

## Open-Source Software SS 2005

**Thema:** Kritik an der Open-Source Software.  
Bezug nehmend auf den Artikel von Michelle Levesque

**Datum:** 27.06.2005

**Name:** Stefan Merkle

Fachhochschule  
Augsburg



University of  
Applied Sciences

Wenn man nach der Kanadischen Forscherin Michelle Levesque geht wird ein Großteil der Computernutzer in Zukunft auf Open-Source Software verzichten, wenn sich in einigen Bereichen nicht grundlegend etwas ändert. Dabei spricht Sie fünf Themen an welche wahrgenommen und gelöst werden müssen. Dies seien die Benutzeroberfläche, die Dokumentation, funktionszentrierte Entwicklung, selbstbezogene Programmierung und "religiöse Verblendung".

Michelle Levesque bezeichnet sich selbst nicht als Expertin aber als „gut informiert“, da Sie schon an zahlreichen kleineren Projekten eine wichtige Rolle gespielt hat.

Sie hat sich an einem bestimmten, aber nicht näher bekannten, Open-Source Projekt beteiligt (im folgenden Text Projekt X genannt) und geht dabei zum Beispiel auf dessen Benutzeroberfläche ein. Sie bezeichnet die Benutzeroberfläche als Albtraum. Sie sei auf gar keinen Fall intuitiv zu bedienen. So nennt Sie zum vergleich folgenden Aspekt. Sie hätte schon Nutzer mit wenig Computererfahrung gesehen die sich innerhalb von Minuten auf der Oberfläche von Mac OS X zurechtgefunden haben, während die gleichen Personen an KDE oder GNOME verzweifelt seien. Sie hätten sich darüber gewundert, warum sie sich mit einer derart "klobigen" Oberfläche beschäftigen sollen, wenn Windows XP wesentlich besser funktioniere.

Dazu noch ein paar weitere Erkenntnisse.

Michelle Levesque: „Ich nehme an, es gibt nicht einen einzigen Grund für die geringe Qualität der Benutzeroberflächen, aber hier kommen einige Erklärungen, die ich Open Source Kreisen umgehen gehört habe. Geeks bevorzugen es so, Funktionalität vor Schönheit, die Kluft zwischen den Geschlechtern in der Open Source Gemeinschaft, es ist für die Programmierer intuitiv – warum sollten sie es also ändern? Der Glaube ein schönes Design der Benutzeroberfläche könne auf später verschoben werden, wenn die hauptsächliche (wichtigere) Arbeit getan ist, der Glaube, das eine schöne benutzerfreundlich und intuitiv gestaltete Oberfläche keine wirkliche Arbeit ist, und einiges Andere. Viele dieser Begründungen klingen erst einmal plausible, und ich nehme an, dass der Grund für das verschmähen schöner Oberflächen eine Kombination mehrerer dieser Gründe ist. Wie auch immer, wenn die Open Source Gemeinde gedeihen möchte, und ihr daran gelegen ist, das möglichst viele Menschen ihre Programme nutzen, ist es eine wichtige Voraussetzung, das sie sich klarmachen, dass die Mehrheit der Benutzer nicht merkt, was für ein besonders cleverer Synchronisations-Algorithmus ihnen gerade das simultane editieren mehrerer Teile ihrer komplexen Datenstruktur ermöglicht. Was der Benutzer wirklich sieht, und wonach er das Projekt beurteilt, ist die Oberfläche. Es ist nicht konkurrenzfähig. Nur Geeks werden dieses Programm jemals nutzen.“

Kaum ein gutes Haar lässt Levesque an die Dokumentation, die nur für Spezialisten verständlich sei. Dabei müsse ein Grundsatz sein, Dokumentationen sollten so geschrieben sein, dass auch Nutzer mit wenig Vorwissen bei Problemen weiterkommen. Open-Source-Programmierer seien anscheinend öfter der Meinung, wenn sie mit ihrem Produkt klarkommen, müsse dies auch für alle anderen Nutzer gelten. Dabei erstellen sie aber Programme für Programmierer.

