Technische Hilfsmittel von Open Source Projekten15

Uli Abend, Joos-Hendrik Böse, Nils Ohlmeier, Sebastian Feuerstack

February 4, 2001

Contents

1 Einleitung **1**

2 Kommunikation in Open Source Projekten **1**
2.1 Vernetzung als Grundlage dezentraler Entwicklungen 1
2.2 Identifizierung von Rollen in Open Source Projekten 2
2.2.1 Kommunikation zur Kommentierung 3
2.2.2 Kommunikation zu Koordination 3
2.3 Anforderung an die Kommunikation der jeweiligen Rollen 3
2.3.1 Anforderung an Kommentierung 3
2.3.2 Anforderung an Koordination 4
2.4 Allgemeine Kommunikationsmodelle 4
2.4.1 Blackboardmodell 4
2.4.2 Einfaches Sender-Empfänger Modell 4
2.5 Entwicklungsmodelle als Hilfsmittel 4

3 Technische Möglichkeiten und Hilfsmittel der Benutzerseite **5**
3.1 Die Akteure der Benutzerseite 5
3.1.1 Der anwendungsbezogene Benutzer 5
3.1.2 Der administrationsbezoge Benutzer 7
3.1.3 Der programmiererfahrende Poweruser 8
3.2 Softwaredistribution 8
3.2.1 Das TAR-Archiv und Komprimierungsfomen 8
3.2.2 Der RPM Package Manager 9
3.2.3 Distributionsform CD und Image 10
3.2.4 OS-Distributionen Debian und Slackware 11
3.3 Beratung und Bezug von Software 11

4 Hilfsmittel der Entwicklerseite **11**
4.1 Der Entwickler im Mittelpunkt 12
4.2 Informationsbeschaffung 12
4.3 Koordination der Ressourcen 12
4.4 Online/ Offline Problematik 13
4.5 Entwicklungstools 13
4.6 Kommunikation 13
4.7 Datenaustausch 15

4.8 Programmierung 15
4.9 Dokumentation 16

5 Probleme und Lösungsansätze auf technischer Seite 16
5.1 Plattformen im Internet 16
5.2 Probleme verteilter Entwicklung 17
5.3 Mögliche Verbesserungen 18
5.4 Existierende Lösungsansätze 19
5.4.1 Sourcetausch- und Treffpunkt-Plattformen 19
5.4.2 WikiWikiWeb 19
5.4.3 WISE 19
5.4.4 Linux Cross-Referencing (LXR) 19
5.4.5 Bitkeeper 19
5.4.6 Issuezilla 20

1 Einleitung

Der Erfolg von Open Source Projekten ist auf die effektive Zusammenarbeit grosser Gruppen zurückzuführen. Im folgenden soll untersucht werden, welche Hilfsmittel dabei in welchem Zusammenhang zum Einsatz kommen. Darüber hinaus wird versucht, Schwächen herauszuarbeiten und Lösungsansätze für diese zu finden.

2 Kommunikation in Open Source Projekten

Mit Ausnahme einiger älterer Projekte wie z.B. Emacs und der GNU C Compiler wurden alle größeren Open Source Projekte dezentral entwickelt. Open Source Projekte sind deshalb auch meist dezentral organisiert und können als eine kooperative Vernetzung der Projektmitglieder zur Erreichung eines gemeinsamen Zieles charakterisiert werden [dezUntStr]. Wobei die einzelnen Mitglieder jeweils persönlich verschiedene Ziele und Motivationen haben an diesem Projekt mitzuarbeiten. Die Summe der aktuellen persönlichen Ziele der einzelnen Entwickler und Nutzer ergibt ein sich mit der Zeit änderndes Projektziel.

Die Kooperation der einzelnen Teilnehmer beschränkt sich in der Regel nur auf einen bestimmten Zeitraum, solange die Ziele und der Nutzen des Projektes für den einzelnen interessant sind. Sind die Ziele eines Projektes für den einzelnen nicht mehr interessant, so wendet er sich von dem Projekt ab und sucht sich ein anderes Projekt oder falls noch kein Projekt existiert, gründet er evtl. selber ein neues.

2.1 Vernetzung als Grundlage dezentraler Entwicklungen

Die gesamten Informationen eines Projektes sind über die Mitglieder verteilt, jedes Mitglied hat spezielles Wissen in einem kleinen Bereich der Entwicklung. Um eine effektive Zusammenarbeit möglich zu machen, müssen die Entwickler möglichst viele Informationen austauschen, um die Entwicklung für alle insgesamt halbwegs transparent zu halten. Es müssen also technische Hilfsmittel zur Verfügung stehen, die einen Austausch von Informationen zwischen den Entwicklern möglich machen.

Da Open Source Entwicklungen frei verfügbar sind, erfolgt die Entwicklung nach dem Basarprinzip, Entwicker können Code von anderen Entwicklern einsehen und

Teile davon für eigene Projekte verwenden. Diese offene locker organisierte Gemeinschaft erfordert ein globales dezentrales Netzwerk, welches den Basar und das Kommunizieren innerhalb und zwischen Projekten ermöglicht. Der Zugriff auf dieses Netzwerk muss jedem Interessierten einfach möglich sein, um eine grosse Menge von Entwicklern und Nutzern zu erreichen.

Mit der Menge der Nutzer steigert sich auch überproportional der Kommunikationsaufwand, da einzelne Tasks in immer kleinere Untertasks runtergebrochen werden und jede Schnittstellt zwischen den einzelnen Teilaufgaben weitere Kommunikation unter den Entwicklern erfordert. Die Technische Grundlage der dezentralen Entwicklung wie sie in Open Source Projekten praktiziert wird ist also als erstes ein Netzwerk. Aktuell handelt es sich dabei um das Internet. Erst mit der Entwicklung und der Verbreitung des Internets wuchsen Open Source Projekte zu der jetzigen Grösse und wurden zu einer wirklichen Konkurrenz zu kommerziellen Entwicklungen und der traditionellen Softwareentwicklung, welche in straffen, meist hierarchisch organisierten Organisationen isoliert, dem Kathedralen Modell entsprechend entwickeln.

Die wichtigsten technischen Hilfsmittel in Open Source Projekten sind die Tools, welche das verteilte Arbeiten der Entwickler und Nutzer unterstützen. Da gerade dieses dezentrale verteilte Arbeiten den Erfolg von Open Source ausmacht. Dies macht auch das folgende Zitat von Tim O'Reilly klar.

> "*The real lesson to be learned from open source communities are the techniques of network collaboration that they've pioneered*" - Tim O'Reilly

Im folgenden sollen die verschiendenen Rollen im Open Source Entwicklungsprozeß identifiziert werden, um dann die entsprechende Kommunikation zu betrachten und deren Charakteristika zu beschreiben. Anhand der Merkmale welche die einzelnen Kommunikationen aufweisen werden dann später Technische Hilfsmittel klassifiziert und beschrieben.

2.2 Identifizierung von Rollen in Open Source Projekten

Open Source Projekte werden von ihren Mitgliedern getragen, die Quelle ihrer kreativen Energie ist nicht Geld, sondern eine Vielzahl von sozialen und persönlichen Vorteilen, die durch die Zusammenarbeit in einer Gemeinschaft von Gleichgesinnten entstehen [Bravehack]. Da die meisten Entwickler nicht das Glück haben gesponsert zu werden und ohne Bezahlung arbeiten, entsteht ein Großteil der Arbeit in der Freizeit und immer dann wenn der Teilnehmer gerade etwas Zeit aufwenden kann oder will. Der zeitliche Rahmen einer Entwicklung ist also relativ schwer planbar und die Kommunikation zwischen den Teilnehmern muß genau dies berücksichtigen. Der einzig wirklich sichere Zustand des Projektes ist der aktuelle Zustand. Ansonsten ist die Entwicklung dynamisch und schwer planbar.

Es ist für einen Entwickler nicht erkennbar wann wer welchen Code produziert, den er eventuell benutzten könnte. Die einzelnen Akteure eines Open Source Projektes sind im wesentlichen *Entwickler*, *User* und *Manager*.

Ein Entwickler ist ein Projektmitglied, welches direkt an der Entwicklung mitwirkt indem es einzelne Programmodule programmiert.

Der User nutzt den aktuellen Stand des Projektes und testet dabei den von den Entwicklern erstellten Programmcode.

Ab einer gewissen Größe wird eine explizite Koordination der Entwicklungsarbeiten sowie des gesamten Projektes nötig, diese Arbeit übernehmen Manager.

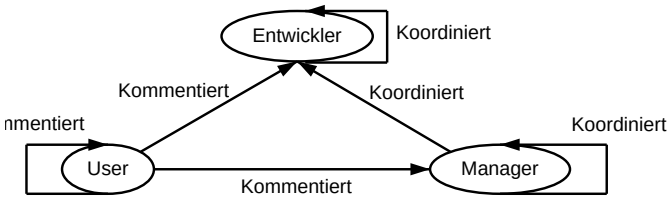
Darüber hinaus gibt es noch weitere Akteure, wie z.B. Administratoren, welche die Infrastruktur eines Projektes pflegen und dafür sorgen das die einzelnen Informationen allen Mitgliedern zugänglich sind, da diese aber nur indirekt am Softwareentwicklungsprozess beteiligt sind, werden sie hier nicht weiter betrachtet. Im wesentlichen lassen sich zwei verschiedene Arten der Kommunikation unterscheiden *Kommentierung* und *Koordination*.

