

„Who-is-who“ der Freien Software

Arbeitsauftrag

Erstellen Sie eine Charakterisierung von RICHARD STALLMAN, LINUS TORVALDS und ERIC RAYMOND anhand den beigefügten Textbezügen. Berücksichtigen Sie dabei insbesondere die jeweilige Meinung zu *Freier Software*, *Open Source* und *proprietärer beziehungsweise kommerzieller Software*. Versuchen Sie, die Motivation der einzelnen Personen, sich für freie Software zu engagieren, herauszuarbeiten und ihre Reputation in der freien Software-Gemeinde zu beurteilen.

I Richard Matthew Stallman

I.1 Richard Stallman aus Wikipedia, der freien Enzyklopädie

Richard Matthew Stallman (* 16. März 1953 in Manhattan, NYC), auch bekannt unter seinen Initialen RMS, ist Gründer des GNU-Projektes und einer der bekanntesten Verfechter Freier Software. Er ist der erste Präsident der Free Software Foundation und ist Preisträger des MacArthur Fellowship, des Grace Murray Hopper Award der Association of Computing Machinery (ACM), sowie einer der Empfänger des Takeda Foundation Award.

Lebenslauf Stallman arbeitete Anfang der 1980er Jahre im AI Lab (Abteilung für Künstliche Intelligenz) des Massachusetts Institute of Technology zusammen mit einer Gruppe von Programmierern die sich selbst als Hacker bezeichneten. Diese Hackergemeinschaft vertrat eine sehr rigorose Philosophie des unbegrenzten Informationsflusses. In den folgenden Jahren gab es in der Softwarebranche einen, aus Stallman's Sicht entscheidenden Wandel: Viele Firmen begannen Software nicht mehr in der bis dahin weitgehendst üblichen Form von Quelltexten auszuliefern, sondern in Form eines rein maschinenlesbaren Formates. Auch statteten von nun an einige Firmen ihre Software mit Lizenzen aus, die es den Anwendern verbot die Programme weiterzuverteilen oder die Programme selbst zu verändern.

Stallman empfand diesen Verlust der Kontrolle von Benutzern über ihre eingesetzte Software als eine Einschränkung seiner Rechte. Um diesem Trend entgegenzusteuern schuf er eine Lizenz, welche unter dem Namen GPL bekannt wurde. Diese Lizenz garantiert Anwendern uneingeschränkte Rechte über ihre Software und stellt sicher, dass diese Rechte, wenn sie einmal gegeben wurden, auch nicht mehr nachträglich entfernt werden können.

Er kündigte seinen Job am MIT und arbeitete für mehrere Jahre daran, ein Betriebssystem zu programmieren welches komplett auf freier Software basierte. In dieser Zeit erschuf er unter anderem die erste Version von GNU Emacs (ein komplexer, programmierbarer Texteditor zu dessen Originalversion er bereits beigetragen hatte), die GNU Bash (die freie Version einer bekannten Kommandozeilenumgebung für Unix) den ersten plattformübergreifenden C-Compiler (gcc), sowie verschiedene für eine Unix-Umgebung benötigte Utilities. Anfang der 1990er fehlte zu einem funktionierenden Betriebssystem jedoch noch ein entscheidender Teil, der so genannte Kernel. Das von Stallman initiierte HURD Projekt, welches diesen Mangel beheben sollte, verzögerte sich zusehends, was, wie er später zugab, in erster Linie auf Designschwächen zurückzuführen war. 1991 veröffentlichte Linus Torvalds einen eigenen Unixkernel namens Linux unter der GNU-Lizenz. Auch wenn Linux ursprünglich nicht als Ersatz für HURD vorgesehen war, war die letzte Lücke zu einem freien Betriebssystem damit geschlossen. Linux schaffte es innerhalb weniger Jahre zahlreiche Entwickler anzuziehen die sich mit Hilfe des Internets koordinierten. Stallman und die Free Software Foundation integrierten daher, wenn auch mit einer gewissen kritischen Haltung, Linux als offiziellen Kernel in ihr GNU-System. Bis heute wurde das GNU/Linux System stark weiterentwickelt und wird inzwischen auch im professionellen Umfeld zunehmend eingesetzt.

Trotz seiner zahlreichen Beiträge zur freien Software, ist Richard Stallman bis heute eine sehr umstrittene Persönlichkeit. Ideologische Differenzen führten inzwischen zu einer gewissen Spaltung in die Open Source (Offene Quellen) und die Free Software (Freie Software) Bewegung. Stallman steht zum Teil in der Kritik der Open-Source-Bewegung aufgrund verschiedener Forderungen, die er als notwendige Bestandteile ansieht um Benutzerfreiheiten zu sichern. Dies ist beispielsweise das Recht des Anwenders auf Kopien, sowie die Forderung, dass diese Rechte wenn sie einmal gegeben wurden auch bei Weiterentwicklungen von Programmen nicht mehr zurückgenommen werden können. Anhänger der Open-Source-Bewegung sehen

darin oft eine zu starke Einschränkung bei den Möglichkeiten der Lizenzgestaltung, die teilweise auch als geschäftshemmend angesehen wird. Die meiste Zeit über arbeiten diese beiden Bewegungen jedoch sehr eng zusammen.

I.2 The Saint of Free Software by Andrew Leonard

When Richard Stallman gave Bill Gates the finger in front of Stanford's computer science building, I got nervous. No, it wasn't the real Bill Gates – it was just his name, engraved in giant letters over the main entrance to the 2-year-old, Gates-funded building. But it didn't seem like a Stanford thing to do. The campus is immaculately manicured, dotted with picture-postcard palm trees and squeaky-clean students. It's just not a flipping-the-bird kind of place.

It didn't strike me as a Richard Stallman kind of place, either. Stallman is a legendary hacker, the founder of the free software movement, a MacArthur "genius grant" recipient and a programmer capable of prodigious exploits. But on this day in Palo Alto he looked unkempt and off-kilter. I had already spent a good part of the afternoon watching in bemused silence as he painstakingly examined his long, stringy brown hair for split ends. I was also mesmerized by his piercing green eyes, radiating the power of an Old Testament prophet. I feared his wrath.

We had come to Stanford in search of a place where Stallman could download his e-mail. Two hours away from catching a long flight to New Zealand – partly for vacation, partly to continue proselytizing his free software "mission" – Stallman was jonesing for one last connection to the Net. Being Richard Stallman, he figured he could just drop in on the computer science department at Stanford. He hadn't visited for several years, but he was good friends with equally legendary Stanford professor John McCarthy – the man who invented the Lisp programming language and coined the term "artificial intelligence". Stallman himself programmed the multipurpose Emacs editing tool, a kind of nuclear-powered Swiss Army knife favored by top-notch programmers and computer scientists. Surely some Emacs acolyte would be delighted to help the one and only Richard Stallman grab his e-mail.

