigger resource demand, the successful BS3 was already available and had reasonable resource requirements, always adapting the programming system and the compilers to BS2 and BS3 meant duplication of work, and last but not least, the company's financial situation was becoming more difficult.

After long, intensive considerations, appraisals, and conversations Kurt Scheidhauer—who was the AEG-Telefunken coordinator responsible for all developments in the division N3 (including the large-scale computer department)—became convinced of these arguments. With his backing and support, work on BS2 was stopped in April 1971 before the second version was finished.[1] The first version of BS2 was also never delivered to customers. After the project was stopped, Esch and all members of his team left AEG-Telefunken. Nearly all went to the German research center *Gesellschaft für Mathematik und Datenverarbeitung* near Bonn.

Upheavals in personnel and organization

As I mentioned earlier, the goals of the BS1 software group were too ambitious, the time schedules were too tight, and the manpower was too low. In addition, the potential customers were skeptical and questioned whether the software development could be successful. Hardware and device development also had huge technical problems and large schedule slips. All these problems led naturally to changes in responsibility for the various projects.

Early in 1968, Köhler separated software development from hardware development. This was an extremely important and innovative action because the needs and culture in the hardware and software groups were

different then, as they still are today. After that, the GR/ER department was only responsible for hardware, and Köhler, head of the sales department, also became head of the new department GR/P, which was responsible for software development (programming).

On 9 September 1968, Klaus Bounin was promoted to the head of operating system development (GR/P1). Wiehle, who up until then had been the head of that group, became a liaison to a group with the Leibniz Computer Center in Munich, which also developed a BS for the TR 440 (*Betriebssystem-München*, BSM).[23–28] This BS was finished, but never deployed. In October 1971, Wiehle left AEG-Telefunken.

I joined the AEG-Telefunken sales division in 1967 to give (potential) customers and the sales people technical support. Wolfried Hanefeld and I wrote the first sales-oriented description of the real software—not just about visions—called the ''golden bible.'' I was then involved in planning BS1 features and coordinating software development as head of the group GR/P1, reporting to Wiehle. In 1969, Köhler became head of the division for large-scale computers (GR). On 20 May 1969, I became head of the GR/P department. At that time, the department oversaw the TR 4 software, the TR 440 software (except for commercial applications), the software activities for a TR 440-successor, and the group of customer-oriented advisory services for sales and marketing.

Double TR 4 system

In the autumn of 1966, the DFG ordered a TR 440 computer to be delivered to the German Computer Center on 1 July 1968. By the beginning of 1968, it was obvious that the system could not be delivered on time.[26] It was also impossible to deliver any preliminary version of BS1 on time. As an emergency solution, the double TR 4 system was born, later derisively also called *doppelschläfrige TR 4*, or double-sleepy TR 4, because there were two TR 4 virtual machines, and the TR 440 initially was running with lower frequency than promised. With this system,[27] two original TR 4 operating systems, along with all TR 4 software, could run in one TR 440. Today this would be described as two virtual machines with TR 4 hardware interfaces running on a TR 440 host. This solution was possible since the TR 440 had a TR 4 mode for the execution of TR 4 instructions because it was already a design goal of BS1 to run a TR 4 BS as a process.

The BS1's slow progress caused great concern as delivery dates came closer.

The TR 440 resources, including main memory and devices, were divided and allocated to the two TR 4 systems. A program, then called *Ämulator* (what today we would call a virtual machine monitor), mapped the TR 4 hardware onto the TR 440 hardware and managed the dynamic allocation of the CPU.

This software was designed and implemented starting in February 1968 by a six-person team. Project manager Manfred Evers reported directly to Köhler about this project. Gisela Hoffmann was an important member of this team right from the start. She became responsible for the double TR 4 system in October 1968 and was in charge of the system's delivery, maintenance, and support. Hoffmann was the first female group leader in our division and also the youngest.

The design phase took four weeks, coding and testing an additional four weeks each.[28] Extensive acceptance testing was conducted during the second half of 1968. At the end of 1968, the computer was delivered to the customer, followed by a three-month trial operation.

Software development took longer than planned because the hardware development group had to implement changes in the adaptation unit to TR 440 for some devices because of hardware bugs. After about an eight-month delay, the double TR 4 system was officially put into operation in February 1969. It received its certification of acceptance at the end of 1969.[1] The double TR 4 system special solution was possible only because of the highly qualified and competent Constance software development team.

Birth of the operating system BS3

Coding of a first development version of BS1 was finished in the autumn of 1969, but the system was a long way from its goals.[1] For example, it could only run a single program; multiprogramming was not yet possible. The BS1's slow progress caused great concern as delivery dates came closer.

The next delivery deadline was June 1970 for the computer center of Bochum University (Ruhr-Universität Bochum), headed by Hartmut Ehlich. The real threaten, however, was that parts of the company, of customers, and of the DFG (German Research Council) had extreme distrust in individual people, as well as in BS1 concepts and target dates. The firm's only chance of success was to make a complete break with the past, at least in my opinion.

At the beginning of October 1969, within a very few days and without informing outsiders, Scheidhauer and I made the lonely decision to stop development of the BS1 and replace it with development of the all-new BS3 operating system headed by Frielinghaus. Alexander Hoyer took over responsibility for compiler development from Frielinghaus, and within a few days, the department GR/P1 responsible for developing the BS1 was broken up and, with only a few exceptions, integrated into a newly formed department responsible for the BS3 development.

This ''swapping horses in midstream'' was a high-risk decision. If unsuccessful, it would have been the end of the large-scale computer division because there was no other product to sell. This decision surprised all the customers but, fortunately, was unanimously acclaimed. The BS3 became a complete success and was used in all TR 440 installations as an interactive time-sharing system. The first delivery of the conversational BS3 to the computer center of Bochum University was on time in June 1970.

In a separate section of this article, I describe the influence of the *DFG-Abnahmekommission* (DFG-AK) on the development of our software, and its impact on the success. The DFG-AK was a DFG committee initially created to do acceptance testing. Alexander Giedke, a member of the DFG-AK, remembered:

> It is no wonder that turning away from BS1 was a surprise for many involved. Even on August 31, 1969 when TR 440 customers met there was not the merest hint about this turn of events. . . . On November 7, at a DFG hearing about the TR 440 a first version of BS3 (batch processing with two Abwickler (ABW) [an ABW is a process which provides an environment for one user program, therefore two ABWs allow two user jobs running in parallel]) was demonstrated, and Telefunken indicated that BS1 was not being pursued any longer. The hearing comprised not only a comprehensive survey, but also led to admissions of the Telefunken company to users and supporters. The demonstration of the BS3 operating system showed that there existed a definitive and feasible concept. . . . In the main this hearing led to the decision to continue work with the TR 440 project.[26]

Operating system BS3

The BS1 development ran parallel to the development of the programming system and compilers. To test the compilers, a testbed was necessary, in this case a BS with basic data management. BS1 was not developed enough for this purpose. In search of a solution, Frielinghaus and his group came across the *Wartungsverteiler* (WV), which was implemented by Jochen Schilling in the hardware division and which Wolfgang Froehlich had suggested to use. Its design was derived from the TR 4 *Verteilerprogramm*. It was used to control test programs, and thus it was a special small BS kernel. An enhanced version of the WV was used as a testbed for compilers. In a 21 June 2007 letter to me, Frielinghaus recalled:

> We built the whole programming system so that BS1 interfaces were called. To test [the compilers] I implemented for my staff a Hilfsabwickler (HIWI), which called services of the WV. Therefore we were not compatible with BS2, but on the other hand it was easy to extend the HIWI to the BS3 ABW.

A HIWI was a special kind of ABW with limited functions for running compiler test programs. The German notation alludes to a HIWI being a substitute for a full-scale ABW running under BS1. The abbreviation HIWI used by the group had a second meaning: it was at that time the abbreviation for a newly hired academic staff member in a university environment.

Based on this test environment, the BS3 operating system was developed under Frielinghaus's leadership. The BS3 system kernel was based on the WV and was implemented initially by Schilling and his group, especially Werner Schwarzmann and Lothar Stolze, but according to specifications of the BS3 group. From 1972 on, the BS3 group took over all the development of the BS3 kernel. In April 1971, Franz Stetter became responsible for the BS3 group, and Frielinghaus became project manager for the TR 550 computer software. Frielinghaus still reported to me because software project management was part of a small matrix organization within GR/P (see Figure 4). The TR 550 was the working title for the TR 440's

successor. The TR 440 software project manager was Joachim Feldmann.

Schilling remembered:

> In about 1965 development of the much more powerful TR 440 computer was begun. Starting in 1968 as head of a group [GR/EV11] I was responsible for the development of software for hardware testing and maintenance, from which evolved the task of programming the operating system kernel for BS3, at first secretly then officially.[29]

Gerd Sapper, member of the Leibniz Computer Center in Munich, took notes on 7 October 1969:

> On Wednesday October 8, 1969 at 9:30 AM a group of programmers of the departments N3/GR/P1, P2, P3, EV11... dealt with making the WV and the HIWI deliverable to TR 440 customers on January 1, 1970.... On July 1, 1970 a version ready for interactive terminal operation was to be delivered. The appointment of extraordinarily qualified personnel to this programming group gives good cause to believe that these dates are realistic.... In the same week I could get a collection of interface documents and lists from W. Frielinghaus.

Along with the BS1 team members, their know-how and practical experience with concepts, solutions, and problems of BS1 development were also directly transferred to the new BS3 team. To some extent, even direct usage of parts of the BS1 code was possible. All the team members were highly competent, had an ambitious goal, and pursued this with unbelievable commitment. They worked a lot of overtime—they could, and did, come to work at anytime during day or night. Discussions with the Employee Council led to at least some relaxation of the requirement for an 11-hour interval between two work days.

By the end of 1969, the BS3 allowed batch processing with two streams of jobs. In the time following, it evolved rapidly through some maintenance versions (MV1 to MV18) into a comfortable time-sharing system with up to three CPUs, remote terminal access, and several computers connected (basic networking).

P Software development Dr. Siegert

P11		Project manager software TR440 and TR86S	Feldmann
P12		Project manager software TR550	Frielinghaus
P2		Integration and quality assurance	Mühlbach
	P21	Internal training	Schütt
	P22	Quality assurance	Froehlich
	P23	Integration and acceptance testing	M. Hofmann
	P24	Program analysis and measurements	Mersmann
	P25	Testing aids	Dr. Schäfer
P3		Application systems	Dr. Knoth
	P31	Applications 1	Gütinger
	P32	Applications 2	Rost
	P33	Applications 3	Schulz
	P34	Applications 4	Benz
P4		Computer center	Wagner
	P41	Operating groups	J. Keppler
	P410	Administration	J. Keppler
	P411	Computer center 1	Westhagen
	P412	Computer center 2	Westhagen (komm.)
	P413	Computer center 3	Bochert
	P414	Computer center 4	Pinggera
	P415	Computer center 5	Bochert
	P42	Service	Möller
	P43	Planning and installation	Jacimowitsch
	P44	Helpdesk and training of operators	Niepelt
P5		Operating system	Dr. Stetter
	P51	Technology and coordination	H. Meyer
	P52	Scheduling (KFK)	Dr. Thurner
	P53	System services 1	Dr. Luhmann
	P54	Abwickler (ABW)	Dr. Burkhardt
	P55	Data management 1	Engelhardt
	P56	Data management 2	Dr. Hoehne
	P57	System services 2	Stadie
P6		Compiler	Schmolz
	P61	Compiler 1	N.N.
	P62	Compiler 2	Mangold
	P63	Compiler 3	Schmolz (komm.)
	P64	Compiler 4	Dr. Krainer
	P65	Compiler 5	Pszolla
	P66	Compiler 6	U. Richter
	P67	Compiler 7	Indefrey
P7		Satellite system	Evers
	P71	Device management	Schichler
	P72	System kernel	Schallenmüller
	P73	Memory management	Heinz
	P74	Secondary storage	Bremke
	P75	Dialog devices	Dr. Mensger
	P76	Data transmission	Dr. P. Meyer
P8		Programming system	Linn
	P81	Programming system 1	Frl. Gebhardt
	P82	Programming system 2	Dr. Kaspar
	P83	Programming system 3	Kuhlmey
	P84	Programming system 4	E. Schmidt
	P85	Programming system 5	Blatt
	P86	Programming system 6	Dr. Kääb
	P87	Programming system 7	Dr. Braess

Figure 4. Software development (GR/P) organization chart (1 June 1974).

