

„Open Source“ und die Zukunft der Software

Albert Endres

Es gibt kaum ein Thema, das zz. die Softwarewelt mehr bewegt als Open Source (zu Deutsch: offener Quellcode). Meist werden dabei vier verschiedene Aspekte miteinander verknüpft.

- Etwas vereinfacht lassen sich die dabei eingenommenen Positionen wie folgt beschreiben:
- Es wird erstens dafür geworben, Software nicht in industriellen Organisationen oder nach einem „bürokratischen“ Prozess zu entwickeln, sondern in spontaner Kooperation von Interessenten (manchmal auch von Enthusiasten), die keinerlei geschäftliche Beziehungen oder andere gegenseitige Verpflichtungen haben;
 - zweitens wird Nutzern nahe gelegt, Software nicht nur in Objektform zu erwerben, sondern auf dem Quellcode zu bestehen;
 - drittens solle man Software möglichst verschenken und nicht verkaufen;
 - und viertens wird dafür plädiert, Programme nur als sog. „freie“ Software weiterzugeben, d. h., ohne Urheberrechts- oder Patentschutz in Anspruch zu nehmen.

Nicht immer kommen alle vier Aspekte gleich stark zur Geltung. Die bekanntesten Ausprägungen dieses Ansatzes sind zz. Linux und Apache. Ausführliche Informationen zu beiden Projekten findet man im Internet. (Die betreffenden URLs sind <http://www.linux.org> bzw. <http://www.apache.org>.)

Vor kurzem schrieb Glass [1], ein amerikanischer Beobachter der Softwareszene, dass er zwei Dinge bezüglich der Open-Source-Bewegung nicht verstehe, nämlich dass Programmierer neuerdings gerne anderer Leute Code lesen und dass sie gerne, ohne Geld dafür zu bekommen, für andere Leute arbeiten.

Ich habe noch einige andere Verständnisprobleme mit dem neuen Kult. Wenn man hinter die

Maskottchen und die knalligen Sprüche schaut, werden nämlich Fragen aufgeworfen, die Grundannahmen des Fachgebiets und der Branche betreffen. Im Folgenden will ich meine Fragen in mehreren Gruppen zusammenfassen, je nach dem möglichen Blickwinkel oder der Interessenlage. Am Anfang jedes Abschnitts stehen einige Vorbemerkungen, die meine Fragen begründen sollen. Das Gegenteil von Open-Source-Software werde ich etwas vereinfacht als „kommerzielle Software“ bezeichnen.

Sicht der Wissenschaft

Seit es Informatik als Wissenschaft gibt, gibt es Bestrebungen, die Eigenschaften eines Programms, also seine Semantik, anders zu beschreiben als durch Auflisten des entsprechenden Quellcodes. Neben der operationellen Semantik, die sich eng an die Implementierung hält, stellen funktionale und axiomatische Semantik stärkere Abstraktionen dar. Selbst kleine Programme vollständig zu testen ist sehr mühsam und bei den meisten Entwicklern nicht besonders beliebt; von großen Programmen ganz zu schweigen. Wie Reif [3] gezeigt hat, ist eine formale Verifikation realistischer Programme heute möglich, setzt aber Expertise und Werkzeuge voraus, über die viele Entwickler nicht verfügen. Steht der Quellcode eines Programmes im Netz zur Verfügung – so wird manchmal argumentiert –, ließe sich seine Qualität auch ohne eigenes Dazutun steigern. Wenn viele Augen den Code lesen, wird schon irgendjemand sich melden, wenn ihm etwas auffällt, was nicht

in Ordnung ist. An die

Albert Endres
Iselerstr. 1, D-71067 Sindelfingen
e-mail: a.endres@computer.org

Stelle der Bemühungen eines Urhebers einschließlich seiner Testorganisation trete ein sozialer Prozess, der am Ende das gleiche oder gar ein noch besseres Ergebnis hätte.