First we tried to sneak in through a side door of the Gates building. Over a lunch of ribs, duck, trout and popcorn shrimp at Palo Alto's MacArthur Park restaurant, Stallman had told me that he didn't despise Bill Gates as much as other free software guerrilla fighters do. But he clearly wasn't eager to legitimize Gates' stature by walking submissively through his totemic gate. Free software and Microsoft don't mix. There had to be a better way.

Except there wasn't. The path to McCarthy's office from the side door entrance was obscure. We sucked in our guts and headed for the main gate.

"Hey," Stallman called out to a graduate student opening the door in front of us, "is it the tradition here to give Bill the finger whenever you go through these doors?"

The student looked over his shoulder, twitched a nervous smile and disappeared inside. Stallman shrugged – and right there on the spot decided to start his own protest movement. As we entered the building, out came what the ancient Romans used to call the "digit impudicus." Stallman flashed me a sly grin. I glanced around, looking for security.

Over the course of about half an hour in the building, Stallman encouraged three other people to join his campaign. No one signed on unreservedly, but two recognized him right away – one from a conference some six years earlier and another from his picture in a recent Forbes magazine article celebrating the surprising commercial success of the free software (or, as it is now more commonly called, "open source") movement. Would the movement to deride Gates have as much success? Stallman didn't know and didn't care. As he

pointed out to me repeatedly through the course of our afternoon together, he doesn't do things because they are socially acceptable or strategically appropriate. Success is not his measure for accomplishment. He does what he does because he thinks it is the morally correct, or simply fun, thing to do. And he brooks no compromise.

Stallman's tendency toward intransigence has made him a controversial figure in the rapidly expanding and ever-more-high-profile world of free software – software for which the source code is freely accessible to all computer users. The expanding market shares boasted by the Apache Web server and the GNU-Linux operating system have encouraged a whole new generation of hackers to plunge into the business world and preach that theirs is the best way to create superior software products – less buggy, more stable and more flexible than their proprietary competitors.

But Stallman just doesn't care about pragmatic arguments – he declares he would prefer to use a free software program even if it wasn't the best solution for his needs. Freedom, for Stallman, is a fundamental moral good – the freedom for computer users to share and cooperate, to copy and change code as they please.

His stance makes some factions of the burgeoning "open source" community uncomfortable – so uncomfortable, in fact, that the very choice of the name "open source" demonstrates an attempt to distance those factions from the unsavory radicalism of Richard Stallman. Never mind that Stallman started the free software movement, or that thousands of lines of code that he personally authored are an integral part of what most people today call "Linux." To the new generation, Stallman is an embarrassment and a hindrance who must, at all costs, be trundled into a back room before he scares off the investors.

As I drove Stallman to the airport, I asked him if he was gratified by the fact that, whatever you call it – free software or open source – there's no denying that the campaign he began at MIT some 15 years ago is one of the hottest stories in computing today. He sighed. He is somewhat pleased, he said, but he is even more anxious. He feels that he is being shoved aside – "that certain people are trying to rewrite history and deny me my place in the movement."

The last chapter of "Hackers", Steven Levy's definitive history of the rise of the personal computer, is titled "The Last True Hacker". It tells the sad story of Richard Stallman, a young, supremely talented programmer who flourished during the glory days of hacking at MIT. Back then, in the 1970s, everyone shared everything, from Chinese food to the latest code tweak. Anything proprietary was greeted with disgust and scorn.

As the '70s became the '80s, however, the hacker heyday came to an end. One after another, talented MIT programmers left academe for the warm embrace of the commercial world. The worst blow came with the creation of Symbolics, a company devoted to creating "artificial intelligence" machines. Symbolics eviscerated MIT's artificial intelligence laboratory, where Stallman worked. To Stallman, it seemed like the end of an era, a recipe for despair.

"I'm the last survivor of a dead culture," he told Levy. "And I don't really belong in the world anymore. And in some ways I feel I ought to be dead."

But being Richard Stallman, he refused to crumple in the face of his own anxiety. Symbolics, says Stallman, spurred him to create the Free Software Foundation and dedicate his life to the construction of GNU (GNU's Not Unix), a Unix-like operating system that would be protected from grasping proprietary hands by a new kind of software license.

"When I saw the prospect of living the way the rest of the world was living," said Stallman, "I decided no way, that's disgusting, I'd be ashamed of myself. If I contributed to the upkeep of that other proprietary software way of life, I'd feel I was making the world ugly for pay."

All GNU programs are protected under the principle of "copyleft." A copylefted program may be copied, it may be changed, it may be modified in any way a programmer pleases, and it can even be sold – as long as

the source code to all changes and modifications is also made accessible to all computer users.

“I decided, let’s try and build something new to replace what was lost,” said Stallman. “Part of what I wanted was to make another hacker community with the same virtue as the previous one, and that virtue, to me, was the freedom to cooperate. You have a certain way of life when you have freedom in a free society. In a totalitarian, non-free society, every aspect of how you deal with people is shaped by your fear. In proprietary society, your dealings with other people are shaped by fear of the information police, currently in its incarnation of the Software Publishers Association.”

Stallman turned out to be anything but the “last” hacker – in part, as Steven Levy argues, as a result of the spread of the personal computer, but also in part due to Stallman’s own efforts. The GNU project, in particular, inspired countless young hackers to devote their talents to writing free software. And eventually, along came Linus Torvalds, who undertook the all-important task of writing the “kernel” – the heart of code that connects all the different pieces of computer hardware and software – for a GNU-type operating system. The Linux kernel made the GNU system complete. Stallman and his fellow FSF hackers are still working on their own much-delayed kernel, HURD, since, as Stallman said, “We’ve put so much work into it we might as well finish it.” But the job that Stallman set out to accomplish is done. The core parts of the “GNU system” have been assembled.

There’s just one problem – the near universal tendency, by the press, by hackers, by free software enthusiasts, to refer to “Linux” as if the term represents the entire system. Linux is just the kernel; the system includes hundreds of other software tools and utilities, many of which were created by GNU hackers.

The evolving semantics of free software are a huge sore point for Stallman. In response, he painstakingly corrects anyone in his presence who utters the word “Linux” when referring to the whole operating system. The proper usage, says Stallman, is “GNU-Linux.” (This sounds like “guh-noo li-nooks.”)

Stallman’s tireless defense of the GNU legacy has launched many a Usenet flame war and contributed to his image as someone obsessed with unimportant details – to the detriment of the greater goals of the free software/open source community. For many open source advocates, arguing about terminology distracts from the main fight – the battle to topple Microsoft.

