c't 5/02

Harald Bögeholz, Andreas Stiller

Der Perfektionist

Donald E. Knuth über MMIX und die Kunst des Programmierens

1 000 000 Jahre alt wurde die Informatik-Legende Donald E. Knuth in diesem Jahr - im Binärsystem, versteht sich. Wir sprachen mit ihm über sein Lebenswerk.

Mit 24 Jahren begann Donald E. Knuth ein Buch über die Grundlagen der Informatik zu schreiben. Schnell war klar, dass das Thema 'The Art of Computer Programming' (TAOCP) doch eher Material für fünf Bände umfasst. Drei sind inzwischen erschienen und gelten als Standardwerk der Informatik, aus dem vierten werden wohl drei Teilbände, die Knuth nach derzeitiger Schätzung 2007 fertig haben will. Nicht nur, um die Wartezeit zu verkürzen, sondern auch, um die Gelegenheit zu haben, Feedback von den Lesern in die Endfassung einzuarbeiten, will er Band 4 Zug um Zug in kleinen Portionen mit jeweils ungefähr 128 Seiten als so genannte Faszikel veröffentlichen. Beta-Versionen der ersten Faszikel stellt er für Interessierte bereits auf seiner Homepage zum Download bereit [**1[1]**].

Die von Knuth erfundene Prozessorarchitektur MMIX stellen wir auf Seite 184 vor. Anlässlich des 'MMIXfest' im Oktober 2001 trafen wir Knuth an der Fachhochschule München.

c't: Herr Knuth, Sie haben einmal geschätzt, dass der erste Faszikel von Band 4 um 1995, '96 herum erscheinen würde. Warum die Verzögerung?

Knuth: Dafür gibt es viele Gründe. Ich konnte nie sehr gut abschätzen, wie lange etwas dauert, darin muss ich noch besser werden. Aber mir war klar, dass ich in den Ruhestand treten muss, um die Dinge nach meinem eigenen Zeitplan zu erledigen. Dann musste ich erst andere Bücher fertig stellen, die 'Selected Papers' ... Und ich habe eine dritte Auflage von The Art of Computer Programming gemacht, das hat den größten Teil von '96 und '97 gekostet. Ich hatte das eigentlich gar nicht vor, ich dachte nur an Band 4. Aber nachdem ich auf dem 16. TeX-User-Meeting über die Makros für meine Errata-Liste gesprochen hatte, kam Silvio Levy auf mich zu und sagte: 'Don, du solltest wirklich neue Auflagen dieser Bücher machen und TeX benutzen. Ich übernehme die ganze Tipparbeit und die Bilder, dann kostet es dich auch nicht so viel Zeit.' Ich hatte natürlich meine Qualitätsvorstellungen und neue Ideen, sodass es ein paar Jahre gedauert hat. Vor Band 4 musste ich also etliche andere Dinge erledigen, MMIX war das letzte.

c't: Wie ist heute Ihr Tagesablauf?

Knuth: Meine Tage, und das wird wohl für die nächsten 20 Jahre oder so ein stabiler Zustand sein, wenn ich gesund bleibe, sind typischerweise TAOCP gewidmet. Das bedeutet eine Kombination von

Lesen und Schreiben. Ich arbeite im Batch-Mode, weil ich nicht gerne verschiedene Dinge aus meinem Kopf raus und rein swappe. Üblicherweise nehme ich mir einen Abschnitt von TAOCP vor, und der sollte in sechs bis acht Wochen fertig werden. Ich will eine Seite pro Tag schaffen. Acht Wochen, 56 Tage ... das macht etwa 60 Seiten inklusive Lösungen für die Übungsaufgaben. Ich hoffe, dieses Tempo durchzuhalten, aber wenn ich 250 Seiten im Jahr schaffe, bin ich auch zufrieden. Bei der Arbeit an 'Concrete Mathematics' habe ich sogar zwei Seiten pro Tag geschrieben, aber das war auf Dauer zu viel. Das könnte ich nur eine Zeitlang durchhalten und würde dabei meine Ehe und meine Gesundheit riskieren.

Ich habe tausende von Aufzeichnungen, alle im Computer indiziert. Da sind 20 000 oder 30 000 Zeilen mit Notizen. Jede Zeile ist entweder ein Verweis auf einen Artikel, den ich lesen will, oder auf eine Idee, die ich mir aufgeschrieben habe. Das alles ist nach dem Inhaltsverzeichnis des Buchs abgelegt, sodass ich weiß, wann ich was im Detail lesen muss. Von den sechs Wochen, in denen ich an einem Abschnitt arbeite, verbringe ich typischerweise die ersten beiden damit, umfassend alles über das Thema zu lesen, beginnend mit den Dingen, die ich mir seit 1962 aufgehoben habe. In den nächsten zwei Wochen schreibe ich mit Bleistift ein Konzept, und die letzten zwei Wochen sind Feinschliff.

c't: Sie schreiben von Hand?

