Zurück zur Halloween Seite

{Eric S. Raymond: Halloween I -- 1.9}

Open Source Software

Eine (Neue?) Entwicklungs Methodologie

ID-Pro-Übersetzung 1.2

Wie man dieses Dokument liest:

Anmerkungen. der Übersetzer in %% (Übersetzung: ID-Pro-Mitarbeiter und Martin Leutz). Kommentare von Eric S. Raymond sind in Grün bzw. kursiv gedruckt, umgeben durch geschweifte Klammern. Die wichtigsten Orginal-Aussagen sind in rot markiert bzw fett gedruckt.

{Der Inhalt dieses unter dem Namen Halloween bekanntgewordenen Dokumentes ist ein internes Strategie-Papier für Microsofts mögliche Antworten auf das Linux-/Open-Source Phänomen.

(Diese kommentierte Version wurde in »Halloween I« umbenannt; da es eine Fortsetzung gibt »Halloween II«, die sich stärker auf Linux bezieht).

Microsoft hat öffentlich bestätigt, daß dieses Papier authentisch ist, es aber gleichzeitig als technische Studie abqualifiziert, die nicht maßgeblich für die Firmenpolitik Microsofts sei.

Allerdings haben - laut der an Ende aufgeführten Liste der Mitwirkenden - einige der führenden Köpfe von Microsoft mitgewirkt, und das Dokument liest sich, als ob diese Untersuchung die Unterstützung des obersten Managements hatte. Es könnte sogar als Strategieentwurf zur Vorlage bei Bill Gates gedacht sein (der Autor .zu sein, daß Gates es lesen würde).

Auf jeden Fall erlaubt es das Papier einen wertvollen Blick auf das, was die Firma wirklich von Open Source hält, unabhängig von Microsofts abwertender Marketingstrategie - was, wie Sie sehen werden, eine seltsame Kombination aus Scharfsinn und Betriebsblindheit ist.

Es bestehenden Spekulationen, dieses Protokoll sei absichtlich durchgesickert, doch dies scheint ziemlich unwahrscheinlich. Das Papier ist dafür zu vernichtend für Microsoft. In Teilen könnte es dem vom Justizministerium angestrebtem Prozeß als Beweis für wettbewerbswidriger Aktivitäten dienen. Außerdem wollte der Autor den Inhalt weder bestätigen noch dementieren, als er auf das Papier angesprochen wurde - was vermuten läßt, daß Microsoft keine Erklärung in der Schublade hatte.

1. *Da der Autor meine Analysen über die Dynamik der Open-Source-Gemeinschaft (*Die Kathedrale und der Basar The Cathedral and the Bazaar) *und* Homesteading theNoosphere Homesteading the Noosphere*) ausgiebig zitiert, scheint es fair zu sein, wenn ich im Namen der Gemeinschaft antworte. :-)*

Wesentliche Zitate

Hier sind einige bemerkenswerte Aussagen des Dokumentes. Nützlich zu wissen: ``OSS '' ist die Abkürzung des Autors für ``Open-Source-Software ''. FUD steht für Fear, Uncertainty, Doubt (Angst, Unsicherheit, Zweifel) - eine typische Taktik von Microsoft, die hierzitiert wird.

OSS stellt eine unmittelbare, kurzfristige Gewinn- und Plattformbedrohung für Microsoft dar, besonders im Serverbereich. Zusätzlich hat die systembedingte Parallelität und der freie Ideenaustausch bei OSS große Vorteile, die sich mit unserem derzeitigen Lizensierungsmodell nicht kopieren lassen - daher besteht durch die Verknüpfung der Entwicklerkreativität eine langfristige Gefahr.

Neue Fallstudien (im Internet) liefern sehr eindrucksvolle Beweise.., daß kommerzielle Qualität durch

OSS-Projekte erzielt werden / überstiegen werden kann.

Um zu verstehen, wie gegen OSS vorzugehen ist, müssen wir uns also nicht auf eine Firma, sondern auf einen Prozeß konzentrieren.

OSS ist langfristig Glaubwürdig... FUD-Taktiken können nicht verwendet werden, um es zu bekämpfen.

Linux und andere OSS-Vertreter sind ein immer glaubwürdigeres Argument dafür, daß OSS-Software mindestens so stabil -- wenn nicht stabiler -- ist wie kommerzielle Alternativen. Das Internet liefert ein ideales, weithin sichtbares Schaufenster für die OSS-Welt.

Linux wurde in praxisrelevanten kommerziellen Umgebungen eingesetzt und hat eine breite öffentliche Zustimmung gefunden... Linux stellt viele andere UNIX-Varianten in den Schatten ... Linux ist auf dem besten Weg, letztendlich den x86 UNIX-Markt zu dominieren.

Linux ist so lange im Vorteil, wie die Dienstleistungen / Protokolle Allgemeingut sind

OSS-Projekte waren aufgrund der weitreichenden Nutzbarkeit weitgehend frei vertriebener und einfacher Protokolle in der Lage, in vielen Server-Anwendungsgebieten einen Anteil zu erobern. Indem wir diese Protokolle erweitern und Neue entwickeln, können wir OSS-Projekten den Markteintritt verwehren.

Die Fähigkeit der OSS-Prozesse, die kollektive Intelligenz von Tausenden Individuen über das Internet zu bündeln und zu kanalisieren, ist schlicht verblüffend. Wichtiger jedoch ist, daß die Verbreitung von OSS mit der Größe des Internets wächst, wesentlich schneller, als unsere eigenen Verbreitungsbemühungen zu wachsen scheinen.

Ich habe ansonsten den Rest des Dokumentes vollständig belassen (noch nicht mal Schreibfehler korrigiert). Also können Sie lesen, was Bill Gates über Open Source liest. Es ist ein wenig lang, aber Sie sollten sich die Zeit nehmen. Eine genaue Betrachtung der gegnerischen Denkweise ist die Mühe wert -- und es gibt ein oder zwei wirklich aufsehenerregende Einblicke in die Unternehmenskultur.

Bedrohungseinschätzung

Ich glaube, die weitaus gefährlichste Taktik, die in diesem Papier befürwortet wird, ist jene, die unter der düsteren Bezeichnung ``Vereinnahmung/Ent-Standartisierung von Protokollen '' läuft. Wenn die Veröffentlichung dieses Dokumentes sonst nichts bewirkt, so hoffe ich, daß es zumindest jedem klarmacht, was diese Taktik bedeutet: der Wettbewerb wird unterdrückt, die Verbraucherwahl eingeschränkt, höhere Kosten und die Auslieferung an ein Monopol.

Die Parallelen zu Microsofts Versuchen Java zu »kapern« und den Versuchen, das ``einmal programmieren, überall laufen''-Potential dieser Technologie zu sabotieren sollten unverkennbar sein. Um zu verhindern, daß diese Taktik zum Erfolg führt, müssen Open-Source Projekte folgende Punkte beherzigen, wie ich glaube:

1. Kunden mögen einen Markt mit frei erhältlicher Ware. Verkäufer nicht.

2. Allgemein verfügbare Dienste und Protokolle sind gut für Kunden; sie sind weniger teuer, sie fördern den Wettbewerb, sie führen zu einer guten Wahl.

3. Die Protokolle exklusiv zu machen bedeutet, die Auswahl einzuschränken, die Preise zu erhöhen und den Wettbewerb zu unterdrücken.

*4. **Daraus folgt:** Damit mit Microsoft gewinnt, **muß der Kunde verlieren.***

*5 **Open Source fördert - und mehr: beruht auf -- allgemein zugänglichen Diensten und Protokollen. Es steht folglich im Einklang mit den Verbraucherinteressen.***

Geschichte

Die erste (1,1) kommentierte Version des VinodV-Papiers wurde über das Wochenende von 31 Oct-1 Nov. 1998 vorbereitet. In Anerkennung des Datums und meiner inbrünstigsten Hoffnung, daß die Veröffentlichen dazu beiträgt, Microsofts schlimmste Alpträume wahr werden zu lassen, habe ich das Dokument als ``Halloween-Dokument '' benannt. In der Version 1,2 sind die Zeichen entfernt, die nicht dem ASCII-Code entsprechen..Die Version 1,3 enthält Microsofts Eingeständnis der Echtheit des Dokumentes.Die Version 1,4 fügte ein wenig mehr Analyse in das Kapitel der Bedrohungseinschätzung hinzu.Die Version 1,5 fügte einiges zur Einleitung hinzu. Die Version 1,6 fügte einiges . hinzu.Die Version 1,7 fügte die Referenz zu den FUZZ-Papieren hinzu.In der Version 1,8 kam ein Link zu dem . Dokument hinzu.Die Version 1.9 fügt eine Notiz zu der . (.Untersützung!) hinzu}

Vinod Valloppillil (VinodV)
Aug 11, 1998 -- v1.00
Microsoft Vertraulich

Inhaltsverzeichnis

Kurzfassung

Open Source Software

- Was ist es?
- Software Lizenzen
- Open Source Software betrifft Microsoft
- Geschichte

Open Source Prozeß

- Open Source Entwicklungs-Teams
- Koordinierung von OSS-Entwicklung
- Parallele Entwicklung
- Paralleles Beseitigen von Fehlern
- Konfliktlösung
- Motivation
- Code-Spaltung

Open Source Stärken

- OSS Herausragende Eigenschaften
- Langfristige Vertrauenswürdigkeit
- Paralleles Ausmerzen von Fehlern
- Parallele Entwicklung
- OSS gleich 'perfekte' API-Verbreitung / Dokumentation
- Veröffentlichungshäufigkeit

Open Source Schwächen

- Managementkosten
- Prozeßthemen
- Organisatorische Vertrauenswürdigkeit

Open Source - Geschäftsmodelle .

- Sekundäre Dienstleistungen .
- Führung bei Markteintritt unter Inkaufnahme von Verlusten .
- Geringes verpflichtendes Supportangebot .
- First Mover Prinzip-Jetzt Bauen, Später Kassieren

Linux

- Was ist das?

Open Source Software

Eine (neue?) Art der Entwicklung

Kurzfassung

Open Source Software (OSS) ist ein Entwicklungsprozeß, der schnelle Erstellung und Umsetzung erweiternder Elemente und Fehlerkorrekturen eines existierenden Codes / einer existierenden Wissensbasis fördert. In den letzten Jahren, korrespondierend mit dem Wachstum des Internets, erwarben sich OSS Projekte eine Komplexität und Reife, wie sie traditionell nur kommerziellen Produkten wie Betriebssystemen und anwendungskritische Server zugeschrieben werden.

OSS stellt damit eine unmittelbare, kurzfristige Gewinn- und Plattformbedrohung für Microsoft dar, besonders im

Serverbereich. Zusätzlich hat die tatsächliche Parallelität und der freie Ideenaustauschs bei OSS große Vorteile, die sich mit unserem derzeitigen Lizensierungsmodell nicht kopieren lassen - daher besteht durch die Verknüfung der Entwicklerkreativität eine langfristige Gefahr.

{ OK, dies macht deutlich, daß Microsoft nicht schläft. }

Jedoch bieten gewisse Schwächen des OSS-Prozesses eine Möglichkeit für Microsoft, einen Vorteil in zentralen Bereichen wie der architektonischen Verbesserung (z.B. Storage+), Integration (z.B. Schemata), Benutzerfreundlichkeit und organisatorischem Support zu erlangen.

{ Diese zusammenfassende Empfehlung ist hauptsächlich deshalb interessant, weil sie es versäumen, die später folgenden Vorschläge zur . zu erwähnen }

Open Source Software: Was ist das?

Open Source Software (OSS) ist Software, bei welcher Binärdateien und Quellcode eines Produktes verbreitet oder zugänglich gemacht werden, üblicherweise kostenfrei. OSS wird oft als Shareware oder Freeware mißverstanden. Aber es gibt signifikante Unterschiede zwischen den Lizenzmodellen und Prozessen, die um jedes Produkt ablaufen.

Software Lizenzen

Lizenz Features	Null-Preis	Neuverteilbar	Unbegrenzter Gebrauch	Quell code Vorhanden	Quellcode Modifizierbar	öffentliche Check-Ins zur Erstel-lung einer Codebasis	Alle Ableitungen müssen frei sein
Software-Art							
Kommerziell							
Probe Software	X (Nicht voll unterstützt)	X					
Nicht-kommerzieller Gebrauch	X (Verbrauchsabhängig)	X					
Shareware	X- (nicht erzwungene Lizenz)	X					
Lizenzfreie Binaries (" Freeware ")	X	X	X				
Lizenzfreie Libraries	X	X	X	X			
Open Source(BSD-Style)	X	X	X	X	X		
Open Source(Apache-Style)	X	X	X	X	X	X	
Open Source(Linux-/GNU-Style)	X	X	X	X	X	X	X

Die umfassenden Lizenzkategorien beinhalten:

Kommerzielle Software

Kommerzielle Software ein klassisches Microsoft-Feld. Sie muß gekauft werden, darf NICHT! weiterverteilt werden und ist typischerweise lediglich als Binärdatei für den Endbenutzer verfügbar.

Limitierte Probesoftware

Limitierte Probesoftware ist gewöhnlich in ihrer Funktion gegenüber der kommerziellen Version eingeschränkt. Sie wird jedoch frei zur Verfügung gestellt und zielen darauf ab, vom Kauf der kommerziellen Version zu überzeugen. Viele solcher Produkte beinhalten eine 60-Tage Lizenz zum testen des Produktes.

Shareware

Shareware Produkte sind voll funktionstüchtige und frei verteilbare Programme, haben aber eine Lizenz, die private und kommerzielle Anwender schließlich erwerben sollen. Viele Internet-Utilities (wie WinZip) nutzen die Vorteile der Sharewareverteilung.

Nicht kommerzielle Benutzung

Nicht kommerzielle Software nicht profit-orientierte Nutzer frei. Firmen etc. müssen das Produkt kaufen. Ein Beispiel dafür ist der Netscape Navigator.

Lizenzfreie Binaries

Lizenzfreie Binaries ist Software, welche in binärer Form frei genutzt und verteilt werden darf. Internet Explorer und Netmeeting sind Beispiele hierfür.

Lizenzfreie Libraries

Lizenzfreie Libraries(Bibliotheken) sind Softwareprodukte, dessen Binaries und Quellcode frei erhältlich und benutzbar sind, aber NICHT! durch den Benutzer verändert werden dürfen. Beispiele hierfür sind Libraries, header files usw.

Open Source (BSD-Art)

Ein kleines geschlossenes Team von Entwicklern entwickelt BSD-typische Open Source Produkte und erlaubt dessen freien Gebrauch und Weiterverteilung von Binaries und Quellcode. Obwohl die Benutzer den Code modifizieren dürfen, übernehmen die Entwickler in der Regel keine der Modifikationen in das Produkt.

Open Source (Apache-style)

Apache übernimmt das Lizenzmodell von BSD - alleridngs werden hier die Überarbeitungen von dritten mit in die Codebasis übernommen.

Open Source (CopyLeft, Linux-style)

CopyLeft oder GPL (General Public License)-basierte Software geht einen entscheidenden Schritt weiter. Während BSD und Apache es erlauben, die Codebasis aufzuspalten und die eigenen Modifikationen wiederum zu lizensieren (um es z.B. kommerziell zu nutzen), verlangt die GPL Lizenz, daß alle Codeableitungen wiederum unter der GPL Lizenz stehen müssen. 'Du bist frei, den Code zu verändern, solange andere Deinen Code auch verändern können.'

{ Es ist interessant zu sehen, wie weit diese letzten drei Unterscheidungen davon entfernt sind, wie die Open-Source-Community sie im Allgemeinen auffaßt. Für uns sind Open-Source-Lizensierung und die Rechte, die sie den Nutzern und Dritten einräumt, das Entscheidende, und die spezifische Entwicklungspraxis variiert adhoc auf die eine oder andere Weise, die nicht gesondert an unsere verschiednen Lizensierungsvarianten gebunden ist. In der Welt von Microsoft ist es in Gegensatz dazu von zentraler Bedeutung, wer die zentrale Code-Basis verändern darf. Das ist ein Spiegelbild einer wesentlich zentralistischeren Sichtweise und verdeutlicht, wie die Vorstellungskraft oder das Verständnis des Autors scheitern. Er begreift unsere Tradition verteilter Entwicklung nicht vollständig. Dies ist wenig überraschend.... }

Open Source Software betrifft Microsoft

Dieses Dokument konzentriert sich auf Open Source Software (OSS). OSS unterscheidet sich von den anderen Lizenztypen (z.Bsp. Shareware) in zwei wichtigen Punkten:

A. Es existiert immer eine Möglichkeit Lizenz-frei die zugrundeliegende Codebasis zu kaufen.

B. Im Gegensatz zu frei verteilten Binaries fördert OSS den *Prozess* rund um die Codebasis und ermutige Entwickler, die Codebasis zu erweitern und anderen Entwicklern zur Verfügung zu stellen.

