Open Source Community Building

Matthias Stürmer, Thomas Myrach

23. November 2005, überarbeitet 19. Dezember 2005

Zusammenfassung

Für die Initiatoren eines Open-Source-Projektes ist es eine langwierige und anspruchsvolle Aufgabe, eine prosperierende Community zu bilden. Im vorliegenden Artikel werden einschlägige Erfahrungen systematisiert, die bei Experten aus acht erfolgreichen Open-Source-Projekten erhoben wurden. Die Resultate können als Handlungsempfehlungen für den Aufbau von Open-Source-Communities aufgefasst werden. Zu Beginn werden wünschbare Charakteristiken einer Open-Source-Community beschrieben. Davon ausgehend werden die Eigenschaften und Verhaltensweisen der Projektverantwortlichen erörtert, sowie günstige Ausgangsbedingungen und Voraussetzungen eines Open-Source-Projektes erläutert. Schlussendlich wird ein systematischer Überblick über verschiedene Bereiche gegeben, welche für einen Community-Aufbau förderlich sind.

Inhaltsverzeichnis

„And do take your time. That's really it, it needs a lot of time, a lot of patience, a lot of openness.“

Guido Wesdorp, Hauptentwickler im Kupu Editor Projekt

1. Einleitung

Das Entwickeln von Software ist eine anspruchsvolle und komplexe Tätigkeit. Viele Software-Produkte sind so umfangreich, dass sie nicht von einer einzigen Person in überschaubarer Zeit geschaffen werden können. Derartige Entwicklungsarbeiten verlangen das koordinierte Zusammenwirken eines Teams im Zuge eines IT-Projektes. Das Aufsetzen einer Projektorganisation und die in diesem Rahmen erfolgende Koordination und Motivation der Teammitglieder ist eine wichtige Führungsaufgabe. Die Art, wie diese Führungsaufgabe wahrgenommen wird, ist mitbestimmend für den Erfolg des Projektes.

Auch in Open-Source-Projekten erzwingt die Komplexität in der Regel die Kooperation verschiedener Personen. Jedoch sind die Spielregeln dabei anders als in kommerziellen Software-Entwicklungsprojekten. Die Partizipation an einem solchen Projekt erfolgt gewöhnlicherweise auf freiwilliger Basis. Durch die Freiwilligkeit bei der Mitarbeit hat der Projektinitiator auch keine oder nur sehr bedingte Weisungsbefugnisse gegenüber den Projektmitarbeitern. Wenn diesen die Aufgaben nicht passen, können sie ohne rechtliche oder finanzielle Konsequenzen deren Bearbeitung verweigern oder sich gar aus dem Projekt zurückziehen. Da die Mitarbeit weder durch die Bezahlung einer Dienstleistung noch eines Lohnes entgolten wird, sind in diesem Fall andere Mechanismen erforderlich, um sich der gewünschten Kooperation zu versichern. Es erscheint plausibel, dass ein Projektverantwortlicher viel stärker auf Maßnahmen und Instrumente zur Förderung der intrinsischen Motiviation zurückgreifen muss, als dies bei kommerziellen Software-Entwicklungen der Fall ist.

Ein weiteres Merkmal, das die Software-Entwicklung im Rahmen von Open-Source-Projekten von vielen kommerziellen IT-Projekten abhebt, ist die ausgesprochen dezentrale Entwicklungstätigkeit, die nicht selten sogar länderübergreifend erfolgt. Anders als in konventionellen IT-Entwicklungsprojekten sind die Mitarbeiter eines Teams nicht an einem Ort zusammengezogen, sondern geographisch verteilt. Die Kommunikation erfolgt in der Regel indirekt. Prinzipiell darf man unterstellen, dass das Führen in verteilten Organisationen spezifische Anforderungen an die Organisation der Zusammenarbeit und die Kommunikation innerhalb eines Teams stellt (Picot/Reichwald/Wigand 2001).

Ein zentrales Element jedes Open-Source-Projektes ist die Community, durch die es getragen wird. Mehrfach wird im nachfolgenden Text betont, dass die Community grundsätzlich wichtiger ist als der Einzelne. Daraus ist aber keineswegs abzuleiten, dass Community-Mitglieder alle gleich seien und dass eine Community praktisch selbstorganisiert und basisdemokratisch funktioniere. Vielmehr lassen sich in jeder Community mehr oder weniger präzise verschiedene Rollen identifizieren und – typischerweise damit korreliert – eine unterschiedliche Stellung von Personen innerhalb der Community bezüglich Status und Macht ausmachen. Wir gehen hier von dem Grundverständnis eines schichtartigen Rollenmodells bei der Beschreibung des Aufbaus einer Open-Source-Community aus. Dies wird in der untenstehenden Abbildung illustriert. Es handelt sich hierbei um eine grob typisierende Abstraktion, die sich im konkreten Fall durchaus etwas anders darstellen kann.

