wdv-notes 196
Stand: 1.MRZ.1993 (3.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

Niklaus Wirth über die Zukunft offener Systeme.

© 1991–1993 Edited by *Karl-Heinz Dittberner*

Theo

Grotesk und chaotisch ist die derzeitige Entwicklung von Computer-Systemen zu „offenen" Systemen, die den direkten Austausch von Software ermöglichen sollen. Jeder Hersteller erklärt sein System zum Standard, fügt noch etwas POSIX hinzu und läßt dann die Anwender mit Halbfertigem allein. – In dieser Situation versucht der Informatik-Professor Wirth Kurs zu halten und die vielen (falschen) Maßstäbe wieder zurechtzurücken. Der Unix-Fachzeitschrift »iX« gab er im Mai 1991 ein Interview, das wegen der sehr grundsätzlichen Bedeutung im folgenden vollständig wiedergegeben wird.

Einfache Lösungen

Niklaus Wirth über die Zukunft offener Systeme und die Perspektiven des Software-Engineering

Komplexe Programmieraufgaben können nach Einschätzung von Niklaus Wirth, seit 1968 Professor für Informatik an der Eidgenössischen Technischen Hochschule Zürich, nur mit Hilfe einfacher Instrumente befriedigend gelöst werden. Mit der Entwicklung von Pascal und der Konzipierung von Modula-2 stellte Wirth entsprechende Werkzeuge dafür auf der Ebene der Programmiersprache bereit.

Der in seinem neuen Forschungsprojekt verfolgte Ansatz ist radikaler: Hardware, Betriebssystem samt Benutzeroberfläche und Hochsprache bilden beim Oberon-System ein integriertes Programmiersystem.
Mit iX-Redakteur Peter Welchering sprach 1991 der Altmeister strukturierter Programmierung über wichtige Trends für die Softwareentwicklung.

iX: Propagiert wird bereits seit längerem, der Trend der Zeit gehe in Richtung der Multiuser- und Multitaskingsysteme. Unix-Derivate für Mehrprozessoranlagen erfreuen sich geradezu unerhörten Interesses. Mit dem Oberon-Projekt wenden Sie sich aber unmißverständlich gegen diese Entwicklung. Ist das nicht ein Anachronismus?

Niklaus Wirth: Man muß sich genau überlegen, wozu man Multiprocessingsysteme eigentlich braucht. Es gibt durchaus sinnvolle Anwendungen dafür – ich müßte da einmal längere Zeit überlegen. Wir haben gezeigt, daß sich ganz einfach viele Komplikationen vermeiden lassen, wenn man sich hier beschränkt. In gewisser Weise ist Oberon sogar ein Multitaskingsystem, denn mit jedem Dokument, das eröffnet wird, ist auch ein Task aktiv. Aber Oberon ist eben kein Multitaskingsystem im herkömmlichen Sinn, und erst recht kein Multiprocessingsystem.

iX: Wo liegen die Unterschiede?

Niklaus Wirth: Das Multitasking wird konventionell nur verwendet, weil lauter eigenständige Applikationen vorliegen, die nicht integriert sind. Mit diesem Multitaskingkonzept soll der Effekt erreicht werden, mit mehreren Applikationen gleichzeitig arbeiten zu können. Daß Multitasking überhaupt benötigt wird, ist in einer Fehlkonstruktion der zugrundeliegenden Architektur begründet. Wenn man lauter abgeschlossene Programme hat, muß man sie eben nebeneinander ablaufen lassen. In Oberon ist das nicht nötig, denn hier ist alles integriert. Oberon ist so konzipiert, daß ein Command sämtliche Ressourcen der Maschine für sich hat, solange es arbeitet.

Wir brauchen hier den komplexen Task-Process-Mechanismus nicht, der etwa grundlegend ist für Unix. Unix ist eigentlich immer noch ein Batch-Prozeß-System. Verschiedene kleine Programme kommunizieren über Pipes miteinander, und Pipes sind im Prinzip Files. Oberon

> *Unix ist eigentlich immer noch ein Batch-Prozeß-System.*

weist Kommandos auf, die über höhere Datenstrukturen miteinander korrespondieren, und zwar sind dies wesentlich mächtigere Datenstrukturen als in Unix, wo alles nur linear ist. Oberon kennt das Konzept des Programms nicht mehr. Auf der einen Seite haben wir einfach *load modules*, die residenten Code bezeichnen, auf der anderen Seite die Kommandos, die zu jedem beliebigen Zeitpunkt aktiviert werden können.