OSS betrifft Microsoft in mehreren Punkten:

1. **OSS-Projekte haben " kommerzielle Qualität " erzielt**
   Eine wesentliche Barriere von OSS war für viele Benutzer die als fehlerhaft wahrgenommene Qualität. OSS vertritt die Ansicht, daß die kontinuierliche Codeinspektion und -Fehlerbeseitigung bei OSS Software in einer höheren Qualität gegenüber kommerzieller Software resultiert.

   Neue Fallstudien (im Internet) liefern in den Augen der Kunden sehr dramatische Beweise, daß kommerzielle Qualität durch OSS Projekte erzielt werden / überstiegen werden kann. Gleichwohl gibt es bisher keinen überzeugenden Beweis für die OSS Code-Qualität, abgesehen von Gerüchten.

   {Diese Sätze zusammengenommen sind sehr widersprüchlich, außer die 'aktuellen Fallstudien' sind allesamt Gerüchte. Aber wenn dies so ist, warum sie dann als 'dramatische Beweise' bezeichnen ? Es scheint, daß hier ein wenig Selbstschutz und Rückendeckung in den Zweiten Satz gebracht werden soll. Nichtsdestoweniger ist der erste Satz ein enormes Eingeständnis seitens Microsoft (auch intern). Auf jeden Fall ist die »Gerüchte« -Behauptung falsch. Siehe: Fuzz Revisited: A Re-examination of the Reliability of UNIX Utilities and Services .. *Hier sind 3 relevante Zeilen dieses Dokumentes: " die Störungsanfälligkeit von Hilfsprogrammen auf den kommerziellen Versionen von UNIX, die wir getestet haben ... reicht von 15-43%. ", "die Störanfälligkeit der Hilfsprogramme auf der frei-verfügbaren Version Linux von UNIX war am zweitniedrigsten mit 9%. ", "die Störungsanfälligkeit der öffentlichem GNU-Hilfsprogramme war in unserer Studie mit nur 7% die niedrigste."}*

2. **OSS-Projekte wurden umfangreich und komplex**
   Ein anderes Hindernis für den Zugang, das von OSS überwunden wurde, ist die Komplexität der Projekte.
   OSS-Teams nehmen sich Projekten an, deren Größe u. Komplexität früher das exklusive Gebiet der kommerziellen, ökonomisch organisierten/motivierten Entwicklungsteams war. Beispiele schließen das Betriebssystem Linux und die Xfree86 Benutzeroberfläche ein.
   OSS-Prozeßvitalität ist direkt an das Internet gebunden, um ungeheure Mengen an Entwicklungs-Resourcen zur Verfügung zu stellen. Einige Beispiele von OSS-Projektgrößen:

Projekt	Anzahl Code-Zeilen
Linux Kernel (nur x86)	500.000
Apache Web Server	80.000
SendMail	57.000
Xfree86 X-Windows Server	1,5 Million
"K" Desktop Umgebung	90.000
Volle Linux Distribution	~10 Million

OSS hat einen einzigartigen Entwicklungsprozeß mit einzigartigen Stärken und Schwächen.

Der OSS-Prozeß ist in Bezug auf die Motivationen seiner Teilnehmer und die einsetzbaren Resourcen zur Bewältigung eines Problemes einzigartig. OSS hat folglich einige interessante, nicht kopierbare Vorteile, welche gründlich verstanden werden sollten.

Geschichte

Open Source Software hat ihre Wurzeln bei den Hobbyisten und der wissenschaftlichen Gemeinde. Sie wurde geprägt durch den spontanen Austausch von Quellcode zwischen Entwicklern/Benutzern.

Internet-Software

Die größte Fallstudie über OSS ist das Internet. Das gößte Teil des frühesten Codes des Internet war und ist noch

auf OSS basierend, wie Tim O'Reilly in einem Interview beschreibt (http://www.techweb.com/internet/profile/toreilly/interview):

TIM O'REILLY: Die wichtigste Nachricht, mit der wir begannen war, " Open Source Software funktioniert", BIND hatte den absolut dominierenden Marktanteil als das erste Anwendungskritische Softwareprodukt im Internet. Apache ist der dominierende Webserver. SendMail läuft vermutlich auf achtzig Prozent der Mailserver und behandelt vermutlich jede einzelne E-mail auf dem Weg durch das Internet

Free Software Foundation / GNU-Projekt

Die Anerkennung für das erste Auftauchen moderner, organisierter OSS gebührt im allgemeinen Richard Stallman vom MIT. Ende 1983 gründete Stallman die Free Software Foundation http://www.gnu.ai.mit.edu/fsf/fsf.html mit dem Ziel, eine kostenlose Version des UNIX Betriebsystemes zu erstellen. Die FSF veröffentlichte eine Reihe von Quellen und Binaries unter der GNU Lizenz. (was für »GNU's Not Unix« steht)

Die ursprünglichen FSF- / GNU-Initiativen verfehlten ihr ursprüngliches Ziel, ein vollständiges OSS Unix zu schaffen. Sie trugen jedoch einige wohlbekannte und weit verbreitete Anwendungen und Programmierhilfsmittel bei, welche heute noch benutzt werden.:

GNU Emacs -- ursprünglich ein leistungsfähiger Zeichen-Modus-Editor. Im Lauf der Zeit wurde Emacs zu einem Front-End für Compiler erweitert. Außerdem wurden ein E-Mail-Reader und einiges mehr zur Verfügung gestellt.

Compiler GNU C (GCC) -- GCC ist der meistgenutzte Compiler in der akademischen und der OSS-Welt. Zusätzlich zum Compiler ist ein weitgehend standardisierter Satz von Bibliotheken als Superset der ANSI-C Bibliotheken vorhanden.

GNU GhostScript -- Postskript Printer/Viewer.

CopyLeft-Lizensierung

FSF-/GNU Software stellte das Lizenzverfahren des 'CopyLeft' vor, demzufolge es nicht nur illegal ist, den Quellcode von GNU-Software zurückzuhalten, sondern der es auch untersagt, den Quellcode von Weiterentwicklungen und Derivaten von *GNU-Software* geheimzuhalten. Das Dokument, das dieses Lizenzverfahren beschreibt, ist als die General Public Lizenz (GPL) bekannt.

Das Wired Magazin bietet folgende Zusammenfassung dieses Entwurfs u. der dahinter stehenden Absicht (http://www.wired.com/wired/5.08/linux.html):

> *Die General Public Lizenz, oder GPL, erlaubt es Benutzern, CopyLeft-Programme - die durchaus auch einem Urheberschutzt unterliegen können - zu verkaufen, zu kopieren, und zu ändern. Sie müssen lediglich anderen die gleiche Freiheit einräumen, die von ihnen vorgenommenen Veränderungen zu kopieren, verkaufen oder weiterentwickeln. Sie müssen den Quellencode ihrer Änderungen ebenfalls frei zur Verfügung stellen.*

Die zweite Klausel -- der frei zugängliche Quellcode von abgeleiteten Arbeiten -- war der umstrittenste (und, möglicherweise erfolgreichste) Aspekt der CopyLeft-Lizensierung

.

Der Open Source Prozeß

Der Prozeß kommerzieller Softwareentwicklung ist geprägt durch die ihm zugrundeliegenden ökonomischen Ziele. Anders bei der Open Source Software, wo der gewünschte kommerzielle Erfolg nicht im Vordergrund steht. Und die Bedrohung richtig zu verstehen, ist daher ein gründliches Verständnis von Arbeitsweise und Motivation der

Open Source-Entwickler notwendig.

In anderen Worten: Um zu verstehen, wie gegen OSS vorzugehen ist, müssen wir uns eher auf eine Entwicklungskonzept als auf eine bestimmte Firma konzentrieren.

{ Dies ist eine sehr wichtige Einsicht, bei der ich mir wünschte, Microsoft hätte sie nicht erlangt. Die eigentliche Auseinandersetzung ist nicht NT gegen Linux oder Microsoft gegen RedHat/Caldera/S.u.S.e etc.-- Es ist Closed Source-Entwicklung gegen Open-Source %Anm. d. Übers.: Closed-Source-Entwicklung = Klassische Softwareentwicklung mit geheimgehaltenen Quellcodes%. Die Kathedrale gegen den Bazar.

Das trifft umgekehrt natürlich auch zu, weshalb Kampf gegen Microsoft nur, weil es Microsoft ist, das eigentlichen Problem nicht trifft -- Microsoft ist nur ein Symptom, nicht die Seuche selbst. Ich wünschte, mehr Linuxentwickler würden dies verstehen.

In der Praxis bedeutet das: Wir müssen damit rechnen, daß die Propagandamaschinerie von Microsoft sich gegen das Konzept und das Prinzip des Open Source richten wird, nicht gegen einzelne Firmen. Also, wappnet euch...

Open-Source Entwickler - Teams

Einige der Schlüsselattribute Internet-basierter OSS-Teams:

Geographisch weit verteilt. Einige der Hauptentwickler von Linux sind gleichmäßig über Europa, USA und Asien verteilt.

{ Es ist sehr interessant, daß der Autor dies bemerkt, aber weder auf den Vorsprung von Linux im Bereich der Internationalisierung eingeht noch überlegt, inwiefern der Erfolg von Linux insbesondere in Europa durch die Furcht vor allzu dominanter Stellung US-Amerikanischer Produkte zu erklären ist. Diese Auslassung zeigt vielleicht eine nutzbare Lücke in Microsofts Strategie.}

Eine große Anzahl von Unterstützer mit einer kleineren Menge von Kern-Entwicklern. Über 1000 Personen haben, um das Beispiel Linux noch mal zu nennen, Patches, Fehlerkorrekturen, usw. beigetragen, und mehr als 200 Einzelpersonen tragen unmittelbar zum Code des Betriebssystemkerns bei.

Kurzfristig keine finanziellen Motive. Diese Individuen betrachten Linux eher als Hobby, dem sie freie Zeit und Energie widmen, während sie gleichzeitig anderen Vollzeittätigkeiten nachgehen. Dies ändert sich nun langsam, seit kommerzielle Versionen von Linux auftauchen.

Koordinierung von OSS-Entwicklung

Kommunikation - Internet-Dimensionen

Die Koordination eines OSS Teams ist extrem abhängig von den Internet-eigenen Formen der Zusammenarbeit. Die typischerweise eingesetzte Methoden nutzen die gesamte Palette der Möglichkeiten, die das Internet bietet:

Email-Listen

Newsforen

24x 7h Überwachung durch internationale Teilnehmer

Web-Seiten

OSS Projekte von der Größe eines Linux und Apache sind nur dann lebensfähig, wenn eine ausreichend große Gemeinschaft von hochqualifizierten Entwicklern angesammelt werden kann, um das Problem anzugehen. Als Konsequenz daraus ergibt sich ein direkter Zusammenhang zwischen der Größe einer Aufgabe, die OSS bewältigen kann, und dem Wachstum des Internets.

Gemeinsame Richtung

Zusätzlich zum Kommunikationsmedium bestimmt eine Anzahl weiterer Faktoren die Richtung eines Teams.

Gemeinsame Ziele

Gemeinsame Ziele sind das Äquivalent zur Zielformulierung eines Unternehmens, die sich als Grundlage der Entscheidungsfindung des gesamten Entwicklungsteams versteht. Eine einzelne, klare Direktive (z.B. »Unix neuerschaffen«> ist weitaus effektiver zu kommunizieren und von einem Team umzusetzen, als mehrere, schwer fassbare Formulierungen (z.B. »ein gutes Betriebssystem bauen«).

Gemeinsame Vorbilder

Vorbilder %Anm. d. Übers.: Wörtlich: »Präzedenzfall« Gemeint ist die Vorgehensweise der Open-Source-Community, bestehende Konzepte zu untersuchen und Ihre Vorteile zu übernehmen.% sind möglicherweise der wichtigste Faktor, wenn es darum geht, das schnelle und stetige Wachstum umfangreicher OSS Projekte wie des Linux Betriebssystems zu erklären. Weil die gesamte Linux-Gemeinschaft jahrelange gemeinsame Erfahrung im Umgang mit vielen anderen Formen von Unix hatte, waren sie sehr leicht in der Lage, zu unterscheiden - ohne Streit --, was funktionierte und was nicht.

Es gab keine Auseinandersetzungen um die im Texteditor zu verwendende Befehlssyntax -- jeder verwendete bereits 'VI' und die Entwickler nahmen einfach Teile dieser Befehlsnamen und entwickelten sie weiter.

Ein geschichtlicher Rückblick mit dem Wissen von heute liefert eine starke implizite Struktur. In zukunftsorientierten Organisationen hingegen wird diese Struktur durch eine starke, visionäre Führung gewährleistet.

{ Auf den ersten flüchtigen Blick liest sich das wie eine devote Huldigung an Bill von jemandem, der erwartet, daß Gates das Memo liest -- Man kann fast den Autor sehen, wie er vor seinem Alter des furchtlosen Führers kniet.

Auf den zweiten Blick zeigt es vielleicht eine ernsthafte und potentiell nutzbare Unterschätzung der Möglichkeiten der Open-Source-Gemeinschaft an, eigene visionäre Führer hervorzubringen. Wir haben Emacs, Perl oder das World Wide Web auch nicht durch `Rückblick` bekommen. - und es ist auch falsch, selbst bei der relativ konservativen Beschaffenheit des Linux-Betriebssystemkerns von einer rückwärtsgewandten Wiederauflage alter Konzepte zu sprechen.

Dementsprechend legt es nahe, daß Microsofts Antwort auf Open-Source gekontert werden kann, indem wir dem Rest der Welt gegenüber die Innovationskraft sowohl bei den Produkten als auch bei der Arbeitsweise hervorheben.}

Gemeinsame Wissensbasis

NatBro unterstreicht die Notwendigkeit einer Basis allgemein akzeptierter Grundkenntnisse als zwingende Voraussetzung für OSS-Entwicklung. . %Anm. d. Übers.: gemeint ist sozusagen ein Handwerkszeug, daß jeder Bastler haben muß - Martin Leutz%. Dieser Punkt steht in einem engen Zusammenhang zu dem weit verbreitetem "Vorbild"- Phänomen. Aus seiner E-Mail:

Ein Schlüsselattribut ist das verbreitete UNIX/Gnu/make Rüstzeug, dessen sich auch OSS bedient und es noch verstärkt. Ich denke, daß der ganze Prozeß nicht funktionieren würde, wenn die Zugangsbeschränkungen höher wären, als sie sind ein mittelmäßig begabter UNIX-Programmierer hat somit auch die Chance, großartige Dinge mit Linux und vielen OSS-Produkten zu tun. Anders gesehen: Ein Entwickler im OSS-Bereich kann sich ohne weiteres betätigen, da Dinge sehr ähnlich konzipiert sind, die Fehlerkorrektur auf gleiche Weise funktioniert, usw.

Während also die Vorbilder das Endziel aufzeigen, definiert das Prinzip einer gemeinsamen Wissenbasis (eines gemeinsamen Baukastens) die Anzahl von Personen, die sich in dem Prozeß auskennen müssen, um das aufgezeigte Ziel zu erreichen .

Die Kathedrale und der Basar

Ein sehr einflußreiches Papier eines Open-Source Fürsprechers -- Eric Raymond -- wurde im Mai 1997 (http://www.redhat.com/redhat/cathedral-bazaar/) publiziert. Raymonds Papier lieferte dem (damaligen) Netscape CTO %Anm. d. Übers.: Chief Technical Officer - Technikchef %. Eric Hahn, wie dieser mehrfach betonte, umgehend die Begründung für die Entscheidung, den Quelltext des Browsers freizugeben.

