Gemeinsam statt einsam

Maximilian Caparo, Michael Dorner, Isabel Drost-Fromm, Christian Kirsch, Johannes Tigges

Wenn Regeln für die unternehmensinterne Softwareentwicklung wie strikte Abschottung der Teams zu Schwierigkeiten führen, können Open-Source-Prinzipien helfen. Diesem „Inner Source"-Vorgehen folgen mittlerweile einige große Unternehmen.

Grenzen zwischen Modulen sind in einer idealen Welt der Softwareentwicklung eindeutig und klar. Es gibt keine Überschneidungen oder Abhängigkeiten zwischen den für sie jeweils komplett verantwortlichen Teams. So weit die schöne Theorie. In der Praxis lassen sich Grenzen zwischen Modulen manchmal nur schwer definieren oder die Weiterentwicklung führt dazu, dass ehemals klare Grenzen verschwimmen.

Folglich bleiben die Entwicklungsteams nicht mehr völlig unabhängig voneinander, ihre Aufgaben beginnen sich zu überschneiden und es entstehen Abhängigkeiten von der Arbeit anderer Gruppen. Dazu kommen häufig Softwareteile, die mehr als ein Team benötigt. Das erforderliche teamübergreifende, gemeinsame Arbeiten an einem System stößt in Unternehmen jedoch oft auf Schwierigkeiten. Gründe dafür liegen in der für Firmen typischen Abschottung der Einzelprojekte voneinander und in der Idee, eine Gruppe müsse vollständig autonom über ein Teilsystem entscheiden. Diese Isolierung verhindert zudem, dass sich Nutzer an der Entwicklung der von ihnen verwendeten Produkte, etwa APIs, beteiligen. Dieses Potenzial geht verloren, und die Nutzer sind frustriert, wenn sie keine Verbesserungen anstoßen können.

Open-Source-Projekte zeigen unter anderem, wie man solche Enttäuschungen vermeiden kann. Jeder hat Zugang zu allen Informationen: Quellcode, Dokumentation und zum Entwicklungsprozess selbst, beispielsweise durch Mailinglisten oder Chats. Dadurch sind die Hürden für eine Teilnahme niedrig und die verwendeten Tools ermöglichen die Zusammenarbeit über Zeit- und Ländergrenzen hinweg.

Selbst aktiv werden statt meckern

Wer statt eines englisch kommunizierenden Programms lieber eines in seiner Muttersprache hätte, übersetzt die Texte. Wer in einem Open-Source-Projekt eine Funktion vermisst oder einen Fehler entdeckt, kann den Code einfach selbst ergänzen oder korrigieren. In größeren Projekten erfolgt so ein Eingriff häufig zweistufig: Die vorgeschlagene Änderung landet erst nach Prüfung durch einen „Maintainer" oder „Committer" in der Software. Die Beteiligung kann viele Formen annehmen: Codekorrekturen, Berichte über Fehler, Beiträge zur Dokumentation, Nutzersupport und Übersetzungen sind gleichermaßen gern gesehen. Zwar gibt es in der

Regel kein Geld für solche Beiträge, doch erhält man Belohnungen durch Anerkennung, etwa durch Namensnennung und mehr Einfluss im Projekt.

Wenn also wie geschildert Abschottung in der unternehmensinternen Softwareentwicklung Schwierigkeiten schafft, könnte Öffnung sie mildern. Diese Öffnung, genauer das Anwenden der erwähnten Prinzipien der Open-Source-Entwicklung auf unternehmensinterne Abläufe, nennt sich Inner Source. Es hat sich bereits in einigen Unternehmen etabliert, unter anderem bei der Robert Bosch GmbH, der US-Supermarktkette Walmart und der deutschen Europace AG.

Die wichtigsten Prinzipien von Inner Source lauten:

- „Offen, auffindbar und transparent“: Alle Projektartefakte müssen unternehmensweit zugänglich und durchsuchbar sein.
- „Beitragen statt Anforderungen stellen“: Projektnutzer gelten als potenzielle Mitstreiter.
- „Schriftlich vor mündlich“: Damit sich jeder jederzeit beteiligen kann, muss man Entscheidungen nachvollziehbar schriftlich treffen.
- „Fehler sind okay“: Da alle Entscheidungen unternehmensweit nachlesbar bleiben, bieten Fehler eine Chance zur Verbesserung.
- „Jedes Mitwirken wird belohnt“: Nicht nur Quellcode, sondern auch Support, Dokumentation und Marketing sind wichtig für ein Projekt und verdienen Anerkennung.
- „Projektgedächtnis“: Alle Dokumente und die Kommunikation in einem Projekt bleiben erhalten und durchsuchbar. Links ändern sich nicht, sodass die Informationen langfristig erreichbar sind.

