Inhärente Ordnung als Prinzip

Ein Gespräch mit dem ACM-Turing-Preisträger Niklaus Wirth

Anfang der 1970er-Jahre entwickelte Niklaus Wirth als Reaktion auf die zunehmende Komplexität der Programmiersprachen die Sprache Pascal, die sich an Hochschulen einer hohen Beliebtheit erfreute. Modula und Oberon folgten. Zudem baute er – Jahre bevor Apple damit erfolgreich war – nach Studienaufenthalten bei Xerox einen Computer mit Maus und GUI an der ETH. Im Gespräch geht er auf die Informatikgeschichte, auf Bildungssysteme und auf aktuelle IT-Entwicklungen ein.

Radomir Novotny

Bulletin SEV/VSE: Vor 14 Jahren haben Sie zwar Ihr ETH-Büro verlassen, sind aber immer noch aktiv und arbeiten zu Hause beispielsweise an einer neuen Oberon-Version. Waren Sie eigentlich kontinuierlich aktiv oder haben Sie Ihren Ruhestand auch zwischendurch genossen?

Niklaus Wirth: Ich war eigentlich immer aktiv, vielleicht nicht so intensiv wie im letzten Jahr. Ich mache meine Arbeit gerne. Es interessiert mich immer noch, meine früheren Sachen aufzubereiten, auf einen neuen Stand zu bringen.

Wie kann man die Leidenschaft so lange bewahren?

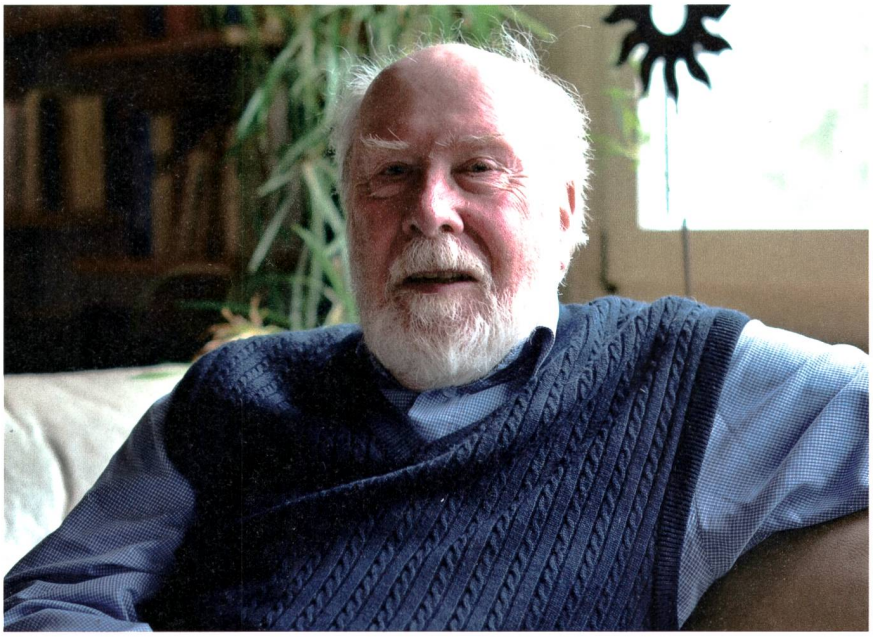
Ich weiss es auch nicht. Neugierde. Von Jugend an war ich interessiert an Technik. Ich wollte immer wissen, wie etwas funktioniert. Der Computer war natürlich Ende der 1950er-Jahre etwas Mysteriöses. Da konnte noch Neuland erschlossen werden.

Sie haben zunächst Elektrotechnik an der ETH studiert. Wie sind Sie zur Informatik gekommen?

