Niklaus Wirth — a Pioneer of Computer Science

Gustav Pomberger, Hanspeter Mössenböck, Peter Rechenberg
Johannes Kepler University of Linz
pomberger@swe.uni-linz.ac.at, moessenboeck@ssw.uni-linz.ac.at, rechbg@soft.uni-linz.ac.at

Abstract

Niklaus Wirth is one of the most influential scientists of the early computer age. His ideas and especially his programming languages have shaped generations of programmers worldwide. This paper tries to acknowledge the scientific achievements of Niklaus Wirth and to honor him as a person. A small part of the paper is also devoted to Wirth's influence on computer science at the Johannes Kepler University of Linz.

1 Introduction

Ask computer specialists anywhere—be it in Europe or America, in Asia or Australia—who the most important computer scientists in the world are, and you can bet that the name Niklaus Wirth will be on every list. Ask programming language experts about the person with the greatest influence on the development of programming languages, and they would agree on Niklaus Wirth. Ask students, teachers and computer hobbyists about the most important programming languages, and you can be sure that even today Pascal will be on every list. Finally, examine the literature of computer science in the elite circle of books with the greatest number of copies printed, the widest distribution, and the most translations, and you will find books by Niklaus Wirth, especially the prominent book on Pascal.

These few examples show that professor Niklaus Wirth is one of the world's leading computer scientists. It is our honor and pleasure to acknowledge his achievements as a scientist, educator and person as well as his influence on research and teaching all over the world.

We begin by reviewing Wirth's scientific career and by recognizing his most important work as well as the honors that the computer world has bestowed upon him. Wirth's work had significant impact on the development of computer science and the work of other researchers worldwide. Needless to say, he has also influenced research and teaching at the Johannes Kepler University of Linz, and part of this paper sketches this influence. Finally, we take particular pleasure in acknowledging Niklaus Wirth as a person and a contemporary.

2 Wirth's Scientific Career

Niklaus Wirth was born in 1934 in Winterthur, Switzerland. After graduating from ETH Zurich in electrical engineering in 1959 he completed his master's degree at Laval University in Quebec, Canada, in 1960 and went on to complete his Ph.D. in programming languages in 1963 under professor H. D. Huskey at the University of California at Berkeley. Wirth served as an assistant professor at the newly founded computer science department at Stanford University from 1963 to 1967. During this time he became intensely involved with questions regarding programming-language development.

In 1967 Wirth returned as a professor to Zurich, first to the university, then in 1968 to ETH, where he co-founded the Department of Computer Science. The development of computer science as an academic major and as a research field at ETH was and is decidedly influenced by Wirth. In March 1999 Wirth retired as professor at ETH with a much-acclaimed parting lecture [47].

The guiding aspiration of Wirth's work has been to exploit to the limits of technical feasibility the possibilities offered by the computer for solving problems, all the while striving for simplicity and comprehensibility. This attitude arises from his insight into interrelations, and presents a sharp contrast to today's prevailing tendency to make programs, systems and organizations increasingly huge and complex, and thus opaque and dangerous [42].

Wirth does not belong to the ever growing number of scientists who publish simple (sometimes trivial) phenomena in an inflated, scientifically padded representation without offering much innovation. Quite the contrary, his work contributed to simplifying the understanding of complex interrelations and to providing clean solutions to existing problems.

The following subsections describe five periods in Wirth's scientific work, each lasting for about a decade and each crowned with major contributions to the development of computer science.

The Early Years

Shortly before 1960 the first Fortran compiler was completed, Algol was defined and some years later implemented. But—strangely enough—this pioneering work was hardly reflected in the open literature. Till the end of the decade only one coherent book about compiler construction appeared [2]. It was understandable that IBM was not interested in the publication of the techniques applied in Fortran compilers, but why did so little appear about the implementation of Algol? Only fragments such as the translation of arithmetic expressions and parsing methods were published. The reason for this reservation was perhaps that the implementers felt their work yet to be immature and poorly suited for publication. General principles and methods were scarcely recognized; and if they were, how could they

be described? Assembler code was unreadable, Algol was known only to a small minority, and Fortran was taboo in the Communications of the ACM.