But Stallman’s motives are hardly petty – though he does acknowledge that one of his goals is to ensure that all the GNU programmers get credit for the work they’ve done.

“The reason I care especially,” said Stallman, “is that there is a philosophy associated with the GNU project, and this philosophy is actually the reason why there is a system – and that is that free software is not just convenient and not just reliable ... More important than convenience and reliability is freedom – the freedom to cooperate. What I’m concerned about is not individual people or companies so much as the kind of way of life that we have. That’s why I think it’s a distraction to think about fighting Microsoft.”

There are factions within the free software community who are careful to employ the GNU-Linux terminology, but in large part Stallman’s protests have been ignored. The word “Linux” rules the headlines, and it is gathering momentum. I asked Stallman if he saw any chance of winning his battle.

“It’s a mistake to ask that question,” said Stallman, fixing upon me a baleful look. “Because that makes it sound like there is one winner and one loser and it’s an all-or-nothing thing. You’re leading yourself into confusion mentally if you formulate it that way. As I see it, I’m sure to have a certain amount of success. And the question facing you or anyone else is, what are you going to do? Instead of worrying about what somebody else is going to do, which is not under your control, the important thing is, what are you going to decide about what is under your control?”

Stallman came to the Bay Area to attend the “Open Source Developer’s Day” on Aug. 21 – a convocation of programmers and free software enthusiasts organized by the computer book publishing company O’Reilly &

Associates. Stallman hadn't been invited to the first such gathering of "open source" leaders, a "free software summit" held in April to coincide with Netscape's decision to release the source code to its browser. But the omission drew wide criticism, and this time around, says Stallman, the organizers dared not ignore him again.

They did, however, ask him to support "unity," said Stallman. They should have known better. To ask Stallman to mince words or keep his mouth shut is to fundamentally misunderstand what the man is all about. Instead of giving lip service to the theme of the conference – the idea that the gathering represented the "coming together" of all the multiple elements within the open source community – Stallman ended up criticizing the core business of the conference organizer. He claimed that the manuals and technical books published by O'Reilly & Associates were doing a disservice to the free software cause. Free software needs free manuals, said Stallman.

"Tim O'Reilly spoke for quite a while earlier in the day about why it wasn't their responsibility that their manuals weren't free and about why their manuals were a good thing even though they are not free," explained Stallman. "What I said was, if you can't write a free manual, please write no manual. Because it will be easier for us to find someone else to write a free manual if your non-free one isn't making a lot of people think the job is done."

When Stallman starts talking about free manuals, people look sidelong at him, as if to say – oh, so this is the communist I've been hearing so much about. It's precisely this kind of talk that makes the open source advocates most antsy. How can the case be made for free software in a commercial marketplace when crazy radicals like Richard Stallman are declaring that everything should be free?

Except that Stallman isn't demanding that everything should be free. He has no problem with people selling free software or selling manuals that are also available for free, or providing contract support or any other software related service for a fee. What he urges is that all software information remain accessible. And in his view, manuals are software – they are part of the program, and need to be freely available and free for modification as the software changes.

"Software should come with documentation," said Stallman. "So manuals are absolutely essential. A free software package has to offer free documentation, because otherwise the documentation can't go to everyone who gets the package. If we don't have a good free manual for a program we are missing something, there's a gap in the system."

Stallman highlights an interesting contradiction. O'Reilly & Associates has been a huge supporter of the concept of open source. It is also justly renowned for the quality of its publications. But the quality is the problem.

"If they were incompetent, it wouldn't be an issue," said Stallman. "But it's hard for us to get free manuals written with all those O'Reilly manuals out there."

Free manuals, flipping off Bill, correcting listeners and colleagues every time they use the word "Linux" sloppily – all this from a man whose appearance flashes back to the height of the '60s counter-culture. No wonder he makes his free software fellows nervous: He's unpredictable, uncontrollable and incorrigible.

And it would be a mistake to attempt to control Richard Stallman – even if that were possible. His incorrigibility is part of what makes free software special.

I asked Stallman if he had paid attention to the uproar in the "Linux community" in mid-August – when a couple of entrepreneurs announced the creation of a "Linux Standards Association" that would charge admission for entry and raised a firestorm of protest.

"I was glad to see that there was so much strong reaction against it," said Stallman, "because what I saw is

II LINUS BENEDICT TORVALDS

that even though most of those people were calling their system 'Linux,' to a certain extent they had absorbed some of the GNU philosophy. They didn't just say that the way we are running our own community is more productive and develops better software. They cared about the way of life for its own sake."

That sense of caring is a major source of free software's strength. There's no questioning that the case for free software is bolstered by the wide perception that GNU-Linux systems crash less often than Windows-based systems, or the fact that Apache is the Web server of choice for demanding system administrators. Pragmatic benchmarks help.

But that's only half the story. It is the intersection of the pragmatic with the ideological that makes free software compelling, that inspires the devotion of volunteers and the passion of advocates. Any attempt to write Richard Stallman out of the history of the movement threatens to eviscerate it, to sap the very emotion that gives it its strength.

Right now, Richard Stallman, no matter how unkempt, could probably walk into any computer science department anywhere in the world and find someone to help him download his e-mail. Among computing's old guard he's still famous, still respected as the author of Emacs and recognized for his accomplishments as the founder of the Free Software Foundation.

I, for one, hope that there is always a place in the computing universe for Richard Stallman. As I drove away from the San Francisco Airport, I thought about Stallman's oft-expressed anxiety, and his fear that "certain people" are striving to deny him his place in the cause he began. At the beginning of the day, I might have dismissed the assertion as paranoia. But by the time I dropped him off, I believed him.

Unkempt and off-kilter though he may be, Stallman embodies the fervor and the faith that make free software worth embracing. If the pragmatists of the open source cause sacrifice him to make free software safe for business, it seems to me, they risk losing their movement's soul.

II Linus Benedict Torvalds

II.1 Linus Torvalds aus Wikipedia, der freien Enzyklopädie

Linus Benedict Torvalds ist der Vater von Linux. Er wurde am 28. Dezember 1969 in Helsinki, Finnland geboren.

Am 17. September 1991 veröffentlichte der Finne Linus Torvalds in einer Newsgroup die Ankündigung für Linux 0.01. Der Quellcode für das neue Unix-Betriebssystem umfasst 241 KByte und das Makefile ist 96 Zeilen lang. Dabei waren noch eine Bash und Update als Binaries.

Am 9. Juli 1996 stellt Linus Torvalds den Linux-Kernel 2.0 auf die offiziellen Server. Von nun an verbreitet sich das Betriebssystem auch auf Computern mit Prozessoren, die nicht von Intel stammen. Am selben Tag wird Tux der Pinguin zum offiziellen Logo für Linux.