Direkt nach dem Studium bin ich mit einem Auswanderungsvisum – nicht mit einem Studentenvisum oder Stipendium – ausgewandert. Ich wollte arbeiten. Dann hat sich herausgestellt, dass es unmöglich war, von hier aus eine Stelle zu finden, und ich bin an eine Universität geraten, an der ich eine Assistenzstelle bekommen habe. Mit 180 Dollar monatlich. Zunächst dachte ich, dass ich mit einem ETH-Diplom das höchste, das man sich erwerben kann, hatte. Etwas, das überall anerkannt ist. Da dem nicht so war, habe ich noch in einem Jahr ein Master-Studium absolviert. Dann hat man mich ermutigt zu doktorieren, woran ich früher nie gedacht hätte. Aber ich habe dann auch gemerkt, dass mich der Ingenieurberuf in der Industrie nicht befriedigt hätte, und ich habe es probiert. Die harten Prüfungen in Berkeley habe ich dann bestanden und machte weiter. Zunächst interessierte mich die medizinische Elektronik, aber da war dort nicht viel los. Danach habe ich mich für Informationstheorie interessiert, aber eher, weil eine Assistenzstelle offen war. Nach einem halben Jahr konnte ich dann zu einem Professor wechseln, der Computer entwickelt hat. Das hat mich fasziniert. So bin ich hineingeraten. Mehr oder weniger per Zufall habe ich dann Kontakt mit einer kleinen Arbeitsgruppe aufgenommen, die im Kellerraum gearbeitet hat. Die haben etwas «Gschpässigs» gemacht: Sie haben an einem Programm gearbeitet, das Programmiersprachen übersetzt – heute nennt man das Compiler. Das war faszinierend, weil das Programm eine Sprache verarbeitet hat, in der es selbst programmiert war. Rekursion, Bootstrapping usw. Dann dachte ich, dies sei ein geeignetes Dissertationsthema, da es neu war. Es war damals eine unglaubliche Bastelei und ich dachte, da könne man etwas Systematik, wissenschaftliche Ordnung hineinbringen.

Sie haben sozusagen die Schweizer Ordnungsliebe ins Programmieren gebracht.

Das ist richtig. Aber ich habe auch gemerkt, dass das recht komplexe Systeme sind – komplexer als Maschinen. Und dass da die einzige Rettung ist, wenn man Ordnung hineinbringt bzw. von Anfang an hat. Das war eigentlich das Thema meiner ganzen Karriere. Software ist ja bekannt dafür, dass sie manchmal versagt. Ich war mir bewusst, dass das ein inhärentes Problem ist, sobald man komplexe Systeme entwickelt. Die Korrektheit muss sozusagen von Anfang an eingeplant werden.



Die Zuverlässigkeit und Einfachheit der Informatik haben bei Niklaus Wirth oberste Priorität.

Diese USA-Zeit hat Ihre Ziele geprägt, Ihren Weg bestimmt.

Absolut.

Wie konnten Sie dies an der ETH einbringen?

Man hat geplant, Informatik an der ETH einzuführen. Damals nannte man es noch Computerwissenschaft. Das hat mich dazu bewogen, 1968 den Ruf an die ETH anzunehmen. Es vergingen dann allerdings 12 oder 13 Jahre, bis ein Lehrgang «Informatik» eingeführt wurde. Konkrete Vorschläge hatten wir schon 1970 eingebracht, aber man ist dann nicht einmal so sehr auf Widerstand gestossen, sondern einfach auf Desinteresse. 1980 kam dann die Meldung an die ETH, nicht an mich, dass gewisse Industriebetriebe Softwareaufträge ins Ausland vergeben müssten, da in der Schweiz zu wenig Kompetenz vorhanden ist. Wörtlich: «Ob wir eigentlich geschlafen hätten?» Das war hart. Weil ein eigener Studiengang aber nicht zustande gekommen ist, konnte ich mehr Zeit für die Forschung, für Programmiersprachen und Compiler, aufwenden. Es hat also auch seine positive Seite gehabt. Bis 1980 waren wir eigentlich nur vier Professoren. Heute sind es über 30.

Sie haben nebst Programmiersprachen auch Hardware entwickelt. Pascal war ja ein durchschlagender Erfolg. Aber von Ihrer Hardware hört man kaum etwas. Wieso diese Diskrepanz?