This was the situation when Wirth's and Weber's paper "Euler: A Generalization of Algol and its Formal Definition" appeared in 1966 [3]. Contrary to all the patchy work before, it contained the syntactic and semantic description of a complete language with a size similar to Algol. Moreover, the description of syntax and semantics were not intermingled as usual at this time but clearly separated, and the semantic description was quite formal. This was new! The paper contained also a lot of other interesting things: An outline and a judgement of contemporary suggestions of semantic descriptions, a definition of precedence grammars (which were invented by Floyd shortly before [1]), and, last but not least, the formal description of an interpreter that served as back end in the Euler compiler.

This description method was so sound that Wirth from thereon always put his faith in it. His technique to separate syntax and semantics may have been a model for Knuth's attributed grammars. It was also new to build the syntax of a programming language in accordance with a particular parsing method, and, of course, to combine the definition and an implementation of the language. This combination characterizes also Wirth's later work: not only to define a language but to implement it as well and to experiment with it before publication. Here we encounter the old and humble craftsmanship which today's computer scientists would need so much.

Two years later Wirth developed PL360 [5], an algorithmic language for the IBM 360 machines. This work showed how steadily he went his own way, building innovations on top of experienced grounds. We believe that PL360 must have been a revelation for the assembler programmers of the IBM 360. It is remarkable that Wirth developed the PL360 compiler as a cross compiler, since no 360 machine was available to him at this time. He wrote:

> At the time when the project to develop a compiler for PL360 was started, no 360 computer was available to the author, nor did the facilities promised with the forthcoming machine look too enticing to use. It was therefore decided to use the available Burroughs B5500 computer for the design and testing of the compiler, which was completed by the author within two months of part time work. It accepted a preliminary version of PL360 which contained the basic features of the presently described language.
>
> The compiler was then reprogrammed in its own language. Through a loader and supervisor program (written in assembly code), the program, recompiled on the B5500, became immediately available on the 360 computer.
>
> The experiment of describing the compiling algorithm in PL360 itself proved to be the most effective test on the usefulness and appropriateness of the language and it influenced the subsequent development of the language considerably. During this process, several features which seemed desirable were added to the language, and many were dropped again after having proved to be either dubious in value, inconsistent with the design criteria, or too involved and

> leading to misconceptions. The leading principle and guideline was to produce a conceptionally simple language and to keep the number of features and facilities minimal.

Don't these sentences embody the program of Wirth's further scientific life? Didn't he apply the same methods in the development of Pascal, Modula and Oberon?

When we read the PL360 paper anew we were astonished how careful he motivated his design decisions. Read for instance the section "Addressing and Segmentation" which discusses base register addressing, a property of the 360 that led to many difficulties and programming errors.

From Euler to Pascal

Wirth's work on Euler attracted the interest of an IFIP working group whose goal was to propagate and further develop Algol60. In 1965 three recommendations for a successor were submitted. During the deciding meeting in the fall of 1966 in Warsaw, the proposal of van Wijngaarden was favored over Wirth's recommendation and was developed further to receive IFIP ratification in 1970 as Algol68. While Algol68 was a completely new language, Wirth's recommendation had strived to improve Algol60. Together with Hoare, Wirth refined his proposal into a language called Algol-W [4] and implemented it at Stanford on an IBM/360 using his systems programming language PL360. Algol-W was widely distributed among universities and contributed to spreading the spirit of Algol.

In 1968 Wirth began with the development of Pascal [7, 8, 10, 11, 14]. This happened to be the beginning of a whole series of great contributions to the development of computer science and laid the foundation of Wirth's reputation as a computer pioneer. He writes of his work on Pascal:

> Freed of the constricting influence of a workgroup consensus, the language Pascal was conceived in Zurich. The foundation was provided by Algol-W and the desire to develop a language that can be implemented efficiently and that meets the requirements for building system programs (compilers, operating systems). Furthermore, we gave top priority to a base of clear, mathematically and machine-independently defined concepts so that the language could be employed to the advantage of instruction in computer science.