Technical highlights of BS3

The BS3 reused several BS1 concepts, including an OL and the ABW. It also used an OP, which was a program running in the environment provided by an ABW; therefore, it obeyed certain rules. Examples were compilers, loaders, shells, or compiled application programs written by users. An ABW was a BS resource and could be dynamically allocated to OPs, or be preempted. From the view of the BS3 kernel, an ABW was a process. The TR 440 CPU had a special ABW mode in which an ABW was executed. Around 1970,[1,30–33] the BS3 had seven ABWs. An ABW's main tasks were managing, starting, and providing services for an OL and providing interfaces for system kernel calls, file management, I/O services, and data access.

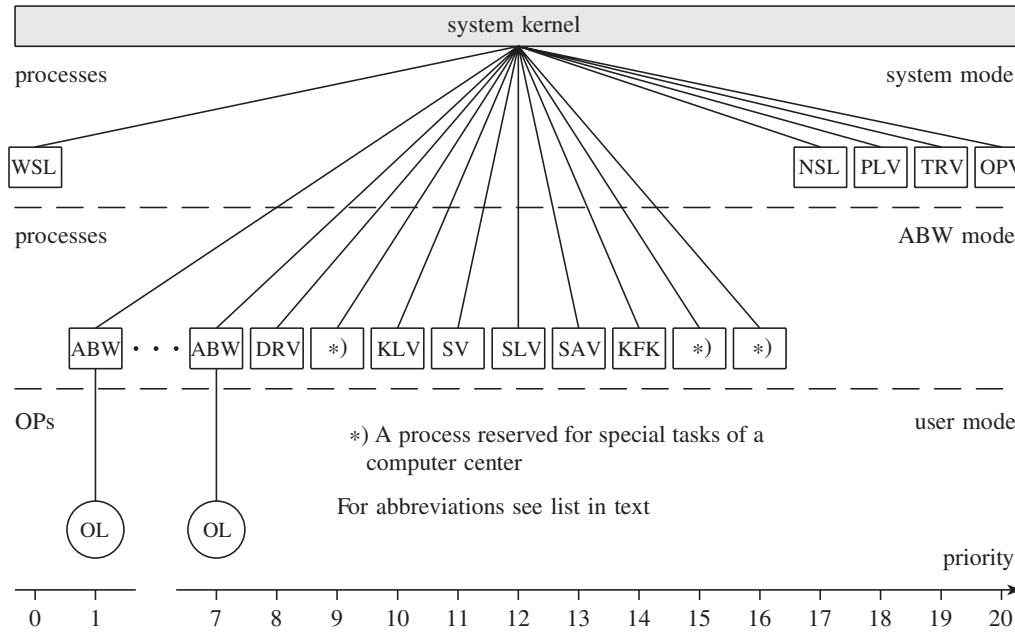


Figure 5. Block diagram of the BS3.[32]

Concurrent execution of OLs was not possible within an ABW, but a sequence of OLs was supported, and services for the management of chains of OLs were provided. Every OL could insert a new OL in this chain at any position. In later versions, OL nesting was possible, so an OL could start another OL and wait until it stopped. If an OL had to be blocked, or would probably have to wait for a long time, then the BS kernel's control function (KFK) sent a message to the ABW. The ABW then preempted the OL, called BS3 kernel services to write the OL's frames to disc, and became free to run another OL.

BS3 processes executed either in privileged system mode or in the ABW mode. A process running in system mode was denoted *Systemakteur*, which needed direct access to hardware components or a device. It had no or short waiting times. Processes to serve slow peripheral devices with long waiting times were run in ABW mode and could be preempted.

Figure 5 shows the processes running in system mode:

- *Operateurvermittler* (OPV), for communication with the operating person;
- *Trommelvermittler* (TRV), administration and drivers of drums implemented by Schwarzmann in Schilling's team;
- *Plattenvermittler* (PLV), administration and drivers of hard discs implemented by Schwarzmann in Schilling's team;
- *Notschleife* (NSL), a process to handle system errors; and
- *Warteschleife* (WSL), a process to be run when no other process was ready to use the CPU.

Figure 5 also shows the processes running in ABW mode:

- *Kontrollfunktion* (KFK), overall scheduling process;
- *Satellitenvermittler* (SAV), connection with the satellite computer TR 86S and its devices, especially all terminals;
- *Lochstreifenvermittler* (LSV), administration and drivers of paper tape devices directly connected to the TR 440;
- *Schreibmaschinenvermittler* (SV), connection to the control typewriter;
- *Kartenleservermittler* (KLV), administration and drivers of punched-card devices directly connected to the TR 440;
- *Druckervermittler* (DRV), administration and drivers of line printers directly connected to the TR 440; and
- ABW.

The BS3 had a system architecture that is still in use today (see Figure 4). Logical addresses of devices were used, which was uncommon at that time. It was not monolithic like other BSs at that time, but was a process-oriented BS, where many parts of

classical BS kernels were implemented as processes. I/O services were separated from the user programs, long before IBM did this. The system kernel was small and managed only processors, main memory, devices, and transport channels to devices. This management covered only reservation and release of components as well as hardware-oriented transport orders to and from devices. Thus, the system kernel provided an abstract hardware interface for processes and concealed special hardware characteristics. The concept was new at the time, but it is fundamental for today's operating systems.

The BS3 had virtual addressing, but like the BS1 it had no paging at first. Instead, whole data regions (*Gebiete*) were swapped between main and secondary memory. There were distinct data regions for, among other things, the program instructions, for constants, for variables, and for the stack. Data regions could perhaps be called segments today. The goal of choosing data regions as transport units was reducing I/O time. To further aid this goal, the transport was executed with just a single I/O instruction, if possible. Therefore, the allocation policy was to store data regions in contiguous blocks of secondary storage, to the extent possible. It was unnecessary to store data regions contiguously in real memory, but they were of course contiguous in virtual address space. This was possible because the transport instructions on the hardware level allowed scattered reading and writing. Thus, several parts of an incoming data stream could be stored in different frames (different real addresses), and correspondingly for outgoing data streams.

The BS3 handled two kinds of user jobs: batch jobs (*Stapelauftrag*) and interactive jobs (*Dialogauftrag* or *Gespräch*).

The KFK, a BS3 system process, handled the strategic planning of resource allocation. Also called logical planning, strategic planning was done on a high level, often before a program can start. For example, if it was known how many hard disk blocks an OL needed, then that number of blocks could be reserved for the OL before starting it; afterward, when running, the OL claimed blocks and only then were real blocks allocated. Responsibility for the KFK lay with Herbert Meißner, who had also been responsible for the BS1's KFK. Resources to be planned by the KFK were not only hardware components, but also software components such as ABWs or other processes. The BS3's KFK did not use the NF concept; it used the classical algorithms of today, such as priorities for batch jobs, preemption of jobs if more important jobs arrive, multiqueue round-robin scheduling for conversational jobs, and good usage of CPUs and main memory. The system had to achieve an adequate compromise, defined by the computer center, for the classical problem of high utilization of the hardware components versus short response times. These goals are incompatible, however, and the situation was aggravated because the financiers demanded an extremely high degree of CPU usage and made the procurement of a new computer dependent on it. On the other hand, especially in a conversational operating mode, high productivity of users demanded good response times from the computer system.

In conversational operation, up to 48 terminals could be active simultaneously—in later BS versions many more. Terminal users could communicate with their programs and influence their behavior. This was possible for interactive programs, which most compiler-generated programs were.

Another BS3 feature was data and file management with permanent data storage. The developers called this feature *langfristige Datenhaltung* (LFD, long-term data storage). Permanently stored data was managed and secured by the BS3 against unauthorized access and loss. Problems and solutions are known today, but the solutions often had to be invented then. Lothar Stolze, who came from Schilling's group to the BS3 group, was the head of the LFD group.

Secondary storage was divided into four segments, the address limits of which were fixed before compilation of the BS. Three of the four segments were used for data regions, LFD, and the documentation system TELDOK (which I explain later on). The fourth segment was not dedicated to a special purpose and could freely be assigned by a computer center.

A hard disk was divided into blocks, each the size of a frame. In each of the four segments, numbering started with zero. The blocks allocated to an object could be scattered.

For administrative purposes, several files were combined into a database. The BS3 supported public, standard, and LFD databases.

The public database consisted of the system's program library, including the compilers, programming system subroutines, utilities mountable to user programs (such

Table 2. Example BS3 system kernel calls (*SSR-Befehle*).

Address			
Symbol	**Numeric**		**Service**
A	0	0	Release an I/O device
B	0	1	Get relative machine time
BP	0	26	Get relative process time
C	0	2	Continue a halted process
D	0	3	Get date
E	0	4	Renew blocked I/O orders
F	0	5	Continue a process in user mode
FA	0	3	Continue a process in ABW mode
G	0	6	Request access to message queue
HLT	0	7	Cancel OL
I	0	8	Get ID (program number) of own OL
J	0	9	Get status of other process
K	0	10	Waiting for short time
KA	0	30	Waiting for input from operator
L	0	11	Delete blocked I/O orders
M	0	12	Output to operator, and waiting for input
N	0	13	Output to operator
NS	0	31	Start NSL
O	0	14	Cancel other process
P	0	15	Request an I/O device
PL	0	27	Transport order for PLV
Q	0	16	Halt a process

as mathematical subroutines), and definitions of macros and commands. The library was set up at system start and allowed read-only access for all users.

A standard database was created at the beginning of each job, batch or conversational. It included the temporary library for the job, including the temporary files, the compiler-generated mountable objects, the lists for backtracking nested subroutine calls in case of a program error, the lists to generate source-code-related dumps, and the descriptions of loadable programs. In addition, a user could create further private databases, which like the standard database were deleted at the end of a job.

The LFD database let a user permanently store files on a hard disk that were not deleted at the end of a job. The file description and other file metadata were also stored in this database. The main index had an entry for each known username. The advantage was that all files for a user were in only one subtree, so each user had his own file namespace. In addition, it was easy to keep unauthorized users from storing files and to control disk quotas.

There were also shared files, access to which was coordinated by the reader-writer algorithm. Separate passwords for reading and writing protected a file from unauthorized access. Processes communicated using the *Depot*, a shared message buffer area in main memory. Its size was 1,000 words, and it was divided into two parts. The first part consisted of message buffers of five words each, and the second was unformatted, so access had to be synchronized. As always, special algorithms and agreements were necessary to ensure freedom from deadlocks and the ability of an operator (person) to take action at any time.

BS3 services were called using the SSR system call instruction. Table 2 shows examples of system kernel calls.

As I already mentioned, file-access services were not part of the system kernel but were located in the ABW. File types were sequential access, random access with record numbers, and random access with short record labels.