2.2.1 Kommunikation zur Kommentierung

Unter Kommentierung von bestimmten Sachverhalten verstehen wir im wesentlichen das Bereitstellen von Erfahrungen über den Gebrauch eines Programmes. Dies kann das Melden von Bugs sein oder auch das Bereitstellen von Anwendungserfahrung. Das Spezialwissen der Einzelnen (wie bringe ich Soundkarte X mit Chip Y zum laufen) wird in Form von Erfahrungsberichten und Lösungsvorschlägen gesammelt und aufbereitet. Software wird nicht nur durch die offenen Quellen transparent [Telepolis1].

2.2.2 Kommunikation zu Koordination

Koordination dagegen dient der Abstimmung von verschiedenen Teilprozessen der Softwareentwicklung des Projektes und weiterer Aktivitäten der Projektarbeit, wie z.B. Organisation von Usertreffen etc..



2.3 Anforderung an die Kommunikation der jeweiligen Rollen

Technische Hilfsmittel müssen die verschiedenen Bedürfnisse der Akteure bei ihrer Kommunikation unterstützen. Dazu soll im folgenden die Charakteristika und Anforderungen der beiden Kommunikationsarten *Kommentierung* und *Koordination* beschrieben werden. Dabei ist zu beachten, dass die Kommunikation sehr abstrakt betrachtet wird. Koordination kann natürlich auch in kleinen Kreisen, z.B. unter den Kernentwicklern im Rahmen von Kommentierungen erfolgen.

2.3.1 Anforderung an Kommentierung

Eine Kommentierung ist in der Regel eine ungerichtete Kommunikation. Hat z.B. ein User ein bestimmtes Problem mit einem Programm und möchte andere User und Entwickler darauf aufmerksam machen, mit dem Ziel ein Tip für einen Workaround oder ein Bugfix zu erhalten, so sendet er ungerichtet sein Anliegen an die Projektgemeinde. Mit ungerichtet ist gemeint, dass seine Nachricht nicht an einen bestimmten Empfänger gerichtet ist, sondern an eine ganze Reihe von Personen erreichen muß, welche die Nachricht überprüfen und evtl. mit dem User Verbindung aufnehmen. Dabei ist nicht sicher, ob sich ein Mitglied der Projektgemeinde verantwortlich fühlt zu antworten. Oft ist es aber der Ehrgeiz der Projektmitglieder keine Frage unbeantwortet zu lassen.

Die Antwort kann ebenfalls ungerichtet erfolgen um wieder der gesamten Entwicklergemeinde die Informationen zur Verfügung zu stellen. Alle Kommunikation die im Rahmen von Kommentaren erfolgt steht der gesamten Gemeinde zur Verfügung.

Kommunikation im Rahmen der Kommentierung sollte also allen Interessierten einfach zugänglich sein und von allen Interessierten mitverfolgt werden können, der Nutzer muß die verantwortlichen Entwickler, Manager und anderen Nutzer nicht kennen.

2.3.2 Anforderung an Koordination

Die Koordination von Entwicklungsaufgaben erfordert eine zielgerichtete Kommunikation zwischen den einzelnen Akteuren. Diese Art von Kommunikation setzt halbwegs klar verteilte Verantwortlichkeiten im Entwicklungsprozeß voraus, da der Empfänger und seine Aufgaben bekannt sein müssen. Innerhalb einer Koordinationskommunikation werden also Teilprozesse auf direktem Wege zwischen einzelnen Akteuren koordiniert ohne, dass die gesamte Projektgemeinde dies mitbekommt. Dies könnte z.B. der Fall sein wenn sich Manager und Entwickler darüber verständigen wann ein neues Release veröffentlicht werden soll und welche Module mit einfließen sollen oder wann ein Code Freeze anzusetzten ist.

2.4 Allgemeine Kommunikationsmodelle

Um Kommunikation zur Kommentierung zu beschreiben kann man das Blackboardmodell heranziehen. Die einfache gerichtete Kommunikation zur Koordination entspricht dem einfachen Sender Empfänger Modell.

2.4.1 Blackboardmodell

Das Blackboardmodell beschreibt nicht nur die Art der Kommunikation, sondern auch ein kooperatives Problemlösungsmodell welches im Open Source Bereich zu finden ist. Entsprechend der Blackboardmetapher erfolgt die gesamte Kommunikation über eine Tafel, welche von den Teilnehmern betrachtet wird. Beiträge werden auf dieser Tafel vermerkt und von anderen Teilnehmern betrachtet, die anderen Teilnehmer können in den aktuellen Stand des Lösungsprozesses, welcher durch den Tafelinhalt beschreiben wird eingreifen. Voraussetzungen dabei sind, dass jeder alles lesen kann und die Teilnehmer können etwas zur Problemlösung beitragen wenn sie wollen, indem sie auf die Tafel zugreifen. Die einzelnen Teilnehmer müssen sich nicht kennen.

2.4.2 Einfaches Sender-Empfänger Modell

Bei der einfachen privaten Kommunikation zwischen zwei Personen müssen sich die Personen jeweils kennen. Der Inhalt der Kommunikation ist für andere in der Regel nicht ersichtlich.

2.5 Entwicklungsmodelle als Hilfsmittel

In der traditionellen Softwareentwicklung stehen eine große Menge von Vorgehensmodellen wie z.B. Spiralmodell etc. zur Verfügung. Dabei werden in der Regel die vorgegebenen Phasen nacheinander abgearbeitet. Der Entwicklungprozess ist dadurch

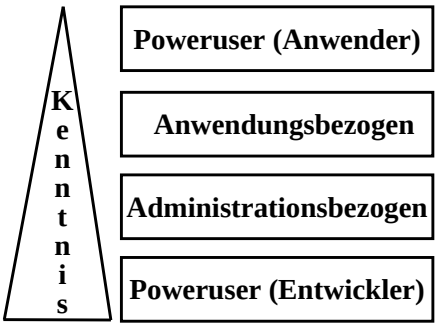
sowohl zeitlich als auch inhaltlich gut planbar. In der Open Source Entwicklung existieren keine Vorgehensmodelle mit denen sich die Entwicklung genau planen ließe, da die Entwicklung einem evolutionärem Prozeß unterliegt auf den unendlich viele Einflüsse wirken. Umfangreiche Spezifikationen werden deshalb selten angefertigt und der Entwicklungsprozeß ist nur schwer planbar [Bravehack].

3 Technische Möglichkeiten und Hilfsmittel der Benutzerseite

Dieses Kapitel soll sich mit der Benutzerseite und denen von ihnen genutzten technischen Hilfsmitteln auseinandersetzen. Im ersten Teil werden dazu die unterschiedlichen Akteure der Benutzerseite voneinander abgegrenzt und die Vermischung der Rollen in Form der einzelnen Akteure erläutert.

Die vom Entwickler eingeleitete Verteilung der Software an die Benutzer wird im Kapitel “Software-Distribution von OSS” dargestellt. Ist dagegen ein Benutzer auf der Suche nach einer Software die seinen Anforderungen an die Funktionalität entsprechen soll, bedient er sich der im letzten Kapitel dieses Abschnitts vorgestellter Hilfsmittel. Auch die Beratung (Support) wird durch Software unterstützt, die das Ende der Betrachtung der Benutzerseite darstellt.

3.1 Die Akteure der Benutzerseite



Grundsätzlich lassen sich, wie auch in der Abb. deutlich wird, die unterschiedlichen Akteure der Benutzerseite nach Ihrer Kenntnisstufe anordnen. Hierunter wird nicht nur die Programmiererfahrung des einzelnen Akteurs, sondern viel Allgemeiner, auch seine Erfahrung im Umgang mit dem Computer und seine Kenntnis der Materie der von ihm hauptsächlich eingesetzten Software verstanden.

3.1.1 Der anwendungsbezogene Benutzer

Der Poweruser ist als sog. “Anwendungsbezogener User” an ganz bestimmter Software interessiert, die er problembezogen in seinem Aufgabenfeld nutzt. Dieses ist z.B. sein Arbeitsplatz, so daß man ihm ein Spezialwissen über den Hintergrund der Problemfeldes zuordnen kann in dem die Software, die er benutzt, ihn funktionell unterstützt.

Wie greift der anwendungsbezogene Benutzer nun in den OS-Softwareentwicklungsprozeß ein? - Im Allgemeinen teilt er sich hauptsächlich durch Ideen und daraus resultierende Vorschläge mit, die er skizzenhaft in Mailinglisten oder über privater E-Mail mit den Entwicklern der Software diskutiert. Der anwendungsbez. Benutzer nimmt aber innerhalb der Mailinglisten auch eine

Beratungsfunktionalität ein, die den Entwicklern hilft grundsätzliche Fragen (z.B. Richtungsentscheidungen bezüglich der Weiterentwicklung) zu lösen, aber er hilft auch anderen Anwendern bei der Lösung ihrer alltäglichen Probleme im Umgang mit der Software. Anwendungsbezogene Poweruser sind im allgemeinen sehr aktiv in der Mailingliste und versuchen alle noch offene Bedienungsfragen anderer User der Software zu klären. Sie pflegen meistens auch die sog. FAQ der Mailingliste, in der die häufigsten Fragen zusammengefaßt beantwortet werden und nehmen so den Entwicklern ein gewisses Maß an Arbeit ab. Auch verfassen sie sogenannte HowTo's, in denen Schritt für Schritt (kochbuchartig) Anweisungen zu häufig auftretenden Problemen mit der Software gegeben werden oder verfassen ganze Tutorials, die z.B. eine Einführung in die Bedienung der Software geben. Sie haben normalerweise kein Interesse an technischen Details eines Problems, sondern suchen nur einen konkreten Lösungsvorschlag oder einen Weg das Problem zu umgehen, da sie keine bis wenig Programmiererfahrung haben.