These goals were more than achieved. Pascal was first used for teaching at ETH in 1972 and grew to be one of the most popular teaching languages in the world. Not only teachers but also practitioners learned to appreciate Pascal's concepts. Hardly any other programming language has been the subject of so much literature.

In the development of Pascal, a major aim was to show that structured languages need not take a back seat to older languages such as Fortran with respect to the efficiency of the generated code as long as proper attention was paid to the implementation of the compiler. Unfortunately the computer industry and industrial

users, who should have had a particular interest in this kind of engineering, took many years to recognize the value of this work, and some never did.

A new Pascal compiler project at ETH served as a proving ground for the technique of stepwise refinement in program development, which was also postulated by Wirth [6]. This provided the welcome opportunity for other universities to assist in porting the Pascal compiler to other computers. The machine-independent parts of the new compiler generated code for a virtual machine, which could then be implemented rather easily as an interpreter on other machines. This virtual machine was called Pascal-P (for portable) [24] and became the basis of the many Pascal implementations that first became available for mainframes (IBM, DEC, UniVac, Siemens). The breakthrough for Pascal, however, came with the rise of the microcomputer and the realization that Pascal-P could be implemented there as well.

Even then Wirth did not confine himself to just the design and implementation of programming languages. In 1970 ETH acquired a mainframe, a CDC 6000. The priority in selecting the machine was to meet the demands of scientific computing (number crunching). While time sharing systems were spreading worldwide, the software of this computer was not well-suited to this approach. Under the direction of Wirth and in cooperation with the ETH computing center, the new time sharing system Venus was developed and successfully met the demands of multiple terminal access without a noticeable reduction in computing performance.

From Pascal to Modula-2 and Lilith

In 1973 Wirth committed himself to the development of concepts and tools for the design of modular multiprogramming systems. The rules postulated here forced the formulation of language elements to express the creation and synchronization of such processes. Embedding these rules into a minimal environment led to the language Modula [17, 18, 19], which was implemented on a PDP-11. The first application was the Hexapus system, which permitted the direct connection of small computers to the computing center.

In a sabbatical year spent at the Xerox Palo Alto Research Center in 1976, Wirth encountered a fundamentally new kind of computer, the personal workstation. It was programmed in a language called Mesa [16], which was based on the concepts of modules and separate compilation. Back at ETH Wirth launched a research project with the goal of developing such a computer together with an appropriate programming language and system software. The result was the workstation Lilith [23, 27], the programming language Modula-2 [25], the operating system Medos [26] and a couple of other tools. Despite the apparently enormous scope of the project, the determined work of up to seven team members achieved the goal in only three years. This is a result that is hard to beat.

In addition to an operating system and a compiler, a program development system requires an interactive editor. All these tools had to be developed from scratch for the Lilith in order to be able to exploit the high-resolution monitor and

the mouse. Wirth himself programmed the prototypical text editor Dina as well as the graphics editor Sil for producing figures and schematic diagrams for documenting the Lilith hardware. Dina proved to be the model for the later editors Andra and Lara [28], which allowed the simultaneous creation of multiple documents with various fonts and were inspired by Xerox' Bravo editor [22]. Although tools of this kind are standard for every computer today, at that time they were revolutionary.

The most significant contribution of the Lilith project was to enable solutions that were unthinkable with commercially available products of the time. The first ten Liliths were put into operation in 1980, years before similar systems became commercially available. Lilith computers were subsequently employed not just at the ETH Zurich, but throughout the world (one of the first Liliths outside of ETH was shipped to the University of Linz). Seldom have the results of a university research project been translated so directly into practical use.

The development of a complete system like Lilith in such a short time was made possible in part by the deliberate availability of only one programming language. The choice was a modified Pascal called Modula-2, a language inspired by Mesa that consciously sacrificed backward compatibility. The new concepts included the module as the programming unit for separate compilation, the coroutine as the basic component of parallel processes, and an encapsulated set of types and procedures to enable access to machine-dependent properties.