Durch den höheren Anteil schriftlicher Kommunikation entsteht quasi nebenbei ein Grundstock „passiver“ Dokumentation, etwa zum Einarbeiten neuer Mitarbeiter. Drei Beispiele mögen deren sinnvollen Einsatz verdeutlichen. So kann erstens ein crossfunktionales Team dank passiver Dokumentation auf Deployment- und Produktionsprobleme zurückgreifen und neuen Kollegen daran Wege zu deren Lösung zeigen.

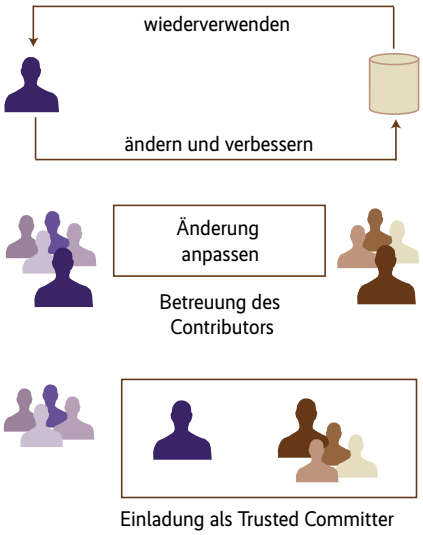
Zweitens kann man Wortmeldungen in einer Diskussion etwa zu neuen Funktionen verlinken. Öffentliche Beispiele dafür finden sich in Projekten der Apache Software Foundation, beispielsweise zum Thema „How do I manage my volunteer energy“. Drittens stellen sich beim Refactoring Fragen nach dem Hintergrund bestimmter Designentscheidungen. Ein durchsuchbares, verlinktes Archiv hilft bei deren Beantwortung. Trotzdem sollte klar sein, dass weiterhin ein Bedarf für klassische Dokumentation besteht. Passive Dokumentation hilft jedoch, sie zu erstellen und Lücken zu füllen.

Jeder darf mitmachen, nur einige entscheiden

Inner Source übernimmt nicht nur Verfahren der Open-Source-Bewegung, sondern verteilt auch die Entwicklungsaufgaben auf zwei Rollen: Ein „Contributor“ arbeitet bei Inner Source aktiv mit und ein „Trusted Committer“ bestimmt die Richtung des Projekts.

Contributor kann jeder Mitarbeiter des Unternehmens werden, indem er einen Beitrag (Contribution) zu einem Inner-Source-Projekt liefert. Dazu gehört in erster Linie Code, ebenso wichtig aber sind Beiträge zum Design, zum Nutzersupport oder zur Dokumentation.

Trusted Committers haben Schreibrechte für alle Artefakte und sind für das Betreuen neuer Contributors sowie deren Ernennung zu Trusted Committers verantwortlich. Sie ähneln insofern dem Maintainer oder Committer in Open-Source-Projekten. Neben der Weiterentwicklung des Projekts verantworten sie die Umgebung, in der sie stattfindet. Dazu gehört, Regeln für die Zusammenarbeit zu pflegen, neuen Code zu prüfen und Contributors zu betreuen. Trusted Committers wirken federführend bei Refactorings und Modularisierungen, beteiligen sich an Diskussionen rund um ihr Projekt und suchen nach Kollaborationsmöglichkeiten.



Trusted Committers integrieren Änderungen im Code, betreuen Contributors und ernennen sie zu neuen Trusted Committers.

Damit haben sie viel mehr Verantwortung als klassische Entwickler. Das kann zu Konflikten führen, etwa wenn teamübergreifende Tätigkeiten nicht mit den individuellen oder teamweiten Vorgaben zusammengehen. Insbesondere zu Beginn der Inner-Source-Arbeit entstehen typische Reibungen. In der klassischen internen Softwareentwicklung bekommt man häufig bei Beginn der Arbeit an einem Projekt Schreibrechte dafür. Bei Inner Source ist das nicht mehr nötig. Denn Contributors reichen dort zunächst ihre Beiträge ein, die Trusted Committers prüfen und gegebenenfalls akzeptieren. Das irritiert oft diejenigen, die sich beteiligen wollen. Deshalb sollte ihnen der Einstieg möglichst leichtfallen.