Am 20. November 2000 erblickte die zweite Tochter von Linus und seiner Frau Tove Celeste Amanda das Licht der Welt.

II.2 Interview mit der Tokyo Linux Users Group aus dem Jahre 1997

Hiro Yamagata: Du bist einer der führenden Verfechter freier Software geworden. Anders allerdings als Richard Stallman kommentierst Du nicht so oft, was freie Software sein soll und was sie uns allen bedeutet. Bist Du überhaupt interessiert an der Förderung der freien Software oder interessierst Dich nur die Software an sich?



Linus Torvalds: Ich bin im allgemeinen ein sehr pragmatischer Mensch: Was funktioniert, funktioniert. Wenn es um Software geht, bevorzuge ich freie Software, weil ich noch nie ein Programm gesehen habe, daß von sich aus meinen Bedürfnissen entsprach. Die Quellcodes zur Verfügung zu haben, kann Dir das Leben retten.

In diesem Sinne bin ich ein Förderer der freien Software und der unter GPL freigegebenen Software. Insbesondere deshalb, weil, wenn Software einmal unter GPL lizenziert ist, dann bleibt sie frei und ich muß mir um weitere Releases keine Sorgen mehr machen.

Allerdings heißt es auch nicht, daß ich gegen kommerzielle Software bin. Kommerzielle Software-Entwicklung hat auch einige Vorteile - die geldbringenden Aspekte geben einige neue Anreize, die es für freie Software nicht gibt. Und diese Anreize sorgen oft für ein polierteres Produkt.

Zum Beispiel bin ich wirklich sehr glücklich über die kommerziellen Linux CD-ROM-Verkäufer wie Red Hat. Die Kommerzialisierung hat für Linux den Anreiz geschaffen, eine gute Version zu produzieren, die man einfach installieren und benutzen kann, und bei der das Thema des Packaging geklärt wurde. Einfach alles ist problemlos erhältlich.

Vor diesen kommerziellen Unternehmungen war es nicht ganz einfach, Linux einzurichten, weil die meisten Entwickler vor allem von ihren eigenen Bedürfnissen motiviert werden und sich wenig um die Einfachheit der Installation und der Benutzung kümmern. Und bei Linux hat man trotz der Kommerzialisierung Zugang zu den Quellcodes, also bekommt man das Beste aus beiden Welten.

Und dann gibt es natürlich noch Software, die kommerziell ist, aber ohne Quellcodes ausgeliefert wird - dies ist die traditionelle kommerzielle Software im Gegensatz zu der Red Hat Linux-Distribution. Dagegen versuche ich auch nicht zu predigen. Ich hasse die Tatsache, daß ich keine Bugs beheben kann, aber manchmal führt kein Weg an dieser Software vorbei.

Hiro Yamagata: Wann und warum hast Du Linux der GPL unterstellt? Hast Du es jemals bereut, daß Linux keine Shareware ist?

Linus Torvalds: Ich habe niemals bereut, daß Linux keine Shareware ist. Ich mag diese pay for use Binär-Shareware-Programme, die in der MS-DOS-Welt so verbreitet sind, überhaupt nicht.

In meinen Augen vereint Shareware die schlimmsten Nachteile von kommerzieller Software (keine Quellcodes) mit dem Schlechtesten der freien Software (es fehlt der letzte Schliff). Ich glaube überhaupt nicht an den Shareware-Markt.

Am Anfang habe ich Linux unter einem Nicht-GPL-Copyright herausgegeben, das in der Tat noch viel restriktiver als die GPL war. Es setzte voraus, daß die Quellcodes immer verfügbar sind und erlaubte nicht, daß mit Linux in irgendeiner Form Geld verdient würde. Mit anderen Worten, nicht nur ich wollte mit Linux kein Geld verdienen, auch niemand anderes sollte dies tun.

Die erste Copyright-Version war eigentlich eine Reaktion auf das Betriebssystem, das ich versuchte zu benutzen, bevor ich Linux entwickelte: Minix. Minix war als Betriebssystem für die Ausbildung gedacht, aber es war viel zu limitiert und zu teuer. Es war auch sehr schwer zu bekommen.

Als ich also Linux entwickelte, wollte ich, daß es einfach mit den gesamten Quellcodes per FTP erhältlich ist. Und ich wollte nicht, daß es für irgend jemanden zu teuer ist.

Nach ungefähr einem halben Jahr habe ich das Copyright zur GPL geändert. Es wurde schnell klar, daß das ursprüngliche Copyright viel zu restriktiv war und einige ganz legitime Dinge verbot, wie z.B. Floppy-Kopierdienste - dies war die Zeit, bevor CD-ROMS wirklich populär wurden. Zuerst war ich in Bezug auf die GPL sehr nervös, aber ich wollte auch dem gcc (auf dem Linux basierte) eine Ehre erweisen, der bekanntlich der GPL unterlag.

Linux unter die GPL zu nehmen, war das beste, was ich je getan habe.

Hiro Yamagata: Offensichtlich arbeitest Du an Linux, weil es Dir auf die eine oder andere Art Spaß macht. Wenn Dir jetzt jemand Geld bieten würde, damit Du Dich auf die Linux-Entwicklung konzentrierst, glaubst Du, es wäre anders gelaufen? Hat dies mit der Wahl Deines augenblicklichen Arbeitgebers zu tun? So wie ich gehört habe, waren viele Leute sehr erstaunt, daß Du nicht zu einer Firma gegangen bist, die direkt mit Linux zu tun hat.

Linus Torvalds: Ich bin nicht zu einer kommerziellen Linux-Firma gegangen, weil ich nie gezwungen sein wollte, etwas zu tun, was ich nicht tun will.

Eigentlich wollte ich nie, daß meine Arbeit zu 100% mit Linux zu tun hat, weil ich die Befürchtung hatte, daß ich mich langweilen würde, wenn Linux alles wäre, was ich tun würde. Also war der Job hier bei Transmeta genau das Richtige für mich. Ich mache etwas Interessantes, das nichts mit Linux zu tun hat, und ich habe trotzdem Zeit, an Linux zu arbeiten. Und ich bin in Bezug auf Linux zu nichts verpflichtet, mein Chef kann mich also nicht bitten, etwas mit Linux zu tun, was ich nicht tun will.

Hiro Yamagata: Was sind Deine Ansichten in Bezug auf Richard Stallmans Idee von Freier Software? In Deinem Vortrag am MIT vor ein paar Jahren klang es nicht so, als wärst Du gegen proprietäre Software. Bist Du dagegen? Welche Applikationen siehst Du eher im Bereich der freien Software und welche in der proprietären Welt?