Als Beispiele läßt sich hier Gimp[GIMP] erwähnen, in deren MLs einige anwendungsbez. Poweruser zu finden sind, die teilweise als Grafiker arbeiten und so den Umgang mit Software (z.B. Photoshop) gewöhnt sind und die Erweiterung der Funktionalität des OS-Vertreters GIMP in diesem Bereich sehr voranbringen.

Auch das LyX-Paket[LyX], welches eine Oberfläche für die Erstellung von Latex-Dokumenten bietet, wird von einigen dieser hier vorgestellten Akteure sehr intensiv gepflegt.

Wie können sich nun derartige Benutzergruppen in den OSS-Prozeß integrieren und welche Hilfsmittel haben sie zur Verfügung? Neben dem Schreiben von Dokumentation können auch Benutzergruppen mitwirken, die kein Spezialwissen über die Software haben. Der Einsatz von Tools (echten Usertools!) soll an dem Beispiel der Übersetzung von Software erläutert werden.

I18n[1] und l10n[2] Anfangs wurde für Programme nur Dokumentation von Entwicklern geliefert. Da diese meistens nur in einer Sprache und teilweise sehr unvollständig war und nur selten der aktuellen Entwicklung angepaßt wurde, schrieben Anwender selbst Dokumenation, wie z.B. HowTo's und Tutorials zu denen von ihnen benutzten Programmen. Mit der zunehmenden Verbreitung und Erfolg von OS-Software kamen allerdings folgende Problematiken auf:

1. Aktualität der Dokumentation
   Durch die ständige Weiterentwicklung von OS-Software mit kurzen Abständen zwischen den Veröffentlichungen (Releases) des Produktes und die stark verteilte Entwicklung von einander unabhängiger Programmierer, fällt es schwer die Dokumentation auf einem aktuellen Stand zu halten. Insbesondere weil die eigentlichen Entwickler der Software nur selten die von ihnen implementierten Funktionalitäten beschreiben[3], sind Handbücher meistens unvollständig und wenig aktuell.

[1] hinter dieser Kurzform versteckt sich der englische Begriff "Internationalization" und umschreibt die Übersetzung der Software und ihrer Dokumentation in verschiedene Sprachen. Die Zahl 18 zeigt hier an, daß zw. "I" und "n" 18 Buchstaben ausgelassen worden sind.

[2] l10n bedeutet "localization" und beschreibt die Anpassung einer Software nicht an die Sprache, sondern an das Land. So muß z.B. eine für Amerika lokalisierte Software, das amerikanische Datumsformat oder auch das Dollar($)-Symbol für Währungen berücksichtigen.

[3] OS-Entwickler sind selten bereit, die von ihnen implementierten Funktionen außerhalb des Quelltextes in einem dem Anwender verständlichen Form zu beschreiben.

2. Übersetzung der Software
   Wesentlich problematischer als die Aktualität der Dokumentation gestaltet sich die Übersetzung der Software in andere Sprachen. Denn sind für die Übersetzung der Anleitung meistens keine weiteren Programmierkenntnisse erforderlich, erfordert die Übersetzung der Software umfangreicheres Wissen über den Aufbau des Quellcodes und die Fähigkeit z.B. Oberflächenelemente an die Anforderungen der Sprache anzupassen. So ist zu berücksichtigen, dass die englische Sprache es erlaubt, eine Aussage mit wesentlich weniger Zeichen (Textlänge) zu beschreiben und so die von einem englischsprachigen Entwickler berücksichtigten GUI-Komponenten in ihrer Größe nicht den Anforderungen an die deutsche Sprache genügen.
   Eine weitere Problematik liegt auch hier in der Aktualität, da Übersetzer und Entwickler nur selten eine und ideselbe Person sind und so der Übersetzer nicht immer bei einer kleineren Änderung ausreichend schnell reagieren kann oder möchte.

3. Verfügbarkeit in Ländern mit anderem Zeichensatz
   Erst durch spezielle Anpassungen im Zeichensatz, insbesondere einer Erweiterung des Umfanges durch das Unicode Consortium[UNICODE], werden auch Sprachen mit einem umfangreicherem Zeichensatz berücksichtigt (z.B. Sprachen des asiatischen Raumes).

Insgesamt läßt sich daher festhalten, daß bei der Dokumentation und Übersetzung von Software auch Anwender ohne Programmierkenntnisse am OS-Entwicklungsprozeß beteiligen können, sofern ihnen die Hilfsmittel in die Hand gegeben werden, die die Komplexität des zu verändernden Quelltextes (schlechter Überblick im fremden Quelltext, keine Kenntnisse der nötigen Konstrukte zur Sprachmanipulation) kapselt und deren Bedienung durch eine Oberfläche erleichtert wird.

Das grundlegende Werkzeug zur Übersetzung von Software ist gettext[GETTEXT], welches durch parsen des Quelltextes in der Lage ist, sämtliche Texte (Fehlermeldungen, Menueinträge usw.), auf die der Anwender bei der Programmbedienung stossen kann, in einer POT-Datei[4], zu speichern. Gettext erkennt schon getätigte Übersetzungen und fügt diese in eine Datenbank ein, aus der noch nicht übersetzte Textstellen mit Übersetzungsvorschlägen zur Hilfe des Übersetzers angeboten werden können. Soll nun die Änderungen eines neuen Releases der Software übersetzt werden, findet Gettext selbständig die geänderten Stellen und schreibt auch nur diese auf Wunsch in eine POT-Datei. Der Entwickler sammelt für das Release der Software alle POT-Dateien (für jede Sprache eines) und fügt sie nach einer Konvertierung ins ".po"-Format der Software hinzu. Anwender können je nach Spracheinstellung ihres Betriebssystems die Sprache des Software anpassen.

Auf Gettext aufbauend existieren inzwischen zahlreiche Werkzeuge, die auch eine Oberfläche bieten und somit die Benutzung von Gettext noch weiter vereinfachen.[KBabel][KTranslator]

3.1.2 Der administrationsbezoge Benutzer

Der administrationsbezogen Benutzer hat ein grundsätzlich anderes Interesse an der Software. Durch seinen Beruf (z.B. als Administrator) sind ihm sowohl die Sicher-

[4] POT steht für Portable Object Template. Es handelt sich um eine Datei, die sämtliche Textstellen einer Software enthält und entsprechend Platz bietet, jeder gefundenen Textstelle einer Übersetzung zuzuordnen.

heit als auch die Stabilität der von ihm benutzten Software die wichtigsten Betrachtungspunkte.

Diese Art von Benutzern haben ein sehr starkes Bedürfnis an hochaktuellen Informationen und benutzen daher sehr ausgiebig Mailinglisten, aber auch News-Seiten im Internet um über den aktuellen Entwicklungsstand der Software immer ausreichend informiert zu sein. Aber hier liegt das Hauptinteresse weniger an neuen Features, die die Software unter Umständen bietet, sondern über Fehler (sprich Bugs) der Software immer sofort informiert zu sein und über eventuell auftretende Lösungsvorschläge (z.B. in Form von Softwarepatchen) sofort Bescheid zu wissen.

Sollte ihnen selbst ein Fehler in der Software auffallen, sind sie die ersten, die sich melden und eine teilweise sehr umfangreiche Fehlerbeschreibung liefern, um den Entwicklern die Lösung dieses Problems so weit es ihnen möglich ist zu erleichtern. Hierzu greifen adminbezog. Benutzer häufig auf sog. Bugtrackingsysteme zurück, die zur Fehlerverwaltung und Dokumentation dienen.

3.1.3 Der programmiererfahrende Poweruser

Den programmiererfahrenden Poweruser treibt ein hohes Interesse an neuen Funktionen in eine Beteilungung an der OS-Softwareentwicklung, da sie oft nach den Grundsatz “Release early and often” erfolgt. Ein Haupthilfsmittel, welches er hierzu benutzt ist ein sog. Versionsverwaltungssystem (z.B. CVS [CVS]), mit Hilfe dessen er immer in der Lage ist, die täglichen Änderungen der Entwickler am Sourcecode mitzuverfolgen und so Features ausgiebig mitzutesten. Fehlt ihm eine Funktionalität der Software, implementiert er diese selbst und nimmt so eine Anpassung der Software an seine spezifischen Anforderungen vor (sog. Patchwork). Das diese Änderungen nur auf sein System zugeschnitten sind, interessiert ihn nicht so sehr. Er gibt seine neu entwickelte Funktion allerdings in Form eines Patches weiter, so dass andere Benutzer diese auch testen und deren Nützlichkeit abschätzen können. Sollten sich eine ausreichende Menge an Benutzern finden, die diese Erweiterung als gut empfinden und der Patch gewisse Anforderungen an die allgemeine Verwendbarkeit (Systemunabhängigkeit) erfüllt, erfolgt das sog. “Check-In” in den CVS-Baum.

3.2 Softwaredistribution

Möchte eine Entwicklergruppe in der OS-Softwareentwicklung ein Projekt an interessierte Anwender verteilen erfolgt dies meist in einem sog. Paket (oder Archiv), also in Form einer einzelnen Datei, in der sämtliche benötigte Programmteile zusammengefaßt und unter Umständen komprimiert vorliegen. Der Benutzer benötigt, ebenso wie der Entwickler zu Erstellung der Archiv-Datei, ein Hilfsmittel zu Handhabung. In diesem Kapitel werden kurz einige gängige Hilfsmittel vorgestellt und insbesondere in Hinblick auf die historische Entwicklung dieser Tools und die gestiegenen Anforderungen, wird ein Schwerpunkt dieses Kapitels der gegenwärtig komplexeste Paket-Manager, der ”RedHat-Package-Manager” (kurz rpm) sein. Abgeschlossen wird die Betrachtung in diesem Teil mit einem Blick auf die verschiedenen Aggregationsstufen der Software-Distribution.