Raymond gliedert sein OSS-Projekt auf, um zu Faustregeln zu kommen, die von anderen OSS-Projekten in Zukunft genutzt werden können. Einige von Raymond's Regeln sind:

Am Anfang eines jeden guten Stückes Software steht, daß ein Entwickler etwas ändern will, was er immer schon machen wollte:

Dies faßt eine Kernmotivation der Entwickler im OSS Prozeß zusammen -- das Lösen eines aktuelles Problems, das ein einzelner Entwickler hat - dies hat dem OSS erlaubt, große, komplexe Projekte zu entwickeln ohne eine konstantes Rückkoppelung einer Marketing- oder Vertriebsorganisation.

Gute Programmierer wissen, was neu entwickelt werden muß. Großartige wissen, was umzuschreiben ist (und was wiederverwendet wird).

Raymond stellt die These auf, daß Entwickler in einem strikten OSS-Prozeß eher dazu neigen, Code wiederzuverwenden, als in einem traditionelleren Entwicklungsumfeld, da sie jederzeit garantierten Zugang zum gesamten Quelltext haben. Allgemein verfügbarer, frei zugänglicher Quelltext reduziert somit den Aufwand, ein bestimmtes Code-Fragment zu finden.

``Plane, Entwürfe wegzuwerfen, Du wirst es sowieso tun.",

Zitat aus Fred Brooks >>The Mythical Man-Month<<, Kapitel 11. Weil OSS-Entwicklungs-Teams häufig extrem weit verteilt sind, gab es für viele wichtige Subkomponenten von Linux mehrere Prototypen, aus denen Linus schließlich durch Auswahl und Verfeinerung ein einzelnen Design ausgewählt hat.

Die Benutzer als Mit-Entwickler zu behandeln ist der Weg des geringsten Widerstandes zur schnellen Codeverbesserung und zum wirkungsvollen Suchen von Fehlern.

Raymond befürwortet ausführliche Dokumentationen und spürbare Entwickleruntertützung in OSS-Projekten, um ihre Vorteile zu maximieren. .Code Dokumentation wird als Bereich zitiert, den kommerzielle Entwickler gewöhnlich vernachlässigen, was in OSS ein fataler Fehler wäre.

Veröffentliche frühzeitig. Veröffentliche häufig. Und höre auf Deine Kunden.

Dies ist ein klassisches Vorgehen nach dem Microsoft-Handbuch. OSS Befürworter werden jedoch anmerken, daß sie tendenziell wesentlich schneller eine Rückmeldung auf ihre Veröffentlichungen bekommen, also dies bei kommerzieller Software der Fall ist .

*{ Dies ist eine interessant arrogante Aussage - derart, als sei *ich* in irgendeiner Weise von der Vorgehensweise von Microsoft, nur die Binärdaten auszugeben, inspiriert worden.*

Aber es suggeriert noch etwas anderes - nämlich, daß der Autor zwar die Wichtigkeit der Freigabe des Quellcodes einsieht, er aber überhaupt nicht versteht, was für eine ungeheure Möglichkeit gerade die frühzeitige Veröffentlichung des Quellcodes wirklich ist. Vermutlich läßt ein Leben im Microsoft-Universum eine solche Einsicht gar nicht erst zu. }

Hat man genügend Beta-Tester und Mitentwickler, so kann fast jedes Problem sehr schnell eingegrenzt werden, und es wird auch jemanden geben, dem die Lösung sofort offensichtlich ist.

Dies ist wahrscheinlich die Kernaussage von Raymond's Einblick in den OSS Prozeß. Er faßt diese Regel in dem Satz zusammen »Fehlersuche ist parallelisierbar«. Eine tiefergehende Analyse folgt weiter unten.

{ Nun, damit hat er recht. }

Parallele Entwicklung

Sobald ein Rahmen an Komponenten festgelegt ist (z.B.: wesentliche API's und Strukturen definiert sind), verwenden OSS-Projekte wie Linux mehrere kleine Teams von Individuen, welche unabhängig voneinander bestimmte Probleme lösen.

Weil die Entwickler typischerweise Hobbyprogrammierer sind, kann eine Vielzahl verschiedener, auch konkurrierender Ansätze verfolgt werden, ohne auf Budgets achten zu müssen. Der OSS-Prozeß zieht seine Vorteile aus der Möglichkeit, die beste Implementierungen aus den verschiedenen produzierten Varianten zu wählen.

Grundvoraussetzung dafür ist:

Eine große Anzahl an Individuen, die bereit sind, ihren Quellcode freizugeben.

Eine Funktionsweise, die auf Modulen basiert (welche im Fall von Linux von der UNIX-Architektur übernommen

wurde).

Paralleles Korrigieren von Fehlern

Das Kernargument von Raymond ist, daß im Gegensatz zu anderen Aspekten der Softwareentwicklung die Fehlerbereinigung ein Vorgang ist, dessen Effizienz fast linear mit der Anzahl der Beteiligten wächst. Es sind nur wenige/ oder gar keine Verwaltungs- oder Koordinierungskosten mit der Fehlerkorrektur von Open-Source-Code verbunden. --dies ist für OSS die entscheidende Möglichkeit, das Brook'sche Gesetz zu umgehen %Anm. d. Übers.: Das Brooksche Gesetz besagt, daß Softwareprojekte, die in Terminverzug sind, durch zusätzlichen Personaleinsatz nicht beschleunigt, sondern eher noch verzögert werden. M. Leutz%

Raymond nimmt Linus Torwalls Beschreibung der Fehlerkorrektur bei Linux auf:

Meine ursprüngliche Formulierung lautete, daß jedes Problem 'für jemanden *transparent* sein wird'. Linus vermutete, daß die Person, die den Fehler versteht und behebt, nicht notwendigerweise - oder sogar meist nicht - die Person ist, die den Fehler zuerst entdeckt und beschreibt. "Jemand findet den Fehler" sagt er, "und jemand anders versteht ihn". Und ich gehe weiter und sage, daß das Finden der Fehler die größte Herausforderung ist". Aber der Punkt ist, daß beides schnell geschieht.

Anders gesagt:

``Fehlerkorrektur ist parallelisierbar". **.** < **.**> beobachtete, daß die Fehlerkorrektur zwar die Kommunikation zwischen Korrektoren und einem koordinierenden Entwickler erfordert, aber zwischen verschiedenen Korrektoren keine wesentliche Kommunikation erforderlich ist. Daher steigen dabei Komplexität und Verwaltungskosten nicht im selben (quatratischen) Maßstab wie dies der Fall ist, wenn zusätzliche Programmierer hinzugezogen werden.

Ein Vorteil der parallelen Fehlersuche ist, daß Fehler und ihre Korrekturen viel schneller als bei traditionellen Verfahrensweisen erstellt und verteilt werden. Ein Beispiel: Als der "TearDrop IP"-Angriff zum erstenmal im WWW bekannt gemacht wurde, war das Problem in der Linuxgemeinde innerhalb von 24 Stunden beseitigt und der Fehlerkorrektur stand zum Herunterladen bereit.

"Impulsive Fehlerkorrektur"

Eine Erweiterung dazu, die ich zu Raymonds Hypothese hinzugefügt habe, ist "impulsive Fehlerkorrektur". Im Fall von Linux, ist das Installieren des OS gleichbedeutend mit dem Installieren der Fehlerkorrektur/Entwicklungs-Umgebung. Infolgedessen ist es sehr wahrscheinlich, daß ein Benutzer/Entwickler einen Fehler, den er in einer Komponente eines Anderen findet (besonders, wenn dieser Programmfehler "einfach" ist), diesen Fehler kurzerhand selber schnell beseitigt und die entsprechende Information über das Internet schnell an den Betreuer des Codes weiterleitet.

In anderen Worten, der beim OSS-Prozeß ist die Einstiegsbarriere zur Fehlerkorrektur sehr niedrig, was an der weit verbreiteten Entwicklungs/Fehlerbereinigungs-Methodologie liegt, die von den GNU Werkzeugen stammt.

Konfliktlösung

Jeder Prozeß bzw. Entwicklung einer bestimmten Größenordnung trifft zwangsweise früher oder später auf Konflikte, welche gelöst werden müssen. Oft ist die Lösung eine willkürliche Entscheidung mit dem Hintergedanken, das Projektes weiterzubringen. In kommerziellen Teams lösen die Hierarchie- und Erfolgsmeß-Stuktur dieses Problem. -- Wie lösen es OSS-Teams?

Im Fall von Linux ist Linus Torwald der unbestrittene Führer des gesamten Projektes. Er delegierte große Komponenten (z.B.: Netzwerk/Treiber etc.) an eine Reihe seine vertrauten "Assistenten", die die Projekte zu einer Handvoll "Bereichsleitern" (z.B.: LAN-Treiber) weiter delegierten.

. wurden durch E. Raymond näher beschrieben.

Einige sehr wichtige große Projekte lehnen das Modell des "wohlwollenden Diktators" völlig ab. Ein Weg, dies zu tun ist, die Co-Entwickler in ein Abstimmungsgremium zu erheben (z.B.: Apache). Ein anderer ist das ständige

wechseln der Führung (rotating dicatorship), wobei die Kontrolle der Entwicklung in regelmäßigen Abständen von einem Entwickler zum nächsten Entwickler übertragen wird. (die Perl-Entwickler organisieren sich auf diese Weise)

Motivation

Dieser Abschnitt bietet einen Überblick über einige der wesentlichen Gründe dafür, dass OSS-Entwickler bei OSS Projekten mitwirken wollen:

Um konkrete Probleme sofort zu lösen

Dies ist im Grunde genommen nur eine andere Formulierung von Raymonds Faustregel: "Jedes gute Stück Software beginnt damit, daß ein Entwickler etwas ändern will, was ihn schon immer gestört hat."

Viele OSS-Projekte -- wie Apache - haben mit einer kleinen Anzahl von Entwicklern begonnen, die ein aktuelles Problem sofort beheben wollten. Anschliessend erfolgte Verbesserungen des Codes stammen oft von einzelnen Entwicklern, die bei der Anwendung in ihren Bereichen z.B. feststellen müssen, daß für bestimmte Netzwerkkarten keine Treiber vorhanden sind etc.

Ausbildung

Der Linux-Kern entstand aus einem Ausbildungs-Projekt an der Universität von Helsinki. Auf ähnliche Weise wurden viele Komponenten des Linux/GNU-Systems (X Windows GUI, Shell Utillities, Clustering, Networking usw.) von Einzelpersonen in Bildungsstätten weiterentwickelt.

> Beispielsweise wächst Linux im Fernen Osten, wie berichtet wird, schneller als die Internetanbindung. -- vor allem dank des Einsatzes im Bildungssektor.
>
> Universitäten sind die ursprünglichen Verfechter von OSS als Lehrmittel.
>
> Forschungs- und Lehrprojekte auf der Basis von Linux sind dank der einfachen Verfügbarkeit von Linux einfach zu verbreiten. Das bedeutet etwa, daß neue Forschungsideen zuerst in Linux implementiert und verfügbar gemacht werden, bevor sie auf anderen Plattformen verfügbar gemacht und eingebunden werden.

{ Und dies von dem gleichen Autor, der später darauf beharrt, daß der Linux-Mob es schwer haben wird, neue Ideen zu . }

Ego Befriedigung

Die am wenigsten faßbare und vielleicht wichtigste Grund, den die OSS Entwickler-Gemeinde nennt, ist schlichte Befriedigung des Egos.

In "The Cathedral and the Bazar" zitiert Eric S. Raymond:

> Die "Nutzenfunktion", die Linux-'Hacker' maximieren ist nicht ökonomisch im klassischem Sinne, aber eine Funktion ihrer Zufriedenheit mit sich selbst und dem Ansehen unter anderen 'Hackern'.
>
> Und natürlich gilt: "Du bist kein Hacker, bis Dich jemand anders "Hacker" nennt.

%Anm. d. Übers.: Zosel: Offensichtlich gibt es für den Autor nur zwei Klassen von Benutzern innerhalb des OSS. Zum einem die Entwickler, die ja nach seinen Aussagen doch ein wenig was können und dann die Benutzer, welche prinzipiell nicht kommerzielles im Sinn haben und er sie alle als 'Hacker' markiert. Für ihr scheint es keine normalen Benutzer im OSS zu geben. %

"Homesteading on the Noosphere" - Inbesitznahme der geistigen Welt

Ein zweites von Raymond veröffentlichtes Papier -- . (.) diskutiert den Unterschied zwischen ökonomisch motiviertem Austausch (z.B.: kommerzielle Software Entwicklung für Geld) und "Verschenken" (z.B.: OSS).

"Homesteading" bedeutet Besitz zu erlangen, indem man entweder derjenige ist, der es als erster "entdeckt" hat, oder derjenige ist, der den letzten bedeutenden Beitrag zur Entwicklung geleistet hat. Die "Noosphere" ist vereinfacht definiert als der "Raum der gesamten Arbeit". Raymond postuliert, daß die Motivation eines

OSS-Hackers darin liegt, einen Anspruch auf das größte Teilgebiet innerhalb des Gesamtproduktes geltend machen zu können. Mit anderen Worten, die Anerkennung für das größte Stück der geschaffenen Arbeit zu erhalten.

{ Dies ist eine subtile, aber bedeutende Fehlinterpretation. Sie führt die Idee territorialer »«Größe«« ein, die in meiner Theorie nirgendwo vorkommt. Es mag ein persönlicher Fehler des Autors sein, aber ich vermute eher, daß sich darin die vom Wettbewerb besessene Microsoft-Kultur widerspiegelt. }

Aus " Homesteading auf dem Noosphere ":

Bestehender Überfluß macht es schwierig, Befehlsstrukturen zu erhalten, und er macht Beziehungen, die auf Austausch basieren, im wesentlichen redundant. In "Geschenk-Kulturen" ist der soziale Status nicht dadurch bestimmt, was Du besitzt, sondern durch *das, was Du weggibst*....

Wenn man es auf diese Art und Weise betrachtet, ist es recht deutlich, daß die OSS-Gesellschaft in der Tat auf 'Schenkens' basiert. In dieser Gesellschaft gibt es keinen ernsthaften Mangel an Lebensnotwendigem - Festplattenplatz, Netzwerkbandbreite, Rechenkraft, etc. Software wird frei geteilt. Dieser Überfluß schafft eine Situation, in der der einzig verfügbare Maßstab für Erfolg die Reputation bei den Anderen ist.

(http://www.techweb.com/internet/profile/eraymond/interview)

SIMS: Also war der Mangel, den du gesucht hast, der Mangel an Aufmerksamkeit und Beachtung?
RAYMOND: Das ist korrekt.

Altruismus

Dies ist eine zwiespältige Motivation, und ich neige dazu zu glauben, daß Altruismus ab einer gewissen Stufe zu einer Form von Befriedigung des Egos (in der von Raymond vorgetragen Art) degeneriert

Ein schwächerer Beweggrund, der zum Teil auf Altruismus zurückgeht, ist das prinzipielle Einprügeln auf Microsoft.

{ Ein faszinierendes Eingeständnis eines Microsofters! Selbstverständlich analysiert er nicht, woher diese Verbindung kommt, denn das könnte der Wahrheit zu nahe kommen... }

Code-Spaltung

Eine entscheidende Gefahr in jedem großen Entwicklungsteam ist das Risiko der Code-Spaltung - erheblich verstärkt durch den chaotischen Prozeß, wenn ein Entwicklungsteam die Größenordnung des Internets hat.

Eine Spaltung des Codes tritt auf, wenn im Verlauf des normalen Geben, Nehmens und Austauschens in einem Entwicklungsprojekt verschiedene, unvereinbare Versionen der Codebasis entstehen.

So wird beispielsweise Codebasis in der kommerziellen Welt das strenge, singuläre Management der Windows NT als einer seiner größten Vorteile gegenüber der "gespaltenen" Codebasis gesehen, die man bei kommerziellen UNIX-Implementierungen findet. (SCO, Solaris, IRIX, HP-UX, usw.).