Michelle Levesque: „Projekt X hat eine ziemlich ausführliche Dokumentation, was in der Open Source Bewegung nicht immer der Fall ist. Unglücklicherweise ist diese Dokumentation für eine antike Version von Projekt X geschrieben, und man hat nur einfach die neueste Versionsnummer auf den Titel geschrieben. Außerdem geht diese Dokumentation davon aus, dass dein System exakt wie das des Schreibers konfiguriert ist. Zusätzlich muss man fortgeschrittene Kenntnisse der Administration besitzen, und wissen, wie man mit der Programmiersprache umgeht, in der die Dokumentation verfasst ist, weil Fehler auftauchen, die in der Dokumentation nicht berücksichtigt werden, und man wissen muss, wie man diese debuggen kann.

Open-Source Projekte scheinen größere Probleme damit zu haben, eine vernünftige Dokumentation anzubieten, wenn sie denn überhaupt eine anbieten. Weil niemand sie vertraglich zwingt, diese Dokumentation mitzuliefern, gibt es eher einen allgemein gehaltenen Rundgang durch das Programm, als ein komplettes Handbuch, das man einem Neuling in die Hand drücken könnte. Stell Dir vor, was folgender Satz für jemand bedeutet, der sehr geringe Unix-Kenntnisse besitzt, und dieses Programm zum ersten Mal auf seinem System installiert. "You will need a list of MD5 checksums for the binary files. If you have the md5sum program, you can ensure that your files are not corrupt by running md5sum -v -c md5sum.txt."

Die häufigste Antwort auf eine Beschwerde darüber ist, wenn sie das nicht verstehen sollten sie dieses Programm auch noch nicht installieren, aber wenn das so ist, wie sollen diese Leute lernen mit dem Programm umzugehen? Dokumentation sollte sich immer an den Bedürfnissen des Schwächsten Users orientieren. Übrigens, jeder der schon einmal einen Fehler in einem Open-Source Programm debuggen musste, weiß, dass die Antworten nach dem „wie“ niemals in der Dokumentation zu finden sind, man findet sie in Usenet Artikeln, auf Bulletin-Seiten und in Chat-Logs. Ein Benutzer, der nicht weiß, wie er ein Problem lösen soll wendet sich an ‚alt.projectx.devel‘ und stellt dort die Frage die schon hunderte Benutzer vor ihm dort gestellt haben. Irgendein Experte hat Mitleid mit den Nöten der Fragesteller, und antwortet, aber er wird diese Antwort niemals dokumentieren. So wiederholt sich der Prozess mit jedem neuen Fragesteller der auftaucht. Hinzu kommt die irriige Annahme, dass Computer-Neulinge in der Lage wären diese alternativen Strömungen der Kommunikation überhaupt zu finden, oder den Ehrgeiz bzw. die Liebe zu diesem Programm zu entwickeln, da sie sich ausgiebig darum bemühen. Ohne brauchbare Dokumentation sind Open-Source Projekte von Natur aus im Nachteil.“

Eine weitere Schwachstelle von vielen Open-Source Projekten ist die funktionszentrierte Entwicklung. Die meisten Projekte bestehen aus einer Vielzahl von einzelnen Tools. Vom Prinzip her ist ein guter Ansichtspunkt, dass ein Tool das speziell für eine Aufgabe entwickelt wurde und ein Interface besitzt welches für diese Aufgabe konzipiert wurde, seine Aufgabe auch am besten erfüllt. Wenn man dann diese ganzen Tools kombiniert, hat man ein Projekt mit dem man viel anfangen kann. Nun versuchen allerdings viele Entwickler Punkte die sie persönlich stören oder als verbesserungswürdig halten zu überarbeiten. So könnte z.B. ein Entwickler der Meinung sein, dass eine 2D-Grafik nicht mehr Zeitgemäß ist, um mit vollem

Ehrgeiz eine 3D-Grafik zu entwickeln. Dabei geraten Kernpunkte wie Stabilität, Sicherheit, Programmierstandards und Benutzeroberflächen ins Hintertreffen. So versucht jeder nur das zu entwickeln, was ihm „Spaß macht“.