Linus Torvalds: Ich sehe die Dinge nicht so schwarz-weiß wie RMS. Ich bin der Meinung, die Leute können tun und lassen, was sie wollen, aber, wie leicht zu erkennen ist, bevorzuge ich freie Software. Ich bevorzuge freie Software nicht aus religiösen Gründen oder so etwas ähnlichem. Nur habe ich viele verschiedene Maschinen, und ich möchte an allen arbeiten können. Wenn ich freie Software habe, kann ich sie auf meinen Alphas und PCs kompilieren.

Auf der anderen Seite neige ich dazu zu glauben, daß einige Dinge bei kommerzieller Software besser funktionieren, weil ein wesentlicher Aspekt des Endprodukts der letzte Schliff ist - und kommerzielle Software hierin wirklich gut ist.

Zum Beispiel sind die Benutzeroberflächen oftmals bei kommerzieller Software besser. Dies ist sicher nicht immer wahr, aber in den meisten Fällen ist die Benutzeroberfläche für eine kommerzielle Firma der wichtigste Teil eines Programms. Ob die Software funktioniert oder nicht ist sekundär, wie die vielen bugdurchsetzten Programme von Microsoft beweisen (nicht daß Microsoft hier der einzige Übeltäter ist).

Also sind Textverarbeitungsprogramme in der kommerziellen Version meist besser, weil ihr wichtigster Teil die Benutzeroberfläche ist.

Auf der anderen Seite ist die freie Software sehr erfolgreich in Bereichen mit technischem Fokus. Dies schließt natürlich den Linux-Kernel ein, aber auch Dinge wie den GNU C-Compiler und andere Entwicklungs-Tools.

Hiro Yamagata: Wir haben schon Linux-Distributionen gesehen, die es dem Benutzer erlauben, Linux zu installieren, ohne zu wissen, wie es innen aussieht. Dies hat eine Menge neuer Benutzer zu Linux gebracht, aber es gibt Leute, die behaupten, dies untergrabe den Geist der freien Software, weil niemand mehr gezwungen wird, unter die Motorhaube zu blicken und zu verstehen, wie es darunter aussieht. Beunruhigt Dich das?

Linus Torvalds: Nein - ich glaube das ist so am besten. Ich glaube nicht, daß sich jeder dafür interessieren muß, wie ein Betriebssystem funktioniert. Es ist das, was mich interessiert, aber ich glaube, ein Programm ist nur so gut wie seine Nützlichkeit.

Ein nutzloses Programm ist niemals gut, egal wie gut es implementiert ist. Die Tatsache, daß es eine Menge Leute gibt, die Linux nur benutzen wollen und sich nicht darum kümmern, wie der Kernel funktioniert, ist nicht nur eine Ehrung für Linux, sondern bringt auch neue Entwicklungen mit sich, an die ich vorher nie

gedacht habe.

Diese Benutzer nutzen die Software anders als ich, und daher sind ihre Bedürfnisse eben auch andere. Und in vielen Fällen zeigten diese Bedürfnisse Dinge auf, die in Linux fehlten oder schlecht gelöst waren. Also haben diese Benutzer Linux entscheidend weitergebracht, obwohl sie sich für das Innenleben nicht interessierten.

Hiro Yamagata: Ich weiß, Du wurdest das sicherlich schon tausendmal gefragt, aber warum wurde Linux so ein Erfolg? Einige Leute meinen, es liegt an Dir. Andere behaupten, es war eine Kombination aus gutem Timing und einer Menge Glück. Was meinst Du?

Linus Torvalds: Es gibt viele Gründe. Gutes Timing und viel Glück sind die offensichtlichen. Aber ich glaube, ich war auch ein guter Manager und wohl auch ein guter Programmierer, und dies war sicherlich auch wichtig, um Linux zu einem erfolgreichen Produkt zu machen.

Ich glaube auch, daß das Linux-Entwicklungsmodell ein gutes Modell ist. Linux hat viel weniger Regeln als andere Entwicklungen und jeder kann beitragen, was er will. Ich bin nur ein allgemeiner Filter für die Patches, aber ansonsten ist es ein sehr freies Entwicklungsmodell.

Hiro Yamagata: Jetzt wo Linux so groß geworden ist, spürst Du viel Druck, Linux auf dem richtigen Kurs zu halten? Was ist Deine größte Sorge in Bezug auf die Zukunft von Linux?

Linus Torvalds: Ich hatte immer den Druck, Linux auf dem rechten Weg zu halten, aber es war immer technischer Druck und daher hat er mir nicht viel ausgemacht. Das Gute an technischen Fragen ist, daß sie eine gute technische Antwort haben. Es sind die nicht-technischen Fragen, die oftmals gar keine Antwort haben. Für die technischen Probleme findet man eine Lösung, solange nur gute Leute daran arbeiten. Und Linux hat die Allerbesten.

Also mach ich mir um die Zukunft von Linux keine Sorgen. Technisch wird Linux besser und besser werden und die nicht-technische Seite macht mir persönlich keine Sorgen.

II.3 “Linux Anecdotes“ von Lars Wirzenius, Linux Expo. 1998, gekürzt

I seem to have acquired a little bit of reputation in the Linux community, despite my efforts in to stay quiet and invisible, so that I don't get so many questions from people having trouble with Linux.

Part of my reputation is that I know Linux pretty well. That part is based on the time when I and Linus Torvalds—I assume you know Linus—shared an office at the University of Helsinki. In reality, I don't know Linux all that well. For example, my one stab at kernel programming resulted in a bug that took three years to track down and fix, and even then it was done by someone hacking OS/2. I'm referring to the `sprintf` function inside the kernel.

I wrote `sprintf` in the summer of 1991. Linus needed some easy way to print messages from the kernel, and did not then know how to use the `stdarg` mechanism to implement variable argument lists to functions. I see some people in the audience shaking their heads. Yes, it's true. There was a time when Linus didn't know everything. Really. Trust me. I was there.

Anyway, I wrote a simple `sprintf` for Linus, to show him how it was done, and he used in the Linux kernel, after some modification. The bug was in the handling of an asterisk as the width for an output field. I forgot to increment a pointer past the asterisk, so the code had no chance of working for this case. No chance at all. This was clear to anyone who tried it. Obviously, I didn't, which means that I probably shouldn't be employed to write software. In 1994, three years later, one Friedemann Baitinger noticed the bug, and sent me a patch. He was using the `sprintf` while debugging a device driver he was writing for OS/2.