The first Modula-2 compiler was completed in 1979 on the PDP-11; it was then ported to the Lilith. Interest in Modula-2 grew quickly because it had obvious advantages over Pascal, especially for the design of complex systems in teams. Furthermore, Modula-2 was simpler and more elegant than Ada, a language developed at the same time with nearly the same goals. Not only was the PDP-11 compiler distributed to hundreds of universities and companies, also compiler companies began to develop and sell their own compilers. The advantages of Modula-2 already became evident in the development of the Lilith software. The compiler, the operating system, the editors, and all other tools were programmed exclusively in Modula-2. This showed that the restriction to a single language was not only possible, but even brought significant advantages.

A personal workstation gains value when it is connected to other computers via a network, especially with servers and shared resources such as printers and file servers. Under the direction of Wirth, a network interface was designed on the basis of Ethernet. This was the first fast local area network in Switzerland and was well ahead of its time.

Although Wirth had already experienced the multifaceted use of laser printers in 1976 at Xerox PARC, these printers only became affordable in 1982, and even then they lacked the interface to computers that could make use of their advantages. That same year Wirth acquired the first two laser printers in Europe. He designed the hardware and software interfaces himself as well as the formatting program

Skylla/Zeus, with which he produced his book *Programming in Modula-2* on the Lilith to camera-ready copy. Again Wirth was a harbinger.

Another important contribution in the seventies was Wirth's proposal for a new syntax notation [21], to become known as EBNF. With Algol, PL/I and other languages a bunch of different notations had been developed, all variations of the Backus-Naur form (BNF), some of them enriched by new metasymbols, some even written in two dimensions (PL/I). Wirth proposed a new notation that at one blow made all the others look baroque and inappropriate. It allowed expressing recursion by repetition, the removal of empty alternatives, and it was easily parsable by a machine. Wirth used this notation for the description of his own languages. We wonder why it did not become a world wide standard. Unfortunately, Ada, Java and other new languages use different syntax notations again.

From Modula-2 and Lilith to Oberon and Ceres

With Modula-2 and Lilith, Wirth set a milestone in computer science. But he would not have been Wirth if he had been satisfied with that and rested on his laurels. Guided by his knowledge and experience from the Modula-2 project, and with the support of his colleague Jürg Gutknecht and a small project team, he developed the programming language and operating system Oberon [34, 35, 36, 39, 40] and the Ceres workstation (Model 1 1984-86, Model 2 1988, and Model 3 1990 [32, 37]). These developments, like their precursors, stood out by advancement and simplification of the status quo without uneasy compromises. Oberon enhanced Modula-2 by the concept of type extension [33], and thus by object-oriented concepts; at the same time it is a leaner Modula-2 trimmed down to what is most important. The language Oberon is seductive in its simplicity and minimality, as is the operating system.

Likewise, the Ceres workstation is exemplary of consistent development in terms of overview and understanding. In his challenging requirements definition Wirth demanded, among other things, energy-saving and quiet operation. He achieved this with such an extremely low energy consumption level that a noisy ventilation fan became superfluous. The Ceres-3 consumes only 15 watts in continuous operation without a monitor.

In order to make Oberon available on all modern computers of that time, a new project [43] was launched in 1989 in which the Oberon compiler was split into a machine-independent front end and a machine-specific back end. The back end was implemented for the IBM PC, for the Macintosh as well as for workstations from Sun, DEC, IBM and HP. This established full source code compatibility of Oberon programs across a multitude of platforms. In the course of this project, the Oberon language was slightly extended to Oberon-2 [38], which supports type-bound procedures. The Oberon-2 compilers were distributed via the newly popular Internet and were picked up by many universities and companies all over the world.

Wirth's recent work

In recent years Wirth has been involved in developing programming languages and tools for Field Programmable Gate Arrays (FPGAs) [45, 46]. These are components that afford an array of processors of a single type. Each processor handles one or more very simple logical operations and is connected with its neighbors. By programming the array, any hardware circuit can be implemented in software. The circuit proves somewhat slower than a real hardware circuit, but has the advantage that it can be reprogrammed at run time. Wirth's assistants employed this technique to build a simple workstation in which individual hardware components assume different functions. Wirth successfully used FPGAs in his lectures on Digital Circuit Design [41], which allowed students to design and test circuits in software. This proved to be an inexpensive alternative to hardware laboratories and shows how Wirth applies his endeavor for efficiency, simplicity and elegance to all conceivable fields.

