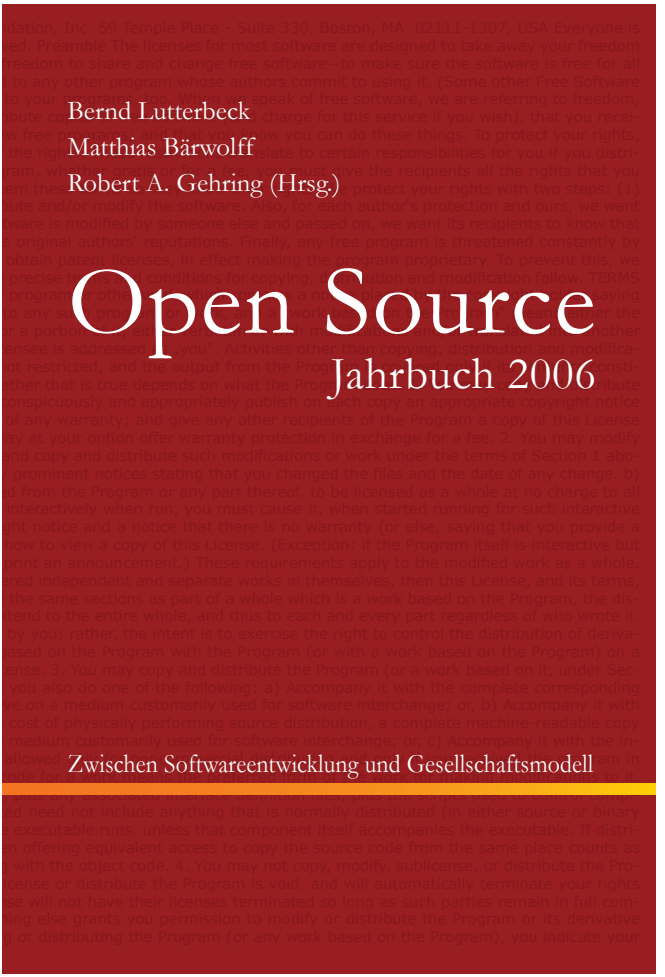
Dieser Artikel ist Teil des

Open Source Jahrbuchs 2006



erhältlich unter www.opensourcejahrbuch.de.

Die komplette Ausgabe enthält viele weitere interessante Artikel. Lob und Kritik zu diesem Artikel sowie weitere Anregungen können Sie uns einfach und unkompliziert mitteilen per E-Mail oder auf www.opensourcejahrbuch.de/feedback/.

Einleitung

FABIAN FLÄGEL

Dieses Kapitel behandelt mehrere Aspekte der Softwareentwicklung, die bei Open–Source-Software (OSS) auftreten. Es legt dar, aus welchen Strukturen die Communitys um Open Source bestehen und enthält Hinweise und Analysen zur Weiterentwicklung und Verbesserung.

Matthias Fink und Wolf-Gideon Bleek räumen im ersten Artikel dieses Kapitels einige in der Literatur vorhandene Mythen und Missverständnisse aus der Welt, die sich teilweise noch recht hartnäckig halten. Wenn manch einer an Open-Source-Programmierung denkt, so hat er vielleicht Gruppen von 18-jährigen, pickeligen *Nerds*[1] vor seinem inneren Auge, die sich als *Jedi* im Kampf gegen die dunkle Seite der Softwareentwicklung sehen, ausgezogen, um die Welt zu verbessern. Wer das glaubt, dem sei der Artikel „Mythen, Märchen, Missverständnisse" ans Herz gelegt. Der Aufbau von Open-Source-Projekten ist im Allgemeinen wesentlich strukturierter und professioneller als gemeinhin angenommen.

Matthias Stürmer und Thomas Myrach liefern im darauf folgenden Beitrag eine empirische Betrachtung von acht bekannten und erfolgreichen Projekten. Im besonderen Fokus ihrer Analyse befinden sich Aspekte des Aufbaus und der Organisation von Open-Source-Communitys. Eine der zentralen Prämissen ihrer Arbeit ist, dass sich Communitys *nicht* von selbst bilden, strukturieren und erhalten bleiben. Sie bestehen aus individuellen Mitarbeitern und Firmen, deren Arbeit geplant und organisiert werden will. Aus den Ergebnissen ihrer Untersuchung haben die Autoren eine Reihe von Erkenntnissen gewonnen, wie man eine Community bilden, halten und entwickeln kann, welche Veränderungen sich ergeben, wenn eine Firma sich hinter ein Projekt stellt und welche Auswirkungen das in der Projektplanung und -entwicklung hat.