Ich war bis 1977 im Gebiet der Software verblieben. Dann hatte ich ein Urlaubsjahr und konnte eine glückliche Wahl treffen, denn ich ging ins kalifornische Xerox-Forschungslabor, wo sie die Idee des Personal Computer entwickelt hatten. In meinem kleinen Büro wurde mir eine solche Maschine, die mich absolut fasziniert hat, zur Verfügung gestellt. Das war sehr grosszügig. Ich erkannte, dass dies eine komplett andere Welt ist. Zu Hause hatte man ein Terminal, das mit einem im Keller stehenden Computer, der von vielen anderen genutzt wurde, mittels dünnem Draht verbunden war. Und da hatte ich nun einen, der mir alleine zur Verfügung stand – mit einem hochauflösenden Bildschirm. Heute sage ich, dass das Computerzeitalter erst damit angefangen hat. Mitte der 1970er-Jahre. Die Leute können sich heute nicht vorstellen, wie man damals mit Computern gearbeitet hat – über ein Terminal mit dünnem Draht. Heute ist alles hoch interaktiv. Da mich das sehr fasziniert hat, wollte ich in der Schweiz nicht zum alten System zurück. Ich habe mich sofort darauf kapriziert, so etwas zu entwickeln, und habe ein Labor von Null auf mit relativ primitiven Mitteln aufgebaut. Wir hatten damals keine Hardware-Entwicklung. Der Initialkredit betrug 56 000 CHF! Wir haben zwei Lilith-Prototypen entwickelt. Ein Nachbau des Alto-Computers mit ein wenig modernerer Technologie. Eigentlich ist er zu früh gekommen, denn die Leute haben die Vorteile nicht erkannt. Erst drei Jahre später kam jemand, der es fördern und entwickeln wollte. Es wurde eine Firma gegründet, an der ich mich nicht beteiligen wollte, da ich mit dem Unterricht und der Forschung ausgelastet war. Das hat dann leider nicht gut funktioniert. Es wurden immerhin ein paar 100 solcher Computer hergestellt. Aber es hat an Kunden und an einem wirklichen Businessplan gefehlt.

Apple hat es dann mit einem vergleichbaren Computer besser gemacht.

Ja, aber das, was wir 1980 hier gehabt haben, das hat Apple erst fünf Jahre später auf den Markt gebracht. Wir und unsere Doktoranden hatten den Vorteil, zukunftsträchtige Software auf moderner Hardware entwickeln zu können.

Die ETH ist eine Universität, die vor allem für den Unterricht und für die Forschung zuständig ist, aber nicht für die Entwicklung industrieller Produkte. Die Marktlösungen müssen von den Industrien kommen.

> «Das Computerzeitalter hat erst Mitte der 1970er-Jahre mit der Maus und dem hochauflösenden Bildschirm angefangen.»

Heute versucht die ETH, via Industry Relations Kontakte zur Industrie zu knüpfen. Wie beurteilen Sie dies?

Das ist positiv. Es gibt ja Stellen an der ETH zur Förderung von Spin-Offs. Das ist eine amerikanische Idee, die in den 1990er-Jahren bei uns eingeführt worden ist. Ich finde das sehr gut, aber man muss aufpassen, dass man es nicht übertreibt.

Heute bauen meiner Ansicht nach zu viele Forscher grosse Forscherteams auf und haben dann die Aufgabe, Drittmittel einzutreiben. Der Professor mutiert zum Fundraiser – für die Forschung bleibt weniger Zeit und für den Unterricht gar keine. Das machen dann die Assistenten. Das ist schade. An der ETH haben sie ja ein Budget von fast einer Milliarde. Da sollte es eigentlich nicht nötig sein, Drittmittel einholen zu müssen. Ich hatte immer um die vier von der Schule bezahlte Etat-Stellen. Heute hat jeder Informatikprofessor mindestens ein Dutzend Assistenten und muss dann die Saläre eintreiben können. Sie werden dann mehr oder weniger von den Bedürfnissen, die an sie gestellt werden, gesteuert. Die Industrie gibt zurecht nicht Geld für nichts aus.

Das ist ein Spannungsfeld, in dem man leben muss. Kurzfristige Erfolge stehen dann im Vordergrund.

Die Industrie denkt nicht auf 50 Jahre hinaus. Das wäre eigentlich die Aufgabe der Universitäten. Vielleicht ist 50 übertrieben. Zehn ist schon viel. Ich behaupte, dass eine Entwicklung wie Pascal heute an Universitäten schlechthin unmöglich wäre. Das muss auf längere Zeit geplant werden, und man muss überzeugt davon sein, dass es längerfristig nützlich ist.

Damals in den 1960er-Jahren war die grosse Frage, ob Programmiersprachen überhaupt eine Zukunft haben. Ist es nicht gescheiter, in Assembler zu programmieren? Das ist effizienter, man kann Instruk-

Als Mäuse noch keine Mikroprozessoren benötigten: Schweizer Dépraz-Mäuse (links) aus den 1980er-Jahren treffen auf dem Wirthschen Arbeitstisch auf kalifornische Xerox-Mäuse.

