



## Niklaus Wirth: Geek of the Week

02 July 2009

by Richard Morris

Av rating: ★★★★★

Total votes: 66

Total comments: 9

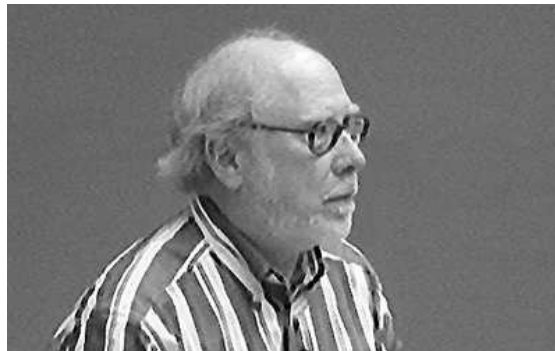
[send to a friend](#)[printer friendly version](#)

It is difficult to begin to estimate the huge extent of the contribution that Niklaus Wirth has made to IT as it exists today. Although now retired for ten years, he remains an abiding influence on the design of computer languages. It is likely that the first structured computer language you ever learned was written by him. He still has fascinating views on contemporary software trends, as Richard Morris found out when he spoke to him.

In 1970 a Swiss mathematician and computer scientist named Niklaus Wirth designed Pascal, possibly the best known, and most influential, programming languages of all time. Originally it was intended as a teaching tool to promote the structured programming style of another leading computing pioneer, Edsger W. Dijkstra, but it quickly transcended its initial remit and became a general-purpose language.

Pascal's simplicity was one of the secrets of its success. Wirth was able to develop a small and elegant Pascal compiler, written in the language itself. It was, at the time, revolutionary.

From the very successful Pascal of the 1970s through to successive languages such as Modula-2 and Oberon, Wirth has built a reputation as an excellent educator who inspires research. He has clear and well thought-out views on computer languages and software development, and he isn't afraid to express them. Even today, his ideas often sound radical.



A true engineer by nature, with interests in both software and hardware, he also influenced the modern computer keyboard after insisting that the design of the PDP - 1 used two extra modifier keys. This computer became famous for being important in the creation of hacker culture at MIT and elsewhere. The vestigial remains of Wirth's keys can be found in the 'alt' keys on PC and 'option' key Apple keyboards.

*As long as programmers cherish their freedom not only to design their own clever software, but also to modify adopted software according to their likings, a proper design discipline remains unlikely. And as long as companies secretly cherish complexity as an effective protection against being copied, there is little hope for dramatic improvements of the state of the art.*

Professor Wirth was born in 1934, and obtained his degree in electrical engineering from the Swiss Federal Institute of Technology (ETH) Zurich in 1959. He received an MSc from Laval University, Quebec, Canada, in 1960 and then studied for his doctorate under the supervision of the pioneering computer designer Harry D. Huskey at the University of California, Berkeley. From 1963 to 1967 he taught as an assistant professor at the newly created computer science department at Stanford University in California and then at the University of Zurich. In 1968 he was appointed full professor of computer science at ETH Zurich. In 1970 he devised the language Pascal, in about 1980 came Modula-2, and in 1988 a language with some object-oriented features named Oberon. He designed the Lilith and Ceres computers, and subsequently became involved with circuit design tools, finally becoming head of the institute of computer systems at ETH in 1990 and retired on April 1 1999.

Over the years, Niklaus Wirth has received numerous honorary doctorates and awards including the 1984 Association for Computing Machinery Turing Award, computer science's nearest equivalent to a Nobel prize, and the 1987 Computer Pioneer Award from the Institute of Electrical and Electronics Engineers' Computer Society.

It is probably the extraordinary feat of designing not one, but three popular computer languages that makes his contribution so exceptional.

He developed his initial version of the Modula programming language as a research exercise, aimed at demonstrating that an operating system for a personal work station could be written entirely in a high-level language. The subsequent and much more widely used Modula-2 language was developed between about 1979 and 1981.

The Oberon project, which contains the most mature fruits of Wirth's design talents, was launched in 1985 by Wirth with Jürg Gutknecht. Oberon had many of the features of Modula-2, but some unwanted aspects were removed and other features were added. The project was originally targeted towards in-house hardware and this drove the language design, but versions of the Oberon language and system were made available for a number of commercial platforms. With Oberon, Wirth has tackled the most difficult feat of all: to create the simplest possible language: Oberon 7 is an intellectual tour de force.