Im dritten Artikel dieses Kapitels beleuchtet Patricia Jung einige Bereiche der Softwareentwicklung, an die man seltener denkt, die aber dennoch wichtig sind und

1 Nerd (engl. für Fachidiot, Langweiler) ist eine meistens abwertend gebrauchte Bezeichnung für Personen, die sich besonders mit Computern oder anderen Bereichen aus Wissenschaft und Technik beschäftigen, deren soziale Kompetenzen aber entweder schwach ausgeprägt sind oder diesen Eindruck zumindest erwecken.

substanzielles Potential für Verbesserungen beinhalten. Ihre Ausführungen legen nahe, die Prozesse der Softwareentwicklung weg vom reinen „Coden" hin zu einer modernen Form der Softwareentwicklung zu verändern. Ihr Blick richtet sich besonders auf weniger beachtete Aspekte im Bereich Open Source: vergleichsweise geringer Frauenanteil, strukturelle Vernachlässigung von *Usability* und Dokumentationsarbeiten. Viele Projekte entwickeln sich nur langsam, weil Programmierer, bedingt durch ihre zentrale und bestimmende Rolle in vielen Projekten, bestimmte Abläufe schlicht unterbinden oder aufhalten. Oft werden Personen, die sich mit Arbeiten etwa in den Bereichen Benutzeroberflächen, Übersetzungen oder Dokumentationen befassen wollen, wenig motiviert und teilweise sogar abgeschreckt. Darunter leidet zwangsläufig die Qualität vieler Projekte.

Des Weiteren widmet sie sich ausführlich dem geringen Frauenanteil in vielen OSS–Projekten. Dabei wird klar, dass viele Bereiche, in denen Frauen ein größeres Geschick und damit auch eine größere Affinität für den Einstieg haben, gerade die Bereiche sind, die gerne von reinen Programmierern vernachlässigt werden. Die Bedeutung von Dokumentation und Tutorialerstellung wird auch von Stürmer und Myrach betont.

Jan Tobias Mühlberg präsentiert im folgenden Beitrag „Software-Engineering und Software-Qualität in Open-Source-Projekten" die vorläufigen Ergebnisse aus der Studie „Innovationsverhalten deutscher Software-Entwicklungsunternehmen"[2], an der auch die FH Brandenburg mitarbeitet. Er vergleicht die Entwicklungsprozesse von OSS und proprietärer Software, wobei er Unterschiede und Gemeinsamkeiten herausarbeitet. Strukturbedingte und gravierende Unterschiede zu kommerziell geführten Projekten sieht er bei der Wartbarkeit von OSS, auf die ab einer bestimmten Projektgröße schon in der Planungsphase geachtet werden muss. Andernfalls muss die Software früher oder später von Grund auf neu geschrieben werden.[3] Mühlberg sieht dieses Problem akut bei einigen großen Projekten und warnt vor der Gefahr eines Projektstillstands durch nicht erweiterbaren Code.

Abgeschlossen wird das Kapitel durch einen Artikel von Eben Moglen, einem der Urväter der GPL v2. Seit er 1993 für Richard Stallman zu arbeiten anfing, tauschten sie Gedanken und Ideen zur GPL v3 aus. Moglen beschreibt seine Sicht auf die Entwicklung der neuen Lizenz. Insbesondere betont er die Wichtigkeit eines demokratischen Prozesses, bei dem die Interessen von Nutzern und Firmen, Konsumenten und Produzenten berücksichtigt werden. Dies ist etwas Besonderes, da die meisten anderen Lizenzen auf eine bestimmte Klientel abzielen. Dieser Entwurf richtet sich an die Community und somit an *alle* – die Lizenz ist demzufolge eine *demokratische* Lizenz.

2 INNODES, siehe http://innodes.fh-brandenburg.de.

3 Als Beispiel sei das Mozilla-Projekt genannt, welches aus dem Netscape-Code entstand. Die Entwickler begutachteten den Code und entschieden dann aufgrund zu großer Probleme mit der Code-Basis, ein komplett neues Programm zu schreiben. Siehe hierzu auch den Artikel „Der Beitrag freier Software zur Software-Evolution" von Andreas Bauer und Markus Pizka im Open Source Jahrbuch 2005.