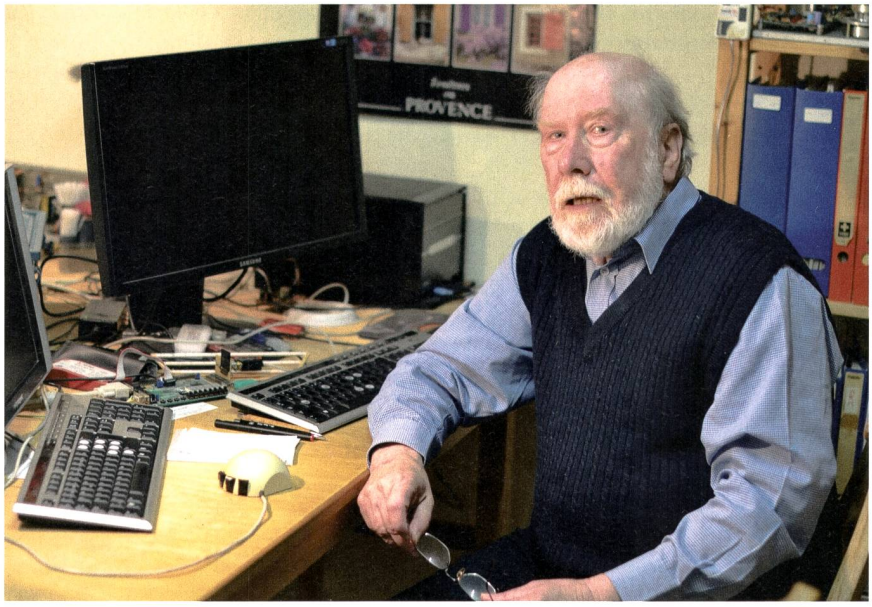
tionen sparen usw. Davon redet schon lange niemand mehr. Aber man musste die fortschreitende Technik abwarten, um zu erkennen, dass sich das «Knübeln» an einzelnen Instruktionen nicht mehr lohnt, sondern dass Zuverlässigkeit und Korrektheit viel wichtiger sind. Um dies mit Assemblercode hinzukriegen – das ist der falsche Weg. Das ist mit modernen Programmiersprachen schwierig genug.

Apropos Programmiersprachen: Stellen Sie eine örtliche Konzentration gewisser Sprachen fest? Beispielsweise dass Lisp eher in den USA verbreitet ist?

Das gibt es schon. Aber heute dominiert das, was von den grossen Firmen kommt – C# von Microsoft, Apple, Linux, Android, Java, Go. Eine funktionale Programmiersprache wie Lisp ist natürlich etwas anderes – eine andere Art, um über Algorithmen nachzudenken. Lisp stand am Anfang: 1962, McCarthy, MIT, Stanford. Es gibt auch heute noch Personen, die auf funktionale Sprachen eingeschworen sind. Das Zentrum liegt aber nicht in Kalifornien, sondern in England und Schottland. Da geschahen die wesentlichen Weiterentwicklungen. Die funktionale Programmierung hat schon attraktive Seiten, und ich versuche auch, sie in meine Art der prozeduralen Programmierung einzubringen. Rein funktionale Programmierung hat es sowieso eigentlich nie gegeben, denn der Computer ist eine Maschine mit Zuständen. Funktionen haben keine Zustände. Man muss da immer ein wenig «betrügen».

Heute wird natürlich alles von den kommerziellen Sprachen dominiert. Leider – ich sage es jetzt ganz offen – muss ich das bedauern, denn die Sprachen sind zwar sehr mächtig, aber man kann Abstraktionen sehr leicht durchbrechen und damit Unsicherheiten einführen wie damals bei der Assemblersprache.

Überhaupt hat sich das Programmieren gewandelt. Es gibt nur eine Minderheit von Programmierern, die neue Algorithmen entwickeln. Meist geht man heute in Libraries und sucht sich seine Programmteile. Die Idee ist gut, aber es kommt darauf an, wie gut die Teile und ihre Schnittstellen programmiert sind. Oft geschieht, dass man dann Systeme zusammenstellt, von denen man 90 % meist gar nicht braucht. Wenn man kompakte Systeme machen will, sollte man sie von Grund auf aufbauen, statt sich in Libraries zu bedienen.