Abspaltung bei OSS -- Bd Unix

Innerhalb des OSS Bereiches ist BSD das beste Beispiel für eine gespaltene Codebasis. Das ursprüngliche BSD-UNIX war ein Versuch der U-Cal Berkeley %Anm. d. Übers.: »U-Cal Berkley« ist die »University of California« in Berkley bei San Francisco%, für Lehrzwecke eine lizenzfreie Version des Unix-Systems zu entwickeln. Allerdings legte Berkeley der Nutzung dieser Codebasis im nicht-akademischen Bereich strenge Restriktionen auf.

{ Die vom Autor wiedergegebene Geschichte über die Aufspaltung von BSD ist schlicht falsch. }

Um eine wirklich völlig kostenlose Version von BSD UNIX zu erstellen, entwickelte ein spontan zusammengestelltes (aber festes) Team von Entwicklern FreeBSD entwickelte. Andere Entwickler, die aus verschiedenen Gründen Differenzen mit dem FreeBSD Team hatten, begannen eigene OS-Variationen zu entwickeln (OpenBSD, NetBSD, BSDI).

Es gibt zwei dominierende Faktoren, welche zur Aufspaltung von BSD führten:

Nicht jeder kann zu der BSD Codebasis beitragen. Dies limitiert die Größe der effektiven "Noosphere" und bietet ein Potential dafür, daß jemand beansprucht, daß sein abgespaltener Code werde im Laufe der Zeit sich gegenüber dem ursprünglichen BSD-Code durchsetzten.

{ Wow. Dies ist eine Einsicht, die ich nie hatte -- daß eine Spaltung wirklich auf dem Glauben basieren kann, durch die Abspaltung einen größeren 'Basar' zusammenbringen zu können als durch das derzeitige Projekt. Es erklärt sicher das EGCS und das BSD-Spinoff-Gruppe-des-Tages Phänomen, jedoch vermutlich nicht die Emacs/XEmacs-Spaltung.

O.K., haben wir jetzt etwas erlernt. Dies mag in der Tat die überraschende Tatsache erklären, daß jene Projekte, die die Entwicklung am offensten gestalten, die geringste Tendenz zeigen, sich aufzuspalten...}

Anders als GPL erlegt die BSD's-Lizenz dem abgeleiteten Code keine Restriktionen auf. Folglich kann jeder seinen veränderten Code, wenn er denn der Ansicht ist, seine Arbeit sei gut genug, vom Code abspalten, dafür Geld verlangen, den Namen ändern usw...

Beide Beweggründe schaffen führen dazu, daß Entwickler versucht sind, eine Spaltung im Code zu erzwingen und die Erträge (finanziell oder Ego) auf Kosten der gesamten BSD-Gesellschaft zu erzielen.

(Fehlen von) Abspaltungen bei Linux

Im Gegensatz zum BSD-Beispiel hat sich der Linux-Kern nicht aufgespalten. Zu den Gründen, warum die Integrität der Linux Codebasis erhalten wurde, zählen:

Allgemein akzeptierte Führung.

Linus Torvalds ist eine Berühmtheit in der Linuxwelt und seine Entscheidungen gelten als endgültig. Für die verschiedenen BSD-Derivate existiert dagegen keine ähnliche berühmte Leitfigur.

Linus wird von dem Entwicklungsteam als fairer, wohlüberlegter Code-Manager geachtet und seine Reputation innerhalb der Linuxgemeinde ist ziemlich groß. Dennoch mischt sich Linus nicht in jede Entscheidung ein. Oft lösen Untergruppen ihre -- zum Teil recht großen -- Differenzen unter sich und verhindern eine Abspaltung.

Offene Mitgliedsstrukturen und langfristiges Beitrags-Potential

Im Gegensatz zum geschlossenen BSD - Kreis kann jedermann etwas zu Linux beitragen, und der "Status" -- und folglich ist die Fähigkeit, einen großen Bestandteil von Linux für sich zu reklamieren -- basiert auf der Menge der seiner vorhergehenden Beiträge.

Indirekt liegt darin eine weitere Abschreckung davor, die Codebasis aufzuspalten. Es gibt so gut wie keinen glaubwürdigen Mechanismus, mit der eine abgespaltene 'Minderheits-Codebasis' die Innovationsrate der primären Linux-Codebasis beibehalten könnte.

Die GPL Lizenz sorgt dafür, daß es keine ökonomischen Anreize gibt, den Code zu spalten

Da Derivate von Linux wiederum frei sein müssen, mindert dies den langfristigen ökonomischen Nutzen einer Minderheit mit einer abgespaltenen Linux-Variante.

Eine Spaltung der Codebasis spaltet auch die "Noosphere"

Die Ego-Motivation treibt die OSS-Entwickler dazu, den größtmöglichen Anteil der größtmöglichen "Noosphere" zu erreichen. Spaltung der Code Basis mindert zwangsläufig die Raum für die Beiträge nachfolgender Entwickler auf der neuen Codebasis.

Open Source Stärken

Was sind die zentralen Stärken der OSS Produkte, die Microsoft Sorge machen sollten?

Herausragende OSS-Eigenschaften

Wie unser Betriebssystem-Geschäft hat auch das OSS-System verschiedene exponentielle Eigenschaften:

Die OSS-Prozesse wachsen mit dem Internet

Das größte Einzelproblem an Anfang jedes OSS-Projektes ist es, genug Entwickler zu finden, die bereit sind, ihre Zeit darin zu investieren. Für den Start eines Projektes von der Dimension eines Betriebssystems war das Internet absolut notwendig, um genügend Leute zusammenzubringen. Wichtiger noch, das Wachstum dieser Projekte wird vom Wachstum der Internet-Verbreitung getrieben. Verbesserungen der Möglichkeiten der Zusammenarbeit befeuern die OSS-Maschinerie.
Anders gesagt, mit dem Wachstum des Internets werden auch die OSS-Projekte wachsen und damit auch OSS Projekte in vielen kleineren Softwarekategorien realisierbar sein.

OSS Prozeß: "Der Sieger kriegt alles "

Wie kommerzielle Software werden auch bei OSS die besten Projekte mit der Zeit konkurrierende Programme verdrängen und die deren besten Eigenschaften übernehmen. So verdrängte Linux BSD-UNIX und absorbierte einen Großteil der Grundideen (genauso wie Grundideen von kommerziellen UNIXes). Dies trägt dazu bei, daß es keine großen Vorteile davon gibt, mit einer Entwicklung zuerst auf dem Markt zu sein.

Entwickler versuchen stets, an dem größtmöglichen OSS-Projekt mitzuwirken

Je größer das OSS Projekt, desto größer wird das Prestige, daß mit dem substantiellen Beitrag zu seiner "Noosphere" einher geht. Dieses Phänomen geht auf die "Sieger kriegt alles "- Natur des OSS Prozeß zurück.

Größere OSS Projekte lösen mehr aktuelle Probleme

Je größer das Projekt, desto mehr Entwickler prüfen/testen den Code. Je mehr Leute testen, desto besser wird das Produkt

Langfristiges Vertrauen

Binaries können vergehen, aber Quellcode lebt für immer

Eine der interessantesten Aussichten von funktionsfähigen OSS Systemen ist langfristige Glaubwürdigkeit.

Langfristige Glaubwürdigkeit definiert:

Langfristige Glaubwürdigkeit besteht, wenn das Produkt auf absehbare Zeit nicht aus dem Markt verdrängt werden wird. Dies zwingt die Konkurrenz dazu, ihren Umgang mit dir zu verändern.

Zum Beispiel: Airbus Industries sammelte anfänglich langfristige Glaubwürdigkeit durch ausdrücklich Unterstützung in der Regierung. Das Ergebnis: Bei Ausschreibungen von Fluglinien wird Boeing kurzfristige, nicht unbedingt gewinnorientierte Erträge akzeptieren, wenn der Mitbewerber Lockheed ist, nicht aber, wenn es gegen Airbus geht. Frei auf den Softwarebereich übertragen bedeuted daß: Ein Produkt/Prozeß ist dann langfristig glaubwürdig, wenn FUD Taktiken nicht dagegen helfen..

OSS ist langfristig glaubwürdig

OSS Systeme sind glaubwürdig, da der Quellcode an unzähligen Stellen und von unzähligen Individuen verfügbar.

{ Wir sind hier tief in der Sicht der Welt á la Microsoft. Die typische Hacker Reaktion wird sein, es eklig zu finden, aber es spiegelt die Art von instrumentalisierter Rücksichtslosigkeit wieder, mit der wir lernen müssen umzugehen.

Der wirklich interessante Aspekt dieser zwei Aussagen is: Aus ihnen geht hervor, daß Microsoft seine FUD- Taktik uns gegenüber aufgeben sollte.

Die meisten von uns haben angenommen, die Monopolklage des Justizministeriums habe Microsoft daran gehindert, mit ihren FUD-Kanonen auf uns zu schießen. Aber wenn seine Gatesigkeit mit dieser Beurteilung übereingestimmt hat, könnte Microsoft der

Ansicht sein, eine etwas inhaltsvollere Antwort zu entwickeln, weil FUD nicht wirkt.
Dies könnte also sowohl eine gute wie auch eine schlechte Nachricht sein. Die gute Nachricht wäre, daß Microsoft sein aggressives Marketing aufgeben würde - eine Waffe, die in der Vergangenheit weitaus effektiver war als ihre eindeutig minderwertige Technologie. Die schlechte Nachricht ist: Diese Stategie uns gegenüber aufzugeben wäre in der Tat die bessere Taktik - sie würden keine weitere Energie verschwenden und hätten womöglich Zeit, eine wirklich effektive Antwort zu entwickeln..}

Die Wahrscheinlichkeit, daß Apache verschwinden wird ist erheblich niedriger als die Wahrscheinlichkeit, daß etwa WordPerfect, verschwinden wird. Das Verschwinden von Apache ist nicht so sehr an das Verschwinden von Binaries gebunden (was etwa von Kaufzyklen etc abhängt), sondern hängt eher am Verschwinden des Quellcodes und des Wissens darüber. Umgekehrt formuliert: Die Kunden wissen genau, daß es Apache in fünf Jahren noch geben wird -- solange ein Grundinteresse von den Anwendern und Entwicklern aufrechterhalten wird..

Ein Apache Kunde sagte beispielsweise als Begründung dafür, daß er seine kommerzielle Webseite auf OSS-Software laufen läßt: "weil es Open Source ist, kann ich ein oder zwei Entwickler dran setzen und es selber auf Dauer betreiben "

Geringe Code-Spaltung fördert langfristige Glaubwürdigkeit

Die GPL und seine Abneigung gegen Code-Spaltung beruhigt Kunden. Sie müssen nicht befürchten, mit der Entscheidung für ein bestimmtes kommerzielles Linux-System in eine Sackgasse zu gelangen und in naher Zukunft ihre Anwendungen nicht mehr weiterentwickeln zu können.

Die "Sackgasse der Entwicklung" ist der Kern der FUD-Strategie bei Software.

{ Sehr wahr -- und gibt hier noch eine offensichtliche Auslassung. Wenn der Autor ehrlich gewesen wäre, hätte er angefügt, daß die OSS-Befürworter diese Argumente problemlos entkräften und Microsoft um die Ohren schlagen können.

Der Verfasser gibt selber zu, daß OSS in diesem Punkt unangreifbar ist. Andererseits ist die explodierende Komplexität von dem umbenannten ``Windows 2000" ein Anzeichen dafür, daß dort eine evolutionäre Sackgasse erreicht ist. Der Autor hat dies nicht weiter verfolgt. Aber wir sollten das tun. }

Paralleles Ausmerzen von Fehlern

Linux und andere OSS-Vertreter sind ein immer glaubwürdigeres Argument dafür, daß OSS-Software mindestens so stabil -- wenn nicht stabiler -- ist wie kommerzielle Alternativen. Das Internet liefert ein ideales, weithin sichtbares Schaufenster für die OSS-Welt.
{ Es ist eine Handvoll von Amateuren, die meisten von uns unbezahlt und fast alle nebenbei, die gegen eine hochgerüstete Multimillion-Dollar-Propaganda arbeiten, die von einigen der höchstrangigen Spezialisten des Technologiemarketings geleitet wird..

Und die Amateure sorgen für ``< FONT COLOR="red"> ein immer glaubwürdigeres Argument ". Microsoft selber gibt zu, daß wir gewinnen! Vielleicht steckt da eine Botschaft über die zugrunde liegenden Produkte drin? }

Insbesondere größere Organisationen, die für kommerzielle Anwendungen auf OSS bauen (z.. ISPs), sehen sich durch die Tatsache bestätigt, daß sie einen Fehler, der die Arbeit beeinträchtigt, selber und damit unabhängig von den Terminen eines kommerziellen Anbieters beseitigen können. (So war beispielsweise UUNET fähig, innerhalb von 24 Stunden nach dem ersten TearDrop-Angriff eine Lösung zu finden, zu erstellen und erfolgreich einzusetzen)

Parallele Entwicklung

Anders gesagt: "Entwicksressoucen in OSS sind im wesentlichen kostenfrei ". Da der Pool von potentiellen Entwicklern groß ist, kann man mehrere Lösungswege und Versionen gleichzeitig verfolgen und am Ende die beste Lösung auswählen.

So wurde etwa der Linux TCP/IP - Bestandteil vermutlich dreimal umgeschrieben. Vor allem Assemblercode - Bestandteile wurden kontinuierlich fein justiert und verbessert.

OSS gleich 'perfekte' API - Verbreitung / Dokumentation

Um wesentlichen versorgt die API-Verbreitung / Weiterbildung von Entwicklern von OSS die Entwickler mit dem zugrundeliegenden Code. Während die Verbreitung von API's in einem geschlossenen Kreis im wesentlichen auf die Frage des Vertrauens reduziert, erlaubt es die API Verbreitung bei OSS dem Entwciklern, sich sein eigenes Bild zu machen.
NatBro und Ckindel zeigen hier eine Aufteilung der Entwickler nach Fähigkeiten. Der Könner fühlt sich mit der OSS-Verbreitung wohl, Neulinge und Fortgeschrittene - die den größten Teil stellen - bevorzugen das "Vertrauensmodell" und die organisatorische Glaubwürdigkeit (z.B. :"Microsoft sagt, API X sieht so und so aus")

{ Ob es wirklich wahr ist, daß die meisten Entwickler das Vertrauensmodell vorziehen, ist eine hochinteressante Frage.

Die Erfahrung von zwanzig Jahren in diesem Bereich sagt nein; im allgemeinen bevorzugen die Entwcikler Code, auch wenn ihre nicht-technischen Vorgesetzte, naiv genug sind, um "Vertrauen" vorzuziehen. Microsoft möchte natürlich gerne daran glauben, daß seine "institutionelle Glaubwürdigkeit" zählen - da ist der Wunsch Vater des Gedankens.
Andererseits könnten sie recht haben. In der Open Source Gemeinschaft können wir es uns nicht leisten, die Möglichkeit außer acht zu lassen. Ich denke, wir können dem Rechnung tragen, indem wir hochklassige Dokumentationen entwickeln. Auf diese Weise kann "Vertrauen" in bekannte Autoren (oder in Verleger mit gutem Ruf, wie O'Reilly oder Addison Wesley) das "Vertrauen" in eine API-definierende Organisation ersetzen.}

Veröffentlichungshäufigkeit

Stark modulierte OSS - Projekte sind in der Lage, einzelne Teile sofort zu veröffentlichen, sobald der Entwickler seinen Code fertig hat. Folglich schreiten die OSS Pläne schnell - und häufig - voran.

Open Source Schwächen

Die Schwächen von OSS Pläne zeigen sich vor allem in drei Bereichen:

Verwaltungskosten

Prozeßthemen

Organisatorische Glaubwürdigkeit

Managmentkosten

Die größte Schwierigkeiten von OSS-Projekten ist es; mit den exponentiell mit Innovationsrate und Größe wachsenden Verwaltungskosten fertig zu werden. Darin ist eine Grenze der Innovationsrate eines OSS-Projektes angelegt.

Ein OSS Projekt zu starten ist schwierig

von Eric Raymond:

> Es ist ziemlich offensichtlich, daß man mit diesem Basar-Stil nicht von Grund auf neuen Code schreiben kann. Man kann testen, Fehler korrigieren, und man kann verbessern, aber es ist ziemlich schwierig, ein völlig *neues* Projekt im Basar-Stil entstehen zu lassen. Linus hat das nicht versucht. Ich auch nicht. Die wachsende Entwicklergemeinschaft braucht etwas lauffähiges und testfähiges, mit dem sie spielen kann.