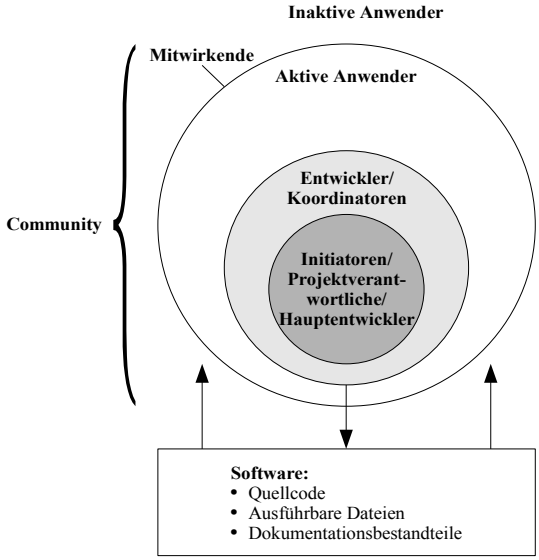

Abbildung 1: Grob typisierte Abstraktion eines Open-Source-Projekts

Wir definieren eine Open-Source-Community als die Gesamtheit aller Personen, die durch irgendwelche Beiträge direkt oder indirekt an der Entwicklungstätigkeit eines Open-Source-Projektes partizipieren. Dies sind ausdrücklich nicht nur die Programmierer, die den eigentlichen Source-Code schaffen bzw. verändern. Ausgegrenzt sind hingegen jene Personen, welche das Software-Produkt lediglich nutzen, ansonsten aber keinen aktiven Beitrag zur Weiterentwicklung des Produktes leisten. Ausgesprochen oder unausgesprochen haben die Personen, welche als Entwickler den Source-Code vorantreiben, oftmals eine privilegierte Stellung innerhalb der Community. Dabei kann sich sowohl aus der Art als auch des Umfangs der Entwicklungstätigkeit ein unterschiedlicher Status ableiten. Wir haben diesbezüglich die Rolle des Hauptentwicklers herausgehoben und diese dem innersten Kreis einer Community zugeordnet. Weiterhin erfordert eine Community auch die Rolle von Koordinatoren, welche vor allen im zwischenmenschlichen Bereich arbeiten und sich um die Anleitung, Motivation und vor allem auch Information von Community-Mitgliedern kümmern. Wie bei den Entwicklern kann diese Rolle auch hier mit einem unterschiedlichen Status verbunden sein. Die Projekteigner sind dem engsten Kreis zuzuordnen. Es handelt sich dabei um Personen, die praktisch uneingeschränkten Zugriff auf den Entwicklungsserver sowie die Software haben und anderen Personen bestimmte Zugriffsrechte einräumen (oder verweigern) können. Sie erfüllen ebenfalls Koordinationsaufgaben. Sowohl die Entwicklungs- als auch die Koordinationsrollen sind sehr häufig vermengt, zumal wenn die Entwicklungstätigkeit eng mit dem Status innerhalb einer Community verknüpft ist. In den Kernbereich eines Projektes sind sehr häufig auch die Projektinitiatoren einzuordnen, sofern sie sich nicht unterdessen zurückgezogen haben.

Im vorliegenden Artikel wird die Perspektive des inneren Kreises eines Open-Source-Projektes eingenommen und die Frage gestellt, auf welche Aspekte bei der Initialisierung und dem Aufbau einer Community besonders geachtet werden sollen. Dadurch richtet er sich primär an Personen, die sich bereits heute zu den Verantwortlichen eines Open-Source-Projekts zählen oder mit dem Gedanken spielen, selber ein Open-Source-Projekt zu initialisieren. Ihnen werden allgemeine Hinweise gegeben, welche förderlichen Tätigkeiten bereits anderen beim Community-Aufbau geholfen haben.

Die Aussagen in diesem Artikel basieren auf den Ergebnissen einer empirischen Untersuchung, die im Zuge einer Lizenziatsarbeit an der Universität Bern im Jahr 2004/2005 durchgeführt wurde (Stürmer 2005). Das Vorgehen folgt einem qualitativen Ansatz der Art, wie es

Eisenhardt (1989) vorschlägt. Es wurden acht Open-Source-Projekte aus dem Bereich CMS und Web-basierte Entwicklungsumgebungen ausgewählt und mit jeweils einem möglichst „hochrangigen“ Community-Mitglied problemzentrierte Experteninterviews durchgeführt. In der untenstehenden Tabelle sind einige wesentliche Informationen zu den ausgewählten Projekten und den Interviews zusammengestellt. Die acht insgesamt zwölfstündigen Interviews wurden aufgezeichnet, transkribiert und ins Englische übertragen (sofern dies notwendig war). Die so erfassten Interviews wurden mit Hilfe des Werkzeuges MaxQDA einer detaillierten Inhaltsanalyse unterzogen und anschließend systematisiert. Wie bei jeder qualitativen Analyse erfordert die hier angewendete Vorgehensweise einige Interpretationen. Diese betreffen allerdings vor allem die Systematisierung, weniger die gemachten Aussagen. Jene lassen sich weitgehend auf Äusserungen der befragten Experten zurückführen; aus Übersichtsgründen wurden hier jedoch die wörtlichen Zitate ausgeblendet. Alle zusammengefassten Aussagen stammen von mindestens zwei Quellen und sind deshalb intersubjektiv belegt.

Angaben über das Open-Source-Projekt								
	Plone	**Magnolia**	**Cocoon**	**Kupu**	**Lenya**	**TYPO3**	**eZ publish**	**Xaraya**
Software-Kategorie	CMS Framework	CMS	Web Application Framework	WYSIWYG Browser Editor	CMS	CMS	CMS	CMS und Application Framework
Programmiersprache	Python	Java	Java	JavaScript	Java	PHP	PHP	PHP
Initiator(en)	Alexander Limi & Alan Runyan	obinary	Stefano Mazzocchi	Guido Wesdorp und andere	Michael Wechner	Kasper Skårhøj	eZ systems	?
Trägerschaft	Plone Foundation	Das Unternehmen obinary	Apache Software Foundation	keine	Apache Software Foundation	TYPO3 Association	Das Unternehmen eZ systems	Digital Development Foundation
Angaben über den Interview-Partner								
	Bernhard Bühlmann	**Boris Kraft**	**Bertrand Delacrétaz**	**Guido Wesdorp**	**Michael Wechner**	**Daniel Hinderink**	**Bård Farstad**	**Gregor Rothfuss**
Aufgabe im Open-Source-Projekt	Koordinatior	Community-Verantwortlicher	Entwickler	Hauptentwickler	Initiator und Hauptentwickler	Marketing-Koordinator	Community-Verantwortlicher	Entwickler
Unternehmen	4teamwork	obinary	codeconsult	Infrae	wyona	dpool	eZ systems	?
Angaben über das Interview								
Datum	**17.11.2004**	**19.11.2004**	**24.11.2004**	**25.11.2004**	**30.11.2004**	**03.12.2004**	**03.12.2004**	**07.12.2004**
Dauer	95 min	108 min	63 min	79 min	106 min	116 min	78 min	103 min
Sprache	Deutsch	Deutsch	Deutsch	Englisch	Deutsch	Deutsch	Englisch	Deutsch
Ort	Bern, CH	Basel, CH	Cugy, CH	Rotterdam, NE (Telefon)	Bern, CH	München, DE (Telefon)	Skien, NO (Telefon)	Boston, USA (Telefon)

Tabelle 1: Angaben über die Open-Source-Projekte, Interview-Partner und Interviews

Die nachfolgende Gliederung folgt der in der Untersuchung vorgenommenen Systematisierung. Zuerst werden grundsätzlich wünschenswerte Charakteristiken erfolgreicher Open-Source-Communities identifiziert. Anschließend werden wichtige Eigenschaften und Verhaltensweisen von Projektverantwortlichen aufgezeigt. Danach werden verschiedene Voraussetzungen für die erfolgreiche Initialisierung von Open-Source-Projekten dargelegt. Schließlich erfolgt eine Reihe von Handlungsempfehlungen zur Förderung des Community-Aufbaus. Diese orientieren sich an einem Untersuchungsraster, in dem die Ebenen der Rekrutierung von Community-Mitgliedern, die Organisation der Zusammenarbeit innerhalb der Community und der Einfluss auf die Entwicklungstätigkeit bezüglich verschiedener relevanter Bereiche betrachtet wird.