The following selection of important milestones[34] connected with delivered MVs (maintenance versions) shows the BS3's rapid evolution:

- MV9 (April 1971): Central (error) log file, reduction of main memory used by BS3.
- MV10 (August 1971): LFD allows daily saves.
- MV11 (February 1972): Further reduction of main memory used by BS3 through segmentation of the kernel, shared regions, and code sharing of compilers and applications; centralized buffer management; one process to serve all devices of a type (multiplexing device drivers); dynamic scheduling of main memory and secondary storage; adjusting priorities as needed; services for removable disks (WSP414); connection of several TR 86S satellite computers in parallel or in cascades; graphic terminals (SIG100) with scroll ball; graphics utilities; and hierarchy of catalogs for all databases.
- MV12 (June 1972): Two or three CPUs supported,[35] restart and rerun; reduction of terminal response times, LFD enhancements, enhanced management of modules of secondary memory, mass storage available (semiconductor-based memory, priced and sized between main memory and hard disks, used by the system as fast drum storage), 96 terminals, and line-oriented displays (SIG50).
- MV13 (December 1972): Enhancements of rerun and restart; 10 ABWs; batch jobs read from removable disks; data stations

(DAS 3200) with card readers, line printers, and a terminal for control connected to TR 86S; plotter; peripheral devices using paper could also be connected to TR 86S; IBM-compatible magnetic tapes; and enhanced generation of the BS.
- MV13N (June 1973): Connecting TR 86S with TR 86S over a long-distance line of 48 Kbaud.
- MV14 (October 1973): LFD provides record-oriented saving and restoring, restart of TR 86S during interactive terminal sessions, demand paging (an ABW service, not part of the system kernel as usual), resource scheduling lets an OL request additional resources, files for sequential access can be larger than 2 Gwords, further enhancements of system generation; segmentation of the ABW, nesting of OLs up to the depth eight, and system kernel is identical for all installations.
- MV15 (June 1974): Saving of preempted jobs over system booting, line-oriented display SIG51, file type with random access and record keys, 22 processes, further enhancements of system generation, and enhancements of system errors' diagnosis.
- MV16 (December 1974): Random-access files larger than 2 Gwords, user restart, removable disk WSP430, operator-oriented management services for removable disks, dial-up terminals, magnetic tapes using ISO standards, several files on a single magnetic tape, and files needing more than one magnetic tape.

This list clearly shows the rapid progress toward reducing system resources, getting higher reliability, automating system generation, connecting the newest devices, and networking. Also of importance were changes in system concepts to be able to implement new functions, achieve better performance, and facilitate enhancements and maintenance (for more and more customers). A big success was operating up to three CPUs simultaneously. Thus, AEG-Telefunken had one of the first—possibly the first—multiprocessor multiprocess time-sharing systems, which was installed in mid-1972 at the Munich Leibniz Computer Center.

TR 86S satellite system and networking

An important and seminal decision at the start of the TR 440 development was to connect character-oriented devices, especially terminals, not directly to the TR 440, but to a satellite computer (TR 86S). The TR 86S was a medium-scale, real-time computer for process control. In a basic configuration, the TR 86S was connected to the TR 440 with a high-speed coaxial cable. At first, only teletypewriters and alphanumerical displays in teletypewriter mode were used as terminals. Later, there were also alphanumerical displays in line mode and graphical displays (vector graphics) with one-button trackballs. Peripheral devices such as paper-tape readers and writers, line printers, punched-card readers and writers, or data stations could be connected directly to the TR 440, or preferably to a TR 86S. A typical data station included a punched-card reader, a line printer, and a typewriter for control by an operator. Some of these devices could be connected over dedicated, leased, or (low-speed) dial-up lines. This was a first, but important step toward decentralized computing, allowing remote access to the computer from devices installed near a user's office.

Later, from about 1972 on, several satellite computers could be connected to a TR 440 or another TR 86S, resulting in a tree-like cascade. The satellite computer's software was built in such a way that it was independent of the TR 86S's position in the tree. The satellite computers could also be connected by leased lines of 48 kilobits per second (kbps), fast at that time (1973). Thus, a TR 86S with terminals or other devices could be installed near a user. In a further development step, a TR 440 or another mainframe computer could be connected to a TR 440 with a TR 86S in between. These were fundamental achievements leading to decentralized computing and computer networks, both central concepts of today's information technology.

The satellite system[36,37] consisted of two main parts: the *Satellitenvermittler* (SAV) in the TR 440 and the TR 86S operating system, also called *Konsolverteilerprogramm* (KVP) or *Satellitenprogramm* (SAP). Design and implementation was done in a group under the Manfred Evers' leadership.

For processes running in the TR 440 under the BS3, the SAV provided an interface for accessing terminals and other devices connected to the TR 86S. The SAV gathered all output for all TR 86S devices and assembled it into large data blocks. The SAV sent these blocks to the SAP at the SAP's request. Analogously, the SAV received a block of input data from a TR 86S, extracted from it the input of the different TR 86S devices, and distributed that to the corresponding processes waiting for it. If the input from a device was not yet

The importance of decentralized computing and networking was clear early on to the people responsible for development.

complete, it was gathered and stored on secondary storage until the last part of the input from the device arrived. The TR 440 always had to accept data blocks from a TR 86S. These restrictions were for the purpose of deadlock avoidance in the TR 86S software.

The main goals in designing the satellite system were

- removing from the TR 440 the high load of interrupts caused by character-oriented I/O to and from interactive terminals,
- removing the burden from the TR 440 caused by the numerous protocols used in accessing terminals and other peripheral devices, in operating transmission lines, and in connecting to other mainframe computers,
- relieving the TR 440 from always having to translate characters according to the many different character sets and character codes used for the devices,
- putting all processing with tight real-time requirements into the TR 86S, and
- providing a uniform device-independent interface for all devices connected to the TR 86S.

The SAP included device drivers for all devices connected to a TR 86S. A device driver served all devices of a class and gathered their input. The SAP aggregated the input into blocks and sent it to the TR 440. Incoming blocks from the TR 440 were decomposed into output strings and given to the appropriate device drivers or forwarded in blocks to a connected TR 86S. Thus, the number of interrupts for the TR 440 was small compared to the number of interrupts caused by all the TR 86S devices. The SAP knew essential parts of the TR 440 commands. Therefore, it was possible to discard input not within a batch or conversational job, interrupt and cancel terminal output, display input characters on a graphics terminal at a location predefined in a previous output sequence, or recognize the final character of an input sequence.

There were tight real-time requirements for input from devices, but not for output. Nevertheless, output delays, especially when recording an input character, could not be allowed to exceed about 100 ms. The necessary real-time behavior was achieved by techniques similar to those used in today's real-time operating systems:

- The SAP was memory resident, meaning no secondary storage connected to a TR 86S.
- The SAP had a highly modular design; each job was accomplished by executing a sequence of modules (a sequence of simple tasks).
- The maximum processing time a module used for its task was known, and the modules were defined so that this maximal time was compliant with the real-time requirements.
- Scheduling was based on fixed priorities allocated to the SAP modules.
- A module could be interrupted only briefly by a device interrupt and was immediately continued after initial basic interrupt handling. As one would expect, there was no preemption by the scheduling algorithms; a module had to release the processor after finishing its task.

New tasks were created by executing a module (task) or by the basic interrupt handler. When a module released the processor then scheduling allocated the processor to the module of highest priority that had a task to execute. The only exception was when a module could not finish its task but had to wait for another module. In this case, a temporary transfer of priorities was made.

The SAP was designed so that deadlocks could not occur and bottlenecks were avoided. As I already described, modules were interrupted only briefly by devices. Internal communication between modules took place by making an entry in a module-specific task queue. Scheduling was based on priorities. Modules handling input from devices had higher priority than modules handling output to devices. The TR 440 was not allowed to send data blocks to the SAP

at any time, but only on request by the SAP. Furthermore, a SAP could tell the TR 440 or another TR 86S on the path to the TR 440 that it could no longer accept data destined for a specific TR 86S further down in the tree. This approach prevented output bottlenecks in the TR 86S. By reducing the output, the progress of terminal sessions was slowed down, which also diminished the input load from devices. Furthermore, it was recommended that a data line going from a TR 86S toward the TR 440 had to have a higher transmission rate than the sum of the transmission rates of all lines coming in from devices or other TR 86Ss, so a back up of input streams could be avoided. Alternatively, handshaking protocols could have been used to control the data flow. This approach was not implemented because it led to additional interrupts.

The importance of decentralized computing and networking was clear early on to the people responsible for development. Our customers also expressed such requirements. In the early 1970s, the TR 440 software development department began exploring and planning all facets of connections between computers, a few years after the ARPA network was begun. For example, fundamental concepts and concrete development steps were documented in a memo at the beginning of 1973.[38] The ideas expressed therein are still valid today—for example, that a longer useful life of a TR 440 could be achieved by integration into a network with load sharing or with computers of different functionality or power. This let the advantages of time-sharing systems be combined with the advantages of number crunchers. At the end of 1973, Evers and I visited the US to get information about current and upcoming features of computer networks.

A 1972 company flyer explained it this way: ''Further evidence for the open user-friendly nature of the TR 440 system is its adaptability to computers of other manufacturers. Dialog between computers is more and more becoming reality.''[39]

Programming system

The programming system (PS) was a kind of layer between the BS and the OPs.[40,41] OPs were user programs, application programs, or standard programs (such as compilers), but the PS was not a single program. It consisted of several units stored in a program library. Some units were programs ready to be loaded and run (OPs); others were program components to be linked with a user program. Some examples of the PS units include linker, loader, I/O subroutines for different programming languages, testing aids, back-tracing programs, dump programs, and a shell (TR 440 command language interpreter).

Frielinghaus led the group responsible for the PS and compilers. In 1969, Alexander Hoyer became the head of this group, until he suffered a severe car accident in 1972, upon which Eberhard Schmolz became the head on 1 August 1972. Schmolz had been with the Telefunken research institute in Ulm and came to AEG-Telefunken in Constance in the late summer of 1968 to work on the Cobol compiler. On 1 April 1973, the group was split into two: one group responsible for the PS, and the other for compilers. Norbert Linn became head of the programming system group, and Schmolz of the compiler group.

The TR 440 command language (shell), designed by Peter Namneck, was used with batch and interactive jobs. A job consisted of data blocks, together with commands that specified what the computer should do (with the data). The shell extracted the commands and executed them. Commands started with a special escape character ◇ to distinguish them from data. The syntax and semantics of commands were the same for batch and conversational jobs, which was unusual at that time. Other features also pointed far into the future and are still fundamental and important for shells today:

- For all applications, the same syntactical structure was used—for example, ◇ `COMPILE, LANGUAGE=...` for all compilations.
- Commands to start or close batch and conversational jobs obeyed the same rules as other commands.
- The parameters (specifications) for a command were named. A parameter consisted of a parameter name and a value, written as `<parameter name>=<parameter value>`. Command and parameter names could be abbreviated, provided the abbreviation was unambiguous. Abbreviations were indicated with a ''.'' following. The parameters were also associated with a position, so the names could be dropped when a predefined sequence of parameters was used. Some parameters were required, and others were optional. For instance, the compile command had to specify the source code and the

language—for example, with abbreviations you could write: ◇ `COM.,S.=mysourcefile,LANG.=ALG60 ...`.