Anyway, back to who I am. I've been a friend of Linus since before Linux even existed. We met as first-

year students in 1988. When he started to write Linux, I naturally followed things with interest and some jealousy. Except for `sprintf`, I didn't really participate, since I'm not a hacker, just a wannabe. When Linux and its community grew, I took part in various non-technical things that needed doing. For example, I helped create the Linux Documentation Project, and co-moderated `comp.os.linux.announce`, also known as `cola`. My moderating `cola` is the reason I'm here. When I had been doing it for five years, that is, in last December, I decided to retire from it and found a successor. In my farewell note to the group, I jokingly said that I wouldn't mind getting a trip around the world, if anyone wanted to send me one. It happens that some people don't understand jokes, and it seems that Marc Ewing is one of them. He told me I would come here to give a talk and he wouldn't take no for an answer. If you feel bored now, you know whom to blame. Linus the god

You all know Linus, at least by reputation. The wonder-child. The coding wizard. The hacker god. Well, it wasn't always like that. What I'm about to say next may shock the most devout Linuxers in the audience, but that's all right. This is a free country, and anyway, I've been promised police protection.

I've already told you that Linus didn't always know everything. I'm not saying he isn't omniscient now. After all, he might now be a god and I'm not lightning-proof. So I'm only going to talk about old times. I'm sure he'll forgive me that.

Not only did Linus not know everything about C, he also didn't know anything about PC's. In fact, he didn't even have one when I first met him. When he bought his first PC, he didn't even start hacking it right away. Instead, he played computer games, especially one called something like *Prince of Persia*. I've never understood that part of him. I mean, what's a computer game worth if it doesn't simulate playing cards? No, give me *solitaire*, if you want me to play with a computer.

Even a few years later, when Linux was already a success, Linus had this strange fascination for silly computer games, such as *Doom* and *Quake*. By then he'd already learned some social skills and knew that one just doesn't admit to liking computer games after the age of 12. So when he was playing *Doom*, he used to explain that he was debugging and stress testing memory management and the X server.

When Linus decides to learn something, he really learns it, and usually quickly. This is why he may now be omniscient. I remember once when we were being questioned about some math home work. I happened to know Linus hadn't done it. But bold as he was even then, he claimed to have done them anyway. As luck would have it, the teacher wanted Linus to present his solution to the class. On the way to the blackboard, Linus read the problem, then stood in front of the board for a second or two, and went on to present a solution that the teacher couldn't understand. Linus can be quite annoying like that.

Once, when Linus was abroad at some conference or another, he modified my shell setup scripts so that when I logged in, it looked as if I was using MS-DOS. That was fun, of course, but it begged for revenge. This happened while we were sharing an office at the university, so once when Linus went out to get something to drink or something, I created an alias for `startx` for him. My alias first ran the real `startx`, and then printed out a kernel 'Oops' message. The first time Linus noticed this made him a bit worried, but he logged out and cleared the screen too fast to read it, but the second time made him really worried. I'd copied the 'Oops' message from `linux-kernel`, and of course it didn't suit Linus's kernel at all. He had gotten as far as decoding the message by hand, and muttering something like "Why is it crashing there? It can't crash there!", when I burst out laughing and told him what I'd done. Linus was quite relieved and never tried any practical jokes on me again. The early times

Let's go back to the spring of 1991. In January, Linus bought a PC. He'd been using a Sinclair QL before that, which, like much British computer stuff, was ingenious and almost unusably different from everything else. Like every self-respecting hacker, Linus had written some software development tools of his own; an

editor and an assembler, I think. He'd also modified the QL hardware a bit, to replace a broken keyboard, and to add a PC-compatible floppy drive. When he bought the PC, he wrote a device driver for the QL so he could move stuff from the QL to the PC.

When he got up to speed with the PC, after having played enough Prince of Persia, he started learning about programming the PC. Especially assembly language programming, since only wimps use high level languages. I remember one day when he was quite proud for having written a `strlen` function in assembly. Gee, I was impressed.

As I said, when Linus decides to learn, he really learns. A few weeks later he showed me two simple concurrent processes. One printed out A's as fast as it could, the other B's. That was much more fun to look at than his `strlen`, but not, I think, immediately as useful.

As time progressed, Linus added keyboard and serial port drivers, so that he could use his modem from the PC. Normal people would have used one of the dozens of existing terminal emulators, but Linus had to write his own. After that, he spent a long time just reading netnews. Sorry, I mean of course that he was debugging his terminal emulation code by reading netnews. The emulator consisted of two processes, one reading the keyboard and writing to the serial port, the other reading the serial port and writing to the screen and emulating a terminal.

At some point during this time Linus decided he wanted a Unix-like system at home, and the obvious choice back then was Minix, since it was the only thing he could afford. As it happened, Linus wasn't very happy with Minix, so he kept improving his terminal emulator, and modifying it to become more like an operating system. I guess we can conclude by now that he succeeded.

The success of Linux wasn't automatic, and things might well have gone differently. For example, if the Hurd had been finished a few years ago, Linux probably wouldn't exist today. Or the BSD systems might have taken over the free operating system marketplace.

However, things went as they did, and Linux prospered. The success has resulted in fame and also material rewards for rewards, including money. One of the first rewards wasn't money, but virtual beer. You may have heard the expression, since it is still used somewhat, but these days it is just a general good wish phrase. Originally, it had a very concrete meaning. Two guys from Oxford, England, calling themselves the Oxford Beer Trolls, wanted to buy Linus some beer, but since it was impractical to move either themselves, Linus, or the beer physically around, they asked me to receive the money via mail, and buy Linus beer with it, and that's what happened. So, virtual beer really means money, preferably money sent to me.

Alas, people started sending Linus money directly. I'm not sure they did it out of gratitude, however, since they usually sent personal checks from the US. As Linus quickly learned, Finnish banks really, really hate checks. Especially personal checks. Particularly personal checks from the US. They invent all sorts of bureaucratic pit-falls and rules and fees to make it difficult and expensive to use checks. If you want to make trouble for a Finn, send him a personal check from the US. And that's not a joke.

Linus also got some other stuff via mail. For example, a pair of 40 megabyte hard disks. That was really nice, since it meant that Linus was finally able to keep some backups. Not that he did, of course. One of his well-known quotes is: "Backups are for wimps. Real men upload their data to an FTP site and have everyone else mirror it." He said that even after dialling his hard disk.

At one point, Linus had implemented device files in `/dev`, and wanted to dial up the university computer and debug his terminal emulation code again. So he starts his terminal emulator program and tells it to use `/dev/hda`. That should have been `/dev/ttyS1`. Oops. Now his master boot record started with "ATDT" and the university modem pool phone number. I think he implemented permission checking the following day.