Before receiving his emeritus status, Wirth once again let his engineering talent flare up. He conceived and implemented hardware and software for the control of model helicopters—of course in Oberon. This project proved that there is no reason for not applying modern high-level languages to even such specialized fields as low-level real-time systems.

Publications and academic honors

Wirth's outstanding character is also reflected in his publications, which, like his other work, are distinguished by clarity and concentration on what is relevant. Wirth has authored numerous publications, among them more than 10 books that have been translated into many languages and have become classics in computer literature.

After this necessarily incomplete list of Wirth's achievements, it should be no surprise that any selection of the top ten computer scientists in the world almost certainly includes his name. The international computer science community has bestowed special recognition on Wirth and honored him with numerous awards, prizes and honorary doctorates. Eight honorary Ph.D.s from seven nations have been conferred upon him, and with at least ten high awards, Wirth is certainly one of the most decorated computer scientists in the world. We mention only the following distinctions as representative of Wirth's many honors:

- The ACM's A.M. Turing Award is the highest distinction that a computer scientist can achieve. Wirth received this award in 1984.
- The IEEE Emanuel R. Piore Award for achievement in the field of Information Processing contributing to the advancement of science and the betterment of society went to Wirth in 1983.
- in 1987 Wirth was decorated with the IEEE Computer Pioneer Award.

- The Prix Max Petitpierre, an award presented to a person whose political, diplomatic or economic activities, or scientific or artistic works, have made a contribution to improving the relationship of Switzerland with the rest of the world, was extended to Wirth in 1990.
- The Marcel Benoit Prize was imparted to Wirth in 1991 in recognition of the computer languages introduced by him that realized new concepts of structured programming and had a lasting influence on the multifaceted use of computers worldwide and in all fields of knowledge.
- In 1994 Wirth was inducted into the U.S. National Academy of Engineering.
- In 1999 Wirth was awarded the Outstanding Research Award in Software Engineering by ACM SIGSOFT and the Leonardo da Vinci Medal from the Societe Europeenne pour la Formation des Ingenieurs (SEFI).

3 The Person Niklaus Wirth

One of the sources of Wirth's many successes, indeed perhaps the primary source, has been a rare combination of personal characteristics. On the one hand, he is by nature an inventor with a tremendous feeling for what will be of importance in the future. On the other hand, he has the ability to combine theory and practice in a most advantageous way. He not only defined his programming languages and tools on paper, but also built compilers and implemented tools so that they could be used by the rest of us. He even designed and built his own computer systems that were tailored to his programming languages, thus merging hardware and software to an inseparable unit. There are few computer scientists who feel at home in both hardware and software engineering to the same extent, and we know of no other person who can combine the construction of software and hardware in such an advanced way.

Another characteristic of Wirth is the inevitability of his continuing development work. We are tempted to say that he is obsessed with striving for simplicity, efficiency and elegance. For example, after it became clear in 1982-83 that existing Modula-2 compilers, including those at ETH, failed to fully exploit the advantages of the language, Wirth decided to design a new compiler, which he implemented himself. The new compiler was based on the one-pass principle, made possible by the larger memory capacity of modern computers, and drastically reduced the frequency of slow access to secondary storage. The new compiler proved attractive in its clear and simple design and its efficiency. It encompassed 5000 lines of code, compared to 10,000 for its predecessor and more than 100,000 for other comparable languages. It compiled itself in less than 2 minutes, compared to 30 minutes for its predecessor and hours for other compilers of comparable languages. These advantages not only become obvious in daily use, they also refute the frequent claim that modern languages need gigantic, complex compilers.

Considering all that Wirth has achieved as an individual researcher and in cooperation with team members whom he often sparked to special achievements, it is easy to be astonished at his great success, and some might feel some degree of envy. Then some tend to look for negative characteristics that are so often associated with such outstanding talent. Could he be reclusive or conceited, or one-sided in his computer orientation, or a lunatic, or unapproachably arrogant? Quite the contrary, we can assure that none of this applies.

