Open Source, ein Erfahrungsbericht: In kleinen Schritten von den ersten Contributions zum Maintainer

Georg Berky und Sandra Parsick

Die Open-Source-Welt erscheint oft riesig und ohne Erfahrungen gesammelt zu haben, ist es für Neulinge nicht offensichtlich, wie sie mit Open-Source-Beiträgen anfangen können. Mit Git als Werkzeug und GitHub als Plattform ist es für Einsteiger um einiges einfacher geworden. Obwohl es für viele immer noch schwierig erscheinen mag, war unsere Erfahrung, dass es leichter ist, als es am Anfang erscheint. Es gibt inzwischen viele Projekte, bei denen auch Neulinge aushelfen und wertvolle Beiträge leisten können.

Georg: Als ich meinen ersten Job in Vollzeit in der IT anfing, war das Erste, das mir auffiel, die schiere Menge an Bibliotheken, die es im Vergleich zu den Projekten gab, die ich aus Studium, Nebenjobs und vom Lehrstuhl kannte. Die meisten davon waren Open-Source-Bibliotheken und als Neuling kam es mir so vor, als säßen dort Leute mit jahrzehntelanger Erfahrung, die Software für alle Softwareentwickler da draußen schrieben – und diese Software war so gut, dass eine Menge Projekte sie nutzen. Ich wusste so gut wie nichts über Open Source, die Ökosysteme, die Community und wie man selbst dazu beitragen kann. Es war damals zugegebenermaßen auch noch etwas schwerer.

Sandra: In meiner ersten Firma, bei der ich nach dem Studium 2008 anfing, wurde zu 100 Prozent auf Open Source gesetzt. Zwar existierte schon GitHub, aber es war nicht so verbreitet wie heute, sodass die meisten Projekte entweder eigene Infrastruktur betrieben oder auf Dienste wie Sourceforge *[1]* setzten. Das bedeutet für jemanden, der etwas beisteuern wollte, Patches zu erstellen und sie über Tickets einzureichen.

Wir wollen euch in diesem Artikel zeigen, wie es bei uns gelaufen ist, was uns überrascht, aber auch motiviert hat, damit weiterzumachen, und wie wir inzwischen Maintainer unserer eigenen kleinen Projekte geworden sind. Begleitet uns auf einer kleinen Zeitreise durch unsere Open Source Contributions!

2011 – 2017

Sandra: Bewusst mit Open-Source-Projekten arbeite ich seit 2008, aber ich habe erst 2011 mit den ersten Versuchen, auch was zu den Projekten beizutragen, angefangen. Ich meldete damals vor allem Bugs, die mir in der täglichen Arbeit auffielen, oder stellte Fragen auf Mailing-Listen, wenn etwas unklar war.

Bei dem einen oder anderen Bug konnte ich auch einen Patch für einen Bugfix beisteuern oder die Dokumentation verbessern. Das war zum Teil recht mühselig, da nicht alle Projekte schon auf GitHub waren und deshalb mit Patch-Dateien gearbeitet werden musste *[2]*. Wer das noch nie gemacht hat: Ein Patch ist eine Datei, die das enthält, was sonst in einem Git-Commit steht, also die Änderungen, die auf den Dateien im Repository ausgeführt werden sollen. Die Patches hat man damals per E-Mail verschickt oder sie an den Bug-Report im Projekt angehängt, damit der Maintainer die Änderungen von Hand in den Code übernehmen konnte.

Wenn Projekte schon auf GitHub waren, dann kamen je nach Projekt noch Formalitäten in Form von Contributor License Agreements *[3]* hinzu. Contributor License Agreements sind für Organisationen wie Apache Foundation, Eclipse Foundation oder für Projekte wie Spring Framework, bei denen Firmen dahinterstehen, wichtig, um Lizenz-Streitigkeiten zu vermeiden.

Später habe ich dann meinen eigenen Blog *[4]* angefangen, um Lösungen zu meinen alltäglichen Problemen zu dokumentieren. Im Blog habe ich so immer mehr Material gesammelt und fing irgendwann an, Vorträge über Open-Source-Projekte zu halten sowie darüber, wie diese meinen Entwickler-Alltag vereinfachten.

2018

Im Jahr 2018 waren wir beide im selben Projekt und uns fiel beim Bearbeiten der aktuellen Story auf, dass uns ein Feature in AssertJ *[5]* fehlte. Wir mussten sicherstellen, dass ein String eine bestimmte Länge hatte, und AssertJ hatte ein solches Feature noch nicht *[6]*.