Ist der Quellcode wirklich so unverzichtbar? Kann es sein, dass wir den Wert der Bemühungen der Wissenschaft, die Eigenschaften von Programmen ohne Rückgriff auf den Quellcode zu definieren, überschätzt haben oder ihr Erfolg bisher gering war? Welche Kriterien der Qualitätssicherung kann ein sozialer Prozess besser erfüllen als Anstrengungen der Entwickler? Sollte man Test- oder Verifikationsverfahren eher zurückhaltend bewerten, da man ja ohnehin eine hundertprozentige Fehlerfreiheit nicht garantieren kann? Sollte man Leuten, die Quellcode nicht lesen können oder wollen, abraten, ein für sie kritisches Programm zu verwenden? Kann es sein – da insbesondere viele Informatik Studierende sich der Open-Source-Bewegung verschrieben haben –, dass in der Lehre Spezifikations- und Testmethoden nicht überzeugend genug vermittelt werden?

Interessenlage einzelner Anwenderfirmen

Nach meiner Erfahrung kommt es bei Entscheidungen für ein Softwareprodukt nicht nur darauf an, dass eine bestimmte Funktionalität gut angeboten wird, es muss auch eine gewisse Schnittstellenstabilität gewährleistet sein, auf die man sich verlassen kann. Das heißt, man muss die Gewähr oder aber zumindest die Überzeugung haben, dass Investitionen, die man selbst tätigt, nicht auf Sand gebaut sind. Diese Investitionen können darin bestehen, dass man selbst Entwicklungen betreibt, die auf diesen Schnittstellen aufbauen. Vielleicht bildet man Mitarbeiter aus (oder lässt sie sich selbst ausbilden), die diese Schnittstellen benutzen.

Eine Softwareschnittstelle, sei es eine Nutzer- oder Programmierumgebung, hatte in der Vergangenheit immer dann Chancen längere Zeit im Markt zu bestehen, wenn sie entweder von verschiedenen Herstellern akzeptiert war oder von einem Marktführer verbreitet wurde, der entsprechende Perspektiven bot. Eine Schnittstelle lebt so lange, wie es Produkte gibt, die diese Schnittstelle unterstützen. Das Schicksal von Produkten hängt davon ab, wie es dem Hersteller oder Lieferanten ergeht, der hinter einem Produkt steht. Viele Soft-

warefirmen legen ihren Kunden gegenüber ihre Entwicklungspläne offen oder machen zumindest Ankündigungen darüber, in welche Richtung ihre zukünftigen Entwicklungen gehen.

Ist der Ruf der Softwarebranche wirklich so schlecht, dass man sich auch bei Standardprodukten lieber auf Graswurzelaktivitäten als auf industrielle Partner verlässt? Sind die Anschaffungspreise für kommerzielle Programme vielleicht unverhältnismäßig hoch im Vergleich zu den Kosten für Geräte, Installation und Betrieb? Ist bei Schnittstellen, die von Open-Source-Software propagiert werden, ein größerer Verbreitungsgrad und eine höhere Lebensdauer zu erwarten als bei den üblichen kommerziellen Programmen? Wenn selbst ein PC-Betriebssystem etwa 20 Millionen Instruktionen umfasst, ist es dann wirklich sinnvoll, zusätzlich zum Objektcode auch den Quellcode zu verwalten und mit dem Objektcode konsistent zu halten? Dasselbe dann bei allen anderen Programmen, die man braucht, die vielleicht noch umfangreicher sind? Ist man wirklich darauf angewiesen, in eigener Regie Anpassungen und Fehlerkorrekturen an nicht selbstentwickelten Programmen vorzunehmen? Besteht damit nicht die Gefahr, dass diese Programmierer sich in Details verlieren und sich dabei mitunter wie Hacker benehmen? Haben unsere Anwenderunternehmen keine wichtigeren Aufgaben für die ach so teuren und knappen Informatiker?

Interessenlage der Gerätehersteller

Alle Rechnerhersteller hatten immer das Problem, dass sie nie ganz ohne Software auskamen. Bei vielen Firmenneugründungen (und das nicht nur in der Vergangenheit) reichte das Anfangskapital oft nur zur Herstellung der ersten Chips oder der ersten Maschinen. Für Software hätte man zusätzlich Geld und Zeit gebraucht. Schön ist es, wenn man dann die potentiellen Käufer auf öffentlich verfügbare (engl.: public domain) Software oder auf die Software von anderen Firmen verweisen kann.