3.2.1 Das TAR-Archiv und Komprimierungsfomen

Bei Tar[TAR] (bedeutet “tape archiver”) handelt es sich um eine Software, die es ermöglicht beliebig viele Dateien und ihre Hierarchiestrukturen in eine einziges Archiv

zu “packen” und ausgehend von diesem auch wieder durch eine Restaurationsvorgang die Ursprungsstruktur wiederherstellen. Mit Hilfe von Tar lassen sich drei wichtige Funktionalitäten abdecken:

1. Storage, der Datenspeicherung mehrerer Dateien und der mit ihnen verbundenen Metainformationen, wie die Größe der Dateien, ihr Erstellungsdatum und die mit ihnen verbundenen Benutzerrechte. Das Handling der Dateistrukturen wird in dieser Archivform erleichtert, so dass ein Transfer z.B. per FTP für den Anwender erleichtert wird.

2. Backup, zur Langzeitsicherung erlaubt Tar durch sequentielle Sicherung der einzelnen Dateien im Archiv für die Sicherung auf Bandlaufwerke (das ursprüngliche Entwicklungsziel bei Tar, daher auch “tape archiver”). Unterstützt werden hier auch inkrementelle Backupformen, die mit Hilfe der im Archiv konservierten Strukturen nur Dateien, die manipuliert worden sind hinzufügt oder löscht.

3. Transport, es lassen sich Systemzustände über Tar festhalten, die zu einem anderen bewegt werden und dort wieder extrahiert werden können. Bestimmte Entwicklungsstände einer Software lassen sich auf diese Weise regenerieren.

Mit Hilfe des Werkzeugs gzip [GZIP], einem Hilfsmittel welches ebenso wie Tar inzwischen zur Grundausstattung eines UNIX basierten Betriebssystem gehört und auch von der Free Software Foundation [FSF] betreut wird, läßt sich der Einsatz von Tar weiter effektiveren. So erlaubt gzip die Komprimierung von Dateien bzw. Archiven und verringert die Dateigröße, abhängig vom Menge, Form des Inhalts (Text,Grafik) um durchschnittlich 40-50%.

Aufgrund ihrer Historie (von der FSF betreute Basistools, Entwicklungsstart ca. 1980) und der damit erreichten Stabilität und Verfügbarkeit von Portierungen für sämtliche Betriebssysteme sind tar und gzip gegenwärtig die beliebtesten Hilfsmittel zur Erstellung einer OS-Software-Distribution auch trotz der im folgenden Kapitel vorgestellten Einschränkungen.

3.2.2 Der RPM Package Manager

Der RPM Package Manager (kurz rpm) [RPM] wurde ursprünglich von RedHat [RedHat] für die von Ihnen vertriebene Linuxdistribution entwickelt und ist gegenwärtig in der GPL-Lizenz verfügbar. Die Entwicklungsziele lagen hauptsächlich in der Ausmerzung der Schwachstellen der im vorangegangenen Kapitel vorgestellten Hilfsmittel tar und gzip:

1. “Upgradability”, also die Vereinfachung der Aktualisierung von Softwarepaketen. Rpm ist aufgrund einer zentralen, lokalen Datenbank, die auf dem zu wartenden System gepflegt wird in der Lage selbständig zu erkennen, ob eine Aktualisierung oder eine Neuinstallation der Software durchgeführt werden soll. Ist bei tar immer eine Reinstallation notwendig, erkennt rpm dagegen die notwendigen Änderungen am System und spielt nur die aktualisierten Komponenten der Software ein, bzw. löscht nicht mehr benötigte Komponenten. Abhängigkeiten, die tar grundsätzlich nicht berücksichtigt, erkennt rpm und fordert vor der Aktualisierung die Installation der entsprechenden Bibliotheken (Libraries).

2. “Powerful Querying”, der RPM Package Manager bietet zahlreiche Suchfunktionalitäten sowohl auf der lokal angelegten Datenbank, die Informationen zu den installierten Paketen beinhaltet, als auch im Internet über RPMFind [RPMFind] mit dessen Hilfe über Suchbegriffe nach RPM-Paketen gesucht und Abhängigkeiten automatisch bei der lokalen Installation aufgelöst werden, aber auch komplexere automatische Aktualisierungen von Softwarekomponenten durchgeführt werden können.

3. “System Verification”, erlaubt die Überprüfung der auf dem System installierten Pakete auf vom Benutzer versehentlich gelöschte Bestandteile oder zerstörte Dateien, wobei aber von Benutzer bewußt an seine Systemanforderungen angepaßte Konfigurationsdateien auf Wunsch nicht zurückgesetzt werden.

4. “Pristine Sources” berücksichtigt immer die ursprünglichen Quelltexte der Software. In der OS-Softwareinstallation werden Anpassungen der Software nicht nur auf Konfigurationsebene, sondern auch auf der Quelltextebene vorgenommen und in Patchform[5] im RPM bzw. dem SRPM[6] verwaltet. Soll nun ein Upgrade der Ursprungssoftware vorgenommen werden, aktualisiert RPM die Komponente ohne die Anpassungen des Benutzers zu verwerfen, so dass er seine Änderungen an der Ursprungssoftware bei Bedarf auch auf die neu eingespielte Version anwenden kann.

Mit rpm steht ein sehr mächtiges Werkzeug zur Verfügung, welches auch eine immer größere Verbreitung findet, deckt es doch die bei tar fehlenden Funktionalitäten, der auberen Deinstallation, der Updatemöglichkeit und der berücksichtigung von Abhängigkeiten ab. Mit dem RPM Package Manager konkurriert ein Tool mit dem Namen “Debian Package Management System” (kurz deb)[deb], welches einen nahezu identischen Funktionsumfang bietet.

3.2.3 Distributionsform CD und Image

Hat die an den Anwender zu verteilende Software einen bestimmten Umfang erreicht, der sich nicht mehr von der Mehrzahl der Benutzer über das Internet in einer vertretbaren Zeit downloaden läßt, greifen die in den vorherigen Kapiteln vorgestellten Werkzeuge zur Paketerstellung nur schlecht und es muß auf alternative Verteilungsformen zurückgegriffen werden. Da nahezu jeder Anwender ein CDROM-Laufwerk besitzt, hat sich die Verteilung von umfangreicheren Softwareteilsystemen über sog. ISO-Images durchgesetzt. Bei eine ISO-Image handelt es sich ebenfalls um eine Datei, die maximal die Größe annimmt, die auf einer einzelnden CD Platz findet (ca. 650-700 MB). Erzeugt wird ein solches Image über Hilfmittel, wie z.B. mkisofs [mkisofs] und läßt sich danach mit jedem Programm, welches das beschreiben einer CD erlaubt benutzen. Ein gutes Beispiel für die Verteilung von solchen ISO-Images ist die Website [LinuxISO], welche zahlreiche komplette Linuxdistributionen zum freien Download anbietet. Als eine Alternative zum direkten Dowload über das Netz bieten sich auch

[5] ein Patch beinhaltet alle Änderungen an einer Ursprungsdatei und vereinfacht so den Austausch von Modifikationen an Dateien, da nicht die gesamte Datei, sondern nur ihre Änderungen verschickt werden. Mit Hilfe des patch-Programmes lassen sicht diese sog. Patchfiles auf eine Datei anwenden und so aktualisieren.

[6] im Gegensatz zu RPM-Dateien, in denen definitionsgemäß nur Binärdateien, also betriebspattformabhängige Dateien verwaltet werden (eine entprechende Endung wie z.B. ppc oder i386 deutet dies an), beinhalten SRPMS nur Quelltextdateien mit den nötigen Kompilierungsanweisungen.

einige Händler, wie z.B. die Buchhandlung Lehmanns Online, die diese Images teilweise in Eigeninitiative aus dem Netz holt und fertig auf CD gebrannt den Kunden anbietet.

3.2.4 OS-Distributionen Debian und Slackware

Die Aggregation von Software kann, wie schon weiter oben angedeutet, in Form kompletter Betriebsysteme verteilt werden. Bekannte Beispiele hierfür sind z.B. Debian [Debian] oder Slackware [Slackware], die komplette sog. Snapshots eines Linuxsystems erstellen, in dem unterschiedliche Softwarepakete zusammengefasst und unter Berücksichtigung deren Abhängigkeiten untereinander, zusammen mit Anleitungen und vorgefertigten Konfigurationsdateien bzw. Konfigurationshilfen auf CD oder als Image dem Anwender angeboten werden. Hier werden, abgesehen von Updates zur Fehlerbeseitigung, immer komplette Zustände eines Systems erstellt, die wobald sich umfangreichere Änderungen ergeben haben, erneut in einer neuen Version auf CD verteilt werden. Da hier immer auch Binärdateien, also direkt ausführbare Programme geliefert werden, ist eine Plattformunabhängigkeit nicht gegeben, so dass für jedes System ein spezfisches Image erstellt wird.

3.3 Beratung und Bezug von Software

Behandelte das letzte Kapitel die Verteilung von OS-Software ausgehend von den Entwicklern der jeweiligen Software, wird abschliessend in diesem Kapitel auf die vom Benutzer angestossene Bezug von Software erläutert. Es werden Hilfsmittel vorgestellt, die dem Benutzer helfen sollen eine Software zu finden, die die von ihm benötigten Funktionalitäten abdeckt. Auch benötigt er Hilfestellung (klassisch Support) bei Fragen zur Benutzug und Installation.