Wie der Zufall es wollte, hatten wir als Co-Organisatoren der Softwerkskammer Ruhr *[7]* ohnehin einen Hackergarten *[8]* geplant. Ein Hackergarten ist ein eintägiges Event, bei dem man sich in ungezwungener Runde trifft, um gemeinsam in kleinen Gruppen Contributions zu Open-Source-Projekten zu machen. Häufig sind auch Maintainer von Projekten anwesend, die vorstellen, was die Projekte tun und wobei sie dringend Hilfe gebrauchen könnten. Auch bei unserem Hackergarten in Dortmund waren einige Maintainer dabei, so auch Karl Heinz Marbaise für das bekannte Build-Tool Maven von Apache, der weltweit größten Open Source Foundation. Während des Hackergartens gab es auch einige Contributions der Teilnehmer zu diesem Projekt.

Wir erstellten beim Hackergarten also ein Issue im GitHub-Projekt von AssertJ, das das fehlende Feature beschrieb, und machten uns im Laufe des Tages an die Arbeit. Der Pull Request (PR) dazu war recht schnell erstellt, aber das war erst der Anfang.

Joel Costigliola, der Maintainer von AssertJ und inzwischen Java-Champion, hatte noch einiges an Feedback für uns. Die Dokumentation passte an einer Stelle noch nicht. Wir hatten den Javadoc einer anderen Methode übernommen und nicht angepasst. An einer anderen Stelle schlug Joel ein besseres Beispiel für die Benutzung des Codes vor. Die Fehlermeldung einer fehlgeschlagenen Assertion war noch nicht klar genug. Joel lobte aber auch gute Beispiele und sprechende Parameternamen, die wir im Code verwendet hatten. Nach mehreren Feedbackrunden und Verbesserungen akzeptierte er unseren Pull Request und wir waren glücklich, unseren ersten PR in nur einem Tag Arbeit fertig gestellt zu haben. Als Bonus obendrauf konnten wir den Code am Montag darauf auch schon im Projekt nutzen.

Die Betreuung von Joel war super. Wir wussten immer, in welche Richtung es mit unserem Feature gehen sollte, und er konnte gut vermitteln, was wir noch verbessern mussten. Sein Feedback zu Stil und Dokumentation war ebenfalls sehr gut. Das motivierte uns, zwei weitere Feature Requests zu erstellen. Einer wurde aus guten Gründen abgelehnt. Für den zweiten erstellten wir einen weiteren Pull Request, der akzeptiert wurde. Seitdem kann man in AssertJ prüfen, dass die Fehlermeldung einer Exception einem regulären Ausdruck entspricht oder einen bestimmten String nicht enthält.

Etwas später stießen wir auf ein offenes Problem im Projekt Commons-Lang von Apache. Das Feature konnten wir auch gut im Projekt gebrauchen, also dachten wir uns: Was bei AssertJ gut lief, könnte auch bei Commons-Lang funktionieren. Wir erstellten wieder einen Pull Request, aber – und auch damit muss man rechnen – der Pull Request wurde nicht gemergt, weil jemand anderes das Ticket gleichzeitig bearbeitet hatte.

Kurz nach dem Hackergarten fand die Aktion „Hacktoberfest" *[9]* im Oktober statt. Sandra machte mit und „contributete" vermehrt Dokumentationen zu verschiedenen Open-Source-Projekten.

Hacktoberfest ist eine Initiative von Digital Ocean, die seit 2013 Open-Source-Projekten helfen möchte, die Beteiligung an ebendiesen zu erhöhen. Dabei wird im Monat Oktober dazu aufgerufen,

mindestens vier Pull Request für ein beliebiges Projekt auf GitHub zu erstellen. Leider führte das Event in den letzten Jahren vermehrt zu Spam-Pull-Request, da die Teilnehmer dadurch ein T-Shirt ergattern konnten. Digital Ocean versucht dagegenzusteuern, sodass sich Open-Source-Projekte seit 2020 aktiv beim Event anmelden müssen, damit die Pull Requests für das T-Shirt gewertet werden.

August 2018 bis Januar 2019

Auch Vorträge und Workshops über Open-Source-Tools und -Bibliotheken sind Open-Source-Arbeit. Sie bringen neue Ideen in die Community, zeigen, welche Ansätze schon zum Erfolg oder Misserfolg geführt haben, und fördern den Austausch unter den Projekten. Ohne die Vorträge und Verbindungen zu anderen Mitgliedern unserer Communities hätten wir das Rad schon in mehreren Fällen zweimal erfunden, weil wir die passende Bibliothek für unser Problem noch nicht kannten.