Auch heute ist Niklaus Wirth noch aktiv: Da der Mikroprozessor für seinen neuen Oberon-Rechner nicht mehr verfügbar ist, hat er einen FPGA-Prozessor (auf Leiterplatte zwischen den Tastaturen sichtbar) als Ersatz enwickelt.

> «Die funktionale Programmierung hat schon attraktive Seiten, und ich versuche auch, sie in meine Art der prozeduralen Programmierung einzubringen.»

Dann würden Sie also auf Libraries verzichten?

Ich mache das, aber ich bin da ein spezieller Fall.

Aber in der Industrie braucht man Sprachen, die alle Bedürfnisse abdecken.

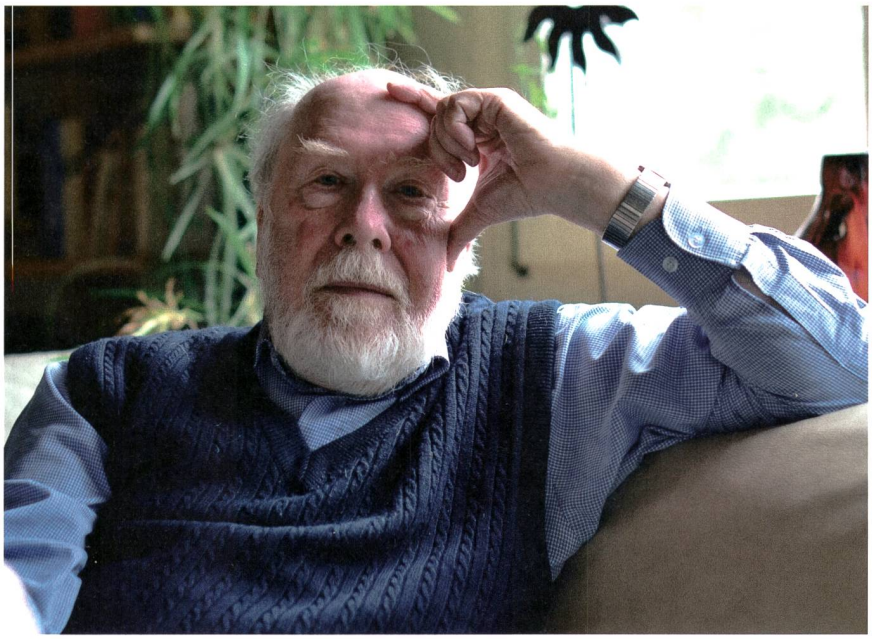
Erstens das, zweitens braucht man Sprachen, die andere auch benutzen – man will sich ja absichern – und drittens kommt es nicht so sehr darauf an, dass sie optimiert und möglichst wenig Ressourcen brauchen und effizient sind, sondern dass das Resultat möglichst schnell realisierbar ist. Time-to-market lautet das Schlagwort.

Heute sieht man auch Tendenzen, bei denen Nachhaltigkeit eine grössere Rolle spielt. Beurteilungskriterien für die Nachhaltigkeit von Software werden entwickelt.

Und wie gut sie erweiterbar ist – das ist auch ein Faktor. Wir haben in der Informatik aber eine gewaltige Hypothek, denn man hat alles mit Werkzeugen aufgebaut, die nicht ideal sind. Das wird man heute nicht mehr los. Benutzer wollen nichts Neues, das nicht kompatibel mit dem Alten ist. Es muss so weitergehen, wie man es gehabt hat. Schon besser, aber die alten Sachen müssen weiterlaufen können.

Sie haben nach Pascal noch Modula 2 und Oberon entwickelt. Wo wird Oberon heute eingesetzt?

Zuerst muss ich noch sagen, dass Modula eine Weiterentwicklung von Pascal ist und Oberon von Modula. Die drei Sprachen bilden eine Familie im gleichen Stil. Pascal hat das strukturierte Programmieren eingeführt, Modula hat die modulare Programmierung ermöglicht, Oberon die objektorientierte. Oberon ist einfacher, aber mächtiger als Pascal. Es sind eigentlich nur Einzelkämpfer, die das Oberon einsetzen, aber es gibt z.B. eine wachsende Verbreitung in Russland. Da wird es an Universitäten und Begabtenschulen gefördert, die den Wert der Systematik und der Einfachheit sehen. Man kann sich dann auf die Sache konzentrieren und muss keine Nebensächlichkeiten mitschleppen. Ich denke, der Zustrom wächst wieder langsam, ist aber klein im Vergleich zur Industrie. Es ist schade, wenn man sich an Universitäten nicht auf solche Werkzeuge konzentriert, weil Grundideen herauskristallisiert werden können. Dazutun kann man später immer noch.