To give a very personal picture of Niklaus Wirth, one of the authors of this paper (Gustav Pomberger) remembers how he first met Wirth and how he became a disciple and a friend of him:

> As I arrived at his institute in 1982 for a research position, my reverence for him made me afraid to knock at his door. But when I first met this great scientist with quickened pulse, the friendly, pleasant-natured and modest man stretched out his hand to me and dissipated my nervousness. He did not speak down to others, as many much less creative persons in such positions often do. Wirth turned out to be not just a brilliant scientist but also a very special person in many fields. Let me share some episodes to show what I mean.
>
> The occasion was the awards presentation of the Austrian Computer Society. I was quite surprised as Wirth recognized after only a few notes that the orchestra was opening the ceremonies with Mozart's Linz Symphony, certainly not one of the composer's more important works. Wirth then began his talk on processor architecture with some comments on the Linz Symphony that showed that he was a profound music lover.
>
> Although Wirth claims not to be into sport and likes to smoke his pipe regularly, he has always been in excellent physical shape. He astounded me on our tour through South America in 1987 as he challenged me to a race at an elevation of 4800 meters in the Peruvian highlands. Although I had done a lot of running not long before, my astonishment grew as my 15 years' senior friend outclassed me without effort. His toleration of the elevation and his condition were sensational.
>
> Wirth is also a language talent—and this is not limited to programming languages. He speaks several languages and amazed me on our South America trip when we arrived in Lima as he began to employ excellent Spanish to converse with a lovely Peruvian lady.
>
> Wirth's interests also extend to politics and literature—not just computer literature, of which I suspect that he reads very little. Over the years he has called my attention to many highly interesting books. I was always amazed how much time he had for things other than computer science, although his scientific productivity is well above average. In many discussions that I recall, Wirth impressed us not only with his astuteness and analytical talent, but ever again with his broad education and his wit and sense of humor.

Often intellectuals prove to be quite clumsy when it comes to any manual work, but not Niklaus Wirth. He shows splendid craftsmanship. His home is filled with not just tools for software and hardware, but also with classical tools for precision engineering and other crafts. For example, he enjoys building model airplanes, but he also demonstrated flight talent on real planes. I recall sharing a glider flight with him. After only a little instruction, he could handle the glider almost perfectly, and some time later he took flight training in the U.S.

4 Wirth's Influence on Teaching and Research in Linz

Wirth's work influenced computer scientists at many universities all over the world. As an example we show, how Wirth's research shaped our own work at the University of Linz.

The oldest of the three authors (Rechenberg) grew up with Fortran and PL/I and had used mainly IBM machines when he came to Linz in 1975 to take a chair for software development. He knew Wirth from his publications and highly valued his work but he did not become a particular friend of Pascal because the Pascal of that time—like Algol—did not allow separate compilation of procedures. This seemed to be a severe disadvantage for practical use because it prevented the modularization of large programs.

This situation changed all of a sudden with the advent of Modula-2 and personal computers. In Modula-2 we found everything we needed for research and teaching in compiler construction and software engineering: A new small language with all the modern constructs of Pascal, and, in addition, a wonderful module concept with encapsulation, import and export.

In the early eighties—at the beginning of the microcomputer age—our group in Linz decided to write a Modula-2 compiler for such a small machine. A "modern" computer was acquired (an Intel 8085 with 128 Kbytes of memory and still without a hard disk) at a price for which you could get 20 high-end PCs today. Since half of the memory was already taken by the operating system it was clear that our compiler would not fit into memory as a whole. We had to divide it into 7 passes. Although the compiler became too clumsy and slow to be used in practice, the project was a great experience and a kind of apprenticeship for us. For the first time we had the opportunity to study Wirth's way of writing compilers in detail. We were lucky to obtain the sources of the ETH 4 pass Modula-2 compiler that had just been developed for the Lilith. We learned about Wirth's way of parsing, error handling, type checking, attribute management and code generation. What we learned was immediately integrated into our lectures, so that our students in Linz obtained first hand knowledge about modern compiler technology.