- Parameters could be regarded as local shell variables, to which values could be assigned. For example, if users wanted to compile several Algol programs, they could declare ◇`*LANGUAGE(COMPILE)=ALG60` and then would not have to specify the language in subsequent compile commands.
- The shell had a separate shell memory for each user in which the variables had predefined initial values that the user could modify. The shell memory was always valid during job execution, and its lifetime could be extended by exporting it at the end of the job and importing it at the beginning of a later one.
- It was possible to define (new) commands with (new) parameters for (new) programs.
- Command sequences could be dynamically changed by using a conditional branch command to jump to another command in the sequence, depending on the state of programs and other shell variables. This allowed, for example, different command sequences depending on whether a program reported an error or not.

Peter Jäger made the following remarks about the TR 440 command language on his Internet page:

> The software products still comprised hundreds or thousands of punched cards or magnetic tapes the size of cake plates. But already the computation of a huge quantity of mathematical and physical values for printed pocket books and data collections astonished the experts. Electronic data processing technology had evolved at a speed which can only be described as breathtaking even from today's viewpoint. Disk stacks the size of a large deep-freeze cabinet already showed what lay ahead. In my view at that time, the Telefunken TR 440 computer was among the most progressive technologies of all. The command language was impressively simple, but nevertheless absolutely reliable. By contrast, the Job Control Language (JCL) which then had to be learned for the IBM was a long way from being user friendly.[42]

In 1986, the German computer magazine *Computerwoche* also reported on the TR 440 installed at Osnabrück University:

> Our [Osnabrück University] computer center has been operating a computer of the type TR 440 since September 1, 1977. According to current plans it will be in use until the middle of 1988. ... The owner of a TR 440 in the year 1986 finds himself in the happy position of being able to benefit from the numerous software packages developed by the manufacturer as well as by various former TR 440 users, even though all around us these computers have been taken out of service, and the TR 440 has become almost quaint. The heyday of the TR 440 was characterized by very good professional cooperation between the computer manufacturer and users of the computer, as well as by an intensive exchange of software between TR 440 computer centers, some of which ported powerful software from the US to the TR 440, and passed it on to other centers. ... An essential reason that we are not at all despondent about still operating the TR 440 today and even for two more years is the excellent user interface. BS3 and the comfort of its command language were far ahead of their time, in their capabilities as well as their underlying concepts. What some suppliers of today praise as a modern and trend setting user interface is for the TR 440 user ''old hat.'' Regarding the aspect ''comfort of the user interface''—so important for a university computer center—the TR 440 is substantially more advanced than nearly all large scale computers which replaced it. A discarded TR 440 was recently delivered to Nikolaus-Kopernikus-University at Thorn in Poland, and a second one will follow; thus not only generations of German students and scientists have been educated using the TR 440, but soon generations of Polish students and scientists will be as well.[43]

Compiler

At the time of the TR 440 development, software was always bundled, meaning that it was included in the price of the central computer hardware—software was ''free of charge!'' Even after IBM unbundled its software in 1969,[44] TR 440 software remained bundled. Therefore, customers demanded a broad range of software products from their manufacturer, and sales people had a tendency to promote sales by offering additional software for free. At that time, customers often implemented their application programs by themselves; this was especially true for customers in the university environment. These were the basic reasons why AEG-Telefunken had to provide compilers for a broad range of programming languages: a functionally rich assembler (Telefunken Assembler, TAS), Algol60, Basic Combined Programming Language (BCPL), Basic, Cobol, Fortran, PL/1, RPGII, and

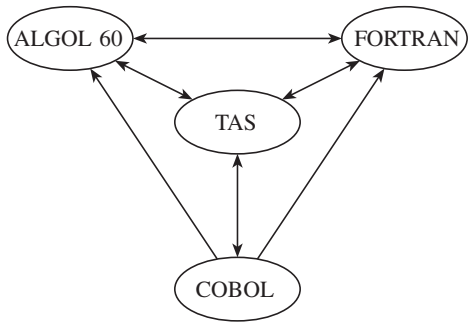


Figure 6. Linking of procedures written in various programming languages.

General-Purpose Simulation Language V (GPSS V), and RPG, an IBM language for business applications, initially designed for generating reports from data files. (Common compiler techniques are outside this article's scope, as are the scope of a language as implemented for the TR 440.) The features of the programming languages met standards and were often also geared to those of other important manufacturers. Our company was active in many national and international standards committees. All compilers mentioned were designed and implemented by the compiler group within large-scale software development (GR/P) except PL/1 and BCPL, both of which I cover later.

Two important features of our compilers were linking of procedures of different languages and testing aids. Both features were groundbreaking then and are still unusual today to the extent that they were implemented for the TR 440.

Linking of different language procedures was done by the linker/loader.[40] A procedure could be linked to a main program possibly written in a different language or to another procedure, maybe also in a different language, and recursively. Linking procedures written in assembly language was straightforward. A programmer of such a procedure had to obey the rules given by the programming language of the calling main program or procedure. A programmer of a main program in assembly language calling a procedure written in another language had to obey the calling conventions and use the data types of that other language. If the calling program or procedure was written in a high-level language and was calling a procedure written in another high-level language, then well-known problems arose: data types had to be compatible, or at least convertible, and calling conventions (including the stack layout) had to be adjusted. Stubs, invisible to the user, were provided by the programming system and inserted by the compilers to convert data and calling conventions between different languages. In some cases, there were even special extensions made to a language to make it possible to use procedures of certain other languages. For example, Algol60 was extended with a `COMMON-area` as regularly needed in Fortran subroutines. Figure 6 shows the possibilities for linking procedures of different languages.

All testing aids were user oriented and always referred to the source language of a main program or procedure, even when different programming languages were used. Here are some of the important testing aids:[40,45]

- At compile time the compiler tried to continue translating at the closest possible point after an error was found. The goal was to find all syntactic and semantic errors (at least of those that could be recognized at compile time) with one compiler run. A user could also request various lists, such as a list of declared variables with all line numbers of the source code where each was referenced.
- There were compiler options to include dynamic controls, such as testing whether indices stayed within their bounds, whether formal and actual parameters were compatible when calling a subroutine, whether loop variables were changed in a way not allowed, and so forth.
- When an error occurred while running a program, a source-code-related error message was given, indicating possible reasons for the error. Furthermore, a source-code-related dump was available, including a list of all valid variables with their names and current values, and the current nesting of subroutines giving the exact location (even within a line of code) of the subroutine call in the source and the name of the called subroutine.
- The source-code-related dump could be output not only when an error had occurred, but also by using a special procedure call with parameters specifying what data to include in the dump. After dumping, the program was continued.
- The execution of any program could be traced from instruction to instruction at the source code level.

- In a source program, a user could define *trace points*, at which a source-code-related dump was initiated. To keep the dumps small and easy to read, it was possible to specify details of what to record, such as which variables to include. The dumps could be stored in the computer, letting the user decide on an error or at the end of the program which of the dumps were of interest (backtrace).
- With interactive programs, *Kontrollereignisse* (KE, control events) played an important role. In the compile command, users could specify a generation of a conversational program and could define KEs, associating each with a source line. In the command used to start a program run, they could specify which KEs were initially active and which were passive. If an active KE was reached, the program was halted and a message with the name of the KE was sent to the terminal. Users could then work entirely at the source code level; they did not need to know any machine-oriented feature. They could query or set the variable values, make KEs active or passive, turn on or off testing aids compiled into their program, and continue or stop their program.
- In conversational mode, users could interrupt their program at any time and then could begin a dialog with the ABW or the user program. For example, they could post new commands having priority or could use any of the testing aids discussed earlier. At the end of the dialog, they could cancel or continue their program.

The compilers designed and implemented in the compiler group used a common structural concept with a common intermediate language (before code generation), influenced by Donald Knuth and by work on compiler compilers as well as by compiler writing systems. In practice, only a common concept with similar intermediate languages remained for the TR 440 compilers because they were not developed at the same time and because code optimization was different for each language, since it was extremely important at that time to attain good benchmark results.

A universal intermediate language (intermediate code) for all compilers, a two-stage optimization concept (language and machine related) on the level of the intermediate code, and common code generation were completely specified for TR 440's successor.

BCPL for writing systems code

Already in 1969, we in the software department believed that it was important with respect to productivity, reliability, and quality to no longer write compilers in assembly language but in a higher-level language, and that memory requirements and runtime of the compilers would be acceptable. Members of the compiler group worked on this subject intensively, examining current languages and even considering designing and implementing their own language for this purpose. Finally, however, we decided to adopt Martin Richards' BCPL compiler, a decision not unanimously welcomed. In 1970, the BCPL compiler was adapted to the TR 440 and enhanced.[46] We were able to get Richards as a consultant for this work. Hoyer had a significant influence on this farsighted and correct decision. At first, BCPL was used only internally for writing compilers and later for parts of the operating system, but in June 1973 (with the BS3 maintenance version MV13N), it was also delivered officially to customers.

The Combined Programming Language (CPL, 1963), BCPL (1967), B (1969), and C (1971) belong to the same family of languages and evolved from each other over time.[47] CPL, a language influenced by Algol, was developed by English universities in Cambridge and London. Richards designed a simplified version called BCPL and implemented a compiler for it during his research stay at MIT. The BCPL compiler was itself written in BCPL, a feature which was important for our selection decision. The compiler generated an intermediate code called OCODE, which could either be interpreted at runtime or translated into machine instructions by a code generator. To port the compiler to the TR 440, we wrote a code generator that was also used to generate executable compiled programs.

For a long time, there were many heated discussions with customers who did not believe that system software could be written using higher-level languages. They feared that too much performance would be lost and that memory requirements would be too high. At that time, all our main competitors' relevant system software was not written in a higher-level language, so we could not present real-world examples to our customers.

Despite this, the large-scale software department forcefully pushed for the use of BCPL as a systems programming language, even for the operating systems, because it expected substantial advantages. These expectations proved true in the course of development. Today, using higher-level languages for system software goes without saying.

A new approach for the PL/1 compiler

With the PL/1 compiler, the software development department made or continued making important strides into the future. As I explained earlier, the PL/1 compiler was not implemented by the compiler group; it was bought in May 1973 and ported to the TR 440. Testing aids and other TR 440-specific functions were added.

The source code of the PL/1 compiler and all rights to it were acquired from MIT for about 600,000 DM (according to current exchange rates about $400,000). The compiler was developed within the Multics project, started as a joint venture between MIT, General Electric, and Bell Labs. At the beginning of 1973, the compiler was finished and running on a GE-645. It was written in PL/1 and, therefore, could be ported to the TR 440 by bootstrapping. In May 1973, after careful preparation, a three-week evaluation of the PL/1 compiler and the bootstrap procedure was conducted at MIT by Eberhard Schmolz and Hanno Krainer.

For the bootstrap process, the PL/1 compiler's source code, itself consisting of many large PL/1 programs, was translated by a working PL/1 compiler into intermediate code and written on magnetic tapes. This was done using IBM 370 computers at the German IBM research center in Böblingen, near Stuttgart. The magnetic tapes were taken to the TR 440 compiler group in Constance and, using the TR 440, translated in one pass into a program that could run on the TR 440. After some transformations, the original PL/1 compiler could be compiled on a TR 440, completing the bootstrapping process. The PL/1 compiler was then ready for use and was the basis for further developments.

Schmolz remembered:

> The results of a night's work filled two or three magnetic tapes. We used the following ingenious stratagem at least ten times: my father lived in Vaihingen, a suburb of Stuttgart. Equipped with written authority, he drove the short distance to the IBM computer center at 6 AM, picked up the magnetic tapes, drove to the Stuttgart main railway station, and gave the tapes to the crew of the express train from Stuttgart to Constance. In Constance we picked up the tapes from the crew at about 9 AM, and could then immediately work with the IBM results of the night before.