Bilder: No

Niklaus Wirth rückt das Ressourcenbewusstsein in den Vordergrund.

Wieso gerade Russland?

Gute Frage, ich habe darauf keine Antwort. Letztlich hängt es von einzelnen Leuten ab. In der Sowjetunion hatten sie ein ausgezeichnetes Bildungssystem. Das wollte man ja bei uns nie wahrhaben. Ein Bildungssystem, das besonders die Begabten gefördert hat. Die Professoren hatten mehr Musse und den Drang, das Wissenschaftliche sauber einzusetzen. Dadurch hat auch Oberon eine Basis, eine Plattform an Interessenten. Ich war 1990 das erste Mal in Russland, wo ich das akademische Informatik-Zentrum bei Novosibirsk besucht habe. Das war für mich sehr interessant. Es hat meine Augen dafür geöffnet, dass es da nicht nur ein politisches System gab, sondern das da Menschen leben wie wir. Dort hat eine Gruppe von jungen Leuten den Lilith-Computer, den ich vor 1980 entworfen habe, aufgegriffen und einen eigenen Computer mit eigener Elektronik gebaut. Das war schon eindrücklich. Allerdings haben sie die wichtigsten Sachen ausgelassen: Die Maus und den hochauflösenden Bildschirm. Sie haben sich auf die Hardware-Entwicklung und die Implementierung von Modula spezialisiert.

Fehlten die technischen Mittel?

Ja. Sie haben mir auch die Maus, die sie entwickelt haben, gezeigt. Ein unmögliches Ding, ein grosses Monstrum, völlig unpraktisch. Auch einen geeigneten Bildschirm hatten sie nicht. Alle hochtechnischen Entwicklungen waren in der Sowjetunion dem Militär vorbehalten. Das hat sich jetzt natürlich geändert, aber bis 1990 war das normal. Sie hatten auch das Geld nicht, und der Zugang zum Westen war verschlossen.

Ich kenne einen theoretischen Physiker in Moskau, der Oberon aktiv verbreitet. Er gibt in Gymnasien Kurse. Seine Spezialität ist die Neutrinophysik. Er hat gesagt, dass er gewisse Systeme mit Oberon programmieren konnte, die er mit anderen Sprachen nicht geschafft hat. Mit diesen Programmen haben sie wesentliche Resultate über Neutrinos erarbeitet. Eigentlich sei dies ein wesentlich wichtigerer Beitrag als die Entdeckung des Higgs-Teilchens, das sich grosser medialer Aufmerksamkeit erfreut. Für das Higgs-Teilchen hätten wir ja in den letzten zehn Jahren schon so viele Daten gesammelt, dass es praktisch sicher gewesen ist, dass die Theorie stimmt. Nun haben wir noch die letzte Bestätigung. Bei den Neutrinos konnte hingegen etwas fundamental Neues berechnet werden. Oberon hat also doch einen Beitrag geleistet…

Spüren Sie ein wachsendes Interesse an der Informatik- und Computergeschichte?

In letzter Zeit schon, aber es ist keine grosse Welle. Man merkt es, weil früher gar kein Interesse vorhanden war. Es waren schon interessante Jahre. Das Computerzeitalter hat aber, wie schon gesagt, 1975 angefangen, nicht mit Babbage und Boole.

Kürzlich wurde Zuse und dann Turing gefeiert, sozusagen als Fixsterne des Informatikhimmels. Wie stufen Sie ihre Bedeutung ein?

Zuse war ein beachtlicher Pionier, vor allem weil er die Sache in völliger Isolation entwickelt hat. Die übrige Welt hat gar nicht registriert, was er geleistet hat. Immerhin hat er es geschafft, eine Firma zu gründen. Zuse-Maschinen wurden kommerziell hergestellt und in der Wirtschaft in kleiner Zahl verwendet. Dann wurde er von den amerikanischen Produkten überrollt. Bezüglich Turing habe ich kürzlich ein sehr interessantes Buch gelesen. Er ist auch eine Randfigur, ein Theoretiker. Er hat grundlegend begriffen, dass die Idee von Programmen, die sich selber modifizieren, neue Aspekte eröffnen könnte. Für ihn war das Thema «Artificial Intelligence» wichtig. Aber ich würde sagen, dass in der heutigen Zeit wenig vorhanden ist, das auf ihn zurückgeht. Gut, das kann man bei Zuse auch sagen, obwohl Zuse auch technische Beiträge geleistet hat. Die Turing-Maschine ist ein theoretisches Konstrukt, das man nie brauchen konnte, ausser um zu beweisen, dass gewisse Sachen nicht berechenbar sind.