Kommen wir nun zum Kritikpunkt der selbstbezogenen Programmierung.

Jeder Entwickler programmiert Anwendungen, von denen er selber denkt, dass sie notwendig sind. Er entwirft sie so wie er sie anwenden würde. Dabei ist in den meisten Open-Source-Projekten gegenüber kommerziellen Projekten das Problem, dass die Zielgruppe nicht genau definiert ist. In kommerziellen Produkten weiß man, wem man sein Produkt verkaufen möchte. In Open-Source-Projekten ist das nur sehr unscharf festgelegt. Man weiß nicht genau, was der Benutzer letztendlich mit dem Programm machen will und ob er es intuitiv genau so machen würde. Entwickler untereinander werden auf Grund gleicher Denkansätze Anwendungen anders als intuitiv betrachten, ohne dabei darauf zu achten, dass sie eben selbst Entwickler sind. Dazu muss erwähnt werden, dass in kommerziellen Projekten fast immer Usability-Experten anwesend sind, welche die Entwickler auf die Schwachstellen ihrer Anwendung rechtzeitig aufmerksam machen können.

Das Ergebnis ist, dass Open-Source-Projekte von Entwicklern für Entwickler sind, welche es nicht verstehen würden, warum die restlichen User nicht alle ihr Programm verwenden, da es doch das Beste ist, um das spezifische Problem oder die Aufgabe zu lösen.

Während der Rest der Welt Open-Source-Software mit Programmen verbindet, welche nur für die technisch versierte Oberliga der Gesellschaft ist.

So ist es speziell bei Open-Source-Programmen wichtig, dass vor Beginn der Entwicklung festgelegt wird, für welche Zielgruppe entwickelt werden soll. Und auch immer sichergestellt wird, dass die Vorgaben eingehalten werden.

Selbstbezogene Programmierung ist eine Falle, in die man sehr schnell treten kann und sollte auf alle Fälle vermieden werden.

Eine Voraussetzung, damit die Open-Source-Entwickler hinzulernen, sei, dass sie ihre "religiöse Verblendung" verlieren. Viele seien so von dem Prinzip der offenen Quellen überzeugt, dass sie nicht von den Vorzügen proprietärer Software lernen können oder wollen. Dabei gebe es auch unter Windows-Programmen welche, die vergleichbarer Unix-basierter Software überlegen sei.

Bei vielen geht es nach dem Prinzip: Wenn es sich um proprietäre Software handelt, dann kann man von ihr nichts lernen bzw. dann ist nichts davon auf Open-Source-Software anwendbar.

Details zu Michelle Levesque:

Forscherin im [Citizen Lab](#) am Munk Centre for International Studies in Kanada.