The name Linux was not coined by Linus himself, strange though that may seem to people familiar with his

self-esteem. It was coined by Ari Lemmke, the administrator at ftp.funet.fi who first made Linux available for FTP. Ari had to coin a name since Linus had failed to give a proper one, so Ari invented one and it stuck. A few days after Linux was put on ftp.funet.fi for the first time, Linus was bubbling with excitement. Ari had sent him the first download statistics for Linux, and there were literally tens of downloads! Ooh, the glory of success.

II.4 From: Linus Torvalds (torvalds@transmeta.com)

The other big news - well, for me personally, anyway - is that I've decided to take a leave-of-absence after 6+ years at Transmeta to actually work full-time on the kernel.

Transmeta has always been very good at letting me spend even an inordinate amount of time on Linux, but as a result I've been feeling a little guilty at just how little "real work" I got done lately. To fix that, I'll instead be working at OSDL, finally actually doing Linux as my main job.

[I do not expect a huge amount of change as a result, testament to just /how/ freely Transmeta has let me do Linux work. My email address will change to torvalds@osdl.org effective July 1st, but everybody is trying to make the transfer as smooth as possible, so we'll make sure that there will be sufficient address overlap etc to not cause any problems]

OSDL and Transmeta will have a joint official (read: "boring". You should have seen the bio - that didn't make it - that I suggested for myself for it ;) press-release about this tomorrow morning, but I just wanted to say thanks to Transmeta. It has been a special place to work for, and hello to OSDL that I hope will be the same.

Snif. I'm actually all teary-eyed.

Linus

II.5 Zitat

Software is like sex: it's better when it's free.

III Eric Steven Raymond

III.1 "Let My Software Go!" by Andrew Leonard

The first time I met Eric Raymond, the co-author of "The New Hacker's Dictionary," he flamed me hairless after I sent him e-mail seeking clarification of a point of research for a project I was working on.

"It sounds to me like you're way out of your technical depth here," his message concluded. "Any real hacker would have known the answer to your question without having to think. Get expert help, or you are likely to produce a book full of breathless nonsense."

Ouch. Stung, I flamed back, and for several days we traded e-mail invective before getting down to serious business. As Raymond told me later, I had run head on into his "idiot filter" - a technique he used to keep the clueless at arm's length. Eric Raymond's idiot filter is usually set pretty high.

My next encounter with Raymond came last fall, while I was reporting a story about the Apache Web server project and the free software movement. Raymond is an influential advocate of the principle of free software - which declares that the best way to produce quality software is to give the world free access to the underlying source code. Raymond's essay "The Cathedral and the Bazaar" is one of the most eloquent explications of the theory that software design is best served by having a community of independent hackers

work together in an atmosphere of complete openness. This time around, my exchange with Raymond was completely civil – I was inquiring about something he loved, something he passionately believed in. No flaming necessary.

Finally, I met him face to face, and discovered a man positively brimming with energy and ebullience. On April 1, Netscape – swayed, company officials say, by many of Raymond’s arguments in “The Cathedral and the Bazaar” – officially joined the free software movement by releasing the source code to Navigator 5.0. To mark the occasion, Raymond left his home in Malvern, Pa., and journeyed out to Silicon Valley – both to attend the release party and to participate in a “free software summit” that included nearly all of the most significant players in the free software game.

I took the opportunity to drive down to Mountain View and take him out to lunch. When I picked him up at the office he was temporarily working out of, I commented that right now must be an exciting time for him. Without a trace of irony, the 40-year-old hacker icon replied, “Oh yes – for my culture and my people, this is the moment we’ve been waiting for for 20 years.”

My culture and my people. The words are grandiose, but the extravagance is not unjustified. Raymond isn’t Moses – the free software movement doesn’t have any one leader – but the Netscape announcement has indeed given the hard-core geeks of the world a glimpse at the Promised Land. And no one defines the parameters “hard-core” and “geek” more efficiently than Eric Raymond.

A self-described neo-pagan libertarian who enjoys shooting semi-automatic weapons, Raymond fits the classic stereotype of the hacker almost too well. Hackers tend to think they know better; free software libertarian hackers tend to think they know best of all. As Raymond told me with pride: “I’m an arrogant son of a bitch.”

The world of hackerdom is full of arrogant sons of bitches, to be sure, but computers are more to Raymond than just a playground to flaunt his ego. In jokingly referring to his “Napoleon complex,” Raymond also told me he suffers from congenital cerebral palsy. That condition encouraged him to look upon computers as a realm in which he could exercise the kind of control denied him in the physical world.

How did you find out that Netscape had embraced the ideas in your essay?

On January 23rd, 1998, I was sitting at my machine, hacking away, fat, dumb and happy, and somebody sent me e-mail that said, gee Eric, I think you better go take a look at this Web page. I think somebody has been reading your paper. So I fired up my browser and sent it to the Netscape press release announcing [the release of the Navigator 5.0 source code]. And I looked at it and I thought, how interesting! Because not only did I immediately see that this was the break my culture had been waiting for for 20 years – but I also saw that a lot of the phrasing looked strangely familiar, as though someone had taken my paper and run it through the marketing meat grinder. An hour later, I got a phone call from Netscape’s chief of PR, and she gave me a 20-minute spiel which broke down to, yup, you have something to do with this, all of our top people read your paper and they loved it and [Netscape CEO] Jim Barksdale is out giving your name to reporters right now.

How did that make you feel?

Well, I hung up the phone, stumbled around in a daze for a while and then sat down, re-engaged my brain and started thinking. Several things were immediately clear to me. Thing No. 1: This is a colossal opportunity. For a very long time, for 20 years, as long as I’ve been involved with Unix and with GNU software and all that stuff – since back when the Internet was 500 techies in a sandbox and the rest of the world didn’t know or care who we were or what we were doing – we knew we had a better way to do things in our software designs and operating systems and the way that we shared work with each other. But we couldn’t get anybody to listen.

Netscape doing this creates a window of opportunity for us to get our message into corporate boardrooms. The flip side of that is that if Netscape tanks, no one is going to listen to us for another decade.

In some respects, Netscape's move seems to represent the resurgence of the old "gift economy" ideals of the Internet – the idea that the Internet could best move forward if everybody gave according to their ability, without necessarily expecting financial return. Has the gift economy influenced your own desire to play a role in the free software movement?

It's what I always wanted to do. The reason is really simple. Like most hackers, I don't really care about money very much. I do what I do primarily for artistic satisfaction, and what I want is to know that other people consider it good art. I mean, it's nice if I consider it good art, I generally know the difference and I can usually tell whether what I have done is beautiful or ugly. But you don't really know that you are evolving in the right direction unless reality and other people confirm that. So, like most other hackers, my most fundamental motivation is that I want other hackers to think that I'm doing good work. And I want them to believe I'm effective and fruitful and a good designer and so forth.