Georg: In der Zwischenzeit hatte ich Groovy als Skriptingsprache für mich entdeckt. Es fing mit einigen Lightning Talks, die ich bei Treffen der Softwerkskammer hielt, an und entwickelte sich weiter zu Talks auf nationalen und internationalen Konferenzen.

Groovy *[10]* ist eine dynamische Programmiersprache auf der JVM, kann aber auch direkt als Interpreter für Shellskripte benutzt werden, als Alternative für *#!/bin/bash* am Skriptanfang. Verbunden mit Grape als Dependency-Manager steht dann das gesamte Java- und Groovy-Ökosystem an Bibliotheken für die eigenen Skripte zur Verfügung. Details dazu findet ihr in der Java aktuell 05/2019 *[11]* oder auf meinem Blog *[12]*.

Den ersten zehnminütigen Talk dazu gab ich beim Jahresabschlusstreffen der Softwerkskammer im Ruhrgebiet während einer Lean-Coffee-Session, bei der alle Teilnehmer ihr Lieblingstool kurz vorstellen konnte. Groovy fand guten Anklang, weil es voll syntaxkompatibel zu Java ist und viele Java-Entwickler sich freuten, nicht mehr in der für sie manchmal umständlichen Schreibweise von Bash entwickeln zu müssen. Groovy unterstützte auch schon vor Java 8 funktionale Ansätze, sodass es für viele eine Gelegenheit war, diese Konzepte auszuprobieren, als viele Projekte noch auf Java 7 liefen.

2020 und Corona: Dependency-Update-Plug-in während Corona

Wir wollten wieder mal zusammen hacken und konnten uns in den User Groups nicht treffen, weil sich gerade alle im Lockdown befanden. Sandra entdeckte bei der Recherche für einen Workshop das Dependency-Update-Plug-in, das von Oliver Weiler *[13]* ins Leben gerufen wurde, und wollte es vorantreiben, da sie die Idee super fand. Sie schlug vor, daran etwas zu machen. Georg hätte ein solches Plug-in auch schon öfter im Projekt gebrauchen können und war begeistert. Ein netter Nebeneffekt für beide: Wir konnten uns mit Kotlin auseinandersetzen, worin das Projekt geschrieben ist.

Das Plug-in ist die „Enterprise"-Version von Dependabot *[14]*, das man in geschlossenen Projekten leider oft aufgrund von Regularien nicht einsetzen kann. Es erstellt pro Maven Dependency einen Branch, in dem nur die Version der Dependency aktualisiert wird, und pusht die Branches automatisch, damit die CI durch Builds prüfen kann, ob ein Dependency Update den Build bricht.

Als wir anfingen, verwendete das Projekt JGit, eine Git-Implementierung in Java, für alle Operationen auf dem Repository. Den ersten Beitrag wollten wir dadurch leisten, dass wir nicht nur JGit, sondern auch das auf der Maschine des Benutzers installierte Git unterstützen, wo Zugangsdaten, SSL-Schlüssel, Author und Ähnliches meistens schon vorkonfiguriert sind und dadurch nicht nochmal in die POM des Projekts geschrieben oder beim Plug-in-Aufruf übergeben werden müssen. Das Plug-in sollte also einen JGit Provider und einen Native Git Provider bekommen, den man über die POM konfigurieren kann.

Die Arbeit an diesem Feature zog sich über das ganze Jahr 2020 hin und brachte Abwechslung und Freude über den Fortschritt in den von Homeoffice und Pandemie geprägten Alltag. Wir waren beide froh, dass wir uns wenigstens virtuell treffen konnten und im Laufe des Jahres nahm der Provider immer mehr Gestalt an, bis wir an den Punkt kamen, an dem wir auch den Plug-in-Aufruf mit Maven und einer echten POM testen mussten, um die verschiedenen Konfigurationsszenarien durchzuspielen.

Wie schreibt man einen Integrationstest für ein Maven-Plug-in? Sollen wir Maven über einen Process Builder aufrufen und den Output parsen? Instabil, weil alles kaputt geht, sobald sich eine Formulierung der Logmeldung ändert. Rückgabecodes? Dazu fanden wir keine Doku – und sind die auf Windows, Linux und Mac die Gleichen? Wir schienen in eine Sackgasse gelaufen zu sein.

