Matthias Stürmer

Community Building

Nach näherer Betrachtung muss man zwangsläufig feststellen, dass TYPO3 eine äußerst aktive Community besitzt. Doch welche Eigenschaften zeichnet eine derart prosperierende Community aus? Welche Aspekte sind wichtig für das zukünftige, nachhaltige Wachstum des Projekts?

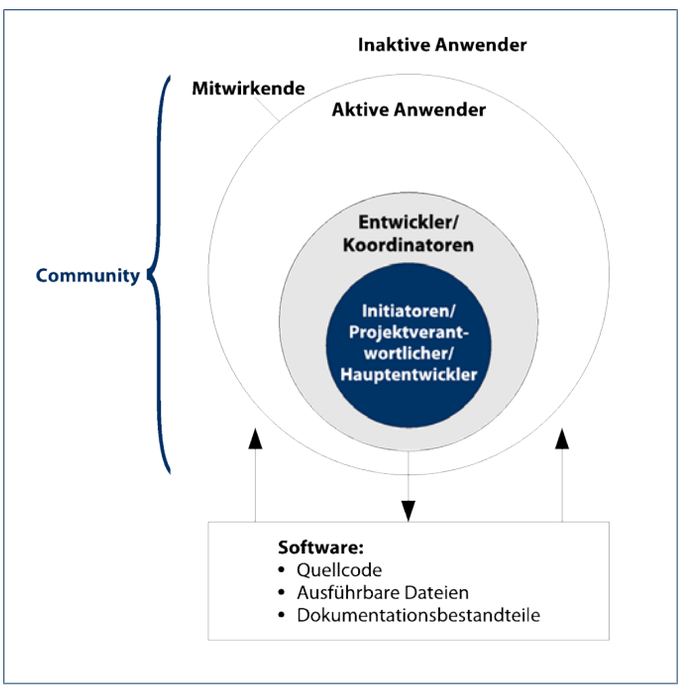
Dieser Artikel basiert auf den Ergebnissen der Lizenziatsarbeit „Open Source Community Building" [1], die zum Ziel hatte Grundlagen zu analysieren sowie Handlungsempfehlungen im Bereich des Community-Aufbaus von Open-Source-Projekten abzugeben. Dafür wurden acht Interviews mit „hochrangigen" Community-Mitgliedern von unterschiedlichen Open Source CMS und weiteren Web Applications durchgeführt und analysiert.

Die Akteure eines Open-Source-Projekts

Bevor im Folgenden die verschiedenen Aspekte einer Open-Source-Community betrachtet und beurteilt werden, empfiehlt es sich klarzustellen, wie der Begriff „Community" in diesem Artikel verstanden wird. In jeder Community lassen sich mehr oder weniger präzise verschiedene schichtartige Rollen identifizieren. Als tatsächliche Open-Source-Community wird die Gesamtheit aller Personen bezeichnet, die durch direkte oder indirekte Beiträge an der Entwicklungstätigkeit eines Open-Source-Projekts mitwirken. Dies sind ausdrücklich nicht nur die Programmierer, die den eigentlichen Source-Code entwickeln. Ausgegrenzt sind hingegen jene Personen, die die Software lediglich nutzen, ansonsten aber keinen aktiven Beitrag zur Weiterentwicklung leisten. Ausgesprochen oder unausgesprochen haben die Personen, die als Entwickler (beispielsweise von Erweiterungen) 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. Diesbezüglich wurde die Rolle des Hauptentwicklers herausgehoben und dem innersten Kreis der Community zugeordnet.

In der Studie „Open Source Community Building" wurden acht Open-Source-CMS-Projekte genauer betrachtet.

Weiterhin erfordert eine Community auch die Rolle von Koordinatoren, welche vor allem 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 Projektverantwortlichen sind dem engsten Kreis zuzuordnen.



Grob typisierte Abstraktion eines Open-Source-Projekts

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. In den Kernbereich eines Projekts sind sehr häufig auch die Projektinitiatoren einzuordnen, sofern sie sich nicht zurückgezogen haben. Die folgende Abbildung visualisiert die angesprochenen Rollendefinitionen anschaulich.

Ideale Communities

Wie zeichnet sich nun eine prosperierende Community aus? Welches sind die erstrebenswerten Charakteristiken der Community-Mitglieder? Die Interviews haben sieben Kriterien ergeben, welche ideale Communities definieren. Das heißt also, dass nicht einfach ein generelles Wachstum der im Projekt involvierten Personen erzielt werden sollte, sondern primär ein solches, das die folgenden Punkte berücksichtigt:

1. Die Community hat keinen Selbstzweck, sondern ein primäres 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 tatsäch-

lich weiter zu bringen. Ein anderes Phänomen sind Einzelpersonen und vor allem Unternehmen, 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.