I have no problem with markets and exchange cultures but I don't really want to live there. And from 1977 on it was clear that there was a flourishing, growing gift culture out there on the Internet that was doing interesting things with software and I always wanted to be part of that. That's where I always wanted to live.

I notice that you no longer prefer to use the term "free software." Instead, you say "open source." Can you explain?

Sure. [After meeting with Netscape] I got together with a bunch of free software hackers and we had our own strategy conference. The issue on the table was how to exploit the Netscape breakthrough. We worked out some strategies and tactics. First conclusion: The name "free software" has to go. The problem is nobody knows what "free" means, and to the extent that they do think they know, it's tied in with a whole bunch of ideology and that crazy guy from Boston, Richard Stallman.

That's somewhat incendiary. [Richard Stallman, lionized in Stephen Levy's "Hackers: Heroes of the Computer Revolution" as the epitome of the true hacker, is the creator of one of the best-known examples of free software, GNU.]

I love Richard dearly, and we've been friends since the '70s and he's done valuable service to our community, but in the battle we are fighting now, ideology is just a handicap. We need to be making arguments based on economics and development processes and expected return. We do not need to behave like Communards pumping our fists on the barricades. This is a losing strategy. So in order to execute that, we needed a new label, and we brainstormed a bunch of them and the one that we finally came up with is "open source."

Why is it so important to have the source code available?

The central problem in software engineering has always been reliability. Our reliability, in general, sucks. In other branches of engineering, what do you do to get high reliability? The answer is massive, independent peer review. You wouldn't trust a scientific journal paper that hadn't been peer reviewed, you wouldn't trust a major civil engineering design that hadn't been independently peer reviewed, and you can't trust software that hasn't been peer reviewed, either. But that can't happen unless the source code is open. The four most critical pieces of infrastructure that make the Internet work – Bind, Perl, sendmail and Apache – every one of these is open source, every one of these is super reliable. The Internet would not function if they weren't super reliable, and they're super reliable precisely because throughout their entire history people have been constantly banging on the code, looking at the source, seeing what breaks and fixing it.

But what about intellectual property rights?

This is 180 degrees removed from any ideology about whether intellectual property rights are good or not. I don't care about that. I'm not interested in having that argument anymore. If your source is open, you

get peer review, you get reliability. If your source is not open, you don't get peer review and you don't get reliability, end of story.

On the open source Web pages, you set forth the reasons why programmers won't starve in a world where software is free. But is that really the issue? Does Netscape or Microsoft care if programmers starve or not? What does Netscape have to gain from going open source?

I have identified several business models for open source. The one that Netscape is working is called market position or loss leader. This is where you have some open source software out there, which you use to create market position for closed source. Since I'm not a fanatic, this is fine with me.

So there is a model in which closed source is appropriate?

I'm not a fanatic. I'm not a Richard Stallman. I'm not against closed source in absolute principle, I just think it's an inferior, shoddy way to do things most of the time. But I've sat down and thought about under what circumstances it makes sense to be closed vs. open. And I've identified a spectrum with two extremes of software that you might want.

On the one hand you have research intensive software. A real good example of that is something everyone is talking about right now – iris scanning for biometrics. It's a research intensive technology that depends on algorithms nobody else has, and it's only being prototyped in relatively small systems where reliability is not a huge concern. On the other end of the spectrum you have what I call implementational kinds of software. The paramount case of that is something like an office mailing list. All the techniques for running mailing lists are very well known. There is no knowledge in particular, there is no special algorithm – the big problems are robustness, reliability and scalability, and that situation is where the open source model really, really shines, because what you want at that point is massive peer review, to get your reliability.

The interesting thing to notice is that individual software technology is always moving from one end to the other of the spectrum. A perfect example is real-time 3-D animation. Five years ago, when [the computer game] "Doom" came out, that was a research intensive technology. Few people knew how to do it well and you could capture a lot of value by adding a new trade secret – it made sense to be closed. Come 1998 and lots of people know how to do it. There are several alternative packages out there, some of them are free, it's being implemented in larger and larger systems where again your problems are scalability, reliability and robustness, rather than just getting the details of the animation right.

Well, the implication of this is at that some point during the last five years the payoff curves crossed over – there came a point when the gain from peer review exceeded the gain from holding the software captive and having it be a trade secret. The interesting question is where is that crossover point? How do you identify that? My thinking now is that every software technology goes through the same evolution. I am beginning to think that this may be the fundamental software management question of the 21st century: Where is the crossover point? And I love to say these things to business people, because this is exactly the kind of optimization problem that gives them enormous erections. And if I can get them thinking in those terms, we've won.

You have some harsh words to say about the way business people have traditionally looked at the software marketplace.

The thing I realized when I sat down and thought about business models is that nobody thinks about the economics of software. Nobody thinks real hard.

Not even those Microsoft guys up in Redmond?

Not even them. Anybody who has studied software engineering knows that programmers do not actually spend most of their time originating software. They spend most of their time on service updates and maintenance. Nobody thinks about the implications of this: that the software industry is actually a service industry

operating under the delusion that it is a manufacturing industry. Software producers are operating under a manufacturing and cost model, under which the way you make money is building a product and getting it out the door. Because they have this model of themselves as a manufacturing industry, all the bright people go to production and the dumb people go to the support desk. That's why when you call a vendor support line you have to fight your way through three layers of idiots to get down to anyone who knows anything. As long as the software industry continues to misperceive itself as a manufacturing industry, instead of a service industry, reliability is going to be awful. But that shift is not going to happen until source is open. That's the difference between closed and open source.

In the closed source world, your short-term profit incentive is to try and keep everything you do a trade secret and extract the absolute maximum rent from that trade secret in terms of initial cost of the software. And then your economic incentive is to put as little money as you can get away with into supporting the fiction that you support your software. OK? Now as a consumer do you want to live in that world, or do you want to live in a world where source is primarily open and the people competing for your dollars are service bureaus? This is why I think that ultimately the closed software model and the whole Microsoft paradigm is doomed, because eventually software consumers are going to wake up and realize that they are being scammed – that the cost and pricing model of the software industry fundamentally does not fit the economics of the situation or the needs of consumers.

At the party Netscape threw for the release of their source code, Marc Andreessen was talking about getting companies like Sun and Oracle to join the free software crusade. Wouldn't this be just another case of Silicon Valley lining up its wagons against Microsoft?

Oh sure. One of the things that I tell corporate types is, "Hi there, Mr. CEO – tell me, do you have any strategic problem right now that is bigger than whether Microsoft is going to either crush you or own your soul in a few years? No? You don't? OK, well, listen carefully then. You cannot survive against Bill Gates playing Bill Gates' game. To thrive, or even survive, you're going to have to change the rules. I'm here to show you how."