Außerdem ist Software ein etwas anderes Geschäft als Hardware, das vielen Geräteherstellern nicht so recht liegt. Software ist nämlich von Natur aus hardwareübergreifend (nicht hardwareunabhängig!). Sie hat Erfolge besonders dann, wenn sie nicht an bestimmte Gerätearchitekturen oder Rechnerhersteller gekoppelt ist. Immer mehr Kunden machen ihre Geräteentscheidung davon abhängig,

welche Anwendungssoftware dafür zur Verfügung steht. Diese wiederum ist von Betriebssystemen oder anderen Systemprogrammen abhängig, die jemand anderes kontrolliert.

Nicht mit allen Softwarefirmen kann man sich gleich gut arrangieren. Je mehr Software es gibt, umso höher sind die Kosten, die darin stecken. Werden diese Kosten über separate Preise abgegolten, belastet dies oft dasselbe Budget, für das Hardware gekauft wird.

Gibt es z. z. ernsthafte Probleme dadurch, dass neue Geräte und Technologien von kommerziellen Programmen nicht rechtzeitig und gut genug unterstützt werden? Besteht Hoffnung, dass diese Situation sich durch Graswurzelaktivitäten deutlich verbessern lässt? Wie groß ist das Risiko, wenn man sich auf zufällig entstandene und im Internet gefundene Programme verlässt? Läge es nicht im Interesse der Rechnerhersteller, wenn der berühmte Anwendungsstau reduziert würde, etwa indem neue Anwendungen portabel und effizient entwickelt oder indem vorhandene Anwendungen für alle Plattformen lauffähig gemacht würden? Wird dieses Ziel von Open-Source-Anhängern gefördert oder nicht? Lohnt es sich wirklich, Softwarekosten als Hardwarekosten darzustellen, also Software mit Hardware zu bündeln? Kann es sein, dass sich einige Gerätehersteller die Zeit vor 30 Jahren zurückwünschen, als das Interesse an Hardware noch nicht von Softwareaspekten überschattet wurde?

Interessenlage der Softwarebranche

Von allen Softwareprodukten sind Betriebssysteme am weitesten verbreitet. Da sie nicht mehr viel kosten, sind sie zwar ein kleines, aber dennoch ein unvermeidbares Übel.

Der Markt der Betriebssysteme wird de facto von einer einzelnen Softwarefirma beherrscht. Ihr Geschäftsgebaren ist nicht ganz frei von Kritik. Das führte zu Emotionen, ja Wut im Bauch, nicht zuletzt bei jungen Leuten. Statt sich auf Gerichte zu verlassen, macht man sich gerne die Hilfe aller derer zu nutze, die auf ihre Art einen Kreuzzug gegen das „böse“ Establishment führen möchten. Man hat ja selbst genug anderes zu tun. Wie der derzeitige Mangel an Softwareentwicklern auf der ganzen Welt zeigt, ist das Potential der Branche riesig.

Offensichtlich ist aber eine bestimmte Art von Entwicklerkapazität vorhanden, auf welche die

Branche keinen Zugriff hat. Diese Programmierer arbeiten angeblich nicht gerne für Firmen, da dies nicht ihrem Lebensziel oder Stilgefühl gerecht wird. Das Internet reizt diese Leute dazu, sich nicht nur an Spielen zu beteiligen, sondern auch sinnvolle Projekte im ortsübergreifenden Verbund durchzuführen.

Ist eine industrielle Arbeitsteilung in der Form, dass Lieferanten für die Qualität ihrer Produkte zumindest minimale Verantwortung übernehmen, in der Softwarebranche fehl am Platze? Führen die Entwicklungsprozesse, wie sie etwa von CMM (Capability Maturity Model) oder ISO 9000 gefordert werden, zu weiter nichts als zu Bürokratie? Wird das Vertrauen in die Verlässlichkeit von Software wirklich gefördert, wenn man Nutzern nahe legt, an jedem komplexen, fragilen Produkt Änderungen zu machen? Sollte man sich darauf verlassen, dass der moderne Kreuzzug, der sich Open Source nennt, seine ideologischen Ziele erreicht, also das „heilige Land von Ungläubigen befreit“? Weiß ein 20-jähriger Informatikstudent z. z. besser als jedes Softwarehaus, was der Markt braucht? Warum riskieren Softwarefirmen nicht noch mehr Innovationen auf den Gebieten, die der Marktführer noch nicht im Griff hat? Falls die Zahl der Programmierer, die lieber in einer sehr lockeren Beziehung und zu Hause arbeiten, signifikant ist, warum überlegt die Softwarebranche nicht, was man tun kann, um diese Entwicklungskapazität auszunutzen? Kann es sein, dass manche jungen Leute falsche Vorstellungen vom Arbeitsstil in der Softwareindustrie haben, soweit sie diesen nur vom Hörensagen her kennen?