As the computers became bigger and cheaper, it was feasible to think about one-pass compilers. In 1984 Wirth developed a single-pass Modula-2 compiler for the Lilith, which was an incentive for us to start a follow-up project, in which we ported

this compiler to our mainframe, an IBM/370, which we still used for teaching at that time. Again we were stunned by the simplicity and elegance of Wirth's code. Since there was no Modula-2 compiler on the IBM/370 so far, we had to cross-develop our compiler on a Macintosh and transfer the generated code over a 300 baud modem line to the mainframe. Although every bug fix caused a recompilation on the Macintosh and a multi-hour transfer of the new object code to the mainframe, we got a first version of the compiler running within just a few weeks—a fact which can partly be put down to the clear structure of Wirth's compiler.

Our compiler projects in the eighties also led to personal contacts between Zurich and Linz. Visits of Rechenberg were followed by a longer stay of Pomberger in Wirth's group in 1982. From 1984 to 1987 Pomberger served as a professor at the University of Zurich, which allowed an even closer contact between Wirth and him. Finally, Mössenböck joined Wirth's group as an assistant professor from 1988 to 1994.

Since 1994, Oberon became the major work horse for many of our research projects in Linz. We ported the Oberon system to the Power Macintosh and to Linux, we used it in Ph.D. projects on object-oriented databases, static program analysis and distributed object systems, and extended it by a number of mechanisms such as metaprogramming, exception handling, and concurrency.

Wirth had also a substantial influence on teaching in Linz. Generations of students were educated with his programming languages. In 1982, Pascal became the primary teaching language, which was followed by Modula-2 in 1986, and finally by Oberon-2 in 1994. The simplicity and clearness of these languages make them ideal for beginners, but Modula-2 and Oberon-2 also have all the features that are necessary to teach advanced software engineering concepts.

More than anybody else from outside, Wirth has shaped teaching and research in Linz. Not only many of the professors and researchers but also the students can be called his disciples. For this reason, Wirth received a honorary doctorate from the Johannes Kepler University of Linz in 1993.

5 Conclusion

Niklaus Wirth is certainly one of the most remarkable characters in computer science. During his 40 years of professional life he has contributed some of the most influential programming languages, but also compilers, system software, computer systems and a number of pioneering software engineering practices [9, 12, 15, 20]. An uncontested pioneer in computer science and a multifaceted individual in every way, Wirth influenced many of us and still continues to enrich the world around him.