2. Charakteristiken prosperierender Communities

Zuerst sollen allgemein gültige Eigenschaften von Open-Source-Communities deren angestrebte Charakteristiken aufzeigen. Es ist also nicht irgendeine Vergrößerung der Community erwünscht, sondern nur eine solche, die den folgenden sieben Kriterien entspricht.

Die Community hat keinen Selbstzweck sondern ein primäre Anliegen: die *Weiterentwicklung* des Open-Source-Projekts, d.h. Programmcodes zu erstellen und zu verbessern, Dokumentationen zu erstellen, Marketing zu betreiben und auf andere Art produktiv zu sein. Dies ist nicht selbstverständlich, denn in vielen Projekten wird wertvolle Energie in endlose Diskussionen und Streitigkeiten gesteckt, ohne die Software tatsächlich weiter zu bringen. Ein anderes Phänomen sind Einzelpersonen und vor allem Unternehmen, welche die Open-Source-Software für sich und für Kunden gewinnbringend einsetzen, ohne z.B. im Rahmen dieser Tätigkeiten erstellte Weiterentwicklungen dem Projekt zur Verfügung zu stellen. Solches Trittbrettfahrverhalten ist in der Community unerwünscht und bringt auch das Projekt nicht direkt weiter.

Eine wichtige Eigenschaft von aktiven Community-Mitgliedern ist deren hohe *Selbstmotivation*. Optimal ist es, wenn Personen eigene Ideen von Anfang bis Ende selber realisieren und dennoch offen bleiben für Kritik und Verbesserungsvorschläge anderer. Üblicherweise müssen Entwickler gewisse Barrieren überwinden, bevor sie Änderungszugriff auf die Software erhalten (von Krogh/Spaeth/Lakhani 2003). Dadurch wird gewährleistet, dass nur Personen, welche schon längere Zeit intensiv am Projekt mitarbeiten und Beiträge liefern, direkten Einfluss auf die Weiterentwicklung der Software nehmen können.

Um die Langlebigkeit eines Open-Source-Projektes gewährleisten zu können, ist es vorteilhaft, Community-Mitglieder aus verschiedenen Organisationen und Regionen zu beteiligen. Einerseits macht diese *Vielfältigkeit* der Mitwirkenden das Projekt unabhängig von lokalen Veränderungen, weshalb unter anderem die Apache Foundation für neue Projekte voraussetzt, dass die Entwickler aus mindestens drei unterschiedlichen Unternehmen stammen {Quelle: The Apache Incubation Policy, 31.10.2005}. Andererseits bringen erst verschiedenartig talentierte Menschen das umfangreiche Wissens- und Fertigkeitsspektrum mit, welches für die Leitung und Umsetzung von Open-Source-Projekten notwendig ist.

Da sich die Mitarbeitenden in Open-Source-Projekten vielfach freiwillig auf unentgeltlicher Basis beteiligen, ist *Korrektheit* in der Kommunikation unentbehrlich. Das heißt konkret, dass z.B. selbst einfache Anfängerfragen von der Community respektvoll beantwortet werden sollten. Andererseits bedeutet es aber auch, dass Neueinsteiger sich an die geltenden Kommunikationsregeln zu halten haben und ihre Fragen zuerst selber in Mailing-List-Archiven oder Wikis zu beantworten versuchen.

Auch ein gewisser Grad an *Altruismus* ist wichtig, denn grundsätzlich gilt: Open-Source-Projekte sind keine „One Man bzw. One Company Shows“ und „die Community ist wichtiger als der Einzelne.“ Auch wenn z.B. jemand sehr viel Quellcode beiträgt und wichtige Tätigkeiten für die Community ausführt, kann dies langfristig das Ende des Projekts bedeuten, wenn er durch seine dominante Art die anderen Mitglieder vertreibt und damit die Community schwächt. Besonders wenn Unternehmen maßgeblich an der Weiterentwicklung eines Projektes beteiligt sind, müssen sie Rücksicht nehmen auf die Bedürfnisse der Community und dürfen zum Beispiel bei der Rückwärtskompatibilität nicht egoistisch vorgehen.

Ein *langfristiges Engagement* von Mitarbeitenden in Open-Source-Projekten ist ein wertvolles Gut. Vielfach kommt es vor, dass neu Dazugestoßene sich über die vorhandene Situation beschweren und Änderungswünsche anbringen aber bald wieder verschwinden. Verbesse-

rungsvorschläge z.B. an der Software-Dokumentation können nützlich sein, viel wichtiger ist es jedoch, dass Personen sich über längere Zeit produktiv in das Projekt einbringen und Aufgaben verantwortungsvoll ausführen. Deshalb erhalten in allen fortgeschrittenen Projekten nach dem Meritokratie-Prinzip ausschließlich solche Mitglieder Zugriff auf wichtige Ressourcen, die sich schon lange aktiv in der Community beteiligt haben.

Auch wenn sich viele Personen aus unterschiedlichsten Motivationen in einem Projekt einbringen, ist eine *gemeinsame Vision* für die Weiterentwicklung der Software unentbehrlich. Neben aller Offenheit für Beiträge neuer Mitwirkender muss sich eine klare Stoßrichtung für die Zukunft des Projekts herauskristallisieren, sonst wird die Diskussion stets in den selben Punkten stecken bleiben. Zu bewältigen ist also die Gratwanderung zwischen Berücksichtigung der Interessen aller Beteiligter und der Fokussierung auf bestimmte Ziele, wobei einerseits die schon entwickelten Stärken der Software und andererseits die vorhandenen Personen und Mittel beachtet werden sollten.

3. Eigenschaften und Verhalten von Projektverantwortlichen

Um ein neues Open-Source-Projekt starten und erfolgreich weiterentwickeln zu können, müssen zahlreiche personelle und sachliche Voraussetzungen gegeben sein. In diesem Kapitel werden zunächst wünschenswerte Eigenschaften und Verhaltensweisen von Projektverantwortlichen betrachtet. Zwar können jederzeit auch nachträglich noch entsprechend geeignete Personen engagiert werden. Es empfiehlt sich jedoch, dass bereits die Initiatoren eines neuen Projektes möglichst viele der folgenden Eigenschaften besitzen. Da die einzelnen Punkte nicht gewichtet wurden, sind sie in alphabetischer Reihenfolge aufgelistet.

Bestimmtheit: Entschiedenes Handeln ist in gewissen Stadien eines Open-Source-Projektes sehr gefragt, besonders wenn es sich um das Release Management handelt. Obwohl sich die Software stets wandelt, müssen vor der Herausgabe einer neuen Version die von den Verantwortlichen kommunizierten Veröffentlichungstermine eingehalten werden können, weshalb in diesen Momenten eine gewisse Bestimmtheit notwendig ist.