Volkswirtschaftliche Sicht

Die Softwarebranche schafft inzwischen auch bei uns Arbeitsplätze. Die Hardwareindustrie tut es auch, aber in etwas geringerem Umfang. Wenn man öffentliche Mittel dafür einsetzt, um nichtkommerzielle Gebrauchssoftware zu verbreiten, macht man nichts anderes, als einer Industrie, die Arbeitsplätze schafft, das Wasser abzugraben. Dies hat bei den z. z. diskutierten Beispielen (Standardbetriebssysteme, Web-Server) weder mit Forschung noch mit der regionalen Wirtschaftsbelebung oder der Korrektur von Wettbewerbsverzerrungen etwas zu tun. Man reduziert das, was eine industrielle Tätigkeit sein könnte, auf ein Hobby. Es werden vielleicht sogar

Leute davon abgehalten, ein normales Arbeitsverhältnis einzugehen oder gar Unternehmer zu werden.

Ich nehme nicht an, dass nichtkommerzielle Gebrauchssoftware während der bezahlten Arbeitszeit entsteht, sei es in der privaten Wirtschaft oder im öffentlichen Dienst. Das würde ich aus betrieblicher Sicht als bedenklich ansehen, wenn nicht sogar als Illoyalität oder Unregelmäßigkeit.

Die Verantwortung der öffentlichen Hand, neue Konzepte zuerst in der Forschung zu fördern, ehe daraus Gebrauchsprodukte entstehen können, ist gerade in der Softwarebranche unbestritten und hat in vielen Fällen (insbesondere in den USA) auch zu großen Erfolgen geführt. Die Open-Source-Diskussion betrifft ausschließlich den der Forschung nachgeordneten Bereich, die wirtschaftliche Verwertung.

Von besonderer Brisanz ist die von der Open-Source-Bewegung aufgeworfene Diskussion um Urheberrechtsschutz und Patentierbarkeit von Programmen. Hier nehmen einige Sprecher dieser Szene sehr extreme Positionen ein. So fordert Stallman [4] seit Jahren die generelle Abschaffung des Urheberrechts, und zwar nicht nur für Software. Man möchte am liebsten den Begriffs des geistigen Eigentums ganz abschaffen und Information und Wissen vergesellschaften. Da sich bei uns Gesetzgebung und Rechtsprechung an Software und elektronische Publikationen nur schrittweise annähern, herrscht noch allenthalben große Unsicherheit.

Welche volkswirtschaftlichen Vorteile oder Nachteile hat es, wenn Software und andere Wissensprodukte in einem freien Markt angeboten werden und konkurrieren dürfen? Bei welchen Aspekten dieses Marktes sollte der Staat regulierend eingreifen? Sollten wir nicht öffentliche Mittel dafür einsetzen, dass neue Konzepte und Anwendungsgebiete erschlossen werden, aus denen wirtschaftlich selbsttragende Tätigkeiten hervorgehen? Wollen wir eine Wirtschafts- und Gesellschaftsstruktur, in welcher der Begriff des geistigen Eigentums ausgehöhlt oder abgeschafft wurde? Brauchen wir nicht eine weltweite Klärung der Urheberrechtsfrage für Information und Wissen, vor allem wenn sie in neuen Medien dargeboten werden, und zwar so, dass dadurch neue wissensbasierte Tätigkeiten gefördert und nicht behindert werden?