By concentrating all software development in one place and in one department, synergies could be achieved and software quality could be improved.

This PL/1 compiler acquisition was remarkable in several respects. First, the software developers were aware of the international scene and did not work cut off by themselves in Constance. Second, the company was cost conscious, since buying the PL/1 compiler was less expensive than implementing it by ourselves and made it available sooner. Third, this was possible only because the compiler group already had experience with bootstrapping and compilers written in higher-level languages from their work adapting the BCPL compiler. Fourth, it was a courageous step because there was not yet another example of this in Germany. The technical risks were high, as became apparent during the bootstrap process. Although PL/1 development at that time was important for IBM, and accordingly their PL/1 compilers were very good, the resources needed for the bootstrap were so high that compiler runs in Böblingen were possible only at night, and several large mainframes had to be coupled together to cope with the load.

Further evidence of the PL/1 project's success was that Siemens ported the TR 440 PL/1 compiler to their 7700 computer family in 1975.

Application software

As I mentioned earlier, TR 440 software remained bundled. This was true for the

For a long time, software development was unable to get sufficient computer time for testing.

application software as well. The portfolio of application software was primarily defined by the customers of university computer centers. Their requirements extended from administrative and planning tasks to judicial information systems. To a lesser extent, application software was influenced by governmental offices.

Part of the applications software was developed for the TR 4, and later for the TR 440, under the supervision of Schlenstedt. TR 440 software for commercial applications was developed by a group in Bonn, headed by Neumann, reporting to the sales department. In June 1974, this group was transferred to GR/P and moved to Constance. The group for application software became GR/EP3 and was headed by Knoth. By concentrating all software development in one place and in one department, synergies could be achieved and software quality could be improved.

To convey a feeling for the breadth of application software, I present a short summary.[48,49]

A comprehensive library of mathematical procedures was supplied, including elementary and special functions, programs for integration and approximation, and linear algebra programs. For plotters and interactive graphics displays, customers got a rich set of subroutines. Programs for sorting and mathematical statistics rounded out this segment.

A modern and convenient database system was implemented to serve as the basis for important nonnumeric applications. It provided data-management and data-access services for large data collections, but not a query language. Data were permanently stored on the computer system. The data structures were specified as in Cobol. Access was sequential, indexed-sequential, and random with record numbers. It was possible to associate multiple indices with an indexed-sequential record. Records that had a logical connection could be linked by a chain of pointers. A compression algorithm for sequences of zeros or blanks was included.

The Telefunken documentation system (TELDOK) used the database system and provided rapid and precise retrieval of stored information (documents) as answers to particular questions. To support this, documents had to be associated with descriptors that could be chosen freely or restricted to words of a thesaurus or dictionary. Every document and every descriptor was assigned a unique internal number by the system. TELDOK was based on a thesaurus, index of documents (ordered by the name of the document), inverted indices (links to documents ordered by their descriptor), abstracts of the documents, and the documents themselves. For retrieval, the search expression consisted of logical combinations of weighted descriptors. Search could also be done within documents already found.

Both the database system and TELDOK, as well as similar products of Siemens (Golem, Sesam, Prisma) and Software AG (Adabas), were partially funded by the German Ministry of Research and Technology.[50]

A management information and enterprise resource-planning system was developed to support management in achieving company goals. In contrast to pure retrieval systems, the contents of the database changed rapidly, because of continually arriving new information.

An application system for production planning and control, PSS, was also based on the database system. As with other management information systems, the contents of the database changed continuously. The system consisted of PSS-STP *Stücklistenprozessor* (handling material lists), PSS-APL *Arbeitsplanprozessor* (planning the logical sequence of work), PSS-BEDA *Bedarfsermittlung* (determining required resources), PSS-MAWI *Materialwirtschaft* (materials management), and PSS-KAP *Kapazitätsplanung* (capacity planning).

BKN was a Fortran program to schedule tasks within a network and calculate critical paths, deadlines, resource allocations, and costs, among other things. It could handle large networks of up to about 10,000 nodes. It was used, for example, to plan TR 440 (hardware) production. It failed to be of value for scheduling software production, mainly because preparing the ever-changing input for this method was extremely time consuming and the real dependencies of

software units always deviated from those of the input since dependencies were not as strict. Therefore, the real progress of software development never matched what was calculated.

The Programming Language for Interactive Teaching (Planit) was a general-purpose system for computer-supported education, including all necessary tools, from the authoring language to the tutor program to the statistical evaluation of the student's answers. Planit was based on a development of the US System Development Corporation (SDC). It had already been ported to IBM and Siemens computers. A first version for the TR 440 was available in 1973.

And finally, the Exapt program was an application for digitally programmed machine tools (drills, lathes, and milling machines). The program handled not only geometric data, but also technical attributes of materials and tools. It was based on the Exapt language, designed in the 1960s by German universities and widely used in Germany for this kind of application. The name Exapt was derived from "extended subset of APT." APT (automatically programmed tools) was designed in the late 1950s by MIT.

Internal computer centers

For a long time, software development was unable to get sufficient computer time for testing. It was not available early, in the necessary quantity, nor on sufficiently stable computer systems. The situation improved after more computers were installed for testing software and after the TR 440 computer centers were transferred to the software development department.

The second TR 440 was installed on the testing floor in December 1968. It was used both for testing additional hardware developments and for software tests. This was an unacceptable situation for software development and led to severe slips of milestone dates.

The installation of TR 440 number four (June 1970), number eight (June 1971), and number 12 (November 1971) led to a significant improvement. These computers were installed at and operated by the AEG-Telefunken computer center in Constance, which operated already three large TR 4s. The software people were still unhappy, however, because frequent hardware changes led to instabilities, and software development was not a priority, just one customer among others. Of course, the permanent conflicts between the computer center and software people had a serious negative effect on software progress.

The TR 440 part of the AEG-Telefunken computer center was transferred as the internal "computer center for development" into the software department (still GR/P) of the newly founded company Telefunken Computer on 1 May 1972, as group GR/P4. Under the leadership of Dietrich Wagner (who joined AEG in October 1967 as a programmer with the AEG computer center in Constance, and later was head of the computer operations group there), comprehensive and professional operation of the computer center was quickly established. Jürgen Keppler was appointed the head of the computer operations groups. Figure 4 shows the organizational chart.

Through the organizational addition of the computer center to software development, priorities could be adjusted flexibly and properly, and all conflicts could be resolved directly at a low level. As a result of this and finally having adequate hardware resources, software development was no longer hampered by a lack of computer time.

The development computer center consisted of five subcenters (RZ1 to RZ5). In each RZ1 to RZ3, a large TR 440 was installed with numerous peripheral devices in accordance with the products offered by the company. RZ4 had five TR 86S satellite systems, likewise with numerous peripheral devices, terminals, and data transmission lines. The sales department computer center operated TR 440 number 33 until it was also integrated into the development computer center as RZ5 on 1 February 1974.

In October 1973, TR 440 number four was replaced by TR 440 number 27, resulting in one TR 440 with a single CPU and two TR 440s with two CPUs each in the computer center. Some TR 440s were either directly connected and had common hard disks or were connected over a TR 86S.

RZ1 to RZ4 were dedicated to software development. During the day, the developers could get direct access to the computer (block time). In the evening and at night, there was a closed-shop operation by operators in several shifts. Accordingly, the computer center employed approximately 30 operators (as of 30 July 1974) and about as many other coworkers. Initially, about 65 operators came from AEG.

There was a special group responsible for preparing and scheduling the jobs for the closed-shop operation. Wagner remembered,

''There were giant decks of punched cards; every day more than a ton of paper was printed on; we had a giant stock of material with storage racks for pallets; the stocks were in an air-conditioned room to prevent electrostatic charge, causing the paper in the fast line printers to stick together.''

The newest stable version of the system, also delivered to the customers, was always run in RZ5. Computer time was provided for customers and all company departments, in particular for sales, demonstrations, trade fairs, acceptance tests, benchmarking, and tests under high, user-oriented load. Software development could get computer time only when there was nothing else to be done.

The software development department's internal development computer center was the largest TR 440 computer center at that time and, in every respect, was progressive, as can also be seen by two more examples:

- A computer connection to Constance was established during the CeBIT trade fair in Hannover (probably in 1974), about 400 miles away, demonstrating remote data processing.
- A large, secure data archive was realized. A central part was a huge *paternoster archive cabinet*—a cabinet containing a chain of compartments moving in a continuous loop, like rosary beads—holding magnetic tapes. The number of a magnetic tape was input on a keyboard, whereupon the tape arrived, ready to be taken. The cabinet was developed in cooperation with a company, which later on sold such cabinets to other computer centers as well.

On 1 February 1975, Wagner left the Computer Gesellschaft Konstanz, and Keppler succeeded him as head of the group.

Project management and organization

With increasingly powerful hardware appearing, software demands also escalated. Software became voluminous, and its implementation required more and more resources, notably large development teams. Many software projects failed; others could be finished only with substantial delays of many months and with huge cost increases. A well-known example for that was IBM OS/360—a batch-oriented multiprocessing operating system—in the second half of the 1960s.[51] At the end of the 1960s, the problems connected with developing large software became more pressing. A discussion of that, and a survey, took place at the famous 1968 Garmisch conference on software engineering, where Köhler and Wiehle represented AEG-Telefunken.

It was no surprise that TR 440 software development also had the typical problems; it was a large, groundbreaking project. I have already discussed some of the technical problems; now I concentrate only on issues of project management.

At the end of the 1960s, it was not known, and not proven, how to manage large software projects. Looking at the international discussions about an appropriate project management model at the time, one could discern two camps.

The first camp believed any project management to be deadly for a project. Instead, a few highly qualified people should be charged with the product development, not be bothered by managers, work without external rules, have absolutely unconstrained working hours, and never be asked about their project status. After some months or years, the team would show up and present a finished and excellent product. This belief was supported by the way many successful projects were completed at that time.

The second camp believed that managing software was as appropriate as managing projects of other engineering disciplines. But this had not been proven in real-world projects; if anything, the long list of large failed software projects was evidence to the contrary. In addition, the specific features successful software project management should have, compared with management in other disciplines, was still a research issue.

Because of the situation the TR 440 software development faced, it became clear to me that a project management process had to be established in order to be successful. As soon as the enormous pressure on the programmers decreased a little after the first deliveries of the time-sharing system, this goal was pursued. Kurt Scheidhauer backed this plan wholeheartedly, but my immediate boss, Köhler, questioned whether there was enough manpower to be able to afford it. After all, there was still considerable deadline pressure on the software team, and introducing management procedures would require a significant amount of manpower. Obviously, the project management process could not be developed without external help and expert knowledge, which at that time was not available in Germany. In the US, however, software companies were forced to use

comprehensive management methods, procedures, and checks if they wanted to sell their products to the military. These were so costly that they could be justified only for military applications. A direct transfer to the commercial market was therefore impossible, but some companies had adapted their software management procedures to the commercial market—for example, Computer Sciences Corporation (CSC). Using that knowledge made sense for us.

In the autumn of 1969, Scheidhauer established contact between CSC, headquartered in El Segundo, California. After technical discussions and conversations with potential consultants in El Segundo, three chief consultants (Christopher Earnest, Robert E. Trainer, and Robert W. Walsh Jr.) were delegated by CSC to our software development department for several months in 1970 to develop the management concepts as a full-time job together with us. They brought their families to Germany. Some other CSC employees (among them Joel Erdwin and Sheldon Sidrane) were in Constance for shorter periods of time, to help with special issues.