Engagement: Vom Initianten eines neuen Open-Source-Projektes wird höchstes Engagement erwartet. Gerade in der Aufbauphase des Projektes kann noch nicht mit breiter Unterstützung zahlreicher Interessenten gerechnet werden, weshalb die Hauptarbeit von den Initiatoren ausgeführt werden muss. Aber auch in späteren Stadien ist mit einem hohen Zeitaufwand zu rechnen, um ein Open-Source-Projekt verantwortungsvoll koordinieren zu können.

Erfahrung: Das kontinuierliche Ändern und Wachsen des Quellcodes fügt der Software zwar stets neue Funktionalitäten zu, stellt die Koordinatoren jedoch gleichzeitig vor eine immer größer werdende Informationsmenge. Nur wer über längere Zeit das Projekt begleitet, sich stets über die Erneuerungen informiert hält und so entsprechende Erfahrungen mit und Kenntnisse über die Software erwirbt, kann schließlich bei auftretenden Fragen Auskunft geben und aktiv die Zukunft des Projektes mitgestalten.

Geduld: Der Aufbau und die Koordination eines Open-Source-Projektes ist eine langfristige Aufgabe. Selbst bei heute erfolgreichen Projekten dauerte es teilweise mehrere Jahre, bis sie eine Eigendynamik erreichten, in der die Community stark zu wachsen begann. Beim heutigen Überangebot an Open-Source-Lösungen hat der Bedarf an geduldigen Projektinitiatoren abermals zugenommen. Bis eine Community tatsächlich zu „leben“ beginnt und Mitglieder z.B. selbständig Einsteigerfragen lösen, müssen die Initiatoren bereit sein, nicht nur selber viele Fragen zu beantworten, sondern oft auch immer wieder die gleichen Problemstellungen zu behandeln.

Hilfsbereitschaft: Gerade die Bereitschaft, sich um alle Fragen aus der Community zu kümmern, zeichnet erfolgreiche Initiatoren aus. Nicht zufällig konnten von Krogh, Spaeth und Lakhani (2003) im Projekt Freenet eine rund 90% Beantwortungsquote aller gestarteten E-Mail-Anfragen feststellen. Wichtig ist also, dass sich Einsteiger willkommen und unterstützt fühlen, wenn sie sich an die Community wenden. Wenn tiefer greifende Problemstellungen aufgeworfen werden, kann aber vorausgesetzt werden, dass sich der Anfragende bereits selber um die entsprechende Behandlung bemüht hat. Ist dies ersichtlich und wird die Frage tatsächlich beantwortet, kann mit einem Multiplikatoreffekt gerechnet werden: Beim nächsten Mal wird die Person, wenn ihre technischen Kenntnisse gewachsen sind, selber jemanden unterstützen.

Offenheit: Die Offenheit in Open-Source-Projekten ist eines der Grundprinzipien und muss deshalb auch beim Handeln der Projektverantwortlichen auf verschiedene Weise erkennbar werden. Erstens ist die Offenheit wichtig, andere aktiv am Projekt mitwirken zu lassen. Zweitens ist auch Offenheit bezüglich der Aufgabenteilung in der Community erforderlich. In jedem Fall motivierend wirkt die dritte Art von Offenheit, die transparente Kommunikation. Und zu guter Letzt müssen „müde“ gewordene Projektverantwortliche auch offen sein, das Projekt wieder zu verlassen und entsprechende Verantwortungen, beispielsweise in Form von Zugriffsdaten, den Zurückbleibenden anzuvertrauen.

Mitteilsamkeit: Wie bereits angesprochen, ist intensive und transparente Kommunikation bei Open-Source-Projekten einer der Schlüssel zum Erfolg. Besonders wegen der dezentralen Organisation solcher Projekte ist eine hohe Kommunikationsintensität der Koordinatoren erforderlich und wird teilweise gar noch höher als deren technisches Können eingestuft (Raymond 1999). Jedoch müssen Mitteilsamkeit und Programmierfähigkeiten bei einer einzelnen Person nicht unbedingt korrelieren. Eine Art Kommunikations-Training kann das Führen eines Weblogs darstellen, denn kontinuierliches Berichterstatten über die eigene Entwicklertätigkeit ist auch Übungssache.

Persönlichkeit: Ausstrahlung ist zwar keine notwendige Bedingung für Projektinitiatoren, kann jedoch beim Aufbau einer Community sehr hilfreich sein. Gerade bei ehrenamtlichen Arbeiten wirkt ein gewisser Charme der Leitenden motivierend, sich auch „für sie“ zu engagieren. Nicht zuletzt sind Projekte anziehend, die von Menschen geleitet werden, mit denen man gerne zusammenarbeiten möchte. Eine wichtige Charaktereigenschaft der Projektverantwortlichen ist dabei eine gewisse Bescheidenheit.

Präsenz: Wird nicht physisch zusammengearbeitet, so ist die virtuelle Präsenz der Hauptakteure umso wichtiger. Sei es in einem Chatraum, in den Mailing Listen oder in einem Forum: die allgegenwärtige Anwesenheit von kompetenten Personen ist sehr wertvoll. Zwar verlangt diese Verfügbarkeit zeitlich und mental einen großen Einsatz. Dieser wird aber durch das hohe Ansehen in der Community und zuweilen auch mit bezahlten Reisen als Konferenzredner oder Workshop-Leiter kompensiert.

Programmiertalent: Insbesondere während der Zeit des Projektaufbaus ist neben den zahlreichen Soft Skills vor allem das Programmiertalent der Initiatoren entscheidend. Zwar wirken sich klare Ziele, transparente Kommunikation, Offenheit und viele andere Führungsqualitäten positiv auf den Projekterfolg aus, am Ende zählt jedoch alleine das Software-Produkt. Deshalb beginnen alle Projektmitarbeitenden am Anfang mit Programmieren, bleiben häufig auch während der Weiterentwicklung hauptsächlich in diesem Bereich aktiv und betätigen sich, wenn überhaupt je, erst später z.B. als Community-Koordinatoren oder im Marketing.

Verantwortungsbereitschaft: Im Gegensatz zu reinen Anwendern und Programmierern von Erweiterungen sind die Koordinatoren am Gesamterfolg des Projektes interessiert. Wegen dieser langfristigen Perspektive zeigen sie mehr Verantwortungsbereitschaft als alle anderen Projektbeteiligten. Diese Einstellung äußert sich sowohl bei kleinen Aktivitäten wie dem Be-

arbeiten von unbeantwortet gebliebenen Nachrichten auf der Mailing List als auch im generellen Arbeitsstil des wohlbedachten Handelns mit Rücksicht auf alle Interessengruppen. In Konfliktsituationen müssen die Projektverantwortlichen vermitteln und gegebenenfalls sich unkorrekt verhaltende Community-Mitglieder zurechtweisen.

