Seminar „Open Source Software Engineering"
Sommersemester 2006

Usability von Open Source Software

Annette Kaudel
kaudel@inf.fu-berlin.de
Betreuer: Christopher Oezbek

16.7.2006

Zusammenfassung

Benutzerfreundlichkeit (engl: Usability) ist eine wichtige Anforderung an jede Software, die von Menschen bedient werden soll. In der kommerziellen Software Industrie wird mittlerweile auch verstärkt auf eine gute Usability geachtet. Bei Open Source Software hingegen findet diese Anforderung, bedingt durch die spezifischen Besonderheiten im Entwicklungsprozess, meist noch wenig Beachtung. Als Folge dessen verwenden der Großteil an Normal-Computer-Nutzern bis jetzt auch keine oder nur wenig Open Source Software.

Worin begründet sich diese geringe Beachtung bei OSS nun genau? Mehrere Ursachen dafür versucht dieser Text aufzudecken.

Inhaltsverzeichnis

1 Einleitung

1.1 Warum dieser Text

Usability ist ein wichtiges Qualitätskriterium an eine Software. Vor allem wenn die letztendlichen Nutzer des Programms selber keine erfahrenen Programmierer, sondern 'Otto-Normal Nutzer' sind, ist dieser Punkt von entscheidender Bedeutung, da die Software sonst für diesen Nutzer unter Umständen komplett unbrauchbar ist. Aus Nutzersicht ist Usability das wichtigste Bewertungskriterium an eine Software. [1]

Obwohl Open Source Software (OSS) mittlerweile starken Zuspruch erhält, so ist dieser doch meist einseitig geprägt. Die Großteil der Nutzer von OSS sind aktuell technisch erfahrene Anwender, zum Teil selber Programmierer. Wohingegen die meisten der technisch weniger erfahrenen Nutzer, die gleichzeitig die Mehrheit der heutigen Computer Anwender bilden, nur indirekt oder wenig bis gar keine OSS Software verwenden. [2] Ein Grund dafür wird der anscheinend schlechteren Usability vieler OSS zugeschrieben. [3]

1.2 Begriffsdefinition Usability

Der ursprünglich englische Begriff Usability leitet sich von 'user-friendly' ab und wird im deutschen meist mit 'Benutzerfreundlichkeit' übersetzt. Allerdings finden sich in der Literatur auch einige weitere Synonyme wie 'Benutzungsfreundlichkeit', 'Gebrauchstauglichkeit', 'Gebrauchsqualität', 'Anwenderfreundlichkeit', 'Ergonomie' u.a. Dies führt teilweise zu Verwirrung, da teilweise nicht alle Begriffe wirklich exakt das Gleiche bezeichnen. Aus diesem Grund und der Tatsache, dass mittlerweile auch im deutschsprachigen Raum der englische Begriff Usability starke Verwendung findet, wird auch in diesem Text weiter der Begriff Usability benutzt.

Usability ist ein Teilaspekt aus dem Wissensgebiet der Mensch-Computer-Interaktion (engl: Human-Computer Interaction, HCI). Ihr Ziel ist in den Software-Entwicklungsprozess Methoden zu integrieren, die ein optimales Arbeiten des späteren Anwenders mit dem Programm ermöglichen.

Laut Jacob Nielsen [4] setzt sich Usability aus den folgenden fünf wesentlichen Anforderungen zusammen.

Erlernbarkeit: (engl.: learnability)

Die Software sollte von Anfang an gut verständlich und der Umgang somit leicht erlernbar sein.

Einprägsamkeit: (engl.: memorability)

Der erlernte Umgang mit dem Programm sollte sich gut einprägen können, um nicht alle Schritte wieder neu erlernen zu müssen.

Effizienz: (engl.: efficiency)

Mit der Software sollte ein effizientes, hoch produktives Arbeiten möglich sein. Der nötige Aufwand zum bewältigen einer Aufgabe sollte so gering wie möglich gehalten werden.

Fehlerrate: (engl.: error rate)

Das System sollte auch bei Fehlern stabil bleiben und auch gegenüber fehlerhafter Eingaben tolerant arbeiten.

Zufriedenstellung: (engl.: satisfaction)

Letztendlich muss die Software den Nutzer zufriedenstellen, so dass die Arbeit mit dem Programm nicht zur Belastung wird.

Die Aufsplittung in diese Teilanforderungen erleichtert auch die Messbarkeit von Usability einer Software.