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

3. Um die Langlebigkeit eines Open-Source-Projekts 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 müssen. Andererseits bringen erst verschiedenartig talentierte Menschen das umfangreiche Wissens- und Fertigkeitsspektrum mit, welches für die Leitung und Umsetzung von Open-Source-Projekten notwendig ist.

4. 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 selbst in Mailing-List-Archiven oder Wikis zu beantworten versuchen.

5. 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 Projekts beteiligt sind, müssen sie Rücksicht nehmen auf die Bedürfnisse der Community und dürfen zum Beispiel bezüglich Fragen über Rückwärtskompatibilität nicht egoistisch vorgehen.

6. 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. Verbesserungsvorschläge z. B. an der Software-Dokumentation können nützlich sein, viel wichtiger ist es jedoch, dass Personen sich über lange 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.

7. 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 an den gleichen Stellen hängen 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.

Möglichkeiten der Wachstumsförderung

Nachdem nun klar ist, wohin es mit der Community eines erfolgreichen Open-Source-Projekts gehen soll, wird im Folgenden anhand von acht umfassenden Bereichen im Detail beschrieben, wie das Wachstum einer Community gefördert werden kann. Diese Wirkungsfelder zeigen verschiedene Aspekte auf, in denen die Projektverantwortlichen konkrete Maßnahmen zur nachhaltigen Vergrößerung der Community unternehmen können. Um die zahlreichen Handlungsmöglichkeiten noch in eine zweite Dimension zerlegen zu können, sind sie in die drei Ebenen Rekrutierung, Zusammenarbeit und Entwicklungstätigkeit zusammengefasst. Aktivitäten auf der Ebene Rekrutierung beabsichtigen, weitere Projektbeitragende für die Community zu gewinnen 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, Softwarearchitektur und Technologien. Acht weitere Bereiche von Maßnahmen, die ausschließlich in einer der drei Ebenen wirken, werden ebenfalls noch kurz angesprochen. Die oben stehende Tabelle stellt einen Überblick empfohlener Aktionen der Projektverantwortlichen dar. Die Handlungsmöglichkeiten der sieben übergreifenden Bereiche werden im Folgenden kurz erläutert.

In der Software-Entwicklung ist seit langem bekannt, dass „Modularität" des Quellcodes seine 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. Ihnen wird 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. Steigt die Anzahl verfügbarer Erweiterungen, 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

Empfohlene Tätigkeiten von Open-Source-Projektverantwortlichen			
Übergreifende Bereiche			
	Rekrutierung	Zusammenarbeit	Entwicklungstätigkeit
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			
	Rekrutierung	Zusammenarbeit	Entwicklungstätigkeit
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

Danksagungen oder finanzieller Unterstützung Projektmitarbeitende ermutigt, Ressourcen für die Erstellung einer anspruchsvollen Projektdokumentation einzusetzen.

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 gewährleisten 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 sollten. Bei Sprints 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 Hinsicht auf Entscheidungsprozesse bieten, personelle Fluktuationen glätten, die Entwicklungs-Infrastruktur 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 etwaiger 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 Gebieten des Community-Aufbaus fördern weitere Tätigkeiten das Wachstum, die Zusammenarbeit und die Entwicklungstätigkeit der Community-Mitwirkenden. So steigern herkömmliche Marketingaktivitäten, wie beispielsweise der Versand von Pressemitteilungen an alle bekannten Technologie-Medien, den Bekanntheitsgrad der Software und wirken imagebildend. 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. 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.

Ausblick

Sind nun nach den erstrebenswerten Community-Charakteristiken auch die Maßnahmen aufgezeigt, welche eine derartige Entstehung unterstützen, ist es interessant, zum Schluss noch kurz einen Blick auf die TYPO3 Community im Spezifischen zu werfen. Dazu sei festgehalten, dass – auch wenn hier nicht explizit ausgewiesen – zahlreiche der oben angesprochenen Themen aus dem beinahe zweistündigen Interview mit Daniel Hinderink von der TYPO3 Community stammen. Es ist deshalb kein Zufall, dass alle der acht umfassenden Handlungsbereiche von der TYPO3 Community vollständig oder zumindest in Teilbereichen bereits heute bearbeitet werden.

Links und Literatur

[1] Open-Source Community Building: http://opensource.mit.edu/papers/sturmer.pdf

DER AUTOR

Matthias Stürmer hat an der Universität Bern Betriebswirtschaft und Informatik mit der Lizenziatsarbeit „Open Source Community Building" abgeschlossen. Im Januar 2006 hat er an der ETH Zürich im Team von Prof. Georg von Krogh eine Dissertation über Open-Source-Software und Open Innovation begonnen. Er initierte 2004 die erste Open-Source-Messeveranstaltung LOTS – Let's Open the Source in der Schweiz mit und wirkt nun im Verein /ch/open mit.