Knuth: Oh ja. Der Grund ist, dass ich schneller tippen als denken kann. Meine Handschrift ist dagegen perfekt auf meine Denkgeschwindigkeit abgestimmt. Wenn ich also tippe, verliere ich die Synchronisation, und dieses Synchronisationsproblem bremst mich, sodass es tatsächlich schneller geht, von Hand zu schreiben. Anschließend tippe ich dieses Konzept ab und bearbeite es dabei.

c't: Mit was für einem Computer arbeiten Sie?

Knuth: Ich habe zu Hause zwei Computer. Mit einem Macintosh surfe ich im Internet und erledige alle grafischen Arbeiten mit Photoshop, Illustrator, Distiller und so weiter. Aber meine Bücher schreibe ich an einem PC mit 600-MHz-Athlon, den ich letztes Jahr mit Hilfe eines Freundes zusammengebaut habe.

c't: Unter Linux?

Knuth: Ja.

c't: Gar kein Windows?

Knuth: Um Himmels willen, nein! Bis das immer gebootet hat ... Aber ich war heute ganz überrascht, inwieweit man Windows tatsächlich für nützliche Dinge benutzen kann. Die Leute bei meinem Vortrag an der TU München hatten zusätzliche Programme installiert, sodass Windows fast so gut wie Linux funktionierte. Wir hatten allerdings nichts Vernünftiges, um PostScript-Dateien anzuzeigen und haben dann doch einen Linux-Rechner genommen. Ich will hier niemandem zu nahe treten, aber ich bin ein großer Fan der Open-Source-Bewegung. Es beeindruckt mich, dass hier tausende von Leuten wirklich gute Arbeit leisten. Jeden Tag passieren zehn Kleinigkeiten, und nach einem Jahr ergibt das massive Fortschritte.

c't: Vielleicht haben Sie das sogar ausgelöst, indem Sie Ihre Programme im Quelltext

veröffentlichten.

Knuth: Na ja, ich habe dazu beigetragen, aber ich bin nicht der Vater der Open-Source-Bewegung. Ich habe eines von mehreren frühen Beispielen dafür geliefert, dass Kooperation an Stelle von Wettbewerb gut funktionieren kann. Hätte ich allerdings meine Reputation und mein gesichertes Einkommen damals noch nicht gehabt, dann wäre ich womöglich mit TeX wesentlich weniger freigebig umgegangen.

c't: Was halten Sie von Software-Patenten?

Knuth: Oh ja, ich habe mitbekommen, dass darüber in Deutschland gerade diskutiert wird. Meine persönliche Meinung dazu ist, dass Algorithmen wie Mathematik sind, also inhärent nicht patentierbar. Es beunruhigt mich, dass die meisten Patente von so einfachen Ideen handeln, dass ich von meinen Studenten erwarten würde, sie als Hausaufgabe zu entwickeln. Manchmal gibt es Ausnahmen, beispielsweise etwas sehr Raffiniertes wie eine Innere-Punkte-Methode für lineare Programmierung, wo man wirklich von einer signifikanten Entdeckung sprechen kann. Für mich ist das aber immer noch Mathematik. Ich komme aus einer mathematischen Kultur, in der wir kein Geld von Leuten kassieren, die unsere Sätze benutzen. Da gibt es den Gedanken, dass Mathematik entdeckt, nicht erfunden wird. Wenn etwas schon da war, wie kann man es dann patentieren?

Wenn etwas schon da war, wie kann man es dann patentieren?

Ich habe einen offenen Brief an das US-Patentamt geschrieben und ihn am Ende der zweiten Auflage des CWEB-Manuals veröffentlicht. Unglücklicherweise haben sie die Seite in der neusten Auflage wieder weggelassen, aber im Web findet man den Brief noch. Im Prinzip sage ich darin das, was Stallman schon seit Jahren sagt, nämlich dass Patente auf Algorithmen etwas Ähnliches wären wie Patente auf englische Wörter. Ich habe als Analogie unser Rechtssystem benutzt: Was wäre, wenn jeder Rechtsanwalt, der einen Fall zitiert, dafür bezahlen müsste? Es würde uns darin einschränken, neue Fälle zu behandeln, oder im Falle eines Software-Entwicklers, neue Software zu programmieren.

Wenn es damals Software-Patente gegeben hätte, hätte ich TeX nie schreiben können. Und so ziemlich jedes Programm, das ich heute für meine tägliche Arbeit benutze, wäre nicht geschrieben worden. Jedes Stück Software, das wirklich wichtig für mich ist, wäre vermutlich nicht entwickelt worden, wenn es damals Software-Patente gegeben hätte.