Andere Autoren [1, 5] sowie der Abschnitt 11 der ISO 9241 Norm kommen zu leicht abweichenden Definitionen und Begriffen, im großen und Ganzen decken sie sich aber mit den hier genannten.

2 Ursachen

Welche Ursachen führen nun dazu, dass OSS mehr Probleme mit Usability hat als proprietäre Software [3]?

Und inwiefern hängt dies mit dem speziellen Charakter und Entwicklungsprozess von OSS zusammen?

2.1 Programmierer sind selber die Nutzer

Der entscheidendste Faktor besteht darin, dass viele OSS von Programmierern in erster Linie für sich selbst geschrieben wird („Open Source developer-user" [6]). Sei es aus einem persönlichen Bedarf [7], Lernziel oder einfach Spaß an der Sache [8, 9]. Letztendlich steht im Gegensatz zu kommerzieller Software nicht unbedingt der Gedanke im Vordergrund einen möglichst großen Nutzerkreis ansprechen zu wollen und die Software an deren Bedürfnisse anpassen zu müssen [6].

> „The relation to usability is that this implies that OSS is in certain ironic ways more egotistical than closed source software." [6]

Aus diesem Blickwinkel betrachtet erscheint die Entwicklung von OSS eigennütziger als bei proprietärer Software.

Diese Punkte wirken sich auch auf die Art der Software aus, die bevorzugt als OSS bereitgestellt wird. So findet sich ein breites Angebot an Entwicklersoftware, wie beispielsweise Editoren oder Compiler. Diese sind, aufgrund ihres spezifischen Nutzerkreises, auch nicht so stark von den Usability Anforderungen für technisch weniger erfahrene Anwender betroffen. Denn hier kann sich der Programmierer auch gut in die Anforderungen des Nutzers hineindenken, da die Zielgruppe ebenfalls Programmierer sein werden [6].

Anders sieht es bei Software aus, die als Hauptzielgruppe den durchschnittlich erfahrenen Computer-Anwender ansprechen soll.

> „That is, while hackers can be very good at designing interfaces for other hackers, they tend to be poor at modeling the thought processes of the other 95% of the population well enough to write interfaces that J. Random End-User and his Aunt Tillie will pay to buy." [10]

So wie Raymond es hier auch beschreibt, stellt es für viele Programmierer oft ein Problem dar, sich in die Bedürfnisse eines weniger erfahrenen Anwenders hineinzuversetzen und die daraus resultierenden Usability Anforderungen umzusetzen [6].

Diese Probleme haben Entwickler von proprietärer Software genauso. Nur wird dort bei der Entwicklung in größeren Firmen, in der Regel extra darauf geachtet und spezielle Maßnahmen zur Lösung werden ergriffen. Bei OSS fehlen dazu oft die nötigen Mittel, sowie der Antrieb die Software an eine möglichst breite Masse von Anwender verkaufen zu wollen.

Gerade einer der Grundgedanken vom OSS Entwicklungsprozess, dass viele Menschen auch mehr Fehler entdecken können [7], funktioniert bei Usability nicht, da die OSS Community zum Großteil aus Programmierern besteht, die sich nur selten aus eigenem Interesse um Fehler im Sinne der Usability kümmern. [6]

> „The OSS approach fails for end user usability because there are 'the wrong kind of eyeballs' looking at, but failing to see, usability issues." [6]

2.2 Usability ist kein technisch anspruchsvolles Ziel

Zwei der oft entscheidenden Motivationsgründe für Programmierer von OSS sind ein Lerneffekt oder die Anerkennung innerhalb der Community [9, 8]. Dies gilt in beiden Fällen allerdings mehr für den technischen Aspekt.

Der Lerneffekt ist in der Regel eher mit dem Wunsch Erfahrung im Programmieren zu sammeln und weniger sich mit Usability zu beschäftigen gedacht.

Auch das Ansehen in der Community hängt eher von den technischen Leistungen als Programmierer ab, als sich als Usability Experte hervor zutun. Als Folge dessen schwindet auch das Interesse Zeit für diese Dinge zu investieren [6].

2.3 OSS Entwicklung noch relativ neu

Im Verhältnis zur Historie des gesamten Software Marktes ist OSS noch relativ jung. Usability war auch dort zu Anfang noch kein großes Thema. Erst mit dem Erreichen der breiteren Masse als Computer-Anwender, wurde dem Thema mehr Beachtung geschenkt. Daraus lässt sich der Schluß ziehen, dass OSS diesen Entwicklungsprozess erst noch nachholen muss [6].