One of his most remarkable projects beside the languages was the design of the Lilith personal workstation, based on the programming language Modula-2. It was influenced by the renowned Alto computer, designed at Xerox Palo Alto Research

### Opinion Home

[Opinion Pieces](#) (129 articles)[Geek of the Week](#) (48 articles)[Tony Davis - Editor](#) (6 articles)[Phil Factor](#) (46 articles)[Damon Armstrong](#) (9 articles)[Douglas Reilly](#) (40 articles)[Adrian Furnham](#) (1 articles)[Claire Brooking](#) (1 articles)[Richard Morris](#) (50 articles)[Tim Gorman](#) (1 articles)[Jesse Liberty](#) (5 articles)[Sarah Blow](#) (1 articles)[Anna Larjomaa](#) (1 articles)[Gaidar Magdanurov](#) (2 articles)[Regular Columnists](#) (1 articles)[Ben Hall](#) (5 articles)[Larry Gonick](#) (21 articles)[Damon Armstrong and Ty Anderson](#)[Matt Lee](#)[Blogs Home](#)

### Simon Sabin Says SQLBits

SQLBits is the largest SQL Server conference in Europe. Because it is held on a Saturday, and is free,... [Read more...](#)

### Level Playing Field

The Federal Government in the States accepts tenders for their IT projects from a wide-range of... [Read more...](#)

### Craig Newmark: Geek of the Week

Occasionally, readers of Simple-Talk will ask quizzically if the 'Geek of the Week' that the editors... [Read more...](#)

### Andrew Tanenbaum: Geek of the Week

Andrew Tanenbaum has had an immense influence on the way that operating systems are designed. He... [Read more...](#)

### Ross Anderson: Geek of the Week

Professor Ross Anderson is one of the foremost experts in Computer Security in the world. He has... [Read more...](#)

### Linus Torvalds, Geek of the Week

Linus Torvalds is remarkable, not only for being the technical genius who wrote Linux, but for then... [Read more...](#)

### Driving up software quality - the role of the tester

Have you ever wondered what a software tester does? Helen Joyce, test engineer at Red Gate software... [Read more...](#)

### Coming Out as a Cancer Survivor - A Guide for Software Developers

A personal perspective on the responsibilities of a cancer-surviving software developer [Read more...](#)

### The Computer that Swore

Database Developers occasionally get crazy ideas into their heads. Phil Factor should know; He... [Read more...](#)

### Bad CaRMA

From hope and euphoria, to desperation, firings and the ultimate demise of a company. Tim Gorman charts... [Read more...](#)

Center in the United States. Although both were technical triumphs, neither resulted in direct commercial success. Niklaus Wirth is primarily remembered as a prolific, expert and successful designer of imperative programming languages and was important in the structured programming movement. His languages have helped in the quest for more understandable programs.

- RM:** "Given that our technological civilization depends on software why is most of it so poor?"
- NW:** "In fact, most of it works correctly most of the time. We have the tendency to consider software for PCs only, and expect it to be (almost) for free. But professional software design is labor-intensive and therefore cannot be cheap."
- RM:** "Do you think better education is the answer to poor software? Surely teaching people better would be cheaper in the long run and certainly avoid the huge bloat we see today and we would be able to use simpler and less power-hungry hardware?"
- NW:** "A proper education certainly would help. However, the belief is wide-spread that programming is easy, can be learned on the job, and does not require specific training or talent. However, it is not programming in the sense of coding that is the problem, but design. Applications have become very demanding and their tasks complex. Mastering this growing complexity is the primary challenge. To tackle this task requires experience and an explicit ability for abstraction on many levels. It is difficult to learn this, and even more so to teach it. Not every programmer is a born designer."
- RM:** "Did you think you would ever design a new language from scratch?"
- NW:** "None of my languages were designed from scratch. Oberon (1988) evolved from Modula-2, Modula-2 (1979) from Pascal and Mesa, Pascal (1970) from Algol-W, and Algol-W from Algol 60. They represent an evolutionary sequence. Each time a new paradigm entered the picture, yet the general structure and syntax were retained. In Algol-W the record and pointer facilities entered the picture, in Pascal the strict application of structuring, not only of programs, but also of data, in Modula-2 it was modularization in combination with type-safe, separate compilation, and in Oberon it was object-orientation in disguise, combined with a rigorous desire to reduce the number of features and facilities, to get rid of bells and whistles.

Whereas the earlier languages were implemented on the basis of an existing environment, Oberon was implemented from scratch. My colleague J.Gutknecht and I programmed the entire system, including the system's kernel, the file and window modules, the display mechanism, the entire text system for document production, and a graphics module for drawing diagrams. The extreme, self-imposed restriction on man-power made concentration on the essentials mandatory, and enforced a homogeneous, integrating design philosophy."