Ich denke, dass Software-Entwickler ein Einkommen verdienen, aber nicht dafür, dass sie sich einen Algorithmus schnappen und ihn dann für sich beanspruchen, sondern für Dienstleistungen wie das Anpassen von Programmen, das Verfassen guter Handbücher und die Unterstützung der Anwender.

In der ersten Auflage von Band 3 hatte ich die Bemerkung stehen, dass es eine starke Analogie zwischen der Erstellung eines Programms und dem Anfertigen einer Maschine aus Holz oder sonstigen Materialien gibt. Das macht es vermutlich unausweichlich, dass es eines Tages Patente auf Algorithmen geben wird. Ich habe das in der dritten Auflage ein wenig umformuliert.

Ich habe jedenfalls nicht die Zeit, bei allen Algorithmen zu überprüfen, ob sie irgendwelche Patente

berühren. Ich versuche es natürlich und verweise auch auf eine ganze Reihe Patente. Aber wenn ich ein Patent nicht erwähne, bedeutet das nicht, dass es nicht existiert.

c't: Halten Sie selbst irgendwelche Patente?

Knuth: Ich habe fünf Patente. Und in der Tat könnte man eines davon als Software-Patent sehen. Es ist wahrscheinlich inzwischen abgelaufen. Es hatte mit der Rasterung von Bildern zu tun. Als ich bei Adobe am 3:16-Projekt* gearbeitet habe, habe ich mit den Leuten dort darüber gesprochen, wie man Bilder rastert. Etwas von dem, was ich vorgeschlagen habe, schien jemandem patentwürdig, also hat er es aufgeschrieben. Die anderen Patente haben mit Dingen zu tun, die in den 60er Jahren in die Burroughs-Hardware eingeflossen sind.

c't: War TeX allein Ihr Werk?

Knuth: Einige Leute haben mich in wöchentlichen Meetings beraten, aber ich habe jede einzelne Zeile von TeX selbst geschrieben. Andere haben Treiber geschrieben, es auf verschiedene Maschinen portiert. Sie haben die Schnittstellen zur Außenwelt geschaffen, um Laserdrucker anzusteuern und so weiter. Aber den eigentlichen Kern habe ich selbst programmiert. Nur so konnte ich herausfinden, worin die Probleme bestanden.

Man versteht etwas nicht wirklich, wenn man nicht versucht, es zu implementieren. Ganz am Anfang habe ich für mich selbst ein kleines Memo darüber geschrieben, wie TeX funktionieren sollte. Das habe ich zwei Studenten als Projekt gegeben und bin für vier Wochen nach China gefahren, wo sie mich nicht erreichen konnten. Ich dachte, die Spezifikation wäre hundertprozentig klar gewesen. Aber als ich zurückkam, war ich überrascht, dass sie erst einen kleinen Prototypen am Laufen hatten.

Dann kam mein Forschungsurlaub, und der hat mir die Augen geöffnet, als ich versucht habe, TeX selbst zu implementieren. Alle fünf Minuten stieß ich auf eine Frage, die die Spec nicht beantwortete. Hätte ich als Student für denjenigen gearbeitet, der diese Spezifikation geschrieben hat, ich hätte andauernd seine Hilfe gebraucht.

Man versteht etwas nicht wirklich, wenn man nicht versucht, es zu implementieren.

Wenn man daraus den Schluss zieht, dass alle Projekte von einem Einzelnen erledigt werden müssen, dann bedeutet das nichts Gutes für das Software-Engineering. Dafür muss es bessere Lösungen geben. Aber wenn es um die erste Version eines neuen Systems geht, dann weiß ich wirklich keinen anderen Weg, als es in fast völlig unabhängige Teile zu teilen und jedes dieser Teile nur einer einzigen Person zu überlassen. Die einzelnen Leute müssen miteinander über die Schnittstellen reden, aber nicht über ihre eigene Arbeit. Wenn man die zweite Generation eines Systems implementiert, hat man ein gemeinsames Vokabular und die Leute kennen sich mit den gleichen Dingen aus, dann ist es etwas anderes. Aber ganz am Anfang ... Für mich ist der Grund, warum manch andere Projekte sich so abstrampeln, ganz einfach: Sie versuchen es mit mehr als einer Person.

c't: Was halten Sie von objektorientierter Programmierung?

Knuth: Die Idee hat mir sofort gefallen, als ich sie zum ersten Mal in Simula gesehen habe. Aber mit Vererbung und dergleichen wird es natürlich schwieriger zu beweisen, dass ein Programm korrekt ist. Ich mag die Idee von Modulen und Klassen, aber irgendwie gefällt sie mir doch nicht so sehr, dass ich sie in meinen eigenen Programmen für nötig halten würde.

c't: Würde sich das nicht prima für einen MMIX-Simulator anbieten? Objekte für die ganzen Funktionseinheiten, den Speicher, die Caches ...