2.4 Fehlende Usability Experten

Bei der Entwicklung großer kommerzieller Software-Produkte werden spezielle Usability Experten mit einbezogen. Diese können die Probleme der Programmiere, sich in den Anwender hineinzuversetzen, ausgleichen. Bei den meisten OSS Projekten sind aber keine finanziellen Mittel vorhanden, solche Experten extra zu engagieren. Diese müssten sich, genau wie die Programmierer, aus eigenem Antrieb an den Projekten beteiligen und ihr Wissen so einbringen.

Laut Nichols [6] gibt es mehrere Vermutungen woran dies bisher scheitert. Dazu zählen beispielsweise, dass auch außerhalb der OSS Community weniger Usability Experten als Programmierer existieren, sie sich in OSS Communities nicht willkommen fühlen, oder sie sich nicht so stark vom OSS Gedanken angesprochen fühlen und dafür interessieren.

2.5 Schwierige Spezifikation

Funktionale Anforderungen an ein Programm lassen sich grundsätzlich leichter und genauer spezifizieren als nichtfunktionale Anforderungen, zu den auch Usability gehört. Somit ist es für die Programmierer auch schwerer diese Anforderungen auch im Programmcode umzusetzen [6]. Dies ist kein spezielles Problem von OSS, allerdings wirkt es sich durch das hinzukommen der weiteren genannten Ursachen noch stärker aus.

2.6 Verteilte Community

Bei vielen OSS Projekten beteiligen sich Personen aus zahlreichen verschiedenen Ländern und Kontinenten. Dies kann sich einerseits positiv auswirken, wenn durch den

kulturübergreifenden Austausch völlig verschieden geprägte Personen ihre Ideen einbringen. Als Folge aus solchen Konstellationen entsteht in der Regel mehr Kreativität.

Allerdings ist für gute Usability auch Konsistenz innerhalb des Programms nötig. Diese bedarf einer gut abgesprochenen Zusammenarbeit, welche nicht unbedingt eine typische Arbeitsweise bei OSS Projekten ist. Jeder fügt einen Teil hinzu oder ändert ein Stück nach seinen Vorstellungen. Besonders problematisch wird dies beim Aufbau von Benutzeroberflächen und Menüstrukturen [6].

2.7 Features versus Einfachheit

Gerade der typische Entwicklungsprozess von OSS Projekten neigt in der Regel dazu viele Features der Software hinzuzufügen[6]. Dies wirkt der Einfachheit des Programms entgegen und kann sich somit negativ auf die Usability der Software auswirken[4].

2.8 indirekte Ursachen

Neben diesen Gründen gibt es auch noch ein paar weitere Gründe, warum OSS von den Normal-Computer-Nutzern bis jetzt nicht so stark bevorzugt wird, die nichts oder nur indirekt etwas mit Usability zu tun haben.

- **proprietäre Software Produkte haben sich bereits etabliert**

  In vielen Bereichen gibt es bereits proprietäre Software die schon einen großen Marktanteil erreicht hat. Dies hat auch zur Folge, dass sich die Nutzer dieser Produkte schon an deren Aufbau gewöhnt haben und diese aufgrund des Gewöhnungseffektes auch bevorzugen [6]. Ein Umstieg auf eine OSS würde ein neues Erlernen der Funktionen bedeuten, was sich somit auch indirekt negativ auf die empfundene Usability des neuen OSS Produktes auswirkt [4].

- **Nutzung vorhandener Daten**

  Als weitere Konsequenz aus der zuvor beschriebenen Etablierung vorhandener proprietärer Software ist auch die Nutzung der daraus entstandenen Daten zu bedenken. Besonders wenn diese in einem programmspezifischen Format gespeichert sind, ist ein Umstieg auf eine neue Software mit erhöhtem Aufwand verbunden. Um dieses Problem so weit wie möglich zu lösen, wird bei vieler OSS eine Konvertierung oder Import gängiger Dateiformate angeboten. Trotzdem verursacht auch dies einen gewissen Mehraufwand und wirkt sich somit auch negativ auf den Usability Aspekt der Effizienz aus.