Um zu einem Begriff oder einer bestimmten Kategorie eine Software zu finden, existieren sog. Softwarekataloge wie z.B. Freshmeat [Freshmeat], der eine zentrale Anlaufstelle zur Suche von OS-Software bietet. Über eine Suchfunktion ist hier ebenso eine gezielte oder schlagwortartige Suche nach Programmen möglich, wie über einen hierarchischen Kategorienkatalog, der den Benutzer über zahlreiche Unterkategorien zum Ziel führt. Die Entscheidung welche Software einer Kategorie nun besonders gut geeignet ist, wird über Kommentare anderer Anwender in Beiträgen ersichtlich aber es erfolgt auch eine implizite Bewertung über die Donwloadanzahl der konkurrierenden Programmpakete. Freshmeat bietet Anwenden auch die tägliche Benachrichtigung zu neu erschienden Paketen über Newsletter per Mail an.

Braucht der Benutzer jedoch Support oder eine Hilfestellung bei der Bedienung seines OS-Systems oder spezifischer Programme, kann er auf in diesem Bereich spezialisierte Websites wie z.B. LinuxNewbie.org [LinuxNewbie] zurückgreifen, die mit Hilfe von HowTo's oder Tutorials Einführungen in die Bedienung geben oder auch eine gewisse Orientierung durch eine Vorauswahl und Bereitstellung gängiger OS-Software bietet. Hier findet der Anwender auch eine Community mit der er über Foren oder Mailinglisten sich über seine speziellen Probleme austauschen kann.

4 Hilfsmittel der Entwicklerseite

Es geht im folgenden um die Betrachtung der Entwicklerseite, insbesondere um den Entwickler an sich, seine Probleme und die Lösungen dieser. Welche Tools verwendet

er? Außerdem wird auf die Frage eingegangen, wo die konkreten Probleme bei der Entwicklung von Online-Projekten im "Offline-Mode" (Dial-Up Links) liegen?

4.1 Der Entwickler im Mittelpunkt

Es stellt sich zunächst die Frage, wer ist Der Open Source Entwickler eigentlich. Um diese Frage zu beantworten, muß man, dem früher aufgebauten Rollenmodell der Open Source Softwareentwicklung entsprechend, den typischen Ablauf betrachten. Um nicht das gesamte Modell zu wiederholen, sondern ein konkretes Bild des Entwicklers zu erhalten, geht die Betrachtung vom Poweruser mit Programmierkenntnissen aus. Dieser User zeichnet sich durch die Fähigkeit aus, selbst in irgendeiner Sprache oder Form einem Rechner Verhaltensweisen beizubringen. Darunter fallen shell/perl-Skripte genauso wie Hochsprachen. Aber auch Macro-sprachen sollen in diesem Zusammenhang als Programmiertechnik gelten. Die Ausgangsposition ist in der Regel so, daß der User ein irgendwie geartetes Problem aufgeworfen hat, welches er gerne mit Hilfe des Rechners lösen möchte, weil er sich daraus einen Effizienzgewinn verspricht oder die Aufgabe schlicht nicht ohne maschinelle Unterstützung zu bewältigen ist.

4.2 Informationsbeschaffung

Seine erste Aktion ist die Suche nach geeigneter Software. Im einzelnen sind für Ihn folgende Punkte beim Finden und bei der Bewertung einer möglichen Lösung von Bedeutung: Was gibt es bereits? Welche Produkte kommen (evtl. auch nur für Teile der Lösung) in Frage? Sind bei einem bestimmten Produkt die essentiellen Features implementiert, oder läßt sich die gewünschte Gesamtfunktionalität durch Komposition von Teillösungen erreichen? Ist die Lösung bereits stabil genug für die Anforderung? Sind bestimmte fehlende Features schon auf der todo list der Entwickler? Wann kann er mit einer zufriedenstellenden Lösung voraussichtlich rechnen? Der entscheidende Punkt, der ihn vom normalen User unterscheidet, ist daß er sich zusätzlich die Frage stellt: Wie kann ich die Entwicklung beeinflussen, falls die Lösung noch nicht vollständig die Erwartungen erfüllt?

Es kann sein, daß ohne ein Mitwirken die Anwendung seine Erwartungen erfüllt, dann ist der User in der Regel zufrieden. Seine Einflußnahme wird sich in Zukunft lediglich auf Bugreports beschränken. Er entdeckt Fehler, die ihn bei seiner regelmäßigen Arbeit mit dem Programm auffallen und wird diese in der Regel den Entwicklern mitteilen und darauf hoffen, daß diese die Probleme im nächsten Release oder per bugfix beseitigen. Sind eventuelle Programmfehler so gravierend störend für Ihn, so wird er möglicherweise auch selbst auf Fehlersuche gehen und versuchen das Problem zu beheben.

Ist allerdings keine Software in aktueller Version für seine Probleme passend, so wird er möglicherweise den Schritt vollziehen selbst die Rolle des Entwicklers einzunehmen. Dazu gibt es zwei Möglichkeiten: Integration in einen Entwicklungsprozeß eines vielversprechenden Ansatzes, oder Komplette Neuentwicklung einer Software für diesen Anwendungsfall.

4.3 Koordination der Ressourcen

In beiden Fällen ergibt sich das Problem der Koordination der Ressourcen, die für den Entwicklungsprozeß entscheidend sind. Im einzelnen sind folgende Punkte von

Bedeutung: Detaillierte Informationen zum Entwicklungsprozeß sind nötig Integration in ein vorhandenes Projekt verlangt genaue Informationen über die Abläufe der Entwicklung. Welcher Entwickler macht was wann und auf welche Weise? Zwischenmenschliche Aspekte sind in diesem Zusammenhang besonders wichtig. Wie reagiert wer auf welche Art von Idee, Anregung, Kritik usw.? Beim Neubeginn eines Projektes sind Informationen zum Entwicklungsprozeß zwar gerade erst im Aufbau, dennoch sollten potentielle Mitentwickler durch gute Informationen dazu ermuntert werden mitzumachen. Es muß eine Einordnung in die Struktur der Entwicklung erfolgen Der Entwickler muß innerhalb des Projektes eine wohl definierte und akzeptierte Position erreichen, denn nur auf diese Weise lassen sich Synergieeffekte nutzen und das Projekt erfolgreich unterstützen. Gleiches gilt für die Neuentwicklung: Hierbei muß dann nicht nur der Initiator seine Position finden, sondern auch hinzugekommene andere interessierte Entwickler. Wissenstransfer und Informationsaustausch zwischen den Entwicklern: Dieser Punkt ist von entscheidender Bedeutung. Es macht wenig Sinn, das Rad erneut zu erfinden, falls es gelingt Wissen und Know-How von einer Person auf eine andere zu transferieren. Ebenso müssen Dokumentationen und sonstige Fachquellen möglichst allen Entwicklern gleichermaßen zur Verfügung stehen. Technische Voraussetzungen müssen geschaffen werden Trotz schwindender Hardwarepreise ist auch dieser Punkt nicht bedeutungslos. Es gilt, daß die Entwickler Zugriff auf benötigte Hardware und auch das Internet haben. Ein Zugang zum Internet ist für nahezu alle Punkte von entscheidender Bedeutung.

4.4 Online/ Offline Problematik

Unter diesem Punkt ist die Problematik der verteilten Zusammenarbeit an einem Online-Projekt bei nicht dauerhafter Internet-Verbindung der einzelnen Entwickler zu verstehen. Das Problem resultiert aus der Kombination von erschwinglicher Hardware auf der einen und zur Zeit noch relativ hohen Kosten für Internet-Verbindungen auf der anderen Seite. War es früher noch so, daß viele Menschen keinen eigenen Computer besaßen und somit entweder auf Rechnern von Arbeitgebern, Universitäten anderen Einrichtungen Ihren Tätigkeiten nachgingen, ist heute ein eigener PC schon fast Standard. Eine dauerhafte Verbindung zum Internet war allerdings für Universitäten und andere Arbeitsplätze (zumindest in der Forschung) eine Grundvoraussetzung. Der PC zu Hause ist hingegen meist nur über zeitmäßig abgerechnete, relativ teure, Wählverbindungen in das Internet zu integrieren. Als Fazit bleibt die Erkenntnis, das man mit in allen Bereichen der Entwicklung damit zu kämpfen hat, nicht ständig online zu sein. Alle im folgenden dargestellten Tools und Hilfsmittel werden daher zum großen Teil daran bemessen, wie gut sie mit dieser Problematik umgehen können.