In the course of developing and finalizing the management rules, regular meetings were held with the CSC consultants, all heads of the software groups and me. In those meetings not only the essentials were discussed and finalized, but also all the details down to the wording level. In that way, we achieved a project management process custom-tailored for our software department,[52] and all software personnel with managerial functions were involved in the definition process. Therefore, the knowledge about the new project management model was spread broadly, and it was accepted without major reservations or resistance. The project management process documentation was finished in July 1970. Immediately afterward, the new guidelines were introduced and implemented step by step, including the necessary organizational changes. The other German and probably even European computer manufacturers had nothing comparable at that time and even years later.

Essential for the success of software development was not only the technical quality, but also the top management's and the customers' confidence in the product and in the people associated with it. The new project management process played an important role in that, particularly regarding documentation, quality assurance, change control, error management, and progress reports.

Documenting was an integral part of development, as important as designing or programming. In addition to the technically oriented material, all other project-related documents had to be included as well, including project approvals and quality or progress reports. The previous behavior, to take just a few notes with some catchwords about the design on a slip of paper, or to keep the design only in one's head, was now no longer acceptable. When we started enforcing the new rules I was often asked by staff members, ''Shall I document, or deliver on time?'' Unfortunately my answer in most cases had to be, ''Deliver on time, and document later.'' All project documents were structured and numbered in a way that filing of documents could be the same for all. At that time, we had to distribute paper copies of the documents. Had we been able to store the documents as shared files on a computer, then handling would have been much simpler and more efficient.

For quality assurance, I formed a new group under the leadership of Kurt Mühlbach, organizationally on the same level as the software development groups (see Figure 4). Documents created within every phase of development were a basis for the work of other teams because they contained interface descriptions. It was therefore important to describe interfaces clearly and to protect them from uncontrolled changes. The same was of course true for software modules, which were also the basis for other work—for example, when testing partially integrated components. The basic idea was for the quality assurance group to do acceptance testing for documents and software units, with features previously frozen. After official acceptance, the software developers could make changes only in response to error reports or approved change requests.

Hence, changing ''accepted'' documents or software units was possible only when a corresponding change request was approved. The change control board met once a week to decide on change requests. Permanent members were the head of the GR/P, the software group leaders, sales department representatives, and the head of the systems support group (Klaus Auerbach). All change requests together with the final decisions were documented. Thus, everyone in the company knew exactly which requests had been received and if, how, and when they were to be implemented. Of utmost importance was that everyone, including all the

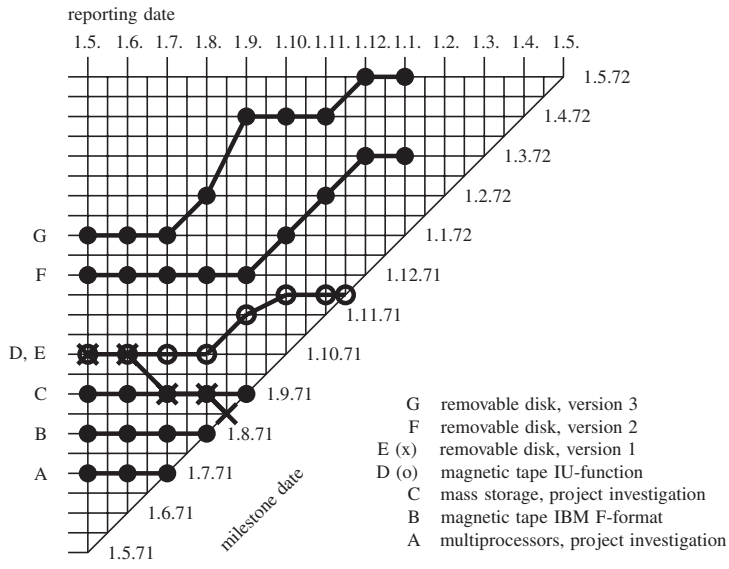


Figure 7. A milestone chart from the TR 440 development.

customers, could submit a change request. Thus, we had implemented a fast and transparent procedure for the software developers, sales people, and customers to articulate wishes and receive decisions.

All problems and errors reported were registered. The time to their resolution was monitored. Errors or problems could induce further change requests. There were error statistics, and errors were associated with software components. In this way, we got an early warning about possible problems with a customer or with a software component that might be badly designed or implemented.

Progress reports replaced development reports. The main differences were that the reports were now on all levels and that content was specified, more formalized, and extended. For example, milestones for each project were now always defined and reported on. Important was a milestone chart, showing the changes of milestone dates at successive reporting dates. See Figure 7 for a real example from the TR 440 development, though perhaps not typical. Because the features to be realized when reaching a milestone could not be changed without quality control, this trend analysis had extreme educative power, indicating problems early, and enabling management to take countermeasures. (A horizontal line might have indicated that the project was running smoothly, but it might also have been a warning sign that nobody cared about the project.)

Figure 4 shows the fully implemented organization, corresponding with the project management features developed with CSC. There was also a small restricted matrix structure with two project managers, one for the TR 440 (Feldmann) and one for its successor, the TR 550 (Frielinghaus). At that time, the GR/P had approximately 300 employees,[53] 62 of them working in the computer center, among them 30 operators. The staff included about 217 programmers, seven group leaders, two project managers, and 13 secretaries. Of the programmers, 43 had doctorates, 92 had a university diploma, 33 were engineers who had graduated from a technical college, 17 had not finished their scientific studies, and 32 had some other education. More than 90 percent of the employees with a university degree had studied mathematics, physics, or another subject of the natural sciences, or engineering. The target staffing level was 313 people.

The financial problems of AEG-Telefunken's large-scale computer division[2] became permanent after 1970 or 1971, so there was a continual reduction of the number of employees until the company was taken over by Siemens AG. New targets for the number of employees came nearly every month. In addition, as more parts of the TR 440 software were completed, people were transferred from that project to work on the TR 550. Because of our technology-oriented rather than project-oriented organization, transfer to work on TR 550 did not mean quitting the old software group; for example, the compiler group developed all compilers for all target machines.

DFG-AK

At the time, the DFG partially funded the computers installed at German universities and research agencies. When AEG-Telefunken signed the contract of purchase for a TR 440 for the German Computer Center (DRZ) in the spring of 1967, the DFG created the DFG-AK, a committee to do user-oriented TR 440 acceptance testing. The leader of the DFG-AK was Dieter Haupt, professor of computer science at the RWTH Aachen University, head of its computer center and chairman of the *DFG-Rechnerkommission*, a DFG committee for evaluating proposals to buy computer systems, technically and financially. A positive assessment was prerequisite for funding.

The DFG-AK members were seven computer experts from the DRZ and the RWTH Aachen. The group's initial mission was to monitor the TR 440's development and to

prepare the DRZ installation of hardware and software. Soon however, the group began to directly influence the concepts and implementation of the TR 440 operating system.

The DFG-AK played an important role in the TR 440 large-scale computer development, notably in software development. In an internal paper, Alexander Giedke described the TR 440 development from the DFG-AK viewpoint.[26]

> In June 1967 the DFG-AK together with another committee, the Apparateausschuss, looked at the concepts Telefunken had developed, and Telefunken's newly coined terms for describing attributes and features of the TR 440 caused difficulties right from the start. For example the term ''process,'' which was said to be of fundamental significance, remained obscure. Because these new terms were not defined in the available documents, having to become acquainted with the new system was for everyone a tough nut to swallow.[26]

The DFG-AK feared too much system overhead and saw other weaknesses in the concepts:

> Telefunken estimated an overhead of about 1%, but the first version of the BS [BS1] already required about 35 K to 50 K of the 128 Kwords of main memory. The necessary displacement out of main memory of the contents of already-completed programs also posed problems.... It was not possible to initiate such displacement by operator services; it had to be done ''automatically.'' But that could hardly be realized in a way that appropriately accounted for the many possible states of operation. Therefore it was no wonder that the way to achieve reasonable operation was absolutely unclear.... Together with Telefunken we looked for other operating concepts, where among other things the benefit function was to be replaced by a simple control using program priorities.... The DFG-AK believed that the BS needed so much main memory that multiprocessing operation of the computer had to be questioned, even with a reduced BS.
>
> At the end of 1967 and the beginning of 1968 Telefunken could not provide machine time for test runs as contracted. Binding documentation was still missing. Only internal, partially confidential, handwritten notes were available to the DFG-AK as the basis for their work.... At the beginning of 1968 it was obvious that the system would not be ready to be delivered on July 1, 1968, as contracted.... A prerelease version BS07 of BS1 was planned for delivery at the end of 1968, but was not yet ready in mid 1968, and as in the past, the DFG-AK still had a negative opinion of it ... Not before July 10, 1968 was time available for the first program tests on the only TR 440 computer, which was also to be delivered to the DRZ in Darmstadt. On December 9, 1968 acceptance testing for the double TR 4 system started.... The state of BS1 development was still of concern. Tests conducted in the autumn of 1969 did not show the expected progress at all. Development was not yet completed, and completion could not be expected in the next few months.
>
> It is no wonder that abandoning BS1 was a surprise for many people involved , since no indication of this turn of events was evident at a user congress held on August 31, 1969.... On November 7, 1969 there was a hearing (DFG, German Ministry of Education and Research, customers/users, other interested persons) in Constance about the TR 440. Here Telefunken made it known that BS1 was no longer being pursued. A first version of BS3, batch processing with two ABWs, was demonstrated.[26]

During this hearing the funding agencies arrived at the conclusion that BS3 was a reliable basis for further development and subsequent installations.

> With BS3 there existed a definitive and feasible software concept. In essence the hearing led to the decision to continue the TR 440 project. Milestones for software development were defined, whose dates and contents were afterwards largely met.[26]

These quotes show that the importance of the DFG-AK was mainly based on four facts: The group spent a lot of time in Constance and had direct contact with the software teams. It was a critical partner with inside knowledge, querying and reviewing the development, and making suggestions for the realization. Therefore, the software developers had to explain and defend their ideas and consider possible weaknesses during design. This required patience and additional effort from everyone involved, but was helpful overall. As a result, this review process as well as the group's selecting and conducting acceptance tests provided a kind of external quality assurance.

- The group represented the views of the customers and funding agencies at that time, and articulated their wishes.

The DFG-AK was a critical partner with inside knowledge, querying and reviewing the development, and making suggestions for the realization.

- The group played an important role in creating and maintaining the confidence of the customers in Telefunken developments.

Giedke summarizes the importance of the DFG-AK as follows:

> For Telefunken, the increase in knowledge presumably was considerable and essential. But the way was painful, especially the path of development of the BS, which was going through a tedious ripening from a design without practical relevance to real world oriented solutions. Without doubt the exchange of experience between the company and the DFG-AK was instrumental in the success of the project, because the DFG-AK acted not only as a surveying committee, but also as a critical consultant.[26]

Besides the DFG-AK, another working group was established in 1970 at Bochum University, supervised by Hartmut Ehlich, a professor and head of the computer center. This group was to prepare for the installation of a TR 440 at Bochum University and was in close contact with the DFG-AK and Telefunken's software people. The group's work was instrumental in the computer being installed rapidly and operating satisfactorily.[54]

Last but not least, the external TR 440 customers' association (STARG) also influenced software development with the following tasks:

- coordinating the TR 440 computer centers and the TR 440 users,
- collecting wishes and proposals concerning the TR 440,
- evaluating proposals for enhancement of TR 440 software from a user's point of view, and
- acting like a fire brigade when computer centers needed help.

In periodic meetings, numerous and substantial program developments of the STARG members themselves were also presented.