Eines muß man ganz klar sehen: Unix ist ein System aus den sechziger Jahren, und es ist konzeptionell uralt. Natürlich wurde es mit allerhand Zusätzen verschönert – bis hin zum X-Window System, das auch schon wieder so ein riesiges Paket ist. Beim direkten Vergleich von X-Window mit unserem Windowsystem ergibt sich eine Größenordnungsunterschied um den Faktor 100. Und ich behaupte: Funktionell können wir das, was das X-Window System macht, im wesentlichen auch machen. Wir erreichen hier sehr viel mehr mit geringerem Aufwand, weil die Grundkonzepte geeigneter sind.

iX: Kenneth Thompson und Dennis Ritchie sind aber in der Entwicklungsphase von Unix doch nicht gerade verschwenderisch mit den wenigen Ressourcen, die sie hatten, umgegangen.

Niklaus Wirth: Das waren schon ganz sparsame Leute. Spannend ist, was in der Folgezeit geschah. Da hat niemand gewagt, die ganze Sache neu zu überdenken. Unix trägt immer noch die Eierschalen des Batch-Betriebssystems für Großrechner mit sich herum. Das ist eben nicht das geeignete Konzept für moderne Arbeitsplatzrechner. Ich sehe

> *Unix trägt immer noch die Eierschalen des Batch-Betriebssystems für Großrechner mit sich herum.*

natürlich schon, daß Unix in Verwaltungen, Firmen oder Banken sehr verbreitet ist. Die haben dort ihre Systeme und Computer auch nicht einfach aus Freude an schönen Konzepten, sondern weil eine bestimmte Arbeit damit getan werden muß. Wie wir das bessere System in die Praxis einführen können, weiß ich noch nicht so genau.

iX: Wo sehen Sie überhaupt eine notwendige Bedingung für die Einführung von Oberon? Speicherplatz kostet nicht mehr so viel, daß man zu übertriebener Sparsamkeit neigen müßte.

Niklaus Wirth: Für die Hardware gilt das gleiche wie für die Software: Man beschränke sich auf das Notwendige. Ich bin von den Möglichkeiten der heute verfügbaren Hardware sehr begeistert. Die erhältlichen Leistungen sind phantastisch.

Es gibt Anwendungen, bei denen wir die Leistungen wirklich brauchen. Der größte Teil der verfügbaren Rechenleistung wird jedoch regelrecht verschwendet, indem man einfach inkompetent konzipierte Software implementiert und dann auch noch mit der Schnelligkeit der Hardware argumentiert, so daß den Kunden die Softwaremängel gar nicht auffallen.

> *Die Hardware wird langsamer schneller als die Software langsamer wird.*

Hier schlägt das Reisersche Gesetz zu: Die Hardware wird langsamer schneller als die Software langsamer wird. Das ist – leider Gottes – eine Tatsache. Und das müßte wirklich nicht sein.

iX: Ihre Vorbehalte gegenüber C liegen auch auf dieser Ebene.

Niklaus Wirth: Eine Sprache an sich ist nicht effizient oder ineffizient. Sie ist allenfalls effizient implementiert – oder eben nicht. Ich denke, daß C zu schnellem Programmieren verleitet. Natürlich weiß ich auch, daß man mit der schönsten Programmiersprache scheußliche Programme konzipieren kann. Da gibt es

Mit C ist es weitaus schlimmer: Man kann einfach alles damit machen, wenn man gerissen ist.

leider so viele Beispiele. Mit C ist es weitaus es weitaus schlimmer: man kann einfach alles damit machen, wenn man gerissen ist. Da wird dann herumgetrickst, ganz schrecklich ist das. Und so etwas fasziniert die Leute auch noch. Ich habe gar nichts dagegen, wenn Heimcomputerbenutzer und Bastler ihre Freude daran haben. Das ist deren Privatsache. Aber wenn das in die Industrie und in die professionelle Softwareentwicklung geht, dann ist das doch kein Zustand!

iX: Also ein C-Programmierverbot für professionelle Softwareentwickler?

Niklaus Wirth: Es soll jeder seinen Spaß am Programmieren mit C haben. Das muß betont werden. Wo es um wirklich wichtige Programme geht, sollten wir besser mit einer Neuorientierung bei den grundlegenden Programmierwerkzeugen beginnen. Sparen wir doch besser das Geld, das wir für Softwarekongresse ausgeben, damit unsere Entwickler sich irgendwelche Heilsverkündigungen anhören.
Privat kann jeder soviel mit C programmieren oder Videos ansehen, wie er mag. In industriellen Zusammenhängen muß das anders bewertet werden. Wenn dann jemand kommt und klagt, daß

Sparen wir doch besser das Geld, das wir für Softwarekongresse ausgeben, damit unsere Entwickler sich irgendwelche Heilsverkündigungen anhören.

Verluste gemacht werden, das Geschäft in die Krise gerät, die Software nicht zuverlässig ist, und dennoch an C festhalten will, das ärgert mich. Das ist geradewegs so, als käme jemand zum Arzt und würde um Hilfe bitten, aber gleichzeitig auch darum bitten, der Arzt solle ihm die Krankheit lassen, weil er sie so gern hat.

iX: Will Professor Wirth diese Leute kurieren?