Raymond `s Argument kann erweitert auf die Schwierigkeit erweitert werden, ein Projekt zu starten und aufrechtzuerhalten, wenn keine klare Ausgangslage oder ein klares Ziel (oder zu viel Ziele!) für das Projekt erkennbar sind.

Basar Glaubwürdigkeit

Ganz offensichtlich gibt es weit mehr Bruchstücke von Quellcodes im Internet als es OSS-Gemeinschaften gibt. Was trennt dann den verworfenen Quellcode von einem blühenden und gedeihenden Basar?

Ein Artikel (http://www.mibsoftware.com/bazdev/0003.htm) liefert die folgenden *glaubwürdigen* Kriterien:

"....in Kriterien von einer Art Mindestanzahl von Teilnehmern zu denken ist irreführend. Fetchmail und Linux haben *jetzt* ungeheuer viele Beta-Tester, aber zu Beginn hatten beide offensichtlich nur wenige.

Beide Projekte hatten eine Handvoll Enthusiasten und ein glaubwürdige Aussichten. Vielversprechend waren sie zum Teil aus technischer Sicht (mit ein wenig Aufwand wird dieser Code klasse!), zum Teil aus soziologischer (wenn du mitmachst, wirst du genau so viel Spaß haben wie wir). Voraussetzung für die Entwicklung eines Basars ist also, daß das Versprechen bunten Basartreibens glaubwürdig ist.

Ich denke, daß einige der folgenden Kriterien erfüllt sein müssen, damit ein Basar glaubwürdig ist:

Große zukünftige "Noosphere" -- das Projekt muß so aufregend genug, daß die geistige Genugtuung des Beitrages die investierte Zeit hinreichend kompensiert. In dieser Hinsicht zeichnet sich beispielsweise das Linux-Betriebssystem aus.

Ein altes Übel endlich beseitigen ... -- das Projekt für ein großes Publikum von *Entwicklern* wichtig / entwickelbar sein. Der Apache-Webserver liefert ein sehr gutes Beispiel dafür.

Den richtigen Anteil des Problems zuerst lösen -- Zu viele auf einmal vorweg zu lösen würde die OSS Gemeinschaft auf die Rolle der Tester degradieren Zu wenig Probleme zu lösen, bevor es für die OSS Gemeinde freizugeben, reduziert den erwarteten Nutzen des Projektes und bietet eine zu schwache Struktur der Komponenten, um die Arbeit effizient zu koordinieren.

{ Diese drei Punkte sind gut durchdacht und verbessern sogar meine Charakterisierung in ``The Kathedrale and the Bazar.". Die Unterscheidung, die er zwischen der `zu erwartenden großen Noosphere" und dem `ein großen Problem lösen' macht, ist insbesondere vielsagend. }

Entwicklung nach Einholen des Vorbildes

Als dieses Problem JimAll beschrieben wurde, beschrieb er es treffend mit "die Rücklichter jagen". Die einfachste Art, das Verhalten einer so großen, nur halb organisierten Masse zu koordinieren, ist, ihnen ein sichtbares Ziel zu zeigen. Rücklichter verleihen einer diffusen Vision etwas Handfestes. In solchen Situationen ist ein Rücklicht, dem man folgen kann, ein Ersatz für eine starke zentrale Führung.

Wenn diese systembedingte Organisation nicht mehr da ist (wenn also ein Projekt erst einmal den Stand der aktuellen Technik erreicht hat), ist eine umso größerer Managementaufwand notwendig, um neue Ziele zu definieren.

{ Quatsch. Alles, was man in der Open Source Welt braucht, ist eine einzige Person mit einer guten Idee.

Der Sinn von Open Source liegt ja eben gerade darin, die Aufwandsschwelle, die Neuerungen erschwert, möglichst niedrig zu halten. Die Erfahrung zeigt, daß gerade der vom Autor so gepriesene "größere Managementaufwand" die schlimmste dieser Schwellen ist. In der Welt des Open Source können Entwickler machen, was immer sie wollen, und das einzige Kriterium ist, ob es genügend Anwender gibt, die die Innovation testen wollen - und ob sie ihnen gefällt. Das Internet vereinfacht diesen Prozess, und die Art, wie die Gemeinschaft ihre Zusammenarbeit organisiert, ist eben genau auf diesen Zweck zugeschnitten.

Der dritte Alternative zum ``Schlußlichter jagen" oder ``starker zentraler Führung" (und effektiver als beide!) ist eine sich entwickelnde kreative Anarchie, in der es Tausende von Führern und Zehntausende von Anhänger gibt, die durch ein Netzwerk gegenseitiger Prüfung verknüpft sind und in der sie sich blitzartiger Erfolgskontrolle unterwerfen müssen. Microsoft kann das nicht schlagen. Ich glaube, sie können es noch nicht einmal richtig verstehen - zumindest nicht gefühlsmäßig.}

Dies ist möglicherweise die interessanteste Hürde für die Linux Gemeinschaft, nun, daß sie in vielen Bereichen den technischen Stand von UNIX erreicht haben.

{ Die Linux Gemeinschaft hat diese Hürde nicht einfach nur genommen, sondern sie in Grund und Boden gestampft. Diese Tatsache ist der eigentliche Grund dafür, daß Open Source auf lange Sicht der geschlossen Entwicklung gegenüber überlegen ist. }

Langweile Arbeit

Eine weitere interessante Frage bei der zukünftigen Entwicklung von OSS ist es, wie sie die weniger aufregenden Arbeiten erledigt bekommen, die notwendig sind, um ein Produkt von kommerzieller Qualität entstehen zu lassen. Im Bereich der Betriebssysteme sind dies kleine, aber essentielle Funktionen wie das Energiemanagement, Unterbrechung/Wiederaufnahmen, Verwaltungsebenen, UI-Nettigkeiten usw.
Für Apache könnte das bedeuten, auch Neulinge mit Anleitungsprogrammen zu Administratoren machen zu

können.

Integrative/architektonische Arbeit

Das Integrieren eines Moduls in das andere ist der größte Kostenfaktor von OSS-Teams. Eine Email -Notiz von Nathan Myrhvold von 5/98 weist darauf hin, daß von alle Aspekten der Softwareentwicklung die Integration auf sichtbarsten Brooks Gesetz unterliegt.

Bis jetzt hat Linux außerordentlich von den Intergrations- und Modularprinzipien der vorhergehenden UNIXe- profitiert. Die Organisation von Apache war zudem durch die vergleichsweise einfache und fehlertolerante Spezifikation von HTTP-Protokollen und die Gestaltung der UNIX Server-Anwedungen erleichtert.
< FONT COLOR="red"> Zukünftige Neuerungen, die Änderungen in der basierenden Architektur- und Intergrationsmodell erforderlich machen, werden für das OSS-Team unglaublich schwer unzusetzen sein, weil damit die Vorläufer und die Wissensbasis gleichermassen beeinträchtigt werden.

{ Diese Vorhersage entspringt der vorherigen Aussage des Autors, die Open Source-Entwicklungen seien zwingend auf Vorläufer angewiesen und notwendigerweise rückwärtsgewandt. Es zeugt von getrübter Wahrnehmung - anscheinend bemerkt er Dinge wie Python, Beowulf, und Squeak *(um nur drei von Hunderten innovativer Projekte zu nennen) nicht.*

Wir können nur hoffen, daß Microsoft weiter daran glaubt - denn das verzögert deren Antwort. Viel wird davon abhängen, wie sie Neuerungen wie zum Beispiel die SMPsierung des Linux Kerns interpretieren.

Interessant ist auch, was der Autor über Innovationen sagt}

Prozeßthemen

Dies sind die Schwächen, die die OSS-Verfahrensweisen für Gestaltung und Rückkoppelung haben.

Wiederkehrende Kosten

Einer der Schlüssel zum OSS Prozeß ist, daß dort wesentlich häufiger neue Versionen entstehen als bei kommerzieller Software (Linux hat z.T. mehrmals am Tag neue Revision des Kerns gehabt!). Allerdings sagen uns die Kunden, sie wollten weniger Versionen, nicht mehr. *{ Ich frage mich, wie diese Antwort wohl ausfallen würde, wenn Microsofts neue Versionen nicht so teuer wären?*

Dafür gibt es kommerzielle Linux Händler - um den Ausgleich zwischen dem schnellen Entwicklungsprozeß und dem Kunden, der nicht jede kleine Änderung mitmachen möchte, zu schaffen. Der Kern mag sich mehrfach am Tag ändern, aber Red Hat bringt nur einmal alle sechs Monate eine neue Version heraus. }

Rückantwort von Laien

Das Linux OS ist nicht für Anwender, sondern eher für andere Hacker entwickelt worden. Der Apache Netzserver ist gleichermaßen für den Einsatz bei den größten und schwersten Seitenbetreibern gedacht, weniger für den Intranet-Server der Abteilung X.

Der Hauptpunkt dabei ist, daß OSS keine ausdrückliche Marketing / Kundenbetreuungs - Instrumente hat, und somit die Wünsche (und damit die Entwicklung weiterer Möglichkeiten) von den Experten bestimmt werden-

< FONT COLOR="red"> Was die Entwicklergruppen vom MSFT wieder und wieder lernen mußten ist, daß Benutzerfreundlichkeit, intuitive Bedienung usw. von Anfang ein Teil eines Produktes sein muß und nicht nachträglich aufgesetzt werden kann.

{ Das muß kommentiert werden - weil es so wahr ist und gleichzeitig so nachhaltig in der Praxis von Microsoft ignoriert wird. Dieser Irrtum zeigt eine verwertbare Schwäche in der angedeuteten Microsoft-Strategie, intuitive Bedienung herauszuheben.

Es gibt zwei Wege, von Anfang an für einfache Benutzbarkeit zu sorgen. Ein Weg - der Microsoft-Weg - ist das Erschaffen monolitischer Anwendungen, die sich durch ihre eigene intuitive Führung definieren. Dies führt zu "Windownitis" -- starres, tapsige, fehleranfällige Ungeheuer mit polierter Oberfläche und hohlem Kern.

Programme, die auf diese Weise gebaut wurden, sehen auf den ersten Blick benutzerfreundlich aus, stellen sich aber auf lange Sicht aus einen Bermuda-Dreieck für Zeit und Energie heraus. Sie halten sich nur durch ein flächendeckendes Marketingbombardement, das dem Benutzer einredet (a) Fehler sind gewollte Leistungsmerkmale, oder (b) alle Fehler sind in Wirklichkeit der Dummheit des Benutzers zuzuschreiben, oder (c) alle Fehler verschwinden mit der nächsten Version. Dieser Ansatz ist im Grundsatz her falsch.).

Der andere Weg ist der, den UNIX/Internet/Web eingeschlagen haben. Hier wird der Motor (der die Arbeit macht) von der Benutzerführung (die Anzeige und Kontrolle ausmacht) getrennt. Voraussetzugn dafür ist, daß Benutzerführung und Motor ein fest definiertes Protokoll haben. Am anschaulichsten wird das bei der Browser/Server-Kombination: der Motor konzentriert sich darauf, Motor zu sein, und die Benutzerführung darauf, Benutzerführung zu sein..

Folgt man dem zweiten Weg, verringert sich die Komplexität und die Zuverlässigkeit steigt. Außerdem kann die Schnittstelle wesentlich leichter entwickelt/verbessert/ und individuell eingerichtet werden gerade weil sie eben nicht mit der Maschine verbunden ist. Es ist sogar möglich, mehrere auf bestimmte Benutzer zugeschnitte Schnittstellen gleichzeitig zu haben.

Und schließlich führt diese Architektur von alleine zu marktfähigen Anwendungen, die getrennt vom Serever benutzt oder verwaltet werden können. Dieser Ansatz funktioniert -- und es ist die der Open Source Gesellschaft eigener Weg, Microsoft in Schach zu halten.

Der Punkt ist: Wenn Microsoft die Open Source Gemeinde mit dem Schlagwort Benutzerführung bekämpfen will, dann laßt sie das ruhig tun -- wir können auch diese Schlacht siegen, indem wir auf unsere Weise kämpfen. Sie können ruhig immer monströsere Windows-Monolithen schreiben, die dich an die Tastatur deines Anwendugsservers schweißen. Wir werden siegen, wenn wir saubere und weit verbreitete Anwendungen schreiben, die die Macht des Internets nutzen und die Benutzerführung zu einer vom Kunden freiwillig getroffenen Entscheidung machen, die weiter wachsen kann.
Zu beachten ist allerdings, daß unser Sieg von sauber definierten Protokollen für die Kommunikation zwischen Motoren und Benutzerführung abhängt (wie HTTP). Deshalb ist der Kram, der nachher über die Ent-Standartisierung von Protokollen folgt, so übel. Wir müssen davor auf der Hut sein. }

Es wird interessant sein zu beobachten, welchen Einfluß die kommerziellen OSS-Anbieter (solche wie RedHat im Linux Bereich, C2Net im Apache Bereich) auf das Prinzip der Rückkoppelung haben werden.

Organisatorische Glaubwürdigkeit

Wie kann die OSS Gemeinde den *Service* liefern, den Kunden von Softwareanbietern erwarten?

Unterstützungs-Modell

Produktunterstützung ist üblicherweise das erste Punkt um den sich potentielle Verbraucher von OSS Gedanken machen, und es ist das dickste Pfund, mit dem kommerzielle Vertreiber wuchern können.

Allerdings wird die große Mehrheit der OSS Projekte durch die Entwickler der betroffenen Komponenten unterstützt. Dieses Netzwerk an Unterstützung so auszubauen, daß es mit kommerziellen Produkten mithalten kann, wird eine beträchtliche Herausforderugn sein. Es gibt viele Größenordnungsunterschiede zwischen Benutzern und Entwicklern in IIS im Vergleich zu Apache.

{ Es spricht Bände, wie vage dieser letzte Satz ist. Hätte der Autor den Gedanken weitergeführt, hätte er zugeben müssen, daß Apache auf dem Markt IIS vernichtend schlägt (Apaches Marktanteil: 54% und steigend; IIS: irgendwo bei 14% und fallend).

Dieses Eingeständnis würde zu einer Wahl zwischen für Microsoft unerträglichen Alternativen führen. ies würde zu einer Auswahl zwischen schlechten (für Microsoft) Alternativen führen. Es könnte beispielsweise sein, daß die informelle Informationskanäle für die Benutzer und die "organisatorische Glaubwürdigkeit" von Apache tatsächlich bessere Ergebnisse erzielen, als sie Mircosofts IIS-Organisation bringen kann. Wenn das wahr wäre, dann fällt es schwer zu glauben, warum das Prinzip nicht auch für andere Open Source - Projekte gilt. Die Alternative ist noch schlimmer - daß Apache so gut ist, daß es weder Unterstützung noch organisatorische Glaubwürdigkeit braucht. Das wiederum würde bedeuten, daß all die Unterstützungs- und Marketingschwadrone eine riesige Fehlinvestition waren, den bröckelnden stalinistischen Apartmentwohnungen vor vierzig Jahren gleich.

Diese zwei mögliche Erklärungen implizieren zwei verschiedene, aber ähnlich gelagerte Strategien für die Verfechter des Open-Source. Die eine ist, Software zu erstellen, die so gut ist, daß sie einfach wenig Unterstützung braucht (aber das wollen wir sowieso, und haben es im wesentlich schon immer getan). Die andere ist, die informellen aber sehr effektiven Informationskanäle (die wir jetzt bereits über Support Mailing-Listen, Newsgroups, FAQs pflegen) noch stärker zu nutzen.}

Kurz- und mittelfristig wird alleine das schon die OSS-Produkte an die Spitze der Benutzergemeinde bringen.

Strategische Zukunft

Der Mangel an strategischer Ausrichtung in den OSS-Entwicklungszyklen ist ein außerordentlich gravierendes Problem dabei, auf breiter Front vom Benutzer angenommen zu werden. Schrittweise Verbesserungen der bestehenden Möglichkeiten eines OSS-Projektes sind glaubwürdig, aber mangels institutioneller Entschlossenheit sind zukünftigen Weiterentwicklungen nicht sicher.