Vision: Gerade weil Open-Source-Projekte vom Engagement vieler Personen abhängig sind und sich diese zuweilen auch verzetteln können, sind klare Zielvorstellungen für das Projekt wichtig. Wer eine solche Entwicklung leitet, muss daher fähig sein, seine Projektvision formulieren und andere Projektbeteiligte dafür motivieren zu können. Dabei ist es entscheidend, begründet zu argumentieren. Dieses visionäre Denken als Charaktereigenschaft der Koordinatoren ist zentral, da nur dadurch der Projektfortschritt aktiv gelenkt werden kann.

4. Voraussetzungen für ein Open-Source-Projekt

Wurden bisher die Charakteristiken der angestrebten Communities sowie begünstigende Führungsqualitäten der Koordinatoren betrachtet, so sollen im folgenden Abschnitt kurz einige Aspekte der Ausgangslage eines Open-Source-Projektes beleuchtet werden.

Selten wird vor dem Start eines Open-Source-Projekts aktiv die *Programmiersprache* der zu schreibenden Applikation gewählt. Meist schränkt bereits die Entscheidung für ein Betriebssystem die Auswahl ein, die Initiatoren besitzen gewisse Sprachpräferenzen oder ein vorangegangener Unternehmensentscheid lässt nur bestimmte Programmiersprachen zu. Unterschiedliche Programmiersprachen beeinflussen auch den Zugang zu potentiellen Community-Mitgliedern: Da etwa die Programmierung in Java grundsätzlich eine komplexere Aufgabenstellung ist als das Schreiben eines PHP-Skripts, kann davon ausgegangen werden, dass Java-Programmierer meistens über eine höhere Informatikausbildung verfügen als ihre PHP-Kollegen. Andererseits ist diese niedrigere Eintrittsbarriere eine der Stärken von PHP, so dass die vielen PHP-Programmierer eine große, potentielle Entwicklerbasis für das aufzubauenden Open-Source-Projekt darstellen. Des weiteren beeinflusst die Wahl der Programmiersprache in gewissen Fällen auch das Image des Projekts.

Obwohl bei der Auswahl der geeigneten *Open Source Lizenz* eines Projekts vor allem juristische Fragestellungen auftauchen, wirkt sie sich auch im Bereich des Community-Aufbaus aus. Die zahlreichen OSI-zertifizierten Lizenzen {siehe http://www.opensource.org/licenses} variieren unter anderem bezüglich Gewährleistung unterschiedlicher Freiheiten und Bedingungen, was wiederum die Entwicklung der Community beeinflusst. Während beispielsweise die Apache Software License (ASL) die Freiheit gewährt, Quellcode-Ergänzungen verschlossen zu halten, schreibt die GNU Public License (GPL) vor, dass jegliche Änderungen der Software an das Projekt zurückzureichen sind. Dies wiederum führt zwingend zu einem stärkeren Ausbau der Community in GPL- als in ASL-Projekten. Weitere Differenzierungsmöglichkeiten bietet die Lizenzwahl für Unternehmen, wenn sie durch Vereinbarungen mit externen Entwicklern die Urheberrechte an der Software bewahren können und dadurch die Grundlage für die Anwendung des erfolgreichen Dual Licensing Modells schaffen (Välimäki 2003).

Als einer der wichtigen Erfolgsfaktoren eines Open-Source-Projekts darf sicherlich die Qualität des *ursprünglichen Quellcodes* bezeichnet werden. Dabei muss dieser nicht unbedingt bereits von Anfang an fehlerfrei ausführbar oder auch nicht besonders sauber und effizient programmiert sein. Um andere Entwickler von der Projektidee begeistern zu können, muss, wie Raymond (1999) formulierte, vor allem das Potential der Software ersichtlich sein. Dies bringt andere Programmierer dazu, sich für das Projekt einzusetzen und an seiner Weiterentwicklung zu arbeiten. Dabei ist vor allem die von den Initiatoren ausgearbeitete Quellcode-Architektur, das Application Programming Interface, entscheidend, da sie das Grundgerüst der

gesamten Software darstellt und in ihren Grundzügen zu einem späteren Zeitpunkt nur mit großem Aufwand verändert werden kann.

Wie im wirtschaftlichen Umfeld ist es auch im Open Source Bereich schwierig, das Ausmaß der zukünftigen *Nachfrage* nach der Software abzuschätzen. Ob ein Projekt auf Interesse der Benutzer und Entwickler stößt, hängt unter anderem stark vom Glück ab, im richtigen Moment die entscheidenden Leute von der neuen Idee zu überzeugen. Man sollte sich vor allem bewusst sein, welches Zielpublikum angesprochen werden soll, denn langjährige Software-Entwickler haben andere Interessen als sporadische Computer-Anwender. Auch kann die Attraktivität des Projekts vom richtigen Zeitpunkt der erstmaligen Publikation im Hinblick auf bestimmte Umweltbedingungen abhängen, wie etwa einem gestiegenen Kostenbewusstsein bei der IT-Nutzung.

Das Spektrum des *Innovationsgrades* neuer Open-Source-Projekte ist breit und reicht von der Nachbildung proprietärer Programme bis hin zu radikalen Software-Innovationen. Vorteil solch vollständiger Neuheiten ist ihre Konkurrenzlosigkeit. Aber auch die erstmalige Replikation geschlossener Software kann äußerst erfolgreich sein; schließlich war die Entwicklung von Linux „bloß“ die Kopie des jahrzehntelang eingesetzten Unix-Betriebssystems. Entscheidend ist eine gewisse Einzigartigkeit, die jedes neue Projekt zeigen sollte.

Je breiter die *Anwendbarkeit* der Software, desto größer ist die potentielle Anwender- und Entwickler-Community. Technisch bedeutet dies, dass eine Applikation auf verschiedenen Plattformen ausführbar oder eine Softwarekomponente in unterschiedliche Programmiersprachen integrierbar ist. Bezüglich der Einsatzmöglichkeiten ist dies der Fall, wenn eine Applikation auf verschiedene Weise angewandt werden kann und in unterschiedlichen Umgebungen Nutzen stiftet.

