**Gemeinsam statt einsam**

* 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 etc. sorgfältig dokumentiert und besprochen werden.
* „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.
* Das Fachwissen wird „dezentralisiert“ – falls ein Teammitlgied das Projekt oder das Unternehmen verlässt, bleibt der Informationsbestand unverändert.
* Barrierefreiheit

Obwohl der InnerSource-Ansatz die Produktivität innerhalb des Unternehmens deutlich steigern kann, muss man ihn vorsichtig herangehen. Denn eine Größe passt nicht allen, können bisher unbedenkliche Konflikte entstehen

* Nicht jedes Open-Source-Vorgehen ist für ein bestimmtes Unternehmen passend. Oft arbeiten Open Source Communities basierend auf ihren spezifischen Anforderungen. Daher müssen vorsichtig nur die passenden Ansätze ausgewählt werden.
* Ein Unternehmen, welches nach dem klassischen Enterprise-Ansatz funktioniert hat unterschiedliche Prioritäten zu den Open Source Communities. Übernimmt ein solches Unternehmen das InnerSource-Vorgehen, bedeutet das, dass die Betriebskultur auch „umgestellt“ wird. Insbesondere werden sich die Barrieren zwischen die Praktiken einzelnen Teams sichtbar machen, da sie bisher nur in ihrer begrenzten Untergruppe aktiv waren. Diese Praktiken müssen so angepasst werden, sodass das Modell funktionieren kann, was an sich von dem Unternehmen und die Mitarbeiter selbst abhängt

[Source](https://engineering.salesforce.com/inner-sourcing-the-ups-and-downs-3d443d5417b9)

**Jeder darf mitmachen, nur einige entscheiden**

* Entwicklungsaufgaben werden auf 3 Rollen verteilt:
  + „Contributor“: Jeder Mitarbeiter im Unternehmen der einen Beitrag (Contribution) zum Inner-Source-Projekt liefert. In erster Linie Code – ebenso wichtig sind Beiträge zum Design, Nutzersupport oder der Dokumentation.
  + „Trusted-Committer“: Schreibrechte für alle Artefakte; Betreuung neuer Contributors, deren Ernennung zu Trusted-Commiter. Neben der Weiterentwicklung des Projekts verantworten sie die Umgebung, in der die Weiterentwicklung stattfindet.
    - Regeln für Zusammenarbeit definieren und zu pflegen
    - Neuen Code prüfen und Contributors betreuen
    - Suche nach Kollaborationsmöglichkeiten
    - Diskussionen rund um das Projekt
    - Wirken federführend bei Refactorings und Modularisierungen
  + Product Owner - Der Product Owner (PO) ist, wie bei den agilen Entwicklungsmethoden, für die Definition und Priorisierung von Anforderungen und deren Implementation verantwortlich. Er kommuniziert oft mit dem Trusted Committer (z.B. um sicherzustellen, dass eine Anforderung oder Beitrag im Endprodukt implementiert wird). Daher wird die Rolle oft von dem Trusted Committer übernommen, vor allem in kleineren Unternehmen.

🡪 Kann zu Konflikten führen:

Entgegen der klassischen Programmierung, bekommen Contributors nicht direkt Schreibrechte am Projekt, sie schreiben ihre Ideen in Beiträge die durch die Trusted Committers geprüft und ggf. akzeptiert werden. Ein beliebtes Werkzeug für die Zusammenarbeit zwischen den Contributor und Committer ist der Pull Request (auch Merge Request genannt), welches das Review und Feedback des Beitrags von allen Beteiligten ermöglicht, bevor er in der Codebasis einfließen darf.

Code Ownership

Hierfür existieren 3 Konzepte:

**Strong Code Ownership** – Die Code-Basis wird in Modulen aufgeteilt und jedes Modul wird an einem Entwickler fest zugewiesen (Owner). Die Entwickler dürfen nur an ihren eigenen Modulen Änderungen vornehmen. Falls eine Änderung für einen fremden Modul erwünscht ist, muss diese vom Owner durchgeführt werden.

**Weak Code Ownership** – ebenfalls hier wird der Code in Modulen aufgeteilt, welche dann an den Entwicklern zugewiesen werden. Der Unterschied besteht darin, dass jeder Entwickler fremde Module bearbeiten kann. Der Owner übernimmt aber die Verantwortung für die Verfolgung der Änderungen.

**Collective Code Ownership** – bei diesem Schema wird auf individuelles Modulbesitz verzichtet. Das Team als Ganzes ist Besitzer der Code-Basis und Änderungen dürfen von jedem vorgenommen werden.

[Source](https://martinfowler.com/bliki/CodeOwnership.html)

Unterstützende Software & Werkzeuge

**Kommunikation** – Manchmal müssen die Entwickler schnell eine Entscheidung treffen. Der E-Mail-Austausch reicht selbst nicht aus, besonders in geografisch verteilte Entwicklerteams, da viele Beiträge in Echtzeit entstehen. Dazu brauch man Werkzeuge, welche sowohl synchrone als auch asynchrone Kommunikation erlaubt. Das meistbenutzte Plattform ist Slack, es sind aber auch zahlreiche Alternativen verfügbar.

**Ticketsystem** – solche Systeme sind bei der Entwicklung von Software zu einer Norm geworden. Sie beinhalten Information für alle Fehler und Bugs, darüber hinaus auch über Backlog, aktueller Projektzustand etc. Die gängigen Werkzeuge sind GitHub Issues und JIRA

**Versionskontrolle** – auch ein Must-have Werkzeug für jedes Softwareprojekt, hier aber mit einem wichtigen Unterschied. Beiträge zum Projekt erfolgen sowohl innerhalb als auch außerhalb des Entwicklungteams, daher ist die zentralisierte Versionskontrolle mit größerem Bearbeitungsaufwand verbunden. Die verteile Versionskontrolle ist daher die bevorzugte Option, da sie ein gewisser Grad an Unabhängigkeit zwischen Beiträge gewährleistet. Solche Features sind auf das Plattform GitHub verfügbar, wie z.B. Forks und pull requests.

**Mailing-Listen und Foren** – bei der Zusammenarbeit können immer Unsicherheiten entstehen, daher sind Foren und Mailing-Listen bei kollaborativer Entwicklung beliebt, da sie Diskussion zwischen den Mitarbeitern schaffen können. Foren, wie z.B. Stack Overflow, sind sowohl für einfache Fragen als auch für komplizierte Anliegen hilfreich. Die Threads von Mailing-Listen können zeigen, welches Projektteil mehr Arbeit benötigt oder wo sich die Aufgaben aufeinanderstapeln. Diese Threads kann man mit anderen Tools kombinieren, um genaue Kennzahlen für das ganze Projektentwicklungsprozess zu ermitteln, wie z.B. Zeitaufwand von der Idee bis zur Bereitstellung, ausgeführte Iterationen, beteiligte Personen etc.

[Source](https://blog.bitergia.com/2017/01/18/open-source-community-tools-for-inner-source-projects/)