{Nein. In der Open-Source Gemeinde werden neue Entwicklungen durch einzelne Hacker vorangetrieben, die ständig auf der Suche nach Neuheiten und Eroberungen sind. Diese treibende Kraft darf sicherlich nicht außer Acht gelassen werden. Das Internet und das Web sind auf diese Weise entstanden -- nicht durch eine `institutionelle Entschlossenheit", sondern weil irgendwer irgendwo dachte ``Hey - das hier wäre nett...´´.

Vielleicht haben wir Glück, daß die `organisatorische Glaubwürdigkeit` in der Microsoft Weltsicht eine so starke Rolle spielt. Die Zeit und Energie, die sie diese vorgebliche Voraussetzung kostet, können sie nicht mher darauf verwenden, etwas wirksames gegen uns zu entwickeln. }

Was bedeutet es für die Linux Gemeinschaft, beim Aufbau des eines vereinten digitalen Nervensystems mitzumachen? Wie kann Linux garantieren, daß Vereinbarkeit mit Anwendungen besteht, die für ältere APIs geschrieben wurden? Wen verklagen sie, wenn die nächste Version von Linux einige Verpflichtungen nicht einhält? Wie geht Linux eine strategische Allianz mit anderen ein?

{Die Frage der Rückwärtskompatibilität ist ziemlich ironisch angesichts der Tatsache, daß ich noch nie von einem Programm gehört habe, das ohne jede Änderungen gleichermaßen unter DOS, Windows 3.1, Windows 95, Windows 98 und NT 4.0 läuft. Der Autor wurde hier von der Wirklichkeit überholt. Er sollte mal Microsofts Kumpel bei Intel fragen, die einen Anteil an RedHat erworben haben, keine zwei Monate nachdem dieses Memo geschrieben wurde.}

Open Source-Geschäftsmodelle

In den letzten 2 Jahren hat OSS eine weitere Wendung vollzogen, nämlich durch das Aufkommen von Firmen, die OSS Software verkaufen und die, was noch viel wichtiger ist, Vollzeitentwickler einstellen, die die Codebasis weiter verbessern. Welches Geschäftsmodell rechtfertigt diese Gehälter?

In vielen Fällen ist die Antwort auf diese Fragen etwas in der Art wie "Warum sollte ich mein(e) Protokoll/Applikation/API einem Standardisierungs-Gremium vorlegen?"

Sekundäre Dienstleistungen

Der Verkäufer von OSS-Ware bietet dem Kunden Verkauf, Unterstützung und Integration. Damit wandelt sich der OSS-Warenverläufer vom Hersteller eines Fertigproduktes zu einem Dienstleister.

Markteinstieg mit Führung, aber Verlusten

Dieses Geschäftsprinzip von OSS kann für zwei Ziele verwendet werden:

Einen kleinen Markt mit einem Knall zu starten

In einen existierenden Markt eindringen, der von etablierten Closed Source - Spielern gehalten wird.

Viele OSS Existenzgründer -- besonders jene im Bereich Betriebssysteme -- sehen die Finanzierung der Entwicklung von OSS Produkten als solch strategischen verlustbringenden Markteintritt gegen Microsoft.

Linux Distributoren, wie RedHat, Caldera und andere, sind ausdrücklich willens, Vollzeitentwickler zu bezahlen, die ihre gesamte Arbeit der OSS Gemeinde freigeben. Indem sie diese Bemühungen gemeinsam unterstützen,

arbeiten RedHat und Caldera zusammen und glauben, die kurzfristigen Gewinne eher durch ein Vergrößern des Linux-Markets insgesamt als durch direkten Wettbewerb gegeneinander steigern zu können.

Ein indirektes Beispiel ist die Einstellung von Larry Wall - dem Anführer und Vollzeitentwickler von PERL - bei O'Reilly & Associates. Der führende Verleger von PERL Nachschlagewerken ist, natürlich, O'Reilly & Associates.

Kurzfristig zahlen sich solche Investitionen aus - besonders, da das OSS Projekt am steilsten Punkt seiner Wachstumskurve ist. Auf lange Sicht werden finanzielle Gründe die Entwickler eher in Richtung eigenständiger Erweiterungen als in Richtung Freigabe von OSS lenken.

Entwicklungen von Nutzern frei verfügbar machen

Dieses ist sehr eng mit dem "Markteintritt mit Verlust" - Geschäftsprinzip. Allerdings versuchen diese Unternehmen nicht ihre Grenzerträge bei Dienstleistungen durch Vergrößerung des Marktes zu erzielen, sondern sie vergrößern den Ertrag in ihrem Teil der Wertschöpfungskette, indem sie Ergebnisse der Anwender frei verfügbar machen.
Das beste Beispiel sind im Moment die Verkäufer schlankerer Server wie Whistle Communications und Cobalt Micro, die aktiv Entwickler für SAMBA bzw. Linux bezahlen.
Beide, Whistle und Cobalt, erzielen Ertrag durch Hardwarevolumen. Folglich ermöglicht die Unterstützung von OSS es ihnen, den heutigen PC-Markt zu umgehen, auf dem eine "Steuer" an den OS-Händler bezahlt werden muß (NT Server Ladenpreis liegt bei $800, wohingegen Cobalts Zielcorstellung bei $1000 liegt).

Die ersten Apache Entwickler wurden von finanziell schwach ausgestatteten ISPs und ICPs eingestellt.

Ein anderes, neueres Beispiel ist das Abkommen von IBM mit Apache. IBM macht den HTTP-Server zum frei verfügbaren Gut und hofft, dadurch Gewinne bei den technisch anspruchsvolleren Anwendungen, die sie mit der Verteilung von Apache koppeln, einzufahren (und nebenbei den Marktanteil von Apache zu erreichen)).

Der Erste sein - jetzt bauen, später kassieren

Eine der herausragenden Qualitäten von OSS -- erfolgreiche OSS Projekte verschlingen weniger erfolgreiche in ihrem Bereich - legt ein "Vorkaufs-" Geschäftsmodell nahe, in dem durch direkte Investitionen in OSS heute spätere Konkurrenzprojekten ausgeschaltet werden können, vor allem, wenn das Projekt von der API-Verbreitung abhängt. Das entspricht dem Erzielen des Vorteils, als Erster auf dem Markt bei OSS zu sein.

Zusätzlich sind die Entwickler Bandbreite, Weiterentwicklungsrate und der Vorteil der Verläßlichkeit des OSS Prozesses ein Segen für die kleinen Existenzgründer, die sich typischerweise keine große hausinterne Entwicklungsabteilung leisten können.

Beispiele von Existenzgründungen in diesem Bereich sind SendMail.com (die eine kommerziell unterstützte Version des SendMail Posttransport-Agenten erstellten) und C2Net (die kommerzielles und verschlüsseltes Apache erstellten)

zu beachten ist, daß kein Fall einer erfolgreichen Existenzgründung beobachtet wurde, aus der ein OSS Projekt *neu enstand*. In beiden Fällen, existierte das OSS Projekt schon *vor* der Existenzgründung.

Sun Microsystems kündigte kürzlich an, daß ihr "JINI" Projekt via OSS veröffentlicht wird und damit möglicherweise eine Anwendung dieses "Vorkaufs-" Prinzips darstellt.

Linux

Die nächsten Abschnitte analysieren die bekanntesten OSS Projekte, darunter Linux, Apache und seit neuestem Netscapes OSS Browser.

Ein zweites Memo mit dem Titel " Analyse der Wettbewerbsfähigkeit von Linux OS" zeigt einen tieferen Einblick in das Linux Betriebssystem. Hier führe ich lediglich die wichtigsten Ergebnisse meiner Untersuchugn auf.

Was ist es?

Linux (ausgesprochen "LYNN-ucks") ist das OSS Betriebssystem mit dem größten Marktanteil. Linux basiert stark auf den mehr als 25 Jahren Erfahrung, die mit dem UNIX Betriebssystem gesammelt wurden.

Die wichtigsten Leistungsmerkmale:

Mehrfachnutzung/Mehrfachvernetzung (Kern & Benutzer)

Multi-Plattform (x86, Alpha, MIPS, PowerPC, SPARC, usw.)

Geschützter 32-Bit-Speicher für Anwendungen; Unterstützung von Virtuellem Speicher (64-Bit in Entwicklung)

SMP (Intel & Sun CPU's)

unterstützt mehrere Dateisysteme (FAT16, FAT32, NTFS, Ext2FS)

Hochleistungsnetzwerk

- NFS/SMB/IPX/Appletalk Netzwerke
- Schnellster in Unix-Unix Vergleichstests

Laufwerksverwaltung

- Stripping, Spiegelung, FAT16, FAT32, NTFS

Xfree86 GUI

Linux ist sowohl als Betriebssystem als auch als Entwicklung real und glaubwürdig.

Wie bei anderen Open Source Software (OSS) Produkten ist auch hier das eigentliche Prinzip von Linux nixht die bestimmte Version des Produktes, sondern vielmehr die damit verbundene Vorgehensweise. Dieser Vorgang verleiht den Linux Investitionen von Kunden die Glaubwürdigkeit und einen Hauch von Zukunftssicherheit.

Verläßlich in kritischen Anwendungsumgebungen. Linux wurde in praxisrelevanten kommerziellen Umgebungen eingesetzt und hat eine breite öffentliche Zustimmung gefunden.

Linux = Das Beste der UNIX-Zucht. Linux übertrifft viele andere UNIXes bei den meisten wichtigen Leistungskriterien (Netzwerkfähigkeiten, Laufwerkseingabe/ausgabe, Prozeß ctx switch, usw.).Um seine Leistungspalette zu vergrößern hat Linux auch großzügig bei anderen UNIXen gestohlen (Shell-Leistungen, Dateisysteme, Grafiken, CPU Ports)

Einziges Unix Betriebssystem, das seinen Marktanteil ausbaut. Linux ist auf dem Wege den x86 Unix-Markt eines Tages zu beherrschen, und es war in den letzten Jahren die einzige Unix-Version, die Marktanteile im Server OS Bereich gewinnen konnte. Ich glaube, daß Linux -- mehr als NT -- in naher Zukunft die größte Bedrohung für SCO sein wird.

Der Linux Prozeß verläuft SEHR schnell. Das Linux Äquivalent des TransmitFile() API beispielsweise brauchte von der Idee bis zur endgültigen Ausführung ca. 2 Wochen Zeit.

{ *Alles wahr. Ich hätte es selber nicht besser sagen können :-).* }

Linux ist eine kurz- und mittelfristige Bedrohung bei Servern

Die primäre Bedrohung für Microsoft, die von Linux ausgeht, betrifft den NT Server.

Die zukünftige Überlegenheit von Linux gegenüber NT Server (und andere UNIXes) basiert auf einer ganzen Reihe von Faktoren:

Linux benutzt weit verbreitete PC Hardware und kann durch die modulare Bauweise des Betriebssystems auch auf kleineren Systemen laufen, als das NT möglich ist. Linux wird oft für Leistungen benutzt wie etwa das laufen von DNS auf alten 486ern in Abstellkammern.

Aufgrund seiner UNIX Herkunft verursacht Linux bei einigen Organisationen niedrigere Umstellungskosten als NT.

Die NT gegenüber empfundenen Vorteile von UNIX's in Bezug auf Ausbaufähigkeit, Verfügbarkeit, Steuerbarkeit und Interopability (????)

Linux kann gewinnen, solange Leistungen / Protokolle frei verfügbar sind

{ Wir bemerken, daß sich hier ein Leitmotiv entwickelt...

Um es ein bißchen anders zu formulieren: Linux kann gewinnen wenn Leistungen offen und Protokolle einfach und transparent sind. Microsoft kann nur gewinnen wenn Leistungen geschlossen, und Protokolle kompliziert und undurchsichtig sind.

Um es noch direkter auszudrücken: frei verfügbare Leistungen und Protokolle sind gut für Kunden; sie fördern Wettbewerb und Auswahl. ***Das heißt, damit Microsoft gewinnen kann, muß der Kunde verlieren.***

Die interessanteste Offenbarung in dieser Mitteilung ist die, wie explizit Microsoft diese Logik schon auszusprechen bereit ist }

Linux wird wahrscheinlich keine Bedrohung beim Desktop werden

Linux wird wahrscheinlich auf mittel- und langfristige Sicht keine Bedrohung auf dem Desktop-Bereich sein, aus mehreren Gründen:

Schlechte Produkte und wenig Beachtung für die Endnutzer. OSS Entwicklungsprozesse sind weit besser darin, Probleme bei individuellen Komponenten zu lösen, als die Probleme integrativer Aspekte wie "einfache Bedienung von Anfang bis Ende" zu lösen.

{ Der einfache und offensichtliche Konter zu diesem ist der Hinweis, daß Microsoft selber ziemlich schlecht beim `von Anfang bis Ende einfach zu bedienen" dasteht; was sie können, ist Systeme zu bauen, die auf den ersten Blick aussehen als seien sie das, aber in Wirklichkeit genau darin versagen hätten (und langfristig durch Fehler und mangelnde Möglichkeiten wesentlich mehr Produktivität kosten als Linux)

Obwohl dies stimmt, weicht es einem wichtigen Problem aus - nämlich, daß Microsofts eigene Aufdringlichkeit in dieser Angelegenheit seine Kritik nicht minder stichhaltig macht. Open-Source Entwicklung ist in der Tat schwach in diesen Bereichen, weil sie Tests mit Nicht-Hackern zur einfachen Bedienung nicht systematisch mit einbezieht.

*Dies wird wirklich den Fortschritt von Linux auf dem Desktop-Segment verlangsamen. Es ist allerdings unwahrscheinlich, daß es für immer stagniert - nicht, wenn Versuche wie*GNOME und KDE *in Ruhe wachsen können. }*

Umstellungkosten für Desktop Installationen Desktops zu wechseln ist schwer und ein Herausforderer muß ein bedeutenden Vorteil vorweisen können. Der Linux-Prozeß konzentriert sich mehr auf die Vorteiler des Nachahmers (z.B. das zu kopieren, was erwiesenermaßen funktioniert) und wird deswegen kaum jene Erstanbieter-Vorteile liefern, die den Anstoß zum Wechsel bieten könnten.

{ Hierin ist die Annahme versteckt, daß Innovation und die Vorteile des Erstanbieters die einzigen Möglichkeiten sind, um die voraussichtlichen Kosten des Wechsels zu decken. Das ist eine gefährliche Annahme für Microsoft; es kann sich herausstellen, daß die überlegene Verläßlichkeit und Stabilität von Linux ausreichend.

Selbst wenn man der Annahme des Autoren folgt, kann die Möglichkeit des Erstanbieter-Vorteils für Linux nur dann endgültig ausgeschlossen werden, wenn das Open Source - Prinzip zu Innovationen unfähig wäre - und wir wissen bereits, daß das nicht stimmt. }

Die UNIX Abstammung wird das Wachstum verlangsamen. Einfache Benutzbarkeit muß von Grund auf

einbezogen werden. Die Hackerorientierung von Linux wird in diesem Bereich niemals den Erfordernisse des normalen Desktop Benutzer gerecht werden.

{ Meine . über das Schaffen von einfacher Benutzbarkeit, und die Möglichkeiten der Open Source Gemeinde, diesem Vorwurf zu begegnen, treffen hier zu. Wir müssen Microsoft auf dem falschen Fuß erwischen, indem wir Systeme bauen, die die Offenheit benutzen, um den Nutzern die rasche Optimierung ihrer Umgebung zu ermöglichen - so, wie es das Web tut}

Linux schlagen

Zusätzlich zu dem Angriff auf die allgemeinen Schwächen der OSS Projekte (z.B. Integrative / Architektonische Kosten) sind weitere spezifische Angriffspunkte:

UNIX Besiegen

Alle bekannten Produktthemen aus der Debatte NT contra Sun gelten für Linux.

Baue zusätzliche Funktionen in bestehende und verbreitete Protokolle / Leistungen ein und schaffe neue Protokolle

Die Basis von Linux ist zur Zeit verbreitete Netzwerk- und Server-Infrastruktur. Indem wir erweiterte Funktionen (z.B. Storage+ in Dateisysteme, DAV/POD für Netzwerkanwendungen) in heutige allgemeingültige Leistungen einbauen, legen wir die Latte etwas höher & ändern die Spielregeln.