Wie bereits im Kapitel 3 mehrmals erwähnt, ist die intensive, *extern wahrnehmbare Kommunikation* im Zusammenhang mit dem Projekt eine wichtige Voraussetzung um neue Mitwirkende anzuziehen. Dies trifft besonders bei der Projektinitialisierung zu. Einerseits muss das Vertrauen der ersten externen Anwender gewonnen werden, was durch stetige Präsenz in allen Kommunikationskanälen unterstützt werden kann. Andererseits ist ein hoher Aufwand in die Erstellung von ausreichender Dokumentation zu stecken, damit das implizite Wissen der Initiatoren weitergegeben werden kann. Hat die Community einmal eine gewisse Größe erreicht, so ist zu hoffen, dass weitere Mitglieder aktiv werden und z.B. ebenfalls beginnen, auftauchende Fragen zu beantworten.

5. Förderung des Community-Aufbaus

In diesem Abschnitt werden den Projektverantwortlichen Handlungsempfehlungen gegeben, um auf den drei Ebenen *Rekrutierung, Zusammenarbeit* und *Entwicklungstätigkeit* das Wachstum der Community positiv zu beeinflussen. Aktivitäten auf der Ebene *Rekrutierung* beabsichtigen, weitere Projektbeitragende für die Community gewinnen zu können und betreffen somit Themen wie Projektattraktivität, Bekanntmachung, Verteilung und externe Kommunikation. Maßnahmen auf der Ebene *Zusammenarbeit* sollen die internen Prozesse verbessern und beeinflussen deshalb Organisation, Koordination, interne Kommunikation und Beziehungen. Aktionen auf der Ebene *Entwicklungstätigkeit* betreffen die Weiterentwicklung der Software und behandeln Themen wie Quellcode, Software-Architektur und Technologien. Wir unterscheiden sieben Bereiche, welche übergreifend auf allen drei Ebenen wirken. Acht weitere Bereiche betreffen spezifisch eine der drei Ebenen; sie werden kurz am Ende dieses Kapitels angesprochen. Tabelle 2 fasst die empfohlenen Aktionen der Projektverantwortlichen in einem stichwortartigen Überblick zusammen.

	↓ Bereich Ebene →	***Rekrutierung***	***Zusammenarbeit***	***Entwicklungstätigkeit***
Übergreifende Bereiche	***Modularität*** Software soll modular programmiert werden	Beschreibung zur Entwicklung von Erweiterungen zur Verfügung stellen	Bei starker Zunahme der Erweiterungen Qualitätssicherung durch erfahrene Programmierer einführen	Erweiterbarkeit der Software durch externe Komponenten von Anfang an vorsehen
	Dokumentation Für verschiedene Interessensgruppen Dokumentationen erstellen	Wenige dafür aktuelle und klar strukturierte Tutorials und Handbücher verfügbar machen	Anreizsystem für die Erstellung hochwertiger Dokumentation schaffen	Erläuterungen im Quellcode und zum Application Programming Interface schreiben
	Release Management Release Management Prozess einführen	Regelmäßig und häufig Software-Releases inklusive aussagekräftiger Ankündigungen herausgeben	Fairen Veröffentlichungsprozess, u.a. mit Feature-Freeze, praktizieren	Rückwärtskompatibilität gewährleisten; wenn unmöglich Migrationsskripte entwickeln
	Umgebung der Zusammenarbeit Eine Umgebung der Zusammenarbeit verwenden	Projekt auf den bekannten Plattformen unter kennzeichnenden Stichwörtern registrieren	Vorhandene bzw. individuelle Entwicklungs- und Zusammenarbeits-Plattform aufbauen	Den Entwicklern ein hochverfügbares Revision Control System zur Verfügung stellen
	Physische Begegnungen Physische Treffen veranstalten	Präsentationen und Workshops an Konferenzen und Messeveranstaltungen geben	Community-Treffen zur technischen und organisatorischen Zusammenarbeit organisieren	Entwickler-Sprints in motivierender Umgebung und mit klaren Zielen veranstalten
	Trägerorganisation Eine organisatorische Trägerschaft für das Projekt gründen	Das Projekt einer bestehenden Trägerschaft anschließen oder eine maßgeschneiderte neue gründen	Organisatorische, rechtliche und repräsentative Aufgaben durch die Trägerschaft wahrnehmen lassen	Entwicklertätigkeiten durch die Trägerschaft unterstützen und beschützen lassen
	Internationalisierung Eine internationale Community anstreben	Auf Englisch kommunizieren und Übersetzungsaufgaben ausschreiben	In einer multikulturellen Community höflich und respektvoll miteinander kommunizieren	Mehrsprachigkeit der Software auf technischer Ebene von Anfang an vorsehen
Ebenenspezifische Bereiche	***Bekanntmachung***	Herkömmliche Marketingaktivitäten betreiben und Pressemitteilungen verfassen		
	Credit System	Alle Beiträge mit deutlichem Hinweis auf Autoren veröffentlichen		
	Kommunikationskanäle		Entsprechend der Community-Größe Kommunikation auf einige wenige Kanäle fokussieren	
	Community-Struktur		Den Bedürfnissen der Community entsprechende Strukturen schaffen	
	Aufgabenliste		Kurze Beschreibungen anstehender Aufgaben mit Aufforderung zur Mithilfe publizieren und aktuell halten	
	Software-Qualität			Nur funktionierende und ausgereifte Software-Änderungen und -Erweiterungen integrieren
	Benutzeroberfläche			Benutzeroberfläche für einfache Bedienbarkeit und Gestaltung ausarbeiten
	Installation			Installationsprozess anhand eingeholter Feedbacks optimieren

Tabelle 2: Empfohlene Tätigkeiten von Projektverantwortlichen zur Förderung des Community-Aufbaus

In der Software-Entwicklung ist seit langem bekannt, dass *Modularität* des Quellcodes dessen Verständlichkeit und Flexibilität erhöht und dadurch den Aufwand für die Weiterentwicklung verringert (Parnas 1972). Besonders in einer meist virtuell zusammenarbeitenden Open Source Community ist es wichtig, dass sie an derselben Applikation programmieren kann, ohne alle Einzelheiten des Gesamtsystems kennen zu müssen. Wenn die Entwicklung von umfangreichen Erweiterungen und Plug-Ins möglich ist und durch eine entsprechende Einstiegsdokumentation unterstützt wird, kann das Projekt einen breiten Kreis von Anwendern und potentiellen Entwicklern ansprechen. Es wird ihnen ermöglicht, die Software auf relativ einfache Weise ihren Bedürfnissen anzupassen, was auch die Eintrittsbarriere für neue Programmierer senkt. Auf der Ebene der Zusammenarbeit schafft Modularität die Grundlage für die Spezialisierung der Programmierer und macht diese unabhängiger von den Hauptentwicklern der Software, da nun konkrete Probleme ohne Eingriff in die Kernapplikation gelöst werden können. Wird die Anzahl verfügbarer Erweiterungen groß, ist die Qualitätssicherung durch erfahrene Entwickler zu empfehlen.