- **fehlende Ressourcen**

Da die meisten OSS Projekte ohne Finanzierung durch einen Softwarekonzern existieren, fehlen auch die finanziellen Mittel nötige Ressourcen anzuschaffen. Dazu gehören im Fall der Usability Entwicklung zum einen Usability Experten, sowie spezielle Labore zur Durchführung von Usability Tests [6]. Dies ergeht kleineren und finanzschwächeren Herstellern von proprietärer Software allerdings nicht viel anders.

- **offener Quelltext ist kein Anreiz für Nichtprogrammierer**

  Die Offenlegung des Programmcodes ist nur ein Anreiz für die Gruppe der Programmierer, die damit die Option haben, das Programm ihren Bedürfnissen anzupassen. Für die meisten Nicht-Programmierer ist dieser Zusatznutzen nur von Belang, wenn sie wiederum Programmierer zur Verfügung haben, die diese Arbeit für sie leisten könnten. Offener Quelltext ist für diese Nutzergruppe also zumindest kein Vorteil gegenüber gleichwertiger proprietärer Software. Andere Faktoren wie Preis, Qualität und Support sind eher entscheidend [11].

3 Lösungsansätze

Nachdem die Ursachen erkannt sind, stellt sich die Frage, wie sich diese am besten beheben oder wenigstens relativieren lassen.

3.1 Unterstützung durch Firmen und Universitäten

In einigen Projekten wird bereits auf die Unterstützung durch große Software-Konzerne oder akademische Einrichtungen zurückgegriffen. Diese können zum einen ihre Ressourcen und Erfahrung einsetzen, um geziehlt zusammen mit der OSS Community ein Projekt noch erfolgreicher zu machen [6].

Beispiele hierfür bieten Projekte wie GNOME, Mozilla und OpenOffice [3].

3.2 Vereinfachte Rückmeldungen

Ein Problem besteht darin, dass die Nutzer-Rückmeldungen an die Entwickler bei OSS Projekten überwiegend von erfahrenen Anwendern geschehen. Das hängt vor allem damit zusammen, dass diese Rückmeldungen einen gewissen Aufwand und Know-How benötigen. So ist für das Melden eines 'Bugs' zuerst einmal die Kenntnis von den zugehörigen Tools wie beispielsweise Bugzilla[1] nötig. Darüber hinaus erfordert es dort auch noch eine Registrierung, bevor ein 'Bug-Report' gemeldet werden kann. Diese Hürden werden von vielen der durchschnittlichen Anwender nicht überwunden [6].

[1] www.bugzilla.org

Um dieses Problem zu lösen wird in einigen Software Projekten bereits auf automatisierte Tools gesetzt die beispielsweise Fehlermeldungen an die Entwickler zurücksenden. So z.B. die Fehlerberichterstattung bei Microsoft Windows XP.

Neben diesen Fehlerbenachrichtigungen könnten auch sogenannte „expectation agents" ein hilfreiches Feedback an die Programmierer geben [6]. Diese werden ausgelöst wenn ein Anwender eine nicht erwartete Anweisung an das Programm durchführt. Zum einen geben sie auch dem Anwender einen Hinweis über die erwartete Benutzung. Zum anderen wird eine Rückmeldung ähnlich der Fehlerberichterstattung an den Programmierer gesendet, welche optional auch noch durch Kommentare des Nutzers ergänzt werden kann.

Bei OSS kommt diesen Rückmeldungen eine bedeutendere Rolle zu, da gerade für den OSS Entwicklungsprozess die Rückmeldungen von vielen Nutzern entscheidend sind [7]. Und durch diese vereinfachte Form auch diejenigen mit einbezogen werden, die sonst nur passive User bleiben würden.

3.3 Automatisierte Usability Tests

Eine Schwierigkeit Usability in OSS Projekten zu überprüfen besteht darin, dass die nötigen Ressourcen für umfangreiche User-Befragungen, und Usability-Tests in nicht kommerziell finanzierten OSS Projekten fehlen. Dieses Problem könnte teilweise durch automatisierte Überprüfungen gelöst werden. Dazu existieren bereits ein paar Software Werkzeugs, die zumindest einige Usability Aspekte automatisiert nachkontrollieren können [6]. Dies hat allerdings seine Grenzen, da gerade Usability sich oft aufgrund von schwer spezifizierbaren Ursachen positiv oder negativ auswirkt.

3.4 Usability Experten einbeziehen

Funktionale Probleme lassen sich in der Regel gut mittels Werkzeugen wie Bugzilla melden und beschreiben. Bei Usability Problemen ist dies schwieriger. Daher würde es helfen eine auf diese Probleme abgestimmte Plattform zu errichten [6]. 'OpenUsability'[2] ist dazu ein erster Ansatz. Hier können Entwickler ihr OSS Projekt anmelden und gezielte Anmerkungen von Usability Experten erhalten.