Niklaus Wirth: Ich will ihnen aufzeigen, daß bestimmte Probleme und Schwierigkeiten aus systematischen Gründen immer wieder auftauchen. Und ich plädiere für eine professionelle Programmentwicklung.

iX: Professionalität scheint bei Ihnen eine Menge mit Objektorientiertheit und der Devise „Small is beautiful“ zu tun zu haben.

Niklaus Wirth: Das schließt zumindest einander nicht aus, auch wenn Professionalität damit bei weitem nicht hinlänglich beschrieben ist. Es hat in der Tat mit dem Oberon-Projekt zu tun. Allerdings würde ich den zweiten Punkt etwas abändern. Ich finde nämlich, daß gilt: *Small is manageable*. Wir sollten uns nie dazu versteigern, komplexe Systeme zu bauen oder gar zu verkaufen, die wir selber nicht wirklich im Griff haben. Die meisten Softwarehersteller sind nicht nur dazu gezwungen, an ihre Grenzen zu

Privat kann jeder soviel C programmieren oder Videos ansehen, wie er mag.

gehen, oftmals müssen sie sogar darüber hinaus tätig werden. Und sie gehen darüber hinaus, weil sonst der Auftrag an die Konkurrenz gegeben wird. Das ist nicht professionell.

Die Objektorientierung ist nun ein anderes Kapitel. Das ist einfach eine Technik, die dort einzusetzen ist, wo wir erweiterbare Systeme brauchen, wo wir neue Funktionen, Objektarten oder Datentypen auf bestehende aufsetzen wollen, ohne alles neu bauen zu müssen. Dafür ist *Objektorientierung* zwar kein guter Name, aber eine gute Technik.

iX: Bei allem Vorbehalt beispielsweise gegenüber Unix und verwandten Systemen ist Oberon aber doch auch ein offenes System.

Niklaus Wirth: Mit Oberon gibt es endlich ein wirklich offenes System. Vor allen Dingen ist es ein stabiles System, das nicht übermäßig komplex ist. Einige Systeme drohen, an ihrer Komplexität zusammenzubrechen. Das nenne ich kein seriöses Software-Engineering.
Ich will damit nicht behaupten, wir hätten bei der Entwicklung von Oberon keine Fehler gemacht. Das wäre vermessen. Aber das System ist so konzipiert, daß man es überblicken kann.

iX: Sie scheinen mit ausgeprägt moralischen Vorstellungen an dieses Projekt herangegangen zu sein.

Niklaus Wirth: Ja, das kann man so sagen. Ich fordere eine bestimmte Haltung des Ingenieurs seinem Ziel gegenüber. Er muß sich und seine Grenzen richtig einschätzen und darf seinen Kunden diese Grenzen nicht verschweigen. Das ist keine hochgestochene Forderung, sondern nur die Ehrlichkeit.

iX: Mit komplexen Systemen haben wir durchaus zu kämpfen. Manchmal kommt sogar die Befürchtung auf, sie seien nicht beherrschbar. Liegen die Gründe dafür in den unrealistischen Versprechungen der Programmierer?

Niklaus Wirth: Gewiß, aber sie werden gezwungen, mehr zu versprechen, als sie halten können, um wirtschaftlich überleben zu können. Eine der wichtigsten Haltungen des Informatikers muß darin bestehen, täglich aktiv im Kampf gegen Komplexität mitzuwirken – und zwar gegen die Komplexität, die er selber verursacht. Ein guter Programmierer muß eben immer noch einmal zusätzlich überlegen, ob er eine Aufgabe nicht noch einfacher, besser und systematischer lösen kann.

Paradoxerweise sind viele Menschen von Komplexität fasziniert und vermuten in Komplexität eine inhärente Intelligenz. Dabei sollte man doch stets nach simpleren Lösungen suchen, die immer weniger Ressourcen in Anspruch nehmen. Es ist ein Grundprinzip jeder Ingenieurstätigkeit, das Verhältnis von Ausbeute und Aufwand zu maximieren. Das gilt für Elektromotoren genauso wie für Computerprogramme. Als Leitsatz für die

Man ist versucht zu sagen: Unix is as complicated as possible.

gesamte Oberon-Entwicklung eignet sich – denke ich – eine Bemerkung Einsteins ganz gut: „Make it as simple as possible, but not simpler“.

In der prozeduralen Programmierung sollte man es so einfach wie möglich machen. Dazu kommt dann noch ein Aspekt, den man Objektorientierung nennen kann. Um noch einmal auf Unix zurückzukommen: Unix ist eben nicht „as simple as possible“. Man ist vielmehr versucht zu sagen: Unix is as complicated as possible. (pw)

Anmerkung des Editors: *Oberon ist – als offenes System – für verschiedene Computer (z. B. für den PC und den Macintosh) frei vom Internet-Knoten »neptune.ethz.ch« der ETH Zürich erhältlich. In den wdv-notes Nr. 160 sind Hinweise zur offiziellen Dokumentation des Oberon-Systems zu finden.*