Knuth: Ich würde liebend gerne eine andere Lösung sehen. Speziell für den Cache-Simulator, der intuitiv eigentlich ganz einfach scheint. Daran habe ich mich nämlich zwei Monate lang versucht und eine Lösung nach der anderen verworfen. Schließlich habe ich das Problem mit massenweise Code erschlagen, indem ich Fall für Fall ausprogrammiert habe. Das war das komplizierteste Programm, das ich je geschrieben habe. Ohne CWEB hätte ich es nicht geschafft.

c't: Wäre das mit einer objektorientierten Sprache nicht einfacher gewesen?

Knuth: Ich habe bei einem C++-Objekt die Syntax noch nie intuitiv auf einen Blick verstanden. Das ist eine andere Gedankenwelt. Es ist natürlich auch eine Frage der Übung. Ich glaube, ich bräuchte eine Sprache, die über C++ hinausgeht, um besser mit Coroutinen und Threads zu arbeiten. Jedenfalls konnte ich den MMIX-Pipeline-Simulator in keiner anderen Sprache schreiben, die ich kannte.

c't: Sie sagen also: Objektorientierte Programmierung ist eine gute Idee, aber in der Praxis doch nicht brauchbar?

Knuth: Nein. Ich will es so zusammenfassen: Wenn es eine gute Idee ist, würde ich es liebend gerne sehen, wenn sie jemand verwirklichen würde. Man sollte sich den Cache-Simulator mal als Studienarbeit vornehmen und ihn so programmieren, wie er eigentlich hätte werden sollen. Ich würde ihn liebend gerne in einem anderen Stil sehen, so, wie er eigentlich sein sollte. Weil ich versucht habe, einen besseren Stil zu finden. Meine Lösung von 1999 ist schon in Ordnung; Immerhin kann ich sie nach zwei Jahren immer noch verstehen und die Leute können sie auf vielfältige Weise anpassen. Aber es ist absolut vorstellbar, dass es eine viel bessere Art gibt, dieses Programm zu schreiben, die ich nur nicht kannte.

c't: War Ihnen Performance wichtig, als Sie den MMIX-Simulator schrieben?

Knuth: Ich hatte nicht den Ehrgeiz, ihn so schnell wie eine richtige Maschine zu bekommen. Aber er sollte angemessen schnell sein, damit man auch längere Programme darauf laufen lassen kann.

c't: Sie sind in der Spezifikation und auch im Emulator sehr detailliert auf die Hardware-Implementierung eingegangen. Haben Sie sich von Hardware-Designern beraten lassen?

Knuth: Oh ja, ich hatte sagenhafte Hilfe. John Hennessy hat eine lange Kritik über mein ursprüngliches Design geschrieben. Dann habe ich mit Dick Sites, dem Designer des Alpha-Chips, zusammengearbeitet. Er hatte viele Vorschläge, und ich bin mit ihm eine Menge Einzelheiten durchgegangen. Auch die Betriebssystem-Gruppe bei Sun und ein paar andere Leute haben zum MMIX-Design beigetragen, insgesamt eine gewaltige Hilfe. Alle Fehler sind natürlich von mir. Wobei es mich überraschen würde, wenn sich noch große finden würden, denn die wären wohl schon

irgendjemandem aufgefallen.

c't: Hat sich schon jemand an einer Hardware-Implementierung des MMIX-Prozessors versucht?

Knuth: Einige Leute haben mir gesagt, dass sie es vorhaben. Ich habe mal mit Dave Ditzel von Transmeta darüber geredet, und Arvind vom MIT hat angekündigt, mich vielleicht eines Tages zu überraschen. Ich bin aber nicht sicher, ob diese Überraschung jemals kommt. Ich bin nicht wirklich unglücklich, dass MMIX noch nicht gebaut wurde. Es würde mich natürlich freuen, eines Tages einen MMIX-Rechner auf meinem Schreibtisch stehen zu haben, weil ich denke, MMIX macht vieles richtig, was andere Maschinen falsch machen. Aber der Traum ist wohl zu schön, um wahr zu sein. (**bo[2]**)

Literatur

[1] **Homepage von Donald E. Knuth[3]**

* *In dem Buch '3:16 Bible Texts Illuminated' beschäftigt sich Knuth mit Bibelversen, und zwar jeweils mit Kapitel 3, Vers 16 aus jedem Buch.*

URL dieses Artikels:
http://www.heise.de/ct/artikel/Der-Perfektionist-288138.html

Links in diesem Artikel:
[1] #literatur
[2] mailto:bo%40ct.heise.de
[3] http://www-cs-faculty.stanford.edu/~knuth

262643 **Copyright © 2011 Heise Zeitschriften Verlag** **Content Management** by **InterRed**
International: **The H**, **The H Security**, **The H Open Source**