4.5 Entwicklungstools

Die Enwicklungstools teilen sich in die Bereiche Kommunikation, Datenaustausch, Programmierung und Dokumentation auf. Zunächst werden wichtige Hilfsmittel der Kommunikation vorgestellt:

4.6 Kommunikation

Das vorherrschende Kommunikationsmedium für Open Source Projekte sind Mailinglisten und deren Archive. Sie bieten viele entscheidende Vorteile: e-mail ist in allen Internet-fähigen Systemen quasi von Natur aus vorhanden. Außerdem bieten

e-mails einen sehr guten Umgang mit der Online/ Offline Problematik. Im Idealfall reicht es aus, die Mails einmal online abzuholen, dann Antworten und neue Messages offline zu schreiben und dann mit einem Mal alle Mails zu versenden. Die Onlinezeit wird minimiert. Das Durchsuchen von Mailarchiven kann zwar in der Regel nur Online ablaufen, da aber ein Abonnent einer Mailingliste in seinem Mailfolder bereits alle Nachrichten gesammelt hat, sind Mailinglistenarchive zumindest für dauerhafte Entwickler nur in seltenen Fällen von Bedeutung. Die Ordnung und das Filtern von Mails wird vom Mailclient erledigt. Dieser kann z.B. auch die Funktionalität enthalten, die Nachrichten nach Themen (Threads) zu sortieren und so abweichend von der normalen Listenstruktur eine bessere Übersicht zu gewähren. Die Hauptprobleme von Mailinglisten sind folgende: Der Traffic in Mailinglisten ist mitunter sehr groß, viele halten sich nicht an Regeln, die ein sinnvolles Arbeiten ermöglichen. Dazu zählen unter anderen: Das Thema der Mailingliste wird nicht berücksichtigt, Flamewars und unsachliche Messages werden versendet, bereits beantwortete Fragen werden zum x-ten Mal gestellt, unwichtige Themen werden diskutiert, usw. Selbst detaillierte Anweisungen zum Umgang mit Mailinglisten, wie etwa die Linux Kernel Mailing List FAQ [lkmlFAQ] werden häufig mißachtet. Spamming erschwert zusätzlich die Kommunikation, viele angemeldete e-mail-Adressen werden von Scannern ermittelt und gezielt für illegale Werbung mißbraucht. Die Reaktionszeit ist nichtdeterministisch, der Anfragende kann nicht davon ausgehen, daß der richtige oder überhaupt irgendjemand seine Mail gelesen hat. Beiträge von nicht eingetragenen Personen sind schwierig zu handeln, sich alternativ für eine kurze Nachricht viele Mails pro Tag liefern zu lassen, ist auch nicht die beste Lösung. Trotz dieser Probleme sind Mailinglisten das bevorzugte Mittel der Kommunikation in Open Source Projekten.

Das bekannteste System zur Verwaltung von Mailinglisten ist Majordomo [Majordomo], zur Darstellung als Webarchiv eignet sich beispielsweise Hypermail [Hypermail].

Ähnlich zu Mailinglisten und Archiven sind Newsgroups. Sie erfüllen letztendlich die gleiche Funktionalität. Das Newssystem sortiert Nachrichten nach Threads und erlaubt gezieltes Suchen und referenzieren auf Nachrichten. Insgesamt scheinen Newsgroups deutliche Vorzüge gegenüber Mailinglisten zu habe, der gravierendste Nachteil ist jedoch der schwache Umgang mit der Online/ Offline Problematik. Newsgroups sind ein klares Onlinemedium, bei dem sowohl das Lesen als auch das Schreiben Online erfolgen muß. Workarounds, die nicht ganz trivial sind, existieren zwar z.B. in Leafnode [Leafnode] oder suck [Suck], erfordern aber etwas mehr Aufwand als die bloße Konfiguration eines Mailclients mit Filterfunktionalität wie etwa kmail [KMail].

Es scheint, als lösten Mailinglisten die Newsgroups mehr und mehr ab, bleibt abzuwarten, wie sich diese Entwicklung angesichts von fallenden Telekommunikationsmonopolen und -preisen fortsetzen wird.

Ein weiteres Kommunikationsmittel ist der Internet Relay Chat [IRC]. Was einer Multitelefonkonferenz auf Textbasis entspricht, hat einige Besonderheiten, die sich als Vor- und Nachteile erweisen. chat ist ein realtime Kommunikationsmedium. Alle Teilnehmer müssen zur gleichen Zeit online sein, was im Vorfeld einige Kommunikation über andere Medien erfordert. Es existieren i.d.R. keine Protokolle von chat-sessions, d.h. das die diskutierten Punkte im Nachhinein entweder über andere Medien verteilt werden oder nie den Kreis der Diskutierenden verlassen. Chatten eignet sich besonders für Diskussionen zu grundlegenden oder zeitkritischen Fragen, die über nicht realtime Medien nicht sinnvoll erarbeitet werden können.

Kommunikation kann auch über WWW-Seiten erfolgen, die allerdings mühselig administriert und aktualisiert werden müssen. Das eignet sich in der Regel nur zur

Information von Unbeteiligten und zur primären Information von Entwicklern.

Eine andere Art von Kommunikation erfolgt über Bugtracking-Systeme. Diese sehr spezielle Form der Kommunikation kann sich als sehr effektiv erweisen. Bugtracking wird an anderer Stelle in dieser Darstellung behandelt.

4.7 Datenaustausch

Zum Austausch von Daten existieren mehrere technische Möglichkeiten. Zunächst gibt es das File Transfer Protocol FTP. Mittels FTP können verschiedene Benutzer auf einen Verzeichnisbaum zugreifen. Es können Rechte vergeben werden, die dem Benutzer erlauben, Dateien lesend und schreibend zuzugreifen. Eine besondere Form ist das Anonymous FTP. Hierbei werden jedem beliebigen Nutzer Zugriffsrechte auf das FTP-Verzeichnis eingeräumt (meist jedoch nur Leserechte). Zum Austausch der Programmdateien eines Softwareprojektes ist FTP jedoch nicht gut geeignet: Ein oder mehrere Verantwortliche müssen persönlich für die Konsistenz der Dateien sorgen und Versionskonflikte manuell auflösen. Schlußendlich eignet sich FTP nur zur Verteilung von Software.

Mit Hilfe von Versionskontrollsystemen wie CVS [CVS] läßt sich die gleichzeitige Arbeit an Dateien sinnvoll verwalten. In einem Repository werden sämtliche Dateien gehalten und die Nutzer erhalten bestimmte Rechte auf diese Dateien. Um Änderungen an einer Datei vorzunehmen, wird diese zunächst ausgecheckt, also heruntergeladen, dann lokal (und offline) bearbeitet und anschließend wieder eingecheckt, also hochgeladen. Die vorgenommenen Änderungen werden dann automatisch in die Datei im Repository eingefügt und protokolliert. Mehrere Benutzer können die gleiche Datei bearbeiten - kommt es dabei zu nicht automatisch auflösbaren Konflikten, muß manuell eingegriffen werden. Die große Stärke von CVS liegt in der guten Offline-Handhabbarkeit und der Einfachheit. Jede Zwischenversion kann im Nachhinein wieder zugegriffen werden, was in Kombination mit regelmäßigen Backups zu einer hohen Datensicherheit führt. CVS ist das Werkzeug der Wahl für die meisten der Open-Source Projekte.

Mit Hilfe von e-Mails können Patches zwischen Entwicklern ausgetauscht werden, die von geringer Größe sind. Das hat den Vorteil, das der adressierte Entwickler den Patch erst in Augenschein nehmen kann. Nicht jeder ist schließlich vertrauenswürdig genug, um Patches selbst im CVS-Tree einzuchecken. Wer beispielsweise mit einem Patch zur Linux-Kernel-Entwicklung beitragen will, muß diesen auf die Mailingliste posten. Nach Inaugenscheinnahme wird dieser dann (eventuell) in den Source-Tree gemergt. Da die meisten Mail-Server eine Restriktion der maximalen Größe von e-Mails besitzen, eignet sich dieses Vorgehen in der Tat nur für kleine Patches, größere Patches oder ganze Files werden in der Regel nur per Mail bekanntgegeben und auf irgendeinem Server (meist via FTP) öffentlich zum download angeboten.

4.8 Programmierung

Neben den low-level Programmierungstools wie Compiler, Debugger oder dem Make Kommando, die hier nicht behandelt werden sollen, gibt es einige Tools, die dem Entwickler das Programmieren erleichtern.

Zunächst werden hier Integrated Development Environments [IDE] betrachtet. IDEs sind prinzipiell Editoren mit der Fähigkeit, verschiedene Dateien zu verwalten. Darüberhinaus erlauben sie den Aufruf von Compiler, Debugger und anderen Tools. Die Ausgaben dieser Programme werden dann aufbereitet und erlauben beispielsweise

das direkte Anspringen von Codezeilen, die Fehler produzieren. Weitere mögliche Funktionen von IDEs sind beispielsweise GUI-Builder, grafische Debugger, Dokumentationsbrowser, Dokumentationseditor oder eine mit den Deklarationen verlinkte Objektübersicht. Keine der aktuellen IDEs kann mit den speziellen Problemen der verteilten Entwicklung besonders gut umgehen, so hat beispielsweise KDevelop [KDevelop] nur schwache CVS-Unterstützung, dennoch erlauben sie ein gutes lokales Arbeiten, solange man die Verteilungsprobleme manuell löst.

Im vorherigen Absatz wurden bereits die sogenannten GUI-Builder erwähnt. Diese erlauben auf intuitive Art das erstellen von graphischen Benutzerschnittstellen. Die meist sehr aufwendige Arbeit, diese von Hand zu erstellen entfällt dann, allerdings muß der Entwickler mit etwas ineffizienterem Code leben. Ein Beispiel ist glade [GLADE] zur Erstellung von GUIs für das GIMP Toolkit [GTK+].

4.9 Dokumentation

Zur Erstellung von Dokumentation eignet sich die Sprache SGML [SGML]. Sie dient als Quelle, die nach der Erstellung in nahezu jedes Format umgewandelt werden kann. Das hat den klaren Vorteil, das die Dokumentation nur in einer Version erstellt und aktualisiert wird. Übersetzungen in andere Sprachen werden ebenfalls in SGML realisiert. Durch Übersetzung nach HTML, TEX, PostScript, PDF und viele andere Formate entstehen für verschiedene User passende Dokumente, die alle den gleichen Inhalt haben. Inkonsistenzen werden so vermieden.