Niedrige Hürden erleichtern die Mitarbeit

Zwar lässt sich Inner Source mit verschiedenen Hilfsmitteln umsetzen. Jedoch vereinfachen Tools wie GitHub/GitLab kombiniert mit unstrukturierter Kommunikation (IRC, HipChat, Slack oder Mailinglisten) den Einsatz. Geeignete Werkzeuge alleine garantieren den Erfolg solcher Initiativen jedoch nicht: „Wir haben unser Projekt in ein privates GitHub-Repo verschoben. Jeder im Unternehmen darf darauf zugreifen. Wir würden uns freuen, wenn Leute uns Pull Requests schicken – aber es kommt einfach nichts.“ So oder so ähnlich sind die Reaktionen, wenn an Autonomie gewöhnte Teams erstmals mit Inner Source in Kontakt kommen. Einfach nur den Quellcode zu ver-

iX-TRACT

- Inner Source bezeichnet das Anwenden von Prinzipien der Open-Source-Entwicklung auf unternehmensinterne Softwareprojekte. Dabei erhalten alle Mitarbeiter lesenden Zugriff auf Code, Dokumentation und den Verlauf des Entwicklungsprozesses.
- Sie können Beiträge dazu liefern und erhalten dafür Anerkennung etwa durch Namensnennung. Schreiben dürfen jedoch nur sogenannte Trusted Committers.
- Wenn sich ein Team am Modul eines anderen beteiligt, müssen die Regeln für Codekonventionen, Architektur et cetera sorgfältig dokumentiert und besprochen werden.

öffentlichen und abzuwarten führt wie in der Open-Source-Welt meist zu wenig.

Eine Faustregel aus Open-Source-Erfahrungen besagt, dass von 100 Nutzern nur einer aktiv und im Projekt sichtbar wird – sei es mit Bug Reports, Korrekturen oder beim Support anderer Nutzer. Auf 100 dieser Aktiven kommt wiederum lediglich einer, der länger dabeibleibt und zum Committer aufsteigt. Auch im Unternehmen braucht es zuerst Nutzer. Haben sie Interesse an Änderungen und Verbesserungen, ist das die beste Voraussetzung für Inner-Source-Beiträge. Insbesondere wenn Anwender neue Funktionen wünschen, hilft es, das Vorhaben in ein Inner-Source-Projekt umzuwandeln.

Wie sich die Hürden für Beiträge niedrig halten lassen, zeigt ein Blick auf Open-Source-Projekte. Dort enthält eine Readme-Datei Entwicklerdokumentation und erklärt, wie die Software zu nutzen ist. Unternehmen geben dieses Wissen oft beim Onboarding weiter. Sinnvollerweise sammeln und notieren sie es beim Einarbeiten neuer Kollegen.

Unternehmenstypisch, aber im Open-Source-Umfeld selten ist die agile Softwareentwicklung. Ihre Prinzipien lassen sich mit dem Inner-Source-Vorgehen kombinieren. So hilft das Prinzip, alle Entscheidungen schriftlich festzuhalten, die agile Vorgabe „individuals and interactions over processes and tools“ umzusetzen.

Der häufig vorgebrachte Einwand, Inner Source und Scrum seien wegen der dort erforderlichen täglichen Besprechung (daily standup) nicht kompatibel, lässt sich einfach entkräften. Denn alle Beteiligten lassen sich auf dem Laufenden halten, wenn man diese Kommunikation zeitlich entzerrt und in ein schriftliches Medium zieht (Slack IRC oder Mailinglisten). Obwohl dadurch nicht mehr sämtliche Status-Updates sofort allen zur Verfügung stehen, bringt es mehr Flexibilität: Kollegen können ihre Arbeiten abschließen, bevor sie am Stand-up teilnehmen. Das berücksichtigt unterschiedliche Arbeitsrhythmen.

Obwohl agile Teams daran gewöhnt sind, flexibel auf Änderungen zu reagieren, müssen Unternehmen nicht agil arbeiten, um Inner Source erfolgreich einzusetzen. Vor allem in traditionell operierenden Firmen können Inner-Source-Projekte ein erster Schritt zu mehr Transparenz und Flexibilität sein. Sie beginnen oft als dedizierte Vorhaben, an denen Interessierte sich beteiligen und zu denen sie ihr Wissen und ihre Erfahrung beisteuern. Dabei sammelt das Unternehmen Erfahrungen mit Inner Source. Bei hinreichender Sichtbarkeit des Experiments und veröffentlichten positiven Erfahrungen können solche Projekte als Leuchttürme fungieren, von denen aus sich die Arbeitsweise in der Organisation verbreitet.