Über die neue Wirtschaft und die Zukunft der Software

Die Antworten zu den oben gestellten Fragen sollte sich jeder geben, der vor der Situation steht, sich für Open Source zu engagieren oder ein Unternehmen, eine Behörde oder einen Meinungsträger diesbezüglich zu beraten. Selbst einige deutsche Bundesministerien scheinen hier Orientierungshilfe zu benötigen. Ich will einige der Fragen indirekt beantworten, indem ich die Open-Source-Diskussion einordne in die allgemeine Wirtschaftsdiskussion und kurz darlege, was davon ich im Hinblick auf die Zukunft der Software und der Informatik für wichtig erachte.

Das Konzept der Open Source wird oft als neue wirtschaftliche Errungenschaft dargestellt. Obwohl Kelly [2] Linux und Apache als Beispiele für seine „Wirtschaft der Geschenke“ (engl.: gift economy) benutzt, haben zwei der vier erwähnten Aspekte (nur Quellcode, kein Urheberrechtsschutz) nichts mit den Regeln dieser neuen Wirtschaft, wie er sie beschreibt, zu tun. Selbst bei dem Slogan „neue Wirtschaft“, der die Wertpapierbörsen zeitweise sehr beflügelte, ist Realismus und Nüchternheit geboten.

Dass Geschenke ein gutes Mittel sind, um Marktanteile zu gewinnen, wissen Hersteller von Konsumartikeln wie Seife und Zigaretten schon lange. Allerdings hat die übertriebene Anwendung dieses Prinzips oft auch ein „Geschmäcke“, wie die Schwaben sagen. Überleben kann man damit nur, wenn man ein gehöriges Polster hat. Kennzeichnend für die neue Wirtschaft ist, dass wissensbasierte Produkte eine zentrale Rolle spielen. Software ist dafür zweifellos ein Beispiel. Außerdem sollen Märkte nicht mehr durch effiziente Produktion und Verteilung gewonnen werden, sondern durch das Erkennen oder Wecken neuer Bedürfnisse und einer raschen Reaktion auf diese. Um wirkliche Bedürfnisse zu befriedigen, muss man nicht unbedingt Geschenke verteilen.

Dass der Open-Source-Kult neue Perspektiven für die Zukunft der Software eröffnet, bezweifle ich. Es handelt sich m. E. teilweise um einen (romantischen) Rückfall in die Zeit vor 1969, als Software primär als kostenlose Beigabe zu Rechnern angesehen wurde. Was wir dagegen heute und in Zukunft brauchen, ist eine richtige Einschätzung des hohen wirtschaftlichen Wertes von Software.

Nur dann wird (auch in unserem Lande) mehr Geld in Software investiert und die Software entwickelt, die unsere Wirtschaft und unsere Gesellschaft braucht. Wenn dies auf industrieller Basis zu langsam und nicht bedarfsgerecht geschieht, können Graswurzelaktivitäten eine willkommene Ergänzung sein; eine Alternative sind sie kaum. Das gilt für Software, die als eigenständiges Produkt in Erscheinung tritt, aber auch für Software, die in ein anderes technisches Produkt (Auto, Fahrstuhl oder Waschmaschine) eingebettet ist. Dass Software längst ein Wirtschaftsgut ersten Ranges darstellt, ist eine Einsicht, der gegenüber sich immer weniger Verantwortliche aus allen Branchen und Gesellschaftsgruppen verschließen können.

In soziologischer Hinsicht haben Graswurzelaktivitäten natürlich enorme Vorteile, die kein Industrieprodukt so leicht wettmachen kann. Es werden dadurch u. U. sehr tief liegende emotionale Einstellungen angesprochen. Wer sie ignoriert, macht sicher einen Fehler. Oft handelt es sich um Untergrundreaktionen gegen den offiziellen Trend, der dann früher oder später korrigiert werden muss. Einem Teil meiner obigen Fragen, besonders den etwas überspitzt klingenden, liegt genau diese Betrachtungsweise zugrunde. Fast jedes technische Gebiet kann zum Betätigungsfeld von Hobbyisten werden. Wie bei allen Erzeugnissen, bei denen sich das Ergebnis professioneller und amateurhafter Vorgehensweise auf den ersten Blick kaum unterscheiden, so ist auch im Falle von Software eine Grenzüberschreitung vom Atelier zum Labor oder vom Bastler zum Ingenieur sehr leicht möglich. Soweit dies eine Randerscheinung ist und hauptsächlich Vorteile bringt, sollte man sich darüber freuen.