Im zweiten Weltkrieg setzte er sich dann aber praktisch ein.

Dafür wurde er bekannt, aber das hat nichts mit der Entwicklung der IT zu tun.

Welche Personen würden Sie eher ins Rampenlicht rücken?

Ganz sicher John von Neumann und Leute aus seinem Kreis. Er hat die meisten Beiträge geleistet. Übrigens: In dem Buch, das ich bezüglich Turing erwähnt habe, ist von Neumann prominenter als Turing selbst. Es ist mir durch das Buch auch bewusst geworden, wie stark die Entwicklung der Computer mit der Entwicklung der Atombombe verbunden war. Computer sind sozusagen im Schlepptau der Atombombe entstanden.

Von Neumann hat die Architektur heutiger Computer geprägt. Könnte man sich heute eine andere, effizientere Architektur vorstellen.

Die wesentliche Idee von Neumanns ist erstens die Idee, dass Programm und Daten denselben Speicher verwenden, wodurch der Computer seine Programme selbst verändern kann, und zweitens, dass ein intrinsischer Mechanismus vorhanden ist, der Instruktionen sequenziell ausführt, wodurch viel weniger Hardware benötigt wird, da in jedem Schritt immer wieder die gleiche

Hardware verwendet wird. Das ist auch heute noch ein fundamentales Prinzip, aber heute ist die Hardware so billig, dass es attraktiv ist, viele von-Neumann-Kerne zu verwenden, die dann miteinander kooperieren. In dem Sinn weicht man durch die Parallelisierung schon vom reinen Single-Prozessor von Neumann ab. Aber die Schwierigkeiten sind natürlich enorm. Ein Prozessor, der Millionen von Instruktionen absolut fehlerlos durchführt, ist schon eine Herausforderung. Bei mehreren Prozessoren ist es noch schwieriger. Eigentlich ist es ein Wunder, dass es so gut läuft und dass die Hardware so zuverlässig ist. Denken Sie: Milliarden von Instruktionen. Wenn eine versagt, geht alles unter. Das ist ein absurdes Konzept. Aber es funktioniert.

Was ist für Sie das Überraschendste an der IT-Welt der letzten Jahre?

Das Ausmass der explosionsartigen Ausbreitung. Das wurde durch die Verheiratung der Computerwelt mit der Kommunikationswelt ermöglicht, die nicht in den letzten paar Jahren stattgefunden hat, sondern um 1980 begonnen hat. Die meisten heutigen Computer kann man in der Hand halten und sie kommunizieren mit anderen Computern. Kommunikation ist total integriert. Das hatte man in den 1970er-Jahren noch nicht so vorausgesehen.

Was bedauern Sie an der IT-Welt?

Dass sie so anfällig ist auf böswillige Störungen. Viren kannte man vor 1990 nicht. Im Prinzip könnte man sie einfach unterbinden, indem man sagt, man könne keine Programme laden, sondern nur Daten. Daten kann man nicht ausführen. Das ist eine so grosse Restriktion, dass sie heute niemand akzeptiert. Heutzutage will man Programme, Programmteile und Programmobjekte importieren, die dann direkt ausgeführt werden. All die Bilder, die sich bewegen, sind heute unverzichtbar, aber der Preis dafür muss bezahlt werden. Heute teilt man den Computer in jeder Sekunde mit der übrigen Welt. Bei meinem Rechner, den ich selber entwickle, habe ich keinen Netzanschluss. Wenn etwas schiefgeht, weiss ich, dass es mein Fehler ist.

Was halten Sie von der Open-Source-Bewegung?