Wie Teams zusammenarbeiten können

Häufig geht es in Unternehmen nicht nur darum, einzelne zur Mitarbeit an Projekten zu motivieren, sondern die Zusammenarbeit mehrerer Entwicklerteams zu organisieren. Das ist etwa dann nötig, wenn Team A einen Service entwickelt, den Team B nutzen möchte. Dazu fehlt ihm jedoch ein Detail. Folgt Team A dem Inner-Source-Ansatz, können die Mitglieder von B den Code auschecken und anpassen.

Was sich ideal anhört, führt jedoch in der Praxis manchmal zu Schwierigkeiten, wenn die Änderungen nicht zur Architektur, zum Codestil oder zu den Best Practices von Team A passen. Um diese Schwierigkeiten zu vermeiden, müssen Teams einerseits ihre Vorgaben möglichst gründlich dokumentieren, sodass sich andere daran halten können. Andererseits sollten Entwickler, die Änderungen oder Ergänzungen wünschen, frühzeitig klären, ob und wie die in die vorhandene Software passen. In Open-Source-Projekten hat sich die Datei CONTRIBUTING.md bewährt. Sie enthält Anforderungen an Beiträge technischer Natur ebenso wie Hinweise zum Vorgehen: Zeiträume, Architekturdiskussionsverfahren und -orte. Das hilft dem Projekt und seinen Trusted Committers sowie den Contributors, Kollisionen bei der Zusammenarbeit zu verhindern.

Bei ersten isolierten Experimenten mit Inner Source treten oft Schwierigkeiten auf. So beklagen Teams, die bisher kaum mit asynchronen Code-Reviews gearbeitet haben, eine langsamere Entwicklung. Hier ist es sinnvoll, dass sich alle Beteiligten über die erwarteten Reaktionszeiten verständigen. Obwohl Inner Source das entfernte Arbeiten vereinfacht oder überhaupt erst ermöglicht, bleibt es spätestens bei der Überbrückung von mehr als sechs Zeitzonen nicht aus, dass Feedback erst am Folgetag vorliegt.

Insbesondere wenn für eine Komponente verantwortliche Teams Beiträge von anderen annehmen sollen, kommt es anfangs häufig zu Vorbehalten: „Dann sind wir für die Fehler der anderen verantwortlich.“ Hier ist eine Frist hilfreich, innerhalb derer die Contributors für Korrekturen ihrer Beiträge verantwortlich bleiben.

Beteiligen sich Mitarbeiter nicht nur in ihren eigenen Teams, sehen ihre direkten Kollegen einen Teil ihrer Leistung nicht. In diesem Fall kann es helfen, wertvolle Beiträge für andere im Unternehmen hervorzuheben. Dazu dienen meist firmenspezifische Kanäle. Ein Beispiel sind Scrum Boards: Für höhere Transparenz ist es sinnvoll, Beiträge zu anderen Projekten im eigenen Board darzustellen.

Generell empfiehlt es sich, die Arbeit nach Inner-Source-Prinzipien durch regelmäßige Retrospektiven zu begleiten. Veröffentlicht man die Erfahrungen und Erkenntnisse über die Grenzen des eigenen Teams hinaus, kann man vermeiden, dieselben Lösungen mehrfach zu entwickeln.

Fazit

Inner Source hilft, die Grenzen des eigenen Teams aufzuweichen. Es stellt Mittel und Methoden bereit, mit denen Mitarbeiter über ihr Team hinaus wirksam werden können – bis hin zum organischen Wachstum von Communitys und Komponenten. Damit einhergehend bekommen Mitarbeiter Einblicke in andere Teams. Sie können von deren Erfahrungen profitieren und von deren Wissen lernen.

Dabei verfolgt Inner Source nicht das Ziel, etablierte Prozesse zu ersetzen. Vielmehr ermöglicht es, sie zu ergänzen und Schwierigkeiten gezielt anzugehen.

(odi@ix.de)

Maximilian Caparo

ist wissenschaftlicher Mitarbeiter in der Open Source Research Group an der Friedrich-Alexander-Universität Erlangen-Nürnberg und forscht zu Inner Source, Open Source und kollaborativer Softwareentwicklung.

Michael Dorner

ist wissenschaftlicher Mitarbeiter an der Professur für Open-Source-Software an der Friedrich-Alexander-Universität Erlangen-Nürnberg und forscht zu Qualitätssicherung in der Softwareentwicklung.

Isabel Drost-Fromm

ist Java-Entwicklerin, Mitglied der Apache Software Foundation und Open Source Strategist bei der Europace AG.

Christian Kirsch

ist freier IT-Journalist.

Johannes Tigges

ist Informatiker und arbeitet bei HERE Technologies am Thema Inner Source sowie als Dozent an einer Fachhochschule.