{ Hier, wie im vorigen Kommentar über ., fangen wir an zu sehen, wie die tatsächlichen Umrisse einer Microsoft Strategie aus dem Dunst der Unternehmenssprache hervorsteigen. Und es ist kein schöner Anblick; in der Tat, es ist häßlich genug um es angemessen erscheinen zu lassen, daß Mitternacht in der Halloween-Nacht heranrückt, während ich schreibe.

Worauf der Autor hinaus will ist nichts geringeres als der Versuch, die gesamte allgemein gültige Netzwerk und Server- Infrastuktur (TCP/IP, SMTP, HTTP, POP3, IMAP, NFS, und andere offene Standards) in Protokolle zu verwandeln, die zwar den gleichen Namen tragen, aber in Kunden- und Marktkontrollprotokolle gewandelt wurden (und das meint der Autor wirklich, wenn er die Microsofter dazu anhält, "die Latte höher zu legen und die Spielregeln zu ändern".
Das "Erweitern bestehender Funktionen" ist eine schönfärbende Umschreibung für die Einfuhr nicht standardgemäßer Funktionen (oder von anderen Protokollen), die dann bis zur Sättigung des Marktes als Standard bezeichnet werden, obwohl sie geschlossen sind, nicht dokumentiert, oder gerade so weit beschrieben, daß die Illusion der Offenheit entsteht. Das Ziel ist es, die neuen Protokolle ein Kriterium für leichtgläubige Unternehmenskunden werden zu lassen und dabei gleichzeitig das Erstellen von Ergänzungen zu Microsoft-Programmen von Dritten unmöglich zu machen (wer es trotzdem schafft, wird aufgekauft).
Das Spiel nennt sich "Umarmen und Ausweiten". Wir haben schon Microsoft bei diesem Spiel beobachtet, und sie sind sehr gut darin. Wenn es gelingt, gewinnt Microsoft ein Monopol. Kunden verlieren.

(Diese Strategie des Verschmutzens von allgemeinen Standards paßt genau zu den Versuchen von Microsoft, Java zu korrumpieren und die Marke Java zu zerstören.)

Open-Source-Fürsprecher können kontern, indem sie darstellen wie und warum genau das zu Lasten der Kunden geht (verringerter Wettbewerb, höhere Kosten, niedrigere Verläßlichkeit, verlorene Möglichkeiten). Open-Source Fürsprecher können dies auch unterstreichen, indem sie die das Gegenteil tun - nämlich zeigen, wie Open-Source und offene Standards den Verkäuferwettbewerb fördern, Kosten verringern, Verläßlichkeit verbessern und Möglichkeiten schaffen.

Wieder einmal ist das Internet unser Vorzeigekind . vorher in der Mitteilung eingeräumt hat. Unsere beste Gegenwehr gegen "umarmen und erweitern" ist der Hinweis auf dem Versuch von Microsoft, das Internet einzuschließen. }

Netscape

In einem Versuch, seine Glaubwürdigkeit im Browserbereich wieder herzustellen, hat Netscape neulich seinen Mozilla-Quellcode freigegeben und versucht, eine OSS-Gemeinde darum zu schaffen.

Organization & Lizensierung

Netscapes Prinzip bei Organisation und Lizenzsierung basiert locker auf dem der Linux-Gemeinde & GPL, mit einigen wenigen Unterschieden. Erstens, Mozilla und Netscape Communicator sind zwei verschiedene Codebasen, die von Netscapes Technikern synchronisiert werden.

- Mozilla = die OSS, frei verteilbarer Browser
- Netscape Communicator = mit Markenname versehene, leicht abgewandelte (so ist per Vorgabe die Homepage auf home.netscape.com gesetzt) Version von Mozilla.

Im Gegensatz zu vollständiger GPL behält Netscape sich das abschließende Recht vor, Modifikationen in die Codebasis von Mozilla aufzunehmen oder die Aufnahme zu verwehren, und die Ingenieure von Netscape sind die ernannten "Bereichsleiter" größerer Bestandteile (bis jetzt).

Vorzüge

Die Anti-MSFT - Einstellung in der OSS Gemeinschaft ausnutzen

Im Vergleich mit anderen OSS-Projekten wird Mozilla als eine der direktesten kurzfristigen Angriffe auf das Territorium von Microsoft betrachtet. Das allein gibt Entwicklern vermutlich den Anstoß, sich an der Mozilla-Codebasis zu beteiligen.

Neue Glaubwürdigkeit

Die Verfügbarkeit des Mozilla-Quellcodes hat Netscapes Glaubwürdigkeit im Bereich der Browser ein wenig gestärkt. Wie BharatS in http://ie/specs/mozilla/default.htm ausführt:

> "Mit der Veröffentlichung des Codes haben sie dafür gesorgt, daß sie niemals völlig verschwinden werden, wie es WordStar damals ergangen ist. Mozilla Browser werden locker die nächsten 10 Jahre überleben, selbst wenn die Benutzergruppe schrumpfen sollte. "

Ein großes Problem lösen

Der Browser ist weitläufig verbreitet und benutzt. Folglich wird die Anzahl der Leuten, die bereit sind, ein unmittelbares Problem schnell zu lösen und/oder einen Fehler zu beseitigen, sehr groß sein.

Schwächen

Entwicklungsgleichstand

Mozilla ist bereits sehr nah am IE4/5. Folglich gibt es kein starkes Vorbild mehr, daß durch seine Führungsrolle das Entwicklungsteam koordiniert.

Netscape hat einige seiner besten Entwickler mit der Vollzeit-Aufgabe beauftragt, die Codebasis vom Mozilla zu pflegen, und es wird eine spannende Sache zu beobachten, wie dieses hilft (wenn überhaupt) die Fähigkeiten von Mozilla weiter in neue Bereiche zu bringen.

Kleine Noosphere

Ein interessanter Nachteil ist die Größe der verbleibenden "Noosphere" des OSS Browsers. Der unabhängige Browser ist im wesentlichen fertig.

Es gibt keine großen hochkarätig Segmente im Browser, die dringend weiterentwickelt werden müssen. In anderen Worten, Netscape hat die 80% der Probleme, die interessant waren, gelöst. Es gibt nur wenig/ keine Befriedigung des Egos, bei den verbleibenden 20 Prozent die Fehler zu beseitigen und sie zum laufen zu bringen.

Netscapes kommerzielle Interessen senken den Effekt der Noosphere-Beiträge.

Linus Torvalds' Verwaltung der Linux Codebasis ist indiskutabel auf das Ziel ausgerichtet, das bestmögliche Linux zu erschaffen. Netscape dagegen behält sich ausdrücklich das Recht vor, Entscheidungen über die Verwaltung des Codes im Sinne von Netscapes *kaufmännischen* Interessen zu treffen. Der Entwickler trägt mit seinem Code also nicht zu einen wichtigen Produkt bei, sondern wird dem Börsenkurs von Netscape untergeordnet.

Integrationskosten

Möglicherweise der größte Problem der Mozillabemühungen ist der Grad an Integration, den Kunden als Leistungsmerkmal eines Browsers erwarten. Wie weiter oben bereits festgestellt, ist die Integrationsentwicklung nicht parallelisierbar und wird daher durch den OSS-Prozeß behindert.

Vor allem ein guter Teil der neuen Bestandteile des IE5+ bedeuten nicht mehr lediglich das Integrieren neuer Komponenten in den Browser, sondern die Integration Browsers in das Betriebssystem. Dagegen anzukommen wird besonders schmerzhaft sein.

Voraussagen

Die Vorhersage ist deshalb, Netscapes Versuch mit Mozilla (anders als die Apache und Linux-Pläne, die momentan recht erfolgreich sind) verläuft wie folgt:

- sie werden den beherrschenden Browser für Linux und einige andere UNIX herstellen.
- auf lange Zeit hinter den IE zurückfallen.

Wenn man bedenkt, daß der Quellcode erst vor kurzem freigegeben wurde (April '98), gibt es bereits Anzeichen dafür, daß das Interesse an Mozilla schwindet. Äußerst unwissenschaftliche Belege finden sich im Rückgang des Umfangs Mail-Verteiler zum Thema Mozilla von April auf Juni.

Mozilla Mail Liste	April 1998	Juni 1998	% ablehnen
gewünschte Eigenschaften	1073	450	58%
UI Entwicklung	285	76	73%
allgemeine Diskussion	1862	687	63%

Interne Kopien der Mozilla Mailing-Listen finden sich unter http://egg.Microsoft.com/wilma/Listen

{ Ha. Dieses "Osterei" ist Linux-basiert }

Apache

Geschichte

Übernommen von http://www.linux.org**.**

Im Februar 1995 war die meist verbreitetste Server Software des WWW der frei verfügbare HTTP-daemon, der von der NCSA, Universität von Illinois, Urbana-Champaign, entwickelt wurde. Die Weiterentwciklung wurde jedoch nach Mitte 94 immer weiter aufgeschoben, und viele Netzverwalter hatten ihr eigenen Erweiterungen und Fehlerkorrekturen entwickelt, die nun verteilt werden mußten. Eine kleine Gruppe dieser Netzverwalter, privat mittels E-Mail zusammengerufen, versammelte sich, um ihre Veränderungen (in Form von Patches - "Flicken") zu koordinieren. Gegen ende Februar`95 bildeten 8 Kernentwickler die Grundlage der ursprünglichen Apache-Gruppe. Im April 1995 wurde Apache 0.6.2 veröffentlicht.

Zwischen Mai und Juni 1995 wurde eine neue Server Architektur entwickelt (Codename Shambhala), die eine modulare Struktur, ein API für bessere Erweiterbarkeit, eine poolbasierte Speicherzuweisung und ein anpassungsfähiger Mehrprozeßmodell(Forking) enthielt. Die Gruppe sattelte im Juli auf dieser neue

Serverarchitektur um und fügte die Leistungsmerkmale der Version 0.7.x hinzu, im August schließlich den Apache 0.8.8 (und seine Nachfahren) ergab.

Weniger als ein Jahr, nachdem die Gruppe geformt wurde, überholte der Apache Server NCSAs HTTPd und wurde der Nummer 1-Server im Internet.

Organisation

Das Apache Entwicklungsteam besteht im Kern aus ungefähr 19 Mitgliedern, dazu kommen Hunderte von Internet-Verwaltern aus der ganzen Welt, die in der einen oder anderen Weise Fehler aufgedeckt und behoben haben. Eine Übersicht über die Fehler von Apache kann hier eingesehen werden: http://bugs.apache.org/Index.

Eine Beschreibung der Code-Verwaltung und der Weise, wie das Apache-Team untereinander entstehenden Streit löst, findet man auf http://www.linux.org**.**

Führung:

Es gibt eine Kerngruppe von Beitragenden (salopp "Kern" genannt), die von den Projektgründern gebildet wurde und die von Zeit zu Zeit erweitert wird, wenn Mitglieder außergewöhnlich gute Mitstreiter nominieren und der Rest des Kerns zustimmt.

Konfliktlösung:

Änderungen am Code werden via dem Postverteiler vorgeschlagen und normalerweise stimmen die aktiven Mitarbeiter darüber ab -- drei +1 (ja Stimmen) und keine -1 (Nein-Stimme oder Veto) sind Bedingung für einen Codewechsel während eines Veröffentlichungszyklus.

Vorteile

Marktanteil!

Apache ist zur Zeit die absolute Nr.1 unter den Webservern im Internet. Der Besitz des Löwenanteils am Markts ist eine äußerst effektive Kontrolle über die Marktentwicklung.

Apaches Marktanteil im Webserver Bereich setzt dem Wettbewerb vor allem folgende Hürden:

- Der kleinste gemeinsame Nenner HTTP Protokoll - bremst unsere Möglichkeiten, das Protokoll so zu erweitern, daß es neue Anwendungen unterstützt.
- schürt die Verbreitung von UNIX-- wohin Apache geht, muß UNIX folgen.

Unterstützung von dritter Seite

Die Anzahl verfügbarer Werkzeuge / Module / Erweiterungen für Apache wächst zunehmend an.

Schwächen

Leistung

Kurzfristig ist IIS auf dem SPEC-Netz deutlich besser als Apache. In Zukunft, wenn sich IIS in den Kern von NT einfügen und damit den Vorteil besserer Integration haben wird, wird sich dieser Vorsprung noch vergrößern.

Apache ist im Gegensatz dazu mit der Verpflichtung belastet, lauffähigen Code für alle möglichen Betriebssysteme zu entwickeln.

HTTP Protokollkomplexität & Anwendungsdienste

Ein Grund, warum Apache so erfolgreich werden konnte, war die Einfachheit des HTTP-Protokolls. Je mehr neue Leistungsmerkmale von Browsern erwartet werden (z.B. Mehrfachserver Unterstützung, POD, usw.), desto mehr bleibt abzuwarten, wie die Apache-Mannschaft da mithalten kann.

Die Unterstützung von ASP ist beispielsweise eines der Hauptgründe für den Einsatz von IIS in Firmen-Intranets.

IBM & Apache

Vor kurzem kündigte IBM die Unterstützung der Apache Codebasis in seinen Web-Anwendung an. Das tatsächliche Ergebnis der Pressetumultes ist aber bis heute noch unklar:

IBM versendet und unterstützt nach wie vor sowohl Apache wie auch Dominos GO Webserver

IBMs Verpflichtung scheint die folgende zu ein:

- Apache zu helfen, auf strategisch wichtige IBM-Plattformen zu gelangen (wie AS/400, etc.)
- Die Apache-Binaries an Kunden zu verteilen, die Apache-Unterstützung verlangt haben
- Unterstützung von Apache-Binaries (nur wenn sie von IBM gekauft wurden?)

IBM hat Entwickler, die sich aktiv an der Weiterentwicklung und an den Diskussionsgruppen des Apache beteiligen.

IBM nimmt eine führende Rolle bei der Optimierung von Apache für NT ein.

Andere OSS-Projekte

Einige andere OSS-Projekte:

Gimp -- http://www.linux.org Gimp (GNU Image Manipulation Program) ist ein OSS-Projekt mit den Ziel, einen 'Adobe Photoshop'-Klon für UNIX Arbeitsplätze zu erzeugen. Was seine Eigenschaften in der Version 1.0 angeht, so ähnelt das allerdings eher PaintBrush

WINE / WABI -- http://www.linux.org WINE (Wine Is Not an Emulator) ist ein OSS Windows Emulationsbibliothek für UNIX. WINE konkurriert (etwas) mit SUNs WABI Projekt, das nicht OSS ist. Ältere Versionen von Office können zum Beispiel mit WINE gestartet werden, obgleich die Leistungsfähigkeit noch bewertet werden muß.

PERL -- http://www.linux.org PERL (Practical Evaluation and Reporting Language) ist die faktische Standard-Scriptsprache für alle Apache WEB Server. PERL ist auf UNIX sehr beliebt, vor allem aufgrund seiner leistungsstarken Text/Zeichenkettenverarbeitung und des Vertrauens auf Befehlszeilenverwaltung aller Funktionen.

BIND -- http://www.linux.org BIND (Berkeley Internet Name Daemon) ist der de facto DNS Server für das Internet. In vieler Hinsicht wurde DNS anhand BIND entwickelt.

Sendmail -- http://www.linux.org SendMail ist heute die Nr.1 der verfügbaren E-Mail Agenten im Internet.

Squid-- http://www.linux.org Squid ist ein OSS-Proxy Server, der auf dem ICP Protokoll basiert. Squid ist ziemlich populär bei großem internationalen ISPs, obwohl die Leistung zu wünschen übrig läßt.

{ Diese URL ist falsch.Siehe: http://squid.nlanr.net}

SAMBA -- http://www.linux.org SAMBA liefert ein SMB Fileserver für UNIX. Vor kurzem hat es die SAMBA Mannschaft geschafft, auch einen NT Controller für UNIX zu entwickeln. SGI engagiert sich ebenfalls stark in SAMBA. Http://www.sonic.net/~roelofs/Berichte/linux-19980714-phq.html: " Gegen Ende des Jahres wird Samba in der Lage sein, alle primären NT Funktionen vollständig zu ersetzten."

{ Die Samba URL ist falsch. Siehe: http://samba.org.au. }

KDE-- http://www.linux.org "K"-Desktop-Environment. Kombiniert integrierten Browser, Oberfläche/Desktop und eine Bürosoftware für UNIX Oberflächen. Die Bildschirmansichten finden sich unter: http://www.linux.org und http://www.linux.org..