Zur Erstellung von API-Dokumentationen eignen sich spezielle Tools, die aus Kommentaren im Code HTML-Dokumente erstellen. Dazu zählt z.B. kdoc [KDoc], das ähnlich wie javadoc funktioniert. Die erstellten HTML-Seiten enthalten Objektdefinitionen mit Querverweisen zu verwendeten Objekten und eine Kurzbeschreibung der Klassen/Funktionen.

5 Probleme und Lösungsansätze auf technischer Seite

Im folgenden soll noch einmal auf die Probleme beim entwickeln von Open Source eingegangen werden, und die moderneren und komplexeren Lösungsansätze vorgestellt werden.

5.1 Plattformen im Internet

In letzter Zeit haben sich im Internet sogenannte Plattformen für Open Source Projekte etabliert. Diese Webseiten sind in erster Linie Anlaufstellen für die Entwickler von Open Source, aber auch alle Interessenten von Open Source Projekten nutzen diese Seiten auf der Suche nach Ihren Problemlösungen. Die absolute Vormachtstellung in diesem Bereich hat Sourceforge [SourceForge] übernommen, welche von VALinux [VALinux] in den USA betrieben wird. In Europa soll als Alternative zu Sourceforge gerade BerliOS [BerliOS] etabliert werden, welches nicht von einem kommerziellen Anbieter wie VALinux betrieben werden soll. Diese Plattformen bieten vor allem für die Entwickler und die Poweruser mit Programmiererfahrung die folgenden interessanten Möglichkeiten:

- Webseiten mit Datenbankanschluss. Hier können die Projekte sich vorstellen, Hilfen bereit stellen und für die Pflege bzw. Erzeugung der Seiten auch auf eine eigene Datenbank zurück greifen.

- CVS bietet die üblichen Möglichkeiten eines Versioning. Die Zugriffskontrolle erfolgt über ssh, und es existiert ein anonymer Zugang, um jeden die Möglichkeit zu bieten jederzeit den aktuellsten Entwicklerstand auschecken zu können.
- Mailinglisten (MLs) werden über den Email-Server der Plattform betrieben, und können nicht nur per Email sondern auch über die Webseiten administriert (abonniert und abbestellt) werden.
- Diskussionsforen die mittels dynamischer Webseiten erstellt werden, also leider nicht mit einem Newsreader nutzbar sind. Diese Foren dienen vor allem der Kommunikation mit den Benutzern der Programme.
- Im Bugtracking können vor allem die Benutzer der Programme gefundene Bugs eintragen, und den Status der gefundenen Bugs verfolgen.
- Plattenplatz bzw. FTP-Server mit Anonymous-Zugriff dienen vorwiegend der Ablage von stabilen bzw. alten Version, oder von aktuellen Snapshots.

Aber auch die normalen Anwender bzw. die Poweruser ohne Programmierkentnisse können diese Plattformen nutzen, indem sie sich über den aktuellen Stand der für sie interessanten Programme informieren oder herunterladen können. Auch bieten diese Plattformen natürlich die Möglichkeit eine Suche über die gehosteten Projekte zu machen, oder einfach nur die verschiedenen Kategorien der Projekte nach interessanten Programmen oder Projekten zu durchstöbern.

Diese Plattformen haben natürlich nicht nur positive Seiten, sondern auch Nachteile. So können fast alle Angebote dieser Plattformen nur online genutzt werden bzw. sie sind nicht sonderlich auf die oben angesprochene Online/Offline Problematik optimiert. Nur das CVS und die MLs sind kostengünstig zu nutzen, indem man kurz online geht um sich bzw. seine Daten auf den aktuellen Stand zu bringen, und danach gleich wieder die Verbindung zum Internet beenden zu können.

Speziell Sourceforge hat aus der Sicht der Open Source natürlich auch den Nachteil, dass sie von einem kommerziellen Anbieter betrieben wird. Dies hat zur Folge, dass es z.B. völlig unklar ist, was mit der Seite selber und den darauf gehosteten Projekten passiert, wenn VALinux in Konkurs gehen sollte, aus Kostengründen der Betrieb eingestellt werden sollte, oder sich VALinux eines Tages entschließen sollte die Daten auf der Plattform anderweitig zu nutzen.

Auch die Tatsache, dass die Plattformen eine zentrale Anlaufstelle darstellen, beinhalten einige Probleme. Zum einen widerstrebt die Idee der Zentralisierung einigen Mitgliedern der Open Source Gemeinde, die doch zu großen Teilen auf Verteiltheit beruht. Zum anderen handelt man sich mit dieser Zentralisierung aber auch technische Probleme ein. So sollte der Ausfall solch einer zentralen Anlaufstelle natürlich auf jeden Fall vermieden werden. Das Problem der Hochverfügbarkeit potenziert sich natürlich noch je mehr Projekte auf solch einer Plattform gehostet sind. Die wünschenswerte technische Konsequenz wäre natürlich eine redundante Auslegung sämtlicher Hardware, was unter dem Sichtpunkt der Kosten allerdings recht unrealistisch erscheint.

5.2 Probleme verteilter Entwicklung

In Ermangelung besserer Lösungen haben Open Source Projekte zur Zeit noch mit einigen Unzulänglichkeiten zu kämpfen. So sind MLs zwar ein einfaches und recht praktisches Kommunikationsmittel für Gruppen, aber sie werden sehr schnell unübersichtlich und sind für den einfachen Anwender nicht zielgerichtet genug. Der normale

Anwender wird z.B. davor zurückschrecken extra eine ML zu abonnieren, nur weil er einen Bug melden möchte oder nur eine mehr oder weniger einfache Frage geklärt haben möchte. Auch das abonnieren und wieder abbestellen von MLs mittels Textkommandos im Body oder Subject einer Email stellt häufig schon eine ziemlich hohe Hürde für die Anwender dar.

Ein weiteres Problem von MLs ist z.B. das man nie genau weiß, ob man von den anderen Teilnehmern gehört bzw. gelesen wurde. Es läßt sich leider nicht heraus finden, ob die Mail gelesen aber ignoriert wurde, ob keiner antworten möchte weil die Frage schon hunderte Mal beantwortet worden ist, keiner der Abonnenten eine Lösung für das Problem weiß, oder jemand gerade versucht das Problem zu lösen und es nur noch eine Frage von ein paar Minuten oder Stunden bis zur Lösung ist.

Wenn Projekte immer größer werden, wächst naturgemäß auch das Mailaufkommen in der zum Projekt gehörenden ML. Deshalb wird diese dann häufig in mehrere MLs aufgeteilt. Damit ergibt sich für die Anwender aber häufig das Problem, dass sie nicht wissen in welcher ML ihre Frage richtig aufgehoben ist, und die etwas stärker engagierten Teilnehmer müssen gleich auch noch mehrere MLs abonnieren, und das damit verbundene Handling koordinieren.

Auch die heutigen Integrated Development Environments (IDE) sind kaum auf die Bedürfnisse von Open Source bzw. verteilter Entwicklung angepaßt. Die Zielgruppe für heutige IDEs sind eher in kleineren, lokal gruppierten Entwicklergemeinschaften, wie man sie im kommerziellen Umfeld findet, zu suchen. Der hohe Kommunikationsbedarf von verteilten Entwicklung ist bisher kaum in IDEs berücksichtigt. So wäre die Integration der MLs in die IDEs aus Sicht von Open Source Projekten durchaus wünschenswert.

5.3 Mögliche Verbesserungen

Für die MLs wäre es vorstellbar, wenn diese in einer Art Netzwerk untereinander verknüpft wären. So sollte z.B. die Meldung eines Bugs in einer ML zu einem Anwendungsprogramm, der auf einen Bug im Compiler zurückzuführen ist, automatisch in die ML für den Compiler umgelenkt werden können, am besten ohne das der Anwender bzw. Autor der Meldung dazu etwas tun muß. Auch wäre eine grafische bzw. webbasierte Überwachung von Bugmeldungen bzw. Fragen für den Anwender möglich. Das ganze hätte dann Ähnlichkeit mit einem Bugtracking-System, mit dem Unterschied das es auch auf einfache Fragen angewendet werden sollte und auf Basis von Email funktionieren sollte.

Für Entwickler wäre es z.B. eine Arbeitserleichterung wenn es eine standardisierte Schnittstelle für den kompletten Datenaustausch des Entwicklers mit den oben angesprochenen Plattformen geben würde. Er müßte nach dem Onlinegehen nur einmal den Datenaustausch anstoßen, und alle von ihm benötigten Daten, Mails, Bugs usw. würden mit der Plattform abgeglichen werden. Diese Schnitstelle könnte in die bevorzugten Tools des Entwicklers integriert werden. Auch ein Wechsel der Plattform würde durch solch eine Schnittstelle erleichtert, wenn alle Plattformen diese Standardschnittstelle anbieten würden. Allerdings dürfte es im Umfeld der Open Source Entwicklung schwierig sein solch einen Standard zu kreieren bzw. zu etablieren, da in diesem Umfeld Standards häufig nur dadurch zu Stande kommen, dass sich eine existierende Lösung durch häufiges kopieren oder den fast ausschließlichen Gebrauch dieser Lösung durchsetzt.

5.4 Existierende Lösungsansätze

Zum Schluß wollen wir noch bereits existierende Lösungsansätze vorstellen, die die Arbeit in der Open Source Entwicklung erleichtern bzw. ermöglichen sollen.

5.4.1 Sourcetausch- und Treffpunkt-Plattformen

- Asynchrony [Asynchrony]
- Cosource [Cosource]
- SourceXchange [Sourcexchange]

Diese Plattformen sollen dazu anregen Sourcecode zu tauschen, sich für neue Projektideen als Entwickler zur Verfügung zu stellen, oder existierende Projekte zu unterstützen. In Kurzform könnte man von B2D- (Business to Development) oder D2D- (Development to Development) Plattformen reden.