## Zusammenfassung

- Benutzeroberfläche
  - Im Vergleich zu anderen Oberflächen nicht intuitiv zu benutzen
  - Zu wenig Schwerpunkte auf die Bedienungsfreundlichkeit
  - Funktion kommt vor Usability
  - Funktion kommt vor Design
- Dokumentation
  - Dokumentation wird nicht bzw. nur ungenügend an neue Softwareversionen angepasst.
  - Oftmals wird nur ein grober Überblick des Projektes dargestellt. Für einen User der keine oder wenig Erfahrungen in der Entwicklung hat ist dies zu wenig.
  - Unzureichende Dokumentation führt dazu, dass Meinungen von Experten eingeholt werden müssen. Dies wiederum ist das Los jedes einzelnen Users der an der Dokumentation scheitert.
  - Dokumentationen sind Prinzipiell sehr technisch orientiert.
  - Oftmals fehlen Dokumentationen aufgrund der fehlenden Pflicht diese anzulegen und zu Pflegen.
- Funktionszentrierte Entwicklung
  - Augenmerk wird auf „Spielereien“ gelegt.
  - Entwickler kümmern sich um Dinge die Ihnen „Spaß“ machen.
  - Wichtige Grundsätze (Programmierregeln, Sicherheit, Projektziele) werden vernachlässigt oder sehr pauschal gehandhabt. Bzw. nicht untereinander abgeglichen.
  - Jeder macht „sein Tool“ und möchte, dass es möglichst perfekt ist
  - Wenig Interesse bekannte Probleme ganz aus der Welt zu schaffen. Meistens werden nur Patches zur Verfügung gestellt
- Selbstbezogene Programmierung
  - Programm von Entwicklern für Entwickler.
  - Zielgruppe muss festgelegt und Programm dafür abgestimmt werden.
  - Falscher Eindruck kann zu Open-Source Projekten entstehen
  - Ein Fehler den man oftmals nicht sofort selber merkt aber sehr schwerwiegende Auswirkungen aufweist.
- „religiöse Verblendung“
  - Strikte Orientierung an Open-Source Gedanken.
  - Keinerlei Offenheit für Ideen und Gedanken die hinter prätentierter Software stecken.

## Meinungen zum Thema:

- Benutzeroberfläche
  - Unter Linux kann jeder die Oberfläche nutzen, die er bevorzugt. Auch ein Windows XP Stil kann verwendet werden. Es gibt mehrere Designs die nur installiert werden müssen.
  - Einige stimmen auch der Meinung von Michelle Levesque zu und betiteln einige Benutzeroberflächen als überladen und schlecht verwendbar.
- Dokumentation
  - Es gibt auch Dokumentationen die speziell für Neulinge geschrieben sind. Allgemein gesagt gibt es für verschieden Zielgruppen Dokumentationen.
  - Bei kommerziellen Anwendungen wie Microsoft Windows® existiert auch keine umfangreiche Dokumentation. Aber eine Hilfe bzw. Anleitung.
- Funktionszentrierte Entwicklung
  - Die meisten Kommentare sind im Punkt „Selbstbezogene Programmierung“ aufgeführt da Sie für beide Bereiche gelten.
- Selbstbezogene Programmierung
  - Zahlreiche Meinungen stimmen der Problematik der technischen Orientierung zu und gehen sogar einen Schritt weiter. Allein durch Begriffe Sachverhalte und systemspezifische Architekturen ist es einem normalen User nicht möglich ein System Problemlos zu konfigurieren, da viele Dinge als selbstverständlich angesehen und sogar in Hilfen deswegen nicht erläutert bzw. genauer erklärt werden. Ohne Grundlegende Kenntnisse ist ein wandeln in der Unix/Linux Welt nicht machbar. Vorbild: Eingebaute aber auch nicht immer perfekte Hilfestrukturen in Windows® die auch der Leihe versteht; dafür oft für Experten mit überflüssigen Informationen gefüllt und schlecht nutzbar.
  - Keinerlei Bezug zu Methoden wie UML, OOA, OOD oder ähnlichem
- „religiöse Verblendung“
  - Alles was nicht Open-Source ist, ist kommerziell und verboten zu benutzen.
  - Wie soll man von patentierter Software lernen?
  - „religiöse Verblendung“ ist wirklich verbreitet.
- Weitere Kritikpunkte
  - Die Open-Source Szene kann nur schlecht mit Kritik umgehen. Dagegen kann man z.B. Microsoft Windows® so schlecht machen wie man möchte
  - Es benötigt viel mehr Zeit und Einarbeitung um ein Open-Source Programm zu verwenden (verstehen => kompilieren => installieren)
  - Für Computerneulinge ist der Einstick in die Linuxwelt sehr schwierig.

Allein bei [www.heise.de](http://www.heise.de) gibt es über 1800 Beiträge zu diesem Thema.

## Quellen:

<http://www.heise.de/newsticker/meldung/46544>

<http://www.citizenlab.org/opensource/>