Die Wichtigkeit von ausführlicher und vielseitiger *Dokumentation* wurde bereits mehrmals erwähnt. Für Anwender wie auch für Entwickler ist sie von zentraler Bedeutung und beeinflusst stark die Eintrittsmotivation zukünftiger Community-Mitglieder. Entscheidend ist, dass ein klarer Dokumentations-Leitfaden anhand von wenigen, dafür aktualisierten und zweckmäßig strukturierten Tutorials und Handbüchern den Einstieg in die Software ermöglicht. Freiwillige für die Erstellung qualitativ hochwertiger Dokumentationen, die z.B. für Lehrbücher verwendet werden können, sind schwer zu finden. Deshalb sollten Projektverantwortliche ein Anreizsystem schaffen, das zum Beispiel in Form von öffentlichen Verdankungen oder finanzieller Unterstützung Projektmitarbeitende ermutigt, Ressourcen für die Erstellung einer anspruchsvollen Projektdokumentation einzusetzen. Dazu gehören auch technische Dokumentationen wie Erläuterungen des Quellcodes oder Application Programmming Interfaces, die der Verständlichkeit und dadurch der Langlebigkeit der Software dienen.

Die Attraktivität eines Open-Source-Projekts wird auch durch dessen „Lebendigkeit“ bestimmt. Diese kommt vor allem durch die Häufigkeit von veröffentlichten Software-Versionen zum Ausdruck. Die Projektverantwortlichen sollten deshalb regelmäßig neue Releases veröffentlichen. Zum *Release Management* gehört neben einer Ankündigung mit Hinweis auf alle wegweisenden Erneuerungen auch ein vollständiges Änderungsprotokoll der neuen Software-Ausgabe. Die Festlegung eines neuen Releases ist eine sensible Aufgabe, bei der unter Einbezug aller Interessengruppen auch Feature Freezes eingehalten werden müssen. Dies erfordert zuweilen einige Durchsetzungskraft der verantwortlichen Personen. Um eine gewisse Kontinuität ausweisen zu können, sollte auch ein zeitlich festgelegter Veröffentlichungsrhythmus von beispielsweise einem halben Jahr in Betracht gezogen werden. Bei jedem Update ist die Rückwärtskompatibilität zu berücksichtigen und nur unter unumgänglichen Umständen zu brechen. Migrationsskripte können in diesem Fall einen Lösungsansatz bieten.

Obwohl SourceForge, Freshmeat, Tigris und andere *Kollaborationsplattformen* dieser Art Einschränkungen unter anderem in den Bereichen Technologie, Server-Zugriff und Gestaltung vorgeben, bieten sie dennoch zahlreiche Möglichkeiten an, unkompliziert ein neues Open-Source-Projekt zu starten und die anfängliche Zusammenarbeit zwischen den Community-Mitgliedern zu unterstützen. Oft beschließen Projektverantwortliche zu einem späteren Stadium, das Projekt auf einen individuellen Server zu transferieren. In jedem Fall ermöglichen diese Plattformen dem Open-Source-Projekt eine breite Sichtbarkeit, wodurch sie im Internet einfacher aufgefunden werden – besonders wenn es durch den intensiven Gebrauch der Plattform-Funktionalitäten eine hohe Platzierung in den Aktivitäts-Ranglisten bekommt.

Physische Begegnungen wie Präsentationen, Vorträge und Workshops an Konferenzen oder Messeveranstaltungen sind eine geeignete Möglichkeit die Funktionalitäten der Software

vorzuführen und neue Anwender und damit potentielle zukünftige Entwickler zu gewinnen. Dabei wirken soziale Kontakte, besonders zu den Hauptentwicklern, vertrauensbildend und erhöhen den Anreiz, am Projekt mitzuwirken. Der Wissenstransfer, der bei solchen physischen Begegnungen auf intensive Weise stattfindet, fördert die Zusammenarbeit auf technischer und organisatorischer Ebene und motiviert zur Weiterarbeit. Zu berücksichtigen ist dabei jedoch, dass abwesende Community-Mitglieder nach Möglichkeit nicht benachteiligt und wichtige Diskussionen und Entscheidungen immer noch online geführt werden. Bei Sprints {Definition Sprint: Ein- oder mehrtägiges Treffen von Programmierern eines Open-Source-Projekts mit dem Ziel der gemeinsamen, intensiven Weiterentwicklung der Software. Quelle: miniGuide to Zope Sprinting, 15.02.2005} sollten im Vorfeld klare Ziele gesteckt werden, um fokussiert auf die nötigen Verbesserungen hinarbeiten zu können.

Die meisten großen Open-Source-Projekte werden heutzutage von einer nicht gewinnorientierten *Trägerorganisation* geleitet. Diese juristischen Institutionen tragen zur Stabilität und Kontinuität des Open-Source-Projekts bei, indem sie sich beispielsweise um rechtliche Aspekte der Lizenzen und Marken kümmern, mehr Transparenz in Entscheidungsprozesse bieten, personelle Fluktuationen glätten, die Entwicklungsinfrastruktur bereitstellen und finanzielle Fragestellungen zentral angehen können. Gegenüber Externen repräsentiert die Trägerschaft den Ansprechpartner des Projekts und kann dadurch auch die Community-Aktivitäten im Bereich Marketing bündeln und einen Image-Aufbau bewirken. Für individuelle Entwickler bietet sie, beispielsweise bezüglich allfälliger Programmierfehler, Schutz gegenüber Anklagen, sichert die Rechte am Quellcode und kann unter bestimmten Voraussetzungen auch Hauptentwickler für gewisse Tätigkeiten finanziell entschädigen.

Den Community-Aufbau eines Projekts *international* anzugehen erfordert unter anderem, sämtliche Kommunikation auf Englisch zu führen. Dies ist für einen Nicht-Muttersprachler etwas schwieriger, bringt aber gewichtige Vorteile mit sich. Einerseits kann ein viel größeres Anwender- und damit auch Entwickler-Publikum angesprochen werden, was ermöglicht, eine multikulturelle Community aufzubauen. Andererseits sind Übersetzungstätigkeiten, wenn technisch vorgesehen, eine gute Einsteigeraufgabe für motivierte Anwender um erstmals aktiv am Projekt mitzuwirken. Des weiteren trägt eine Community mit vielseitiger Herkunft zu einer ausgeglichenen Kommunikation bei und kann bei respektvollem Umgang gewisse lokale kulturelle Phänomene abschwächen.