The end of large-scale computer development

To be successful in the large-scale computer business, it was absolutely necessary to show the customers paths to the future. Therefore, as early as 1969, AEG-Telefunken set up a project group to deal with a successor computer system to the TR 440, with the working title TR 550 (among others). The organizational structure was a small matrix organization. Technological responsibilities were with the development groups; for example, all operating systems were to be developed by the BS group. Orthogonal to that was the project-oriented coordination by TR 440 and TR 550 project managers in the hardware and software departments, complemented by temporary ad hoc working groups for specific issues.

Different approaches to developing the TR 550 system were surveyed and often pursued further. At the same time, there were many discussions about close cooperation with other computer manufacturers, within and outside of Europe. (See other work in this issue for details.[2] Also see the ''Further Reading'' sidebar for other works n this topic.) The background of these efforts was twofold. The AEG-Telefunken management knew that because of restricted financial resources it was nearly impossible to develop large-scale computers alone in the long run, and the German government therefore pushed for a cooperation or merger with a European computer manufacturer, first of all Siemens AG.

On 18 July 1974, the large-scale computer development in Constance was taken over by Siemens AG, forming a separate company called *Computer Gesellschaft Konstanz* (CGK). Before this takeover, manpower was reduced to a number stipulated by Siemens. Because Siemens had just formed Unidata together with CII (France), ICL (Great Britain), and Philips (The Netherlands), and as CII was responsible for large computers, there was no room for a large-scale computer manufactured by CGK.

Consequently, Siemens immediately stopped all work on a successor to the TR 440. Software development was reduced to absolutely necessary maintenance so that

TR 440 systems could be operated by the computer centers as long as possible (until a Unidata system was available for replacement) and so that already produced TR 440s could be sold. Because CGK was founded only because of heavy pressure by the German government, neither Siemens nor Unidata had any interest in it—there was even strong rejection.[55] The Siemens management was not prepared and did not know how to make best use of the competent and highly qualified people in Constance.

There were discussions on all levels, down to the group level, about the software developers' future work. The software development groups with Siemens Munich saw no way to give larger software projects to CGK, perhaps fearing competition. There were several consequences of this:

- A small software group remained with CGK, maintaining TR 440 software and doing contract work for Siemens.
- Many people in software development left CGK, including myself at the beginning of 1975, after which I became a professor of computer science at the University of Technology Munich. Frielinghaus then took over software development until he also got a new management job with Siemens Munich, whereupon Klaus Auerbach became the head of CGK software development.
- Approximately 90 software developers were delegated as individuals or as small teams to software groups of Siemens Munich, therefore commuting on weekends between Constance and Munich (a four- or five-hour journey at that time). Among those were Feldmann and Schmolz, both of whom later attained higher management positions with Siemens.
- Together with the large-scale computer division, AEG-Telefunken had also sold their optical document reader division to Siemens. The large high-speed document reader machines, previously developed by AEG-Telefunken, became CGK's central product, and CGK turned into a profitable company.

For Germany, the transfer of knowledge that had been gained with the TR 4 and TR 440 development was important for the rapid buildup of computer science and computer know-how. This transfer included numerous students using the TR 440 at universities and staff leaving AEG-Telefunken to work for universities, research institutes, governmental agencies, and industrial companies. The final report of Telefunken Computer covering the TR 440 development—funded by the *2. DV-Programm der Bundesregierung* (the second data processing program of the German government)—said:

> At the beginning of the TR 440 project only a few experienced people were available. The main reasons were that electronic data processing was a very young discipline, especially in Europe, and at that time there was next to no education in computer science. . . . In addition to the primary result of the project, namely the large scale TR 440 computer system, we have to note as an indirect result the technical and scientific knowledge which was gained in the complex areas of electronic data processing and of managing large projects. This knowledge has been passed on by intensively pursued training and education.[1]

For Germany, the transfer of knowledge that had been gained with the TR 4 and TR 440 development was important for the rapid buildup of computer science and computer know-how.

Acknowledgments

First of all, I thank Chris Earnest, who critically reviewed and corrected my English version of this article and gave invaluable hints for improvements so that things became clearer and more understandable for readers. My grateful thanks go to Manfred Evers, Joachim Feldmann, Wolfgang Frielinghaus, Wolfgang Froehlich, Alexander Giedke, Fritz-Rudolf Güntsch, Heinz-Gerd Hegering, Gisela Hoffmann, Eike Jessen, Jürgen Keppler, Dieter Michel, Albert Noltemeier, Gerd Sapper, Kurt

Further Reading

For those interested in additional reading on the TR 440, AEG-Telefunken, and any related topics, the following resources provide a wealth of information.

E.G. Coffman Jr. and P.J. Denning, *Operating Systems Theory,* Prentice-Hall, 1973.

Computer Gesellschaft Konstanz, ''Die Kopplung von Fremdsystemen an das Rechensystem TR 440 über KOMSYS'' [Coupling Computers of Other Manufacturers with the TR 440 Computer using KOMSYS], 440.B9.07, Ausgabe 0975, Sept. 1975.

Computer Gesellschaft Konstanz, ''Teilnehmer-Rechensystem Kurzbeschreibung'' [Multiaccess System: Short Description], . 440.B0.04, Ausgabe 0375, Mar. 1975.

''Computer Gesellschaft Konstanz: Siemensverträglich'' [CGK: Peaceable with Siemens], *Computerwoche,* no. 16, 1976.

''TR 445 DP und Cyber 7276 im Verbund—Großrechnerkoppelung zwischen Universitäten'' [TR 445 Double Processor and Cyber 7276 in a Network: Connection of Large Scale Computers of Universities], *Computerwoche,* //no. 17, 1977.

''Umstellungshilfe mit BMFT-Förderung—Ausmusterung der TR 440 erleichtert'' [Transition Aids with BMFT's Subsidies—TR 440s Putting out of Service Easier], *Computerwoche,* no. 11, 1982.

''Nixdorf zwischen MDT und IBM. Eine deutsche Erfolgsstory—vom Einmannbetrieb zum internationalen Super-Systemhaus (Teil 2)'' [Nixdorf between MDT and IBM. A German Success Story—from One-Person Business to a Large International System Company (Part 2)], *Computerwoche,* no. 46, 1987.

P.J. Denning, ''The Working Set Model for Program Behavior,'' *Comm. ACM,* vol. 11, no. 5, 1968, pp. 323–333.

P.J. Denning, ''Virtual Memory,'' *ACM Computing Surveys,* vol. 2, no. 3, 1970, pp. 153–190.

P.J. Denning, ''Third Generation Computer Systems,'' *ACM Computing Surveys,* vol. 3, no. 4, 1971, pp. 175–216.

J.B. Dennis and E.C. Van Horn, ''Programming Semantics for Multiprogrammed Computations,'' *Comm. ACM,* vol. 9, no. 3, 1966, pp. 143–155.

E.W. Dijkstra, ''The Structure of the THE-Multiprogramming System,'' *Comm. ACM,* vol. 11, no. 5, 1968, pp. 341–346.

R. Engelhardt et al., ''Rechnerverbund beim TR 440'' [Networks with TR 440], *Elektronische Rechenanlagen,* vol. 17, no. 1, 1975, pp. 13–17.

R. Engelhardt, J. Huber, and S. Luhmann, ''Datenhaltung im Teilnehmer-Betriebssystem TNS 440'' [Data Management in the Multiaccess Operating System TNS 440], Telefunken Computer, Beiträge 11, 1972.

H. Fischer, P. Namneck, and L. Stolze, ''Datensicherheit auf Großrechnern'' [Data Security in

Scheidhauer, Eberhard Schmolz, Gerhard Seegmüller, Franz Stetter, Günther Stiege, Hanni Stolze, Heinz Voigt, Dietrich Wagner, and Hans-Rüdiger Wiehle. They helped me prepare this article with different kinds of support, from supplying documents to diligently reading and enhancing parts of the article. Besides this, they told me anecdotes and gave important suggestions. In particular, Wiehle closely examined the sections on the TR 4 and TR 440 BS1—a special thanks for this.

I also owe many thanks to Wolfgang Füßl and Hartmut Petzold of the Deutsches Museum (German Museum [of Technology]) in Munich. The Deutsches Museum Library has accepted into its archive the TR 440 documents that I had or got from others, so the documents, including most of those cited here, are available for future scientific research.

Finally, I thank the anonymous reviewers for their helpful comments

References

1. Telefunken Computer, ''Vielfachzugriffssystem TR 440, Schlußbericht'' [Multiaccess System TR 400, Final Report], II. Datenverarbeitungsprogramm der Bundesregierung, Teilprogramm 4, Dec. 1973.
2. E. Jessen et al., ''The AEG-Telefunken TR 440 Computer: Company and Large-Scale Computer Strategy,'' *IEEE Annals of the History of Computing,* vol. 32, no. 3, 2010, pp. 20–29.
3. Telefunken, ''Großrechenanlage TR 4, Kurzbeschreibung'' [Large Scale Computer TR 4, Short Description], KB006/1, 1962.
4. E. Ulbrich, ''Struktur und Arbeitsweise der Telefunken-Digitalrechenanlage TR 4'' [Structure and Principles of Operation of the Telefunken Digital Computer TR 4], *IEEE Trans. Electronic Computers,* 1963.
5. Computer Gesellschaft Konstanz, ''Zeugnis für Wolfgang Frielinghaus'' [Job Reference for Wolfgang Frielinghaus], 1976.
6. H.R. Wiehle et al., ''Ein Betriebssystem für schnelle Rechenautomaten'' [An Operating

Large-Scale Computers], *Elektronische Rechenanlagen,* vol. 21, no. 6, 1979, pp. 297–304.

H. Forster et al., ''Planung und Verwaltung von Betriebsmitteln im Teilnehmer-Betriebssystem TNS 440'' [Resource Planning and Managing in the Multiaccess Operating System TNS440], Telefunken Computer, Beiträge 9, 1972.

J. Fotheringham, ''Dynamic Storage Allocation in the ATLAS Computer, Including an Automatic Use of a Backing Store,'' *Comm. ACM,* vol. 4, no. 10, 1961, pp. 435–436.

B. Gebhardt, ''Interaktives Arbeiten am Terminal'' [Interactive Work at a Terminal], Telefunken Computer, Beiträge 10, 1973.

C.A.R. Hoare, ''Towards a Theory of Parallel Programming,'' *Proc. Int'l Seminar Operating Systems Techniques,* 1971.

C.A.R. Hoare, ''Monitors: An Operating System Structuring Concept,'' *Comm. ACM,* vol. 17, no. 10, 1974, pp. 549–557.

E. Jessen, ''Das TR 440-Großrechensystem an deutschen Hochschulen,'' [TR 440 Large-Scale Computer System at German Universities], press release, AEG-Telefunken, 11 Nov. 1971.

E. Jessen and E. Ulbrich, ''TR 440 als Teilnehmersystem'' [TR 440 as multiaccess system]. In Datenverarbeitung mit Mehrfachzugriffssystemen [data processing with multiaccess systems]. Haus der Technik, Essen, 1968.

E. Jessen, D. Michel, and H. Voigt, ''Structure, Technology, and Development of the AEG-Telefunken TR 440 Computer,'' *IEEE Annals of the History of Computing,* vol. 32, no. 3, 2010, pp. 30–38.

T. Kilburn et al., ''One-Level Storage System,'' *IRE Trans.,* vol. EC11, no. 2, 1962, pp. 223–235.

J.E. Morrison, ''User Program Performance in Virtual Storage Systems,'' *IBM Systems J.,* vol. 12, no. 3, 1973, pp. 216–237.