5.4.2 WikiWikiWeb

Mit Hilfe dieser Software ist es möglich eine Art Whiteboard in HTML-Form zu realisieren. Dass heißt jeder Besucher einer Webseite mit WikiWiki Unterstützung kann über einen Link diese Editieren und so verändern. Auch das erstellen vom Malflächen auf denen jeder Besucher zeichnen kann soll mit Hilfe von Java-Applets möglich sein. [WikiWiki]

5.4.3 WISE

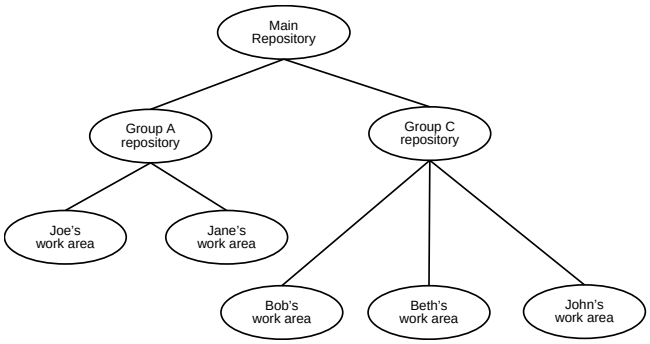
Diese von der NASA entwickelte Software soll das Management von Softwareprojekten mittels einer Weboberfläche ermöglichen. Allerdings wird die Software von den Entwicklern noch als Alpha bezeichnet, die nur auf Grund der hohen Nachfrage öffentlich zur Verfügung gestellt worden ist. [WISE]

5.4.4 Linux Cross-Referencing (LXR)

Diese aus Perl und CGI-Skripten bestehende Software durchsucht den kompletten Code eines Projektes nach den für Programmiersprachen typischen Identifiern wie Variablen, Konstanten und ähnlichem, und erstellt daraus dann Webseiten in denen diese Identifier als Links auf die anderen Codeteile ausgeführt sind. Mit Hilfe dieser Software kann man z.B. den Linuxkernel, oder den Code des Mozilla-Projektes durchsuchen. [LXR] [LxrMozilla]

5.4.5 Bitkeeper

Statt nur ein Repository für das Versioning zu benutzen, können mit dieser Software viele Repositorys in einem hierarchischen Baum organisiert werden, so dass Entwicklungen in den Blättern erst zur Wurzel weiter gereicht werden, wenn sie einen stabilen Stand erreicht haben. Damit soll es einfacher sein, komplexe Projekte kompilierbar und stabil zu halten. Allerdings steht Bitkeeper unter einer recht merkwürdigen Lizenz. [Bitkeeper] Nachfolgend ein kleines Beispiel:



5.4.6 Issuezilla

Diese Software stellt die Umsetzung der Idee des Bugtracking auf Verbesserungsvorschläge (issue) dar. Es beruht auf der Software des Bugtrackingsystems Bugzilla. Ein eingegebener Vorschlag kann auf Webseiten bearbeitet, anderen Entwicklern zugewiesen werden, und verfolgt werden. [Issuezilla] [Bugzilla]

References

[Asynchrony] Die Homepage von Asynchrony, 7.1.2001, http://www.asynchrony.com

[BerliOS] BerliOS Site Documentation, 7.1.2001, http://developer.berlios.de/docs/site

[Bitkeeper] Die Homepage von Bitkeeper, 7.1.2001, http://www.bitkeeper.com

[Bravehack] Bravehack, 8.1.2001 http://unixpr.informatik.fh-dortmund.de/~dbadmin/bravehack/kapitel05.html

[Bugzilla] Die Homepage von Bugzilla, 7.1.2001, http://www.mozilla.org/bugs

[Cosource] Die Homepage von Cosource, 7.1.2001, http://www.cosource.com

[CVS] Die Homepage von CVS, 8.1.2001, http://www.cvshome.org

[deb] Debian Package Management System, HowTo welches die Grundlagen vermittelt, 10.1.2001, http://www.debian.de/doc/FAQ/ch-pkg_basics.html

[Debian] Die Hompage von Debian, einer Linuxdistribution für verschiedene Plattformen, 10.1.2001 http://www.debian.de

[dezUntStr] Der Pinguin als Vorbild dezentraler Unternehmensstrukturen, 8.1.2001, http://www.uni-jena.de/~w6arro/pinguin.html

[Freshmeat] Freshmeat, ein zentraler Softwarekatalog im Internet für OS-Software, 10.1.2001 http://www.freshmeat.net

[FSF] Free Software Foundation, 10.1.2001, http://www.gnu.org/fsf/fsf.html

[GETTEXT] Homepage des "Internationalization-Tools" gettext, welches von der FSF betreut wird,15.1.2001 http://www.gnu.org/manual/gettext/html_mono/gettext.html

[GIMP] Homepage eiens OS-Bildbearbeitungstools, ähnlich Photoshop, 10.1.2001 http://www.gimp.org

[GLADE] Die Homepage von GLADE, 8.1.2001, http://glade.pn.org

[GTK+] Die Homepage von GTK+, 8.1.2001, http://www.gtk.org

[GZIP] Die Dokumentation zu Gzip, 10.1.2001 http://www.gnu.org/manual/gzip/

[Hypermail] Die Homepage von Hypermail.pl, 8.1.2001, http://web.nwe.ufl.edu/~northrup/hypermail.html

[lkmlFAQ] Linux Kernel Mailinglist FAQ, 8.1.2001, http://www.tux.org/lkml

[IRC] Informationen über IRC, 8.1.2001, http://www.irc.org

[Issuezilla] Die Homepage von Issuezilla, 7.1.2001, http://www.openoffice.org/bugs/issues.html

[KBabel] Die Hompage von KBabel, welches eine Oberfläche für Gettext anbietet, 15.1.2001 http://i18n.kde.org/tools/kbabel/

[KDevelop] Die Homepage von KDevelop, 8.1.2001, http://www.kdevelop.org

[KDoc] Die Homepage von KDoc, 8.1.2001, http://www.ph.unimelb.edu.au/~ssk/kde/kdoc

[KMail] Die Homepage von Kmail, 8.1.2001, http://kmail.kde.org

[KTranslator] KTranslator bietet eine Oberfläche für Gettext, 15.1.2001, http://www.geocities.com/bilibao/

[LinuxNewbie] LinuxNewbie.org ist eine Website die sich auf Support für Einsteiger in Linux spezialisiert hat,10.1.2001 http://www.linuxnewbie.org

[LinuxISO] Die Website LinuxISO bietet zahlreiche Linuxdistributionen als ISOImage zum Download an, 10.1.2001 http://www.linuxiso.org/

[Leafnode] Die Homepage von Leafnode, 8.1.2001, http://www.leafnode.org

[LOB] Lehmanns Online, Internetpräsenz einer Buchhandlung, die OSS in Eigeninitiative in CD-Form anbietet,10.1.2001 http://www.lob.de

[LXR] Die Homepage von Cross-Referencing Linux, 7.1.2001, http://lxr.linux.no

[LxrMozilla] Mozilla Cross-Reference, 7.1.2001, http://lxr.mozilla.org

[LyX] Homepage des LyX-Projektes, einer grafischen Oberfläche für TeX,10.1.2001 http://www.lyx.org

[Majordomo] Die Homepage von Majordomo, 8.1.2001, http://www.greatcircle.com/majordomo

[mkisofs] Homepage von mkisofs, einem Tool zur Erstellung von ISO-Images zum brennen auf CD, 10.1.2001 http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/mkisofs.html

[RedHat] Die Homepage des amerikanischen Linux-Distributors Redhat,10.1.2001 www.redhat.com

[RPM] Die Homepage des RPM PackageManagers, 10.1.2001 http://www.rpm.org

[RPMFind] RPMFind.net ist die zentrale RPM-Datenbank im Internet vom W3C, http://rpmfind.net/linux/rpm2html/rpmfind.html

[Slackware] Die Homepage einer seit den Anfängen von Linux existierenden Distribution http.//www.slackware.com

[Sourcexchange] Die Homepage von SourceXchange, 7.1.2001, http://www.sourcexchange.com

[SGML] What you really need to know about SGML, 8.1.2001, http://people.debian.org/~bortz//SGML-HOWTO/potato/x59.html

[SourceForge] SourceForge Dokumentation, 7.1.2001, http://sourceforge.net/docs/site

[Suck] Die Homepage von suck, 8.1.2001, http://home.att.net/~bobyetman

[TAR] Die Dokumentation zu Tar, 1.1.2001, http://www.gnu.org/manual/tar/html_chapter/tar_toc.html

[Telepolis1] Open Source als Modell, 8.1.2001 http://www.heise.de/tp/deutsch/html/result.xhtml?url=/tp/deutsch/inhalt/te/4562/1.html&words=Kommunikation\%20Open\%20source

[UNICODE] Hompage des Unicode Consortiums, einer Organisation die eine Standarsierung eines erweiterten Zeichensatzes vorantreibt,15.1.2001 http://www.unicode.org/unicode/consortium/consort.html

[VALinux] Die Homepage von VA Linux, 7.1.2001, http://www.valinux.com

[WISE] Die Homepage des Web Integrated Software Enviroment, 7.1.2001, http://research.ivv.nasa.gov/projects/WISE

[WikiWiki] Die Homepage WikiWikiWeb, 7.1.2001, http://c2.com/cgi/wiki