Das kann ich nicht in einem Satz beantworten. An den Universitäten haben wir seit eh und je Open Source gemacht. Wir haben von Anfang an Pascal-Compiler ausgeliefert. Es ist die Grundidee der akademischen Welt, dass man Gedanken und Informationsmittel austauscht. Dass dies aber nun in der kommerziellen Welt stattfindet, ist ein Novum. Open Source lebt von vielen Volontären. Das Problem scheint mir, dass es nicht klar ist, wer wofür verantwortlich ist, wenn etwas nicht mehr stimmt. Aber bis jetzt hat es offenbar funktioniert. Viele arbeiten mit Linux statt mit Windows. Ich finde es eine durchaus positive Entwicklung.

> «Open Source ist eigentlich nicht nötig, es genügen geschlossene Module mit wohldefinierten Schnittstellen.»

Idealerweise sollte man mit Modulen arbeiten können, auf die man aufbauen kann. Die Grundidee der Module ist, dass man nicht wissen muss, wie diese organisiert sind. Man stützt sich auf die bekannte Schnittstelle ab. Open Source ist eigentlich nicht nötig, es genügen geschlossene Module mit wohldefinierten Schnittstellen. Dann muss man nicht wissen, wie es drinnen aufgebaut ist. Man muss es auch nicht modifizieren können. Die Verantwortung bleibt beim Modulentwickler. Ich bin auch heute noch davon überzeugt, dass so zukünftige zuverlässige Systeme aufgebaut sein müssten. Open Source – heute ja, aber eigentlich verletzen sie ein grundlegend wichtiges Prinzip.

Was freut Sie an heutigen IT-Entwicklungen?

Wenig... Heute ist die Hardware so billig. Wir haben noch mit Systemen im Kilobyte-Bereich gearbeitet. Lilith hatte 64 kB. Das war damals viel. Der erste PDP-11, als ich hier ans Institut kam, hatte 16 kB. Man kann heute mit Ressourcen ganz anders umgehen, und leider auch verschwenderischer. Die «Abundance of Hardware» hat auf die Disziplin der Softwareentwicklung einen negativen Einfluss. Es gibt durchaus Gebiete, die diese Hardware brauchen – man denke an die grossen Systeme zur Berechnung von Physikdaten, die in Gigabyte-Quantität anfallen, oder an die Wetterprognosen und Fahrplansysteme. Aber heute werden mit grossem Aufwand PC in Prospekten, die ins Haus flattern, angepriesen, die 64 GB haben. Da wird man dazu aufgefordert, mit der Hardware verschwenderisch umzugehen. Und das ist nicht nur in der IT schlecht, auch sonst.

Wie finden Sie Ansätze wie beispielsweise den Raspberry Pi, diesen kreditkartengrossen Rechner, mit dem sich Studierende hardwarenah auseinandersetzen können?

Das ist eine interessante Entwicklung, vor allem im Bereich des Unterrichts. Was ich mache, ist das gleiche, abgesehen davon, dass ich einen selbstgebauten Prozessor statt eines kommerziell erhältlichen einsetze und statt einer riesigen Sprache wie Java oder C# mein Oberon darauf laufen lasse. Das ist alles auf die «Educational Usage» ausgerichtet, bei der das ganze System und neuerdings auch die Hardware in einem Buch detailliert beschrieben ist.

Heute ist aus meiner Sicht problematisch, dass die Leute nicht mehr wissen, wie ein Computer funktioniert. Das war bis zu einem gewissen Grad immer so, aber heute hat sich dies verschärft. Selbst manche sogenannte IT-Spezialisten haben keine Ahnung, welcher Prozessor drin ist, wie er aufgebaut ist, wie man ihn programmiert. Selbst Fachleute werden zunehmend abhängig von dem, was geliefert wird.

Wenn Ihr neues Oberon-Buch veröffentlicht wird, möchten Sie es kommerzialisieren?

Nein. Sie haben Open Source erwähnt, und ich habe gesagt, wir haben schon immer mit Open Source gearbeitet. So wird es auch da sein. Wenn das Buch fertig ist, werde ich es im Internet gratis zur Verfügung stellen. Es wäre schön, wenn das Buch auch in Buchform erscheinen könnte.

Das ist die heutige Frage, eBook oder gedrucktes Buch.

Es hat beides seine Vorteile, aber ich habe ganz gerne etwas in der Hand. Dass es im Internet auch noch ist, ist okay. Ist manchmal sogar sehr vorteilhaft. Aber ich hätte das System, das ich seit einem Jahr entwickle, nicht ohne ein Hardcopy-Buch auf dem Schreibtisch entwickeln wollen.

Herzlichen Dank für das Gespräch.