- RM:** "How soon after you created Pascal did you see it start to take over the industry?"
- NW:** "During the first seven years we received many requests from universities for help in implementing Pascal on their specific computers, main-frames in computer centers. To facilitate this activity, we specified a hypothetical stack computer, which could easily be reprogrammed on the computers in question, and whose code was called Pascal-P, the P standing for portable.

This technique turned out to be very successful, and it caused Pascal to spread among universities for teaching programming. However, industry remained aloof. They claimed that retraining their work force to a "non-standard" language would be impossible, and shifting to a "university product" suicidal. The break-through came with the arrival of the micro-computers, cheap machines that entered schools and homes. These computers did not feature large memories. Hence, the Pascal system with its compact interpreter was the ideal vehicle to introduce a high-level language, adequate for teaching programming in a structured fashion.

Pascal did not spread in industry, but rather among people who had not to be retrained from the use of an unstructured, feature-ridden language."

- RM:** "Was the development of Object Pascal a positive thing? Is the result still, recognizably 'Pascal'?"
- NW:** "During my sabbatical at Xerox Research in Palo Alto in 1985, Larry Tesler approached me with his proposal for an object-oriented version of Pascal. He was a member of the team around Alan Kay implementing Smalltalk. I believe it was a valuable contribution to provide the object-oriented paradigm in a language with static typing. It contained several additions, but essentially retained the character and appearance of Pascal."
- RM:** "What of the valuable features of Pascal and Modula-2 do you think were brought into Java and C#? Is there a ghost form of Pascal in these two languages?"
- NW:** "There is more in them than merely a ghost. Most of the statement structures and data structures appear in them, and, I believe, also the important concept of static typing of all constants, variables, and procedures. The similarity is not obvious, as both languages have adopted the syntax of C which gives them a quite different appearance. Its terseness was intended to reduce the amount of typing, which seemed important in the era preceding the mouse."
- RM:** "Oberon for .NET seems an interesting development, because of Oberon's simplicity, particularly in Oberon-7. It seems a great tool for education. Do you see a time when the project be revived?"
- NW:** "I do not know, and I have never been involved in this project. The integration of Oberon in .NET seems to be a problem infinitely more complex than the Oberon compiler itself!"
- RM:** "How do you see the future for procedural programming languages? "
- NW:** "Procedural programming is still the most common paradigm, and it will remain so, because the semantic gap between procedural languages and computers is smaller than for any other paradigm. Instruction sequences are represented by statements, and the state space by variables.

I consider object-oriented languages also as procedural. They emphasize the grouping of related data into objects, and the attachment of methods to objects. The close relationship between object-oriented and procedural views becomes apparent if we relate the respective terminologies: Basically the terms class, object, and method stand for the classical type, variable, and procedure. Instead of calling a procedure, we now send a message."

- RM:** "A number of highly current successful windows tools and applications have been written in Pascal and Delphi, and yet we seem to hear very little of the languages they were written in. Why is this, do you think?"



Over 150,000 Microsoft professionals subscribe to the Simple-Talk technical journal. Join today, it's fast, simple, free and secure.

JOIN SIMPLE TALK!

**NW:** "Once a house stands, nobody asks with which hammers and nails it was built. If the design was successful, the competition need not know which tools were used. They were, after all, part of the success."

**RM:** "I once read that you were not a great fan of C++. Has your opinion changed? "

**NW:** "This is an understatement. As a teacher, I have a great aversion against over boarding complexity. C++ is a language that was designed to cater to everybody's perceived needs. As a result, the language and even more so its implementations have become monstrously complex and bulky, difficult to understand, and likely to contain errors for ever.

Yet, C++ will remain in use, mostly as a software legacy. The more complex an object, the larger the investment in learning to use it, and the greater the resistance to abandon it.

We recently have heard considerable grumbling about the complexity, bulkiness, and lack of reliability of the widely used software systems for PCs and servers, particularly from people who are concerned about their trustworthiness. Being constantly connected to a world-wide network, the systems are vulnerable, be it through vicious attacks or through back doors. In order to protect a system from attacks, all possible cracks must be removed, and in order to eliminate hidden back doors, the entire system must be in the open to the designer.


Proprietary software preclude these possibilities, and therefore the Open Source approach has gained many supporters. I remain skeptical. As it stands now, also the systems available in open source are complex, and who wants to read and comprehend millions of lines of code before using these programs?

Besides all the good things, the open source movement ignores and actually hinders the perception of one of the most important ideas in designing complex systems, namely their partitioning in modules, and their formation as an orderly hierarchy of modules. The key idea is that the designer of a module using (importing) other modules, need not know any of the source code of them. He must rely solely on a clear specification of the interface of these modules. Most of the time, modules lack such clear, complete, and unambiguous interface specifications. And if the complete source is available anyhow, so the story goes, who needs it?