Majordomo -- Der vorherrschende Postverteiler im Internet ist in PERL geschrieben und ebenfalls ein OSS-Projekt.

Microsofts Antwort

Generell ist eine weitaus ausführlichere Diskussion notwendig, um die Antwort von Microsoft auf OSS zu formulieren. Das Ziel dieses Dokuments ist es, über OSS zu informieren und zu analysieren. Daher zeige ich hier nur eine sehr oberflächliche Auswahl an Möglichkeiten und Problemen auf.

Produkt-Verwundbarkeit

Wo wird Microsoft in naher Zukunft am ehesten eine Bedrohung durch OSS Projekte spüren?

Server gegen Client

Der Serverbereich ist durch OSS-Produkte mehr gefährdet als der Client-Bereich. Gründe hierfür sind u.A.:

Clients wechseln wesentlich öfter zwischen Anwendungen hin und her -- der durchschnittliche Client-Desktop wird für eine wesentlich größere Vielfalt von Anwendungen als ein Server benutzt. Folglich sind Integration, Benutzerfreundlichkeit, "Einbauen & Fertig" usw. Schlüsselattribute.

Server sind aufgabenspezifischer. OSS-Produkte arbeiten am besten mit klar definierten Zielen oder Vorbildern - etwa, bestimmte Protokolle zur Verfügung zu stellen

Dienstleistungsserver sind weniger "verpflichtend" als Cients. Dienstleistungen wie Speichern, Druckern, E-mail-Verteilung usw durch Open Source-Alternativen zu ersetzten, beeinträchtigt die gewohnte Umgebung des Endanwenders auf dem Client nicht. Bei diesen Dienstleistungen betreiben Organisationen häufig auch experimentelle Wegwerf-Lösungen.

Server werden professionell gewartet. Dadurch kommen die Vorteile von OSS bei der Individualisierung zu tragen, und es mildert die Schwächen im Bereich mangelnder Konzentration auf den Endnutzer.

Übernahme von OSS-Errungenschaften - Ideennetzwerk der Entwickler

Die Fähigkeit der OSS-Prozesse, die kollektive Intelligenz von Tausenden Individuen über das Internet zu bündeln und zu kanalisieren, ist schlicht verblüffend. Wichtiger jedoch ist, daß die Verbreitung von OSS mit der Größe des Internets wächst, wesentlich schneller, als unsere eigenen Verbreitungsbemühungen zu wachsen scheinen.

{ *Das heißt, Microsoft unterliegt auf diesem Markt im Bezug auf Technologie UND Marketing -- und weiß es!* }

Wie kann Microsoft einen Teil der geballten Geisteskraft, die auf OSS-Produkte gerichtet ist, für sich gewinnen?

Einige Anfangs-Ideen sind:

Übernahme der Vorteile paralleler Fehlerkorrektur durch offenere Code-Lizensierung. Die Ausgabe Quellcodelizenzen von NT an Organisationen wie Universitäten und bestimmte Partner etwas wenig restriktiv handhaben.

Einstiegswerkzeuge frei oder preiswert anzubieten. Die Nebenwirkung von Werkzeugen ist, daß eine gemeinsame Wissensbasis/Grundlage geschaffen wird, die von Entwicklern eifrig verbreitet wird. NatBro verweist darauf, daß die breite Verfügbarkeit eines einheitlichen Werkzeugkastens für Entwickler von entscheidender Bedeutung für die Koordinierung von Linux/UNIX ist.

Teile des Quellcodes freigeben. Versuchen, das Interesse der Hacker daran zu wecken, zusätzlichen Mehrwert einem Microsoft-basierten Code hinzuzufügen. Teile des TCP/IP Stacks könnte hierfür der erste Kandidat sein. OshM verweist darauf, daß die Herausforderung darin liegt, einen Teil der MS-Codebasis zu finden, dessen Noosphere groß genug ist, um Interesse zu erzeugen.

Bessere Ausbaufähigkeit bieten. Der "enthusiastische Entwickler" von Linux liebt es, undokumentierte API's und Internes zu schreiben und zu verstehen. Die Veröffentlichung einiger interner APIs mit der Klassifizierung als "nicht unterstützt" könnte eine Möglichkeit sein, externe Innovationen hervorzubringen, die unsere Systemeinvestitionen fördern. Insbesondere die Garantie, daß mehr Komponenten von mehr Entwicklern schreibfähig / automatisierbar sind, wird dazu beitragen, daß besonders ausgiebige Benutzer mit unseren Komponenten 'spielen'.

Eine Gemeinde / Noosphere schaffen. MSDN erreicht eine extrem große Population. Wie können wir soziale Strukturen schaffen, die die Vorteile eines Netzwerkes entwicklen und damit diese riesige Entwicklerbasis fördern? Was etwa, wenn wir auf Microsoft.com eine zentrale VB (Virtual Basic) - Vitrine hätten, in der VB-Entwickler ihre Kreationen samt Quellcode ausstellen und mit anderen tauschen dürfen? Ich vermute, viele VB-Entwickler sähen ihr Ego ungeheuer gestärkt davon, daß man ihren Namen / ihren Code von Microsoft.com herunterladen kann.

Beobachtung von OSS-Newsgruppen. Neue Ideen kennenlernen und die besten/hellsten Leute einkaufen.

Übernahme von OSS-Errungenschaften -- Microsoft interne Verfahrensweisen

Was kann Microsoft von dem OSS Beispiel lernen? Spezieller: Wie können wir eine eigene interne OSS Entwicklungsumgebung nachahmen? Verschiedene Anmerkungen zu diesem Memo haben immer wieder darauf hingewiesen, daß wir Microsoft als idealisierte OSS-Gemeinschaft betrachten sollten, es aber aus verschiedenen Gründen nicht tun:

Verschiedene Entwicklungsweisen. Das Einrichten einer NT Entwicklungsumgebung ist äußerst komplex unterscheidet sich sehr von der Umgebung, die das Bürosoftware-Team benötigt.

Verschiedene Werkzeuge / Quellcode-Manager. Einige Mannschaften benutzen SLM, andere VSS. Verschiedene Fehler-Datenbanken. Verschiedene Entwicklungsprozesse.

Keine zentrale Sammelstelle/Codezugriff. Es gibt keine zentrale Server, um den Code von Projekten außerhalb deines eigenen Bereiches zu finden, installieren und zu begutachten. Bereits die Errichtung einer zentralen Sammelstelle für Fehlerkorrektur-Symbole wäre ein großer Fortschritt. NatBro:

"Ein Entwickler bei Microsofts, der am Betriebssystem arbeitet, kann nicht Dinge ändern, die ihn etwa bei Excel schon immer gestört haben - und dar Exel-Entwickler kann seine Anliegen an dem Betriebssystem nicht umsetzten. Herauszufinden wie erstellt, Fehlerkorrektur betrieben und installiert wird würde sie Monate kosten, und sie können vermutlich sowieso keinen vernünftigen Zugang zum Quellcode bekommen. "

Breit angelegte Entwicklerkommunikation. Postverteiler, die sich mit bestimmten Komponenten und Fehlerberichten beschäftigen, sind normalerweise ausschließlich für die Teammitglieder zugänglich.

Robustere Komponenten. Linux und andere OSS-Projekte machen es Entwicklern leicht, ohne Beinträchtigung anderer mit Komponenten zu experimentieren. DavidDs:

"Die Leute müssen ihre Teile der Projekte unabhängig von den Übrigen entwickeln, so daß die internen Schnittstellen zwischen den Komponenten gut dokumentiert und gut verfügbar sind, und außerdem robuster, da sie keine Idee haben, wie die Teile genannt werden. Das Linux Entwicklungsmodell hat sich bis zu einem Stand entwickelt, der es mehr Entwicklern ermöglicht teilzunehmen ohne ein Übermaß an Integrationsproblemen zu verursachen, da auf jeder Ebene eine gewisse Robustheit gewährleistet ist. Dies hilft der langfristigen Stabilität sehr, und man merkt es. "

Der Trick ist es nun natürlich diesen Nutzen ohne die damit verbundenen Kosten eigen zu machen. Die folgenden Kosten sind typischerweise die Gründe dafür, warum solche Barrieren überhaupt entstanden:

Integration. Ein Vollzeitentwickler hat schon genug Arbeit, ehe er auch noch die Lösungen anderer Mitarbeiter aus der Firma analysieren und integrieren muß.

Wiederholende Kosten & Zusammenhänge. Das mögliche Problem von kleinen Codespaltungen zwischen verworfenen Versionen des Betriebssystems, daß ein Excel-Entwickler noch benutzt, und dem 'Kern'-Betriebssystem das andere Excel-Entwickler benutzen.

Erweitern der OSS-Vorteile -- Service-Infrastruktur

Die Unterstützung einer Plattform- und Entwicklergemeinde erfordert eine ungeheure Serviceinfrastruktur, die OSS nicht bieten kann. Dies schließt PDC's, MSDN, ADCU, ISVs, IHVs, usw. ein.

Das OSS-Gegenstück zu MSDN ist natürlich nur ein lockerer Verbund von Webseiten mit API Dokumenten wechselnder Qualität. MS hat die Gelegenheit, das Web als Basis für die Weiterentwicklung *richtig* zu nutzen.

OSS-Angriffe blocken

Generell siegt Microsoft dadurch, daß es die Kernschwächen der OSS Projekte angreift.

Ent-Standardisierung von Protokollen und Anwendungen

OSS-Projekte waren aufgrund der weitreichenden Nutzbarkeit weitgehend frei vertriebener und einfacher Protokolle in der Lage, in vielen Server-Anwendungsgebieten einen Anteil zu erobern. Indem wir diese Protokolle erweitern und Neue entwickeln, können wir OSS-Projekten den Markteintritt verwehren.

David Stutz zeigt einen sehr gut Punkt auf: In der Konkurrenz mit Microsofts Desktop Integrierung können verbreitete Protokolle ein Mittel der Integration für OSS-Projekte werden. Es ist eine Menge an Gehirnschmalz in verschiedenen IETF-Arbeitsgruppen aufgewendet worden, die ziemlich rasch das architektonische Modell der Integration dieser OSS-Projekte entwerfen.

{ *Mit anderen Worten, frei zugängliche Protokolle müssen verschlossen und IETF zerstört werden, damit 'Protokolle und Anwendungen ent-standardisiert' werden und OSS gestoppt wird.*

Wieder einmal ist die beste Antwort der Open Source - Verfechter der Hinweis an den Kunden, daß die Ent-Standardisierung dem Verkäufer nutzt und dem Kunden schadet.}

Einige Beispiele von Microsoft-Initiativen für die Erweiterung von Protokollen:

DNS Integration mit Verzeichnisdiensten. Die Verzeichnisdienste werden erweitert, um DNS durch dynamische Aktualisierungen, Sicherheit, Authentifizierungsmechanismen aufzuwerten.

HTTP-DAV. DAV ist komplex und die Protokollspezifikation erlaubt eine unbegrenzte Implementierungskomplexität bei verschiedenen Anwendungen (so ist das Konzept für Exchange über DAV gut, aber sicher nicht die einzig offensichtliche Möglichkeit). Apache wird Schwierigkeiten haben, die richtigen ersten Bereiche von DAV zu wählen und zu implementieren.

{Welch wunderbare, beleidigende Ironie! Vier Tage nachdem Halloween I im Internet verfügbar war, veröffentlichte Greg Stein (ein Ex-Mitarbeiter von Microsoft) ein funktionierendes . als Open-Source Software. }

Strukturierte Datenspeicherung. Verändert die Spielregeln im Fileservermarkt (eine Schlüsseltechnologie für Linux/Apache). Strukturierte Datenspeicherung erzeugt zwingende Vorteile auf der Client-Seite, die auch auf den Serverbereich ausgedehnt werden können.

MSMQ für verteilte Anwendungen. MSMQ ist ein gutes Beispiel einer verteilten Technologie wo der größte Mehrwert in den Diensten und der Implementierung zu finden ist und nicht in dem Protokoll. Dies gilt auch für MTS, DTC, und COM+.

Erzwinge Integration -- besonders auf dem Server

Das Aufkommen von spezialisierten Servern ist eine besonders starke und langfristige Bedrohung, die unsere Einkommmensquellen direkt beeinflußt. Einer der Schlüssel dafür, diese Bedrohung zu bekämpfen, ist die Erschaffung integrativer Szenarien, die für die Server wertvoll sind. David Stutz zeigt auf:

"Der Punkt ist: Wer immer die besten netzwerkorientierten Integrationstechniken und Prozesse anbietet, wird den Servermarkt beherrschen. Es gibt eine Annäherung von eingebetteten Systemen, mobiler Vernetzbarkeit und durchlässigen Netzwerkprotokollen, die die Anzahl von Servern (besonders spezialisierte Server ??) explosionsartig steigen lassen wird. Der Allzweck-Client ist ein vielversprechender Geschäftsbereich - ob ihn das Geschäft mit den spezialisierten Servern in den Schatten stellen wird?"

Systemverwaltung. Die Funktionalität zur Systemverwaltung betrifft tendentiell alle Punkte eines Produktes / Plattform. Folglich ist dies nichts, was einfach nach dem Baukastenprinzip auf eine existierende Codebasis aufgesetzt werden kann. Systemverwaltung muß von Anfang an entworfen werden oder sie ist das Ergebnis eines bewußten Bewertungsprozess der Komponenten eines Projektes. .

Einfache Bedienbarkeit. Wie Systemverwaltung muß dies oft von Grund auf geplant sein und hat folglich hohe Entwicklungskosten. In der Konsequenz werden OSS Projekte in diesem Bereich Probleme haben

Szenarien bewältigen. ZAW, Einwählnetzwerke, Assistenten, usw.

Client Integration. Wie können wir den Client benutzen, um auf der Server-Seite ähnliche Integrationszwänge zu erzeugen? Zum Beispiel: Als ein Stück Zwischenware verlangt MSMQ eng synchronisierte Codebasen auf der Client- und Server-Seite .

Die Kontrolle der Zwischenware ist wesentlich. Da Server und ihre Protokolle den Marktwert in Gefahr bringen, ist es offensichtlich, durch höherwertige Funktionalität einen Teil des Server-Betriebssystemmarktes zu sichern.

Organisatorische Glaubwürdigkeit

Release- / Service-Pakete-Prozess. Durch Konsolidierung und Regelung des anstrengenden Prozesses von ständiger Aufmerksamkeit auf neueste Entwicklungen hat Microsoft einen wesentlichen Vorteil für den Kunden den OSS Projekten gegenüber.

Langfristige Verpflichtungen. Durch Mittel wie Unternehmensvereinbarungen, langfristige Forschung, Grundsatzerklärungen der Führungsebene usw. ist Microsoft in der Lage, sich an einer langfristigen Vision zu orientieren und eine deutlicher sichtbaren langfristigen Richtung zu folgen als ein OSS Projekt.

Andere interessante Links

http://www.lwn.net/ - faßt wöchentliche Ereignisse in der Linux Entwicklungswelt zusammen.

http://www.Slashdot,org- tägliche Nachrichten / Diskussionen in der OSS Gemeinschaft

http://www.linux.org

http://www.opensoursce.org

http://news.freshmeat.net/ - Informationen über die neuesten Open-Source Versionen und Projektaktualisierungen

Danksagungen

Viele Leute haben Daten beigesteuert, Korrekturgelesen, geistreiche Email eingesandt, und Analysen sowohl dieses Papiers als auch von Linux durchgeführt:

Nat Brown

Jim Allchin

Charlie Kindel

Ben Slivka

Josh Cohen

George Spix

David Stutz

Stephanie Ferguson

Jackie Erickson

Michael Nelson

Dwight Krossa

David D'Souza

David Treadwell

David Gunter

Oshoma Momoh

Alex Hopman

Jeffrey Robertson

Sankar Koundinya

Alex Sutton

Bernhard Aboba

Entwicklungsgeschichte

Datum	Revision	Kommentare
8/03/98	0.95	
8/10/98	0.97	gestartet Revision Tabelle in Kommentare gefasst von JoshCo
8/11/98	1.00	mehr fixiert, Druckexemplare für PaulMa Überprüfung

Zurück zur Halloween Seite