Dass kommerziell entwickelte Software für den Endnutzer teuer sein muss, ist ein selbst unter Experten hin und wieder anzutreffender Fehlschluss. Das hängt damit zusammen, dass bei Software immer noch zu oft an eine Spezialanfertigung für eine einmalige Anwendung und nicht an einen wiederverwendbaren Standardbaustein gedacht wird. Je mehr ein bestimmtes Softwareprodukt gebraucht wird, also je öfter es wiederverwandt wird, umso höher dürfen seine Entwicklungskosten sein und umso preisgünstiger kann es angeboten werden. Das ist zwar paradox, gilt aber für alle Produkte, für welche die Entwicklungs- und nicht die Reproduktionskosten entscheidend sind. Das war zwar in der

alten Wirtschaft selten, ist aber in der neuen Wirtschaft die Regel.

Solche Gesetzmäßigkeiten kommen da am ehesten zum Tragen, wo es einen leistungsfähigen und elastischen Markt gibt. Weder eine Blockierung von Marktmechanismen (etwa durch Monopole) noch ihre Umgehung (etwa durch Ausblenden kommerzieller Aspekte) helfen hier weiter. Angesichts des Misserfolgs planwirtschaftlicher Experimente kann man davon ausgehen, dass es sowohl für Softwarehersteller wie für Softwarenutzer zuträglicher ist, wenn Marktmechanismen verstärkt zur Anwendung kommen. Dazu passt keine Sozialisierung durch die Hintertür; auch dann nicht, wenn sie als freiwillige Option daherkommt.

Echte Fortschritte in der Zuverlässigkeit von Software sind, auch im Interesse des Fachgebiets Informatik, sehr wünschenswert. Ich erwarte diese vor allem dadurch, dass man noch mehr als bisher zwischen den unterschiedlichen Anforderungen differenziert und dass man die geballten wissenschaftlichen und technischen Bemühungen auf die wirklich kritischen Komponenten eines Systems konzentriert, und zwar zur Entwicklungszeit. Danach sollte die Veränderbarkeit von Software tunlichst verhindert werden, etwa durch Verschlüsselung, oder indem man sie in Hardware gießt. Anzunehmen, dass die Qualität von Software oder generell die von technischen Produkten sich ohne eigene Anstrengungen der Entwickler einstellt, also allein durch die Aufmerksamkeit der Nutzer oder gar aufgrund von Wachstums- und Selektionsprozessen, ist eine merkwürdige Utopie. Vor ihr sollte man warnen, und dies insbesondere bei jungen Menschen.

Fazit

Wie in diesem Beitrag angedeutet, kommt das Interesse an dem Phänomen Open Source aus ganz unterschiedlichen Lagern. Jeder sucht sich den Aspekt heraus, der ihn anspricht. Es ist oft ein etwas seltsames Zusammentreffen von Partnern. Die Frage ist, wer tut wem einen Gefallen. Vielleicht ist aber der Nutzen gar nur eine Illusion oder (für einem Kult) nur eine Nebensächlichkeit. Das bedeutet aber nicht, dass man die Realität des Phänomens übersehen kann. Genau so wie die Stadt Berlin nicht umhin kann abzuwägen zwischen „Love Parade“ und klassischem Kultur- und Unterhaltungsangebot, muss

man sich auch hier überlegen, wofür man sich mehr engagiert.

Lernen können wir aus der Open-Source-Bewegung, dass für manche Formen der Softwareentwicklung und der anschließenden Verteilung und Nutzerbetreuung sehr flexible Arbeitsformen möglich sind und dass sich das Internet dabei als sehr nützlich erweisen kann.

Literatur

1. Glass, R.L.: The Sociology of Open Source: Of Cults and Cultures. IEEE Software 17(3), 104–105 (2000)
2. Kelly, K.: New Rules for the New Economy. New York: Penguin 1998
3. Reif, W.: Formale Methoden für sicherheitskritische Software – Der KIV-Ansatz. Info. Forsch. Entw. 14(4), 193–202 (1999)
4. Stallman, R. : The Right to Read. Comm. ACM 40(2), 85–87 (1997)