References

1. Floyd R.W.: Syntactic Analysis and Operator Precedence. Journal of the ACM, 10(3):316-333, 1963
2. Randell B., Russell L.J.: Algol 60 Implementation: The Translation and Use of Algol 60 on a Computer, Academic Press, 1964
3. Wirth N., Weber H.: Euler: A Generalization of Algol and Its Formal Definition. Comm. ACM 9(1): 11-23, and 9(2): 89-99, 1966
4. Wirth N., Hoare C.A.R.: A Contribution to the Development of Algol. Comm. ACM 9(6): 413-432, 1966
5. Wirth N.: PL360, A Programming Language for the 360 Computers, Journal ACM 15(1): 37-74, 1968
6. Wirth N.: Program Development by Stepwise Refinement. Comm. ACM 14(4): 221-227, 1971
7. Wirth N.: The Programming Language Pascal. Acta Informatica 1(6): 35-63, 1971
8. Wirth N.: The Design of a Pascal Compiler. Software—Practice & Experience 1(4): 309-333, 1971
9. Wirth N.: Systematic Programming. Prentice Hall, 1973
10. Hoare C.A.R., Wirth N.: An Axiomatic Definition of the Programming Language Pascal. Acta Informatica 2: 335-355, 1973
11. Jensen K., Wirth N.: Pascal—User Manual and Report. Springer-Verlag, 1974
12. Wirth N.: On the Composition of Well-Structured Programs. ACM Computing Surveys 6(4): 247-259, 1974
13. Wirth N.: On the Design of Programming Languages. IFIP Congress 1974, 386-393
14. Wirth N.: An Assessment of the Programming Language Pascal. Trans. Software Engineering 1(2): 192-198, 1975
15. Wirth N.: Algorithms + Data Structures = Programs. Prentice Hall, 1975
16. Mesa Language Manual. Xerox PARC, Oct. 1976.
17. Wirth N.: Modula: A Language for Modular Multiprogramming. Software—Practice & Experience 7(1): 3-35, 1977
18. Wirth N.: The Use of Modula. Software—Practice and Experience 7(1): 37-65, 1977
19. Wirth N.: Design and Implementation of Modula. Software—Practice and Experience 7(1): 67-84, 1977
20. Wirth N.: Towards a Discipline of Real-Time Programming. Comm. ACM 20(8): 577-583, 1977
21. Wirth N.: What Can We Do about the Unnecessary Diversity of Notations for Syntactic Definitions? Comm. ACM 20(11): 822-823, 1977
22. Lampson B.W.: Bravo Manual. Alto User's Handbook. Xerox PARC, 1978
23. Wirth N.: The Personal Computer Lilith. Proc. 5th Intl. Conf. on Software Engineering 1981, 2-16
24. Nori K.V., Amman U., Jensen K., Nägeli H.H., Jacobi Ch.: Pascal-P Implementation Notes. In: Pascal -The Language and its Implementation, Ed. D. W. Barron, John Wiley & Sons, 1981
25. Wirth N.: Programming in Modula-2. Springer-Verlag, 1982
26. Knudsen S.E.: Medos-2 : A Modula-2-Oriented Operating System for the Personal Computer Lilith. Diss ETH 7346, ETH Zürich, 1983
27. Wirth N.: From Programming Language Design to Computer Construction (Turing Award Lecture). Comm. ACM 28(2): 159-164, 1985
28. Gutknecht J.: Concepts of the Text Editor Lara. Comm. ACM 28(9): 942-960, 1985
29. Wirth N.: Algorithms and Data Structures. Prentice Hall, 1986
30. Wirth N.: Microprocessor Architectures: A Comparison Based on Code Generation by Compiler. Comm. ACM 29(10): 978-990, 1986

31. Wirth N.: Hardware Architectures for Programming Languages and Programming Languages for Hardware Architectures. Proc. ASPLOS '87, 2-8
32. Eberle H.: Development and Analysis of a Workstation Computer. Dissertation, ETH Zurich, 1987.
33. Wirth N.: Type Extensions. ACM TOPLAS 10(2): 204-214, 1988
34. Wirth N.: The Programming Language Oberon. Software—Practice and Experience18(7): 671-690, 1988
35. Wirth N., Gutknecht J.: The Oberon System. Software—Practice and Experience19(9): 857-893, 1989
36. Wirth N.: Designing a System from Scratch. Structured Programming 10(1): 10-18, 1989
37. Wirth N.: Ceres-Net: A Low-Cost Computer Network. Software—Practice & Experience 20(1): 13-24, 1990
38. Mössenböck H., Wirth N.: The Programming Language Oberon-2. Structured Programming 12(4): 179-195, 1991
39. Reiser M., Wirth N.: Programming in Oberon: Steps Beyond Pascal and Modula. Addison-Wesley, 1992
40. Wirth N., Gutknecht J.: Project Oberon—The Design of an Operating System and Compiler. Addison-Wesley, 1992
41. Wirth N.: Digital Circuit Design. Springer-Verlag, 1995
42. Wirth N.: A Plea for Lean Software. IEEE Computer 28(2): 64-68, 1995
43. Brandis M., Crelier R., Franz M., Templ J.: The Oberon System Family. Software—Practice and Experience 25(12): 1331-1366, 1995
44. Wirth N.: Theory and Techniques of Compiler Construction. Addison-Wesley, 1996
45. Wirth N.: The Language Lola, FPGAs and PLDs in Teaching Digital Circuit Design. Proc. Ershov Memorial Conference 1996, 2-20
46. Wirth N.: Hardware Compilation: Translating Programs into Circuits. IEEE Computer 31(6): 25-31, 1998
47. Erfindungen der Informatik: Ein Plädoyer für Zuverlässigkeit anstelle von Innovationitis. Abschiedsvorlesung von Niklaus Wirth. Neue Zürcher Zeitung, 29.1.1999