Neben den erläuterten sieben übergreifenden Bereichen des Community-Aufbaus fördern weitere Tätigkeiten das Wachstum, die Zusammenarbeit und die Entwicklungstätigkeit der Community-Mitwirkenden, welche lediglich spezifischen Ebenen zugeordnet werden können (Tabelle 2). So steigern herkömmliche *Marketingaktivitäten*, wie beispielsweise der Versand von Pressemitteilungen an alle bekannten Technologie-Medien, den Bekanntheitsgrad der Software und wirken Image-bildend. Des weiteren werden Mitwirkende durch ein *Credit System* motiviert, qualitativ hochwertige Beiträge für das Projekt zu liefern. Die bewusste Wahl der *Kommunikationskanäle* wie Website, Chat-System, Mailing Listen, Wiki und Foren helfen, die Diskussionen auf einige wenige Kanäle zu konzentrieren. Hingegen hat sich die antizipative Wahl von *Organisationseinheiten* der Community nicht als sinnvoll erwiesen, weshalb in diesem Bereich eher reaktives Handeln empfohlen wird. Eine Ausnahme stellt die Veröffentlichung einer *Aktivitätenliste* dar, welche anhand kurzer Beschreibungen anstehender Aufgaben Community-Mitglieder zur Mithilfe einlädt. In jedem Fall ist die *Qualität der Software* sicherzustellen, was bedingt, dass ausschließlich funktionierende und ausgereifte Software-Änderungen und -Erweiterungen ins Projekt integriert werden. Auf technologischer Ebene spielen auch die ansprechende *Benutzeroberfläche* und der einfache *Installationsprozess* wichtige Rollen und dürfen nicht als Nebensache abgehandelt werden.

6. Schlussfolgerungen

Mit der Beschreibung wünschbarer Charakteristiken von Open-Source-Communities sowie der für sie besonders wichtigen Projektverantwortlichen und der Schilderung begünstigender Voraussetzungen für neue und den Förderungsmöglichkeiten bei bestehenden Projekten wurde in dieser Arbeit ein Versuch unternommen, verschiedene für den Aufbau einer lebendigen und produktiven Open-Source-Community relevante Themen anzusprechen und zu systematisieren.

Die Aussagen basieren auf einer kondensierten und systematisierten Expertensicht auf Organisationsprobleme beim Aufbau einer Open-Source-Community, die durch eine qualitative Untersuchung gewonnen wurde. Da es sich bei den Experten um erfahrene Aktivisten aus bekannten Open-Source-Projekten handelt, gehen wir davon aus, dass deren Erfahrungen grundsätzlich repräsentativ sind und sich daher auch weitgehend auf andere Projekte übertragen lassen. Dies gilt umso mehr, als alle eingefangenen Aussagen unmittelbar plausibel erscheinen und nachvollziehbar sind. Dennoch ist grundsätzlich nicht auszuschließen, dass aufgrund der mengenmäßig eingeschränkten Datenerhebung bei lediglich je einem Projektverantwortlichen von insgesamt acht Open-Source-Projekten im Bereich von CMS und Web-Produktivitäts-Werkzeugen gewisse Aspekte überbetont und andere gänzlich ausgelassen wurden.

Es ist zu hoffen, dass durch die hier präsentierte Systematik und die formulierten Handlungsempfehlungen der eine oder andere Projektverantwortliche inspiriert und motiviert wird, die eigenen Handlungen und die Rahmenbedingungen des Projektes kritisch zu hinterfragen und sich allenfalls zu einem den Fortschritt seines Projektes förderlichen Verhalten anleiten zu lassen.

Quellen

[Eisenhardt 1989]
Eisenhardt, K., Building Theories from Case Study Research, in: Academy of Management Review, Vol. 14, No. 4 (1989), S. 532-550.

[von Krogh/Spaeth/Lakhani 2003]
von Krogh, G.,Spaeth, S.,Lakhani, R., Community, joining, and specialization in open source software innovation: a case study, in: Research Policy 32 (2003) 4, S. 1217 – 1241.

[Parnas 1972]
Parnas, D., On the Criteria To Be Used in Decomposing Systems into Modules, in: Communications of the ACM, Vol. 15, No. 12 (1972), S. 1053-1058.

[Picot/Reichwald/Wigand 2001]
Picot, A., Reichwald, R., Wigand, R.T., Die grenzenlose Unternehmung – Information, Organisation und Management. 4. Auflage, Wiesbaden: Gabler 2001, S. 410 ff.

[Raymond 1999]
Raymond, E., The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary, Sebastapol: O'Reilly and Associates 1999.

[Stürmer 2005]
Stürmer, M., Open Source Community Building, Lizenziatsarbeit Universität Bern
URL: http://opensource.mit.edu/papers/sturmer.pdf [2005-04-01].

[Välimäki 2003]
Välimäki, M., Dual Licensing in Open Source Software Industry, Systemes d'Information et Management 1/2003
URL: http://opensource.mit.edu/papers/valimaki.pdf [2005-02-10].

Autoren

Matthias Stürmer hat an der Universität Bern im Jahr 2005 sein Betriebswirtschafts- und Informatik-Studium mit der Lizenziatsarbeit „Open Source Community Building“ abgeschlossen. Ab 2006 wird er an der Eidgenössischen Technischen Hochschule Zürich im Team von Prof. Georg von Krogh eine Dissertation im Forschungsbereich von Open Source Software beginnen. Seit mehreren Jahren verwendet Matthias Stürmer als Softwareentwickler Open Source Software im Rahmen von Kundenprojekten. Außerdem hat er die erste schweizerische Open Source Messe-Veranstaltung LOTS – Let's Open the Source mitinitiiert. Im Frühling 2005 arbeitete er als Zivildienstleistender in Peru in einem Lehrerseminar, wo er Linux und andere Open Source Software einführte.

Thomas Myrach ist Direktor des Instituts für Wirtschaftsinformatik an der Universität Bern und ordentlicher Professor für dieses Fach. Er hat an den Universitäten Kiel und Bern Betriebswirtschaftslehre und Informatik studiert. Sein langjähriger Schwerpunkt sind datenbankzentrierte Informationssysteme für betriebliche Anwendungen sowie Methoden der Anwendungsanalyse. In diesem Bereich sind eine Reihe von Publikationen entstanden, darunter zwei Monographien. In den letzten Jahren beschäftigt er sich schwerpunktmäßig mit der Vision des E-Business und den Veränderungspotentialen, welche Netzwerktechnologien eröffnen. In diesem Zusammenhang sind auch einige Arbeiten über Open-Source-Software durchgeführt worden.