K. Radius, ''Probleme der Entwicklung von Großrechenanlagen'' [Problems in Developing Large-Scale Computers]. Vortrag vom 3. Jul. 1968 bei der Arbeitsgemeinschaft für Forschung des Landes NRW, AEG-Telefunken DVO 060, Oct. 1968.

D.M. Ritchie and K.L. Thompson, ''The Unix Time-sharing System,'' *Comm. ACM,* vol. 17, no. 7, 1974, pp. 365–375.

Südkurier, ''Telefunken Computer und Unidata nähern sich an'' [Telefunken Computer and Unidata Get Closer to Each Other], *Südkurier,* 12 Dec. 1973.

Telefunken Computer, ''Time-Sharing Computing-System—Introduction,'' VS1, N31.B0.04 E, 1972.

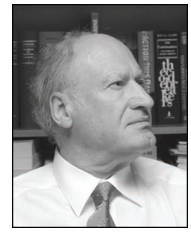
H.R. Wiehle, ''External Characteristics of Computer Operations: Toward Large Conversational Time-Sharing Systems,'' *IEEE Annals of the History of Computing,* vol. 32, no. 3, 2010, pp. 4–18.

system for Fast Computing Automatons], *Elektronische Rechenanlagen,* 1964.

7. G.R. Sapper, ''Telefunken TR 4,'' 2004; http://www.qsl.net/dj4kw/index.htm.
8. G. Seegmüller, ''Some Remarks on the Computer as a Source Language Machine,'' *Proc. Int'l Federation of Information Processing* (IFIP) *Congress,* vol. 62, North-Holland, 1962, pp. 524–525.
9. W. Wulf et al., ''Curriculum Vitae Prof. Dr.-Ing. Fritz-Rudolf Güntsch'', Medieninformation Nr. 8, Pressestelle TU Berlin, 2002.
10. S. Rosen, ''Electronic Computers: A Historical Survey,'' *ACM Computing Surveys,* vol. 1, no. 1, 1969, pp. 7–36.
11. J. Bellec, ''Information Technology Industry Time Line,'' 2006; http://perso.orange.fr/jeanbellec/information_technology_1.htm to http://perso.orange.fr/jeanbellec/information technology 5.htm.
12. J.B. Dennis, ''A Multiuser Computation Facility for Education and Research,'' *Comm. ACM,* vol. 7, no. 9, 1964, pp. 521–529.
13. Ferranti, ''Die Atlas Rechenanlage'' [The Atlas Computer], Ferranti; Faltblatt No. 1 in einer Reihe von Skizzen unserer Rechenanlagensysteme, [Flyer 1 in a Series of Short Introductions of Our Computer Systems], 1962.
14. F.J. Corbato et al., ''An Introduction and Overview of the Multics System,'' *Proc. AFIPS 1965 FJCC,* vol. 27, part 1, Spartan Books, pp. 185–196.
15. E. Jessen, ''Stellungnahme zum Entwicklungsvorhaben TR 400'' [Statement to the Development Project TR 400], internal letter via Güntsch to Peltz, AEG-Telefunken, 4. Nov. 1964.
16. H.R. Wiehle, ''Operating Systems at AEG-Telefunken,'' Proc. Conf. Pioneering Software in the 1960s in Germany, The Netherlands, and Belgium,'' 2006.
17. H.R. Wiehle, ''Vorträge der Grundprogrammentwicklung (E44) für den Vertrieb über die Grundprogrammierung des TR 440'' [Lecture by the System Programs Development Group (E44) for Sales People about TR 440 System Programs], internal memo, Telefunken, 1967.

18. E. Jessen, ''Das Betriebssystem des Rechners TR 440'' [The Operating System of the TR 440 Computer], *Teilnehmer-Rechensysteme,* W. Händler, ed., Oldenbourg Verlag, 1968, pp. 114–123.
19. P. Namneck, H.-J. Siegert, and H.R. Wiehle, ''TR 440 Grundprogramme 1—Einführung'' [TR 440 System Programs 1: An Introduction], AEG-Telefunken, Großrechner, Oct. 1968.
20. R.F. Rosin, ''Supervisory and Monitor Systems,'' *ACM Computing Surveys,* vol. 1, no. 1, 1969, pp. 37–54.
21. AEG-Telefunken, ''TR 440 Betriebssystem BS2 (erste Ausbaustufe)—Einführung'' [TR 440 Operating System BS2 (First Stage): Introduction], Schrift DBS182/0470, 1970.
22. G. Stiege, ''Zum Betriebssystem BS2'' [About Operating System BS2], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen, AEG-Telefunken Berlin, 1970, pp. 112–115.
23. K. Lagally, *Das Projekt Betriebssystem BSM* [The Project Operating System BSM], tech. report 7509, Technische Universität München, Institut für Informatik, 1975.
24. G. Goos, J. Jürgens, and K. Lagally, *The Operating System BSM Viewed as a Community of Parallel Processes,* tech. report 7208, Technische Universität München, Fakultät Mathematik, 1972.
25. A. Jammel and H. Stiegler, ''Managers versus Monitors,'' *Information Processing,* B. Gilrichst, ed., North-Holland, 1977, pp. 827–830.
26. A. Giedke, ''Die Entwicklung des TR 440,'' unpublished draft, 1974, German Museum library.
27. AEG-Telefunken, ''Bedienungshandbuch Doppel-TR 4-System auf TR 440—Ämulator-Erläuterungen und Verteilerprogramm'' [Operation Manual Double-TR 4-TR 440 System—Ämulator Explanations and Distribution Program], user manual N3/GR 70, Sept. 1968.
28. G. Hoffmann, ''Das Doppel-TR 4-System'' [Double TR 4 System], Vortrag, handwritten, unpublished, 1969, German Museum library.
29. J. Schilling, ''Digitalrechner-Geschichte: TR 4—Ein paar Gedanken zur Rechner-Historie in Deutschland'' [Digital Computer History: TR 4 – A Few Thoughts about the Computer History in Germany], 2004; http://www.pc-profiler.ch/geschichte.htm.
30. AEG-Telefunken, ''Unterlagensammlung TR 440 - Betriebssystem-Kern'' [Collection of Documents about TR 440: Operating System Kernel], internal document, N31.B1.11, Jul. 1970.
31. F.v. Sydow, ''Die TR-440-Staffel—vom mittleren Rechensystem bis zum dialogfähigen Teilnehmer-Rechensystem'' [The TR 440 Family of Computers: From Middle Computer to Conversational Time-Sharing System], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen AEG-Telefunken Berlin, 1970, pp. 101–104.
32. J. Piper et al., ''Das Teilnehmer-Betriebssystem BS3'' [The Multiaccess System BS3], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen AEG-Telefunken Berlin, pages 115–122, 1970.
33. H. Niesporek, ''Die AEG-TELEFUNKEN Datenverarbeitungsanlage TR 440'' [The AEG-Telefunken Computer TR 440], 1972.
34. Telefunken Computer, ''Leistungserweiterung des TR 440-Systems durch MV8 bis MV16'' [Enhancements (Performance and Services) of the TR 440 System from MV8 to MV16], internal memo, VS33, 19 Sept. 1974.
35. G. Stadie, ''Der TR 440 mit zwei Rechnerkernen und Massenkernspeicher'' [TR 440 with Two CPUs and Mass Storage], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen, AEG-Telefunken Berlin, 1970, pp. 132–133.
36. M. Evers, ''Datenfernverarbeitung im Teilnehmersystem des TR 440'' [Remote Data Processing in the TR 440 Multiaccess System], Telefunken Computer, Vortrag, Best. Nr. N31.B9.01, 1972.
37. M. Evers and W. Hoheisel, ''Das Satellitensystem des Telefunken-Rechensystems TR 440'' [Satellite System of the Telefunken Computer TR 440], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen AEG-Telefunken Berlin, 1970, pp. 122–124.
38. H.-J. Siegert, ''Rechnerverbundnetz'' [Computer Network], software development documents, Telefunken Computer, 29 Jun. 1973.
39. Telefunken Computer, ''Telefunken Computer—Logik setzt sich durch'' [Telefunken Computer — Logic Wins], Firmenprospekt, 1972.
40. E. Schmidt et al., ''Zum Programmiersystem des Telefunken-Rechensystems TR 440'' [The Programming System of the Telefunken Computer TR 440], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen, AEG-Telefunken Berlin, 1970, pp. 124–131.
41. Telefunken Computer, ''Programmiersystem—Einführung'' [Programming System: Introduction], VS1, N31.B0.02, 1972.
42. P. Jäger, http://www.jaegers-home.de/person.htm.
43. ''DV-Oldies: Antiquität oder leistungsgerechtes System'' [Data Processing Oldies: Antique or System with Still Adequate Services], *Computerwoche,* no. 35, 1986.
44. *IBM Highlights, 1885–1969,* tech. report 1406HA02, IBM, Dec. 2001.
45. H. Krainer, ''Testmöglichkeiten im Teilnehmer-Rechensystem TR 440'' [Testing (User Programs)

in the Multiaccess System TR 440], Telefunken Computer, Beiträge 10, 1972.
46. H. Pszolla and F. Eberhardt, ''BCPL, eine Sprache zum Schreiben von Compilern'' [BCPL, A Language for Writing Compilers], Beiträge 1, Telefunken Computer Konstanz, 1972.
47. O'Reilly Media, ''History of Programming Languages,'' 2004; http://www.oreilly.com/pub/a/oreilly/news/languageposter_0504.html.
48. H. Voltz, ''Anwendungssysteme für den TR 440'' [Application Systems for the TR 440], Datenverarbeitung 3, Beihefte der Technischen Mitteilungen AEG-Telefunken Berlin, 1970, pp. 136–140.
49. Telefunken Computer, ''Beispiele zur problemorientierten Software des Teilnehmer-Rechensystems TR 440'' [Examples of Application-Oriented Software for the Multiaccess System TR 440], Beiträge 7, 440.ZZ.50.07, Dec. 1972.
50. ''Software-Subventionen noch ansteigend—MDT 1978 auf dem Bonner Fördergipfel'' [Software Subsidies Still Rising: MDT 1978 at the Bonn Summit for Subsidies], *Computerwoche,* no.5, 1978.
51. M. Campbell-Kelly and W. Aspray, *Computer: A History of the Information Machine,* Basic Books, 1996.
52. AEG-Telefunken, ''Software Projektmanagement'' [Software Development Project Management], internal document, 23 Jun. 1970.
53. H.-J. Siegert, ''Stand der Software-Entwicklungen—Vortragsnotizen zu Kooperationsgesprächen mit der Siemens AG'' [State of Software Development: Lecture Notes for Cooperation Talks with Siemens AG], internal document, Telefunken Computer, 30 Jul. 1974.
54. H. Zoller, ''RUB trauert um Professor Hartmut Ehlich'' [RUB (Bochum University) is Mourning Professor Hartmut Ehlich], 2002; http://www.ruhr-uni-bochum.de/pressemitteilungen-2001/msg00402.html.
55. H. Janisch, ''30 Jahre Siemens-Datenverarbeitung—Geschichte des Bereichs Datenverarbeitung 1954-1984'' [30 Years of Siemens Data Processing: History of the Data Processing Department from 1954 to 1984], Siemens AG, 1988.



Hans-Jürgen Siegert is a retired professor of computer science from the Munich University of Technology and was a director of the Institute of Informatics and also responsible for its computer center. His interests include operating systems, real-time systems, networking, robotics, simulation, and project management methods. Siegert studied physics and received a doctor's degree in engineering from Stuttgart University. He is an IEEE Life Member and a member of ACM and GI (Gesellschaft für Informatik e.V.).Contact him at siegert@in.tum.de.