3.5 Nutzen von Usability bekannt machen

Ein entscheidender Lösungsansatz ist es auch der OSS Community den Nutzen von Usability für ihre Software deutlich zu machen und Interesse für deren Umsetzung zu wecken.

[2] www.openusability.org

Ein möglichst großer Nutzerkreis ist anders als bei kommerzieller Software zwar kein finanziell entscheidender Faktor, aber die meisten Programmierer wünschen sich auch bei OSS Projekten, dass ihre Software möglichst viele Anwender findet. Zumal OSS auch nicht immer gleichbedeutend mit 'kostenlos' ist [12].

4 Fazit

Der OSS Entwicklungsprozess hat sich vielfach als ein leistungsfähiger Weg herausgestellt hochwertige Software hervorzubringen. Einige davon zeichnen sich auch durch eine sehr gute Usability aus, dennoch gibt es aktuell viele Projekte bei denen es noch Schwierigkeiten im Sinne der Usability zu geben scheint. Diese Probleme in Zukunft besser zu lösen ist eine weitere Aufgabe, die von der OSS Community angegangen werden wird.

Als weiterer Aspekt hängt auch die Barrierefreiheit teilweise mit den hier genannten Punkten zusammen.

Es würde sich die Chance bieten gerade die Besonderheiten im OSS Entwicklungsprozess auch für die Umsetzung von Usability zu nutzen.

Literatur

[1] H. Balzert, *Lehrbuch der Software Technik*, 2nd ed. Spektrum Akademischer Verlag, 2000, vol. Software-Entwicklung.

[2] D. M. Nichols, K. Thomson, and S. A. Yeates, "Usability and open-source software development," in *Symposium on Computer Human Interaction*, E. Kemp, C. Phillips, Kinshuk, and J. Haynes, Eds., 2001. [Online]. Available: http://www.comp.lancs.ac.uk/computing/users/dmn/docs/oss.pdf

[3] C. Benson, M. Müller-Prove, and J. Mzourek, "Professional Usability in Open Source Projects: GNOME, OpenOffice.org, NetBeans," in *Conference on Human Factors in Computing Systems (CHI '04)*, Vienna, Austria, 2004, pp. 1083 – 1084. [Online]. Available: http://mprove.de/script/04/chi/hc10-benson.pdf

[4] J. Nielsen, *Usability Engineering*. Academic Press, 1993, ISBN 0-12-518406-9.

[5] I. Sommerville, *Software Engineering*, 6th ed. Addison-Wesley, 2001, ISBN 3-8273-7001-9.

[6] D. M. Nichols and M. B. Twidale, "The Usability of Open Source Software," *First Monday*, vol. 8, no. 1, Jan. 2003. [Online]. Available: http://firstmonday.org/issues/issue8_1/nichols/

[7] E. S. Raymond, "The cathedral and the bazaar," *First Monday*, 1998. [Online]. Available: http://www.firstmonday.org/issues/issue3_3/raymond/

[8] B. Luthiger, "Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern," in *Open Source Jahrbuch 2004*. Lehmanns Media, 2004. [Online]. Available: http://www.opensourcejahrbuch.de/2004/pdfs/II-2-Luthiger.pdf

[9] A. Hars and S. Ou, "Working for free? – Motivations of participating in Open Source Projects," in *The 34th Hawaii International Conference on System Sciences*, 2001. [Online]. Available: http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf

[10] E. S. Raymond, "The Revenge of the Hackers," in *Open Sources: Voices from the Open Source Revolution*, 1999. [Online]. Available: http://www.oreilly.com/catalog/opensources/book/raymond2.html

[11] J. Feller and B. Fitzgerald, "A Framework Analysis of the Open Source Software Development Paradigm," in *The 21st International Conference in Information Systems (ICIS 2000)*, 2000, pp. 58–69. [Online]. Available: http://www.josephfeller.com/publications/ICIS2000FellerFitzgerald.pdf

[12] R. M. Stallman, *Free Software, Free Society: Selected Essays of Richard M.Stallman*. GNU Press - Free Software Foundation, 2002, ch. Chapter 1 - The GNU Project, pp. 15 – 31. [Online]. Available: http://gnu.paracoda.com/doc/book13.html