As long as programmers cherish their freedom not only to design their own clever software, but also to modify adopted software according to their likings, a proper design discipline remains unlikely. And as long as companies secretly cherish complexity as an effective protection against being copied, there is little hope for dramatic improvements of the state of the art.

This article has been viewed 6166 times.

**Author profile:** Richard Morris



Richard Morris is a journalist, author and public relations/public affairs consultant. He has written for a number of UK and US newspapers and magazines and has offered strategic advice to numerous tech companies including Digital Island, Sony and several ISPs. He now specialises in social enterprise and is, among other things, a member of the Big Issue Invest advisory board. Big Issue Invest is the leading provider to high-performing social enterprises & has a strong brand name based on its parent company The Big Issue, described by McKinsey & Co as the most well known and trusted social brand in the UK.

Search for other articles by Richard Morris

**Rate this article:** Avg rating:  from a total of 66 votes.

Thank you for rating this article. Due to caching your rating might take up to five minutes to display.

## Have Your Say

Do you have an opinion on this article? Then add your comment below:

You must be logged in to post to this forum

[Click here to log in.](#)

---

**Subject:** Fascinating read  
**Posted by:** Diarmuid (not signed in)  
**Posted on:** Friday, July 03, 2009 at 4:34 AM  
**Message:** Very interesting, some excellent points there. I was thought Pascal in college, then worked as a C++ programmer for several years.  
 "The more complex an object, the larger the investment in learning to use it, and the greater the resistance to abandon it."  
 Too true. I found it difficult to leave behind C++, having finally acheived a certain level of knowledge. Plus I could no longer look down on the non C++ people!  
 I think its hard to leave behind any language though, as its a long time to become "good" at a new one.

---

**Subject:** A Genius.  
**Posted by:** Phil Factor (view profile)  
**Posted on:** Saturday, July 04, 2009 at 8:10 AM  
**Message:** I was programming in P/1 when I first came across Pascal in the seventies. The contrast was enormous and I remember programming in Pascal as a very liberating experience. If one could somehow bottle Niklaus Wirth's great wisdom, it would look like Pascal. The legacy lives on in C#. I loved the article and Niklaus's continuing sagacity.

---

**Subject:** Good  
**Posted by:** Arcko (view profile)  
**Posted on:** Sunday, July 12, 2009 at 9:42 PM  
**Message:** Agree with the opinion of open source

---

**Subject:** Common sense and wisdom  
**Posted by:** Anonymous (not signed in)  
**Posted on:** Monday, July 13, 2009 at 3:52 AM  
**Message:** Inspiring article. His example of clear thought and common sense is what makes great designers.

btw.. "The key idea is that the designer of a module using (importing) other modules..

What was wrong with the word "using"...? Even Niklaus seems to be behind in

IT verb-replacement - how heartening.

---

**Subject:** Good read  
**Posted by:** Jon Gilbert (not signed in)  
**Posted on:** Monday, July 13, 2009 at 5:03 AM  
**Message:** Really nice to read this interview - thanks a lot - and I agree with a lot of NW's (possibly controversial) views :-)

---

**Subject:** Nicklaus Interview  
**Posted by:** Arthur Fuller (not signed in)  
**Posted on:** Monday, July 13, 2009 at 7:09 AM  
**Message:** What a superb interview this was, and what a superb candidate! The only thing one might revise is "Geek of the Week". Surely Nicklaus deserves "Geek of the Decade". I loved his concise puncturing of the OOP parade. When you strip the covers off, all it is is arrays of arrays, with some pointers tossed in.

---

**Subject:** Excellent Insight  
**Posted by:** Anonymous (not signed in)  
**Posted on:** Monday, July 13, 2009 at 11:47 AM  
**Message:** Simplify!

---

**Subject:** Thanks, Richard!  
**Posted by:** Anonymous (not signed in)  
**Posted on:** Thursday, July 16, 2009 at 9:08 AM  
**Message:** Great interview! Good to go back to the roots, to see where we've been! Helps us to keep a better course on where we want to go! ... oh well, back to work ...

---

**Subject:** Programming - just digging a trench... may be the grave?  
**Posted by:** [muet](#) (view profile)  
**Posted on:** Monday, July 20, 2009 at 2:53 AM  
**Message:** What a compassion! Real progress in deed advances much slower than the increase of computing and storage power. To feel the pain enjoy reading <http://www.inf.ethz.ch/personal/wirth/Articles/Miscellaneous/IEEE-Annals.pdf> - If fathers could be adopted: Niklaus Wirth is one of mine!

---

[Site map](#) | [Become an author](#) | [Contact us](#)

[Privacy policy](#) | [Terms and conditions](#) | ©2005-2009 Red Gate Software