Karl Heinz to the Rescue: Die Maven IT Extension

Zum Glück erinnerten wir uns zurück an den Hackergarten, bei dem Karl Heinz Marbaise *[15]* das Maven-Projekt vorgestellt hatte. Maven selbst bietet ein riesiges Ökosystem an Plug-ins und ein Großteil der Arbeit des Build Tools wird von Plug-ins erledigt. Es war also kein Wunder, dass dort bereits jemand ein Tool für Plug-in-Tests entwickelt hatte, aber unser Glück war, dass es Karl Heinz selbst war. Seine Maven IT Extension *[16]* war wie gemacht für unseren Use Case.

Wir hatten sogar doppeltes Glück, denn Karl Heinz hatte sofort Lust, uns beim Dependency-Update-Plug-in zu helfen. Jetzt waren wir zu dritt und machten uns daran, das Plug-in mithilfe von Karl Heinz zu testen. Dadurch fanden wir weitere Use Cases für Karl Heinz' Plug-in und beschlossen, einen Abstecher in dessen Projekt zu machen, um später mit weniger Aufwand und höherer Lesbarkeit unsere Tests schreiben zu können.

Die Maven IT Extension ist eine Extension für JUnit 5 und stellt dem Benutzer einen Rahmen zur Verfügung, Maven-Projekte mit POM und weiteren Dateien als Ressourcen ins Projekt zu legen, Maven Goals auf dem Projekt auszuführen und die Ergebnisse von Setup und Plug-in-Lauf programmatisch mit Java zu untersuchen. Für jede Testmethode wird das Projekt in ein eigenes Zielverzeichnis kopiert, bevor Maven darauf aufgerufen wird. So bleiben die Tests isoliert und auch parallel ausführbar.

Wir erweiterten zusammen die Extension um die Möglichkeit, einfacher auf das Zielverzeichnis zuzugreifen, das für das Dependency-Update-Plug-in zum Git Repo werden musste. Wir wollten prüfen, ob die richtigen Branches angelegt wurden.

Durch die Zusammenarbeit mit Karl Heinz lernten wir noch einiges über die Interna von Maven, konnten die Tests für unser Plug-in verbessern und außerdem als echte Benutzer Feedback für die Maven IT Extension geben. Beide Projekte profitierten von der Zusammenarbeit und außerdem hatten wir viel Spaß beim wöchentlichen Entwickeln im Lockdown.

2021

Insgesamt ein Jahr später waren wir dann mit dem Native Git Provider und allen Tests fertig und bereit für den finalen Pull Request. Was dann kam, überraschte uns jedoch: Oliver bat uns, das Projekt komplett zu übernehmen, weil er weniger Zeit dafür hatte als wir. Wir mussten erst mal über das Angebot nachdenken, sagten dann aber zu und zogen das Projekt in Georgs GitHub-Account *[17]* um.

Wir waren also plötzlich Maintainer eines Open-Source-Projekts und dadurch kamen neue Aufgaben auf uns zu. Wir mussten einen automatisierten Build einrichten, Code Coverage messen, veraltete Dependencies aufspüren und die Lieferung nach Maven Central organisieren. Maven Central, unser Projekt landet in Maven Central… Das fühlte sich alles surreal an, aber zum Glück hatten wir wieder Hilfe von Karl Heinz, der das als Open-Source-Veteran schon öfter gemacht hatte.

Georg brauchte als Inhaber des Projekts in GitHub einen Account bei Sonatype. Für diesen wird eine Maven GroupId *io.github.georgberky* freigeschaltet, indem man seinen GitHub-Account verifiziert. Ab dann kann man GnuPG-signierte Artefakte mit dieser GroupId und den Sonatype Credentials ins Staging Repository pushen. Dort laufen noch ein paar Tests, etwa, ob unsichere Dependencies vorhanden sind. Wenn alle Tests bestanden sind, wird das Artefakt nach Maven Central befördert.

Als Co-Maintainer bekamen Karl Heinz und Sandra noch Push-Rechte auf die Unter-GroupId für das Maven-Plug-in *io.github.georgberky.maven.plugins.depsupdate*. Damit können sie das Plug-in pushen, aber nicht alles unter Georgs GroupId veröffentlichen.

Ende Mai schlossen wir die Arbeiten am Native Git Provider im Rahmen des „hack-commit-push"-Events *[18]* ab, was uns ermöglichte, die Bekanntheit des Plug-ins und der Extension zu vergrößern. Gegen Ende des eintägigen Open-Source-Events hatten wir unser erstes Release fertig und veröffentlichten es auf Maven Central *(siehe Abbildung 1)*.

Oliver hatte parallel ein Relocation-Release *[19]* gemacht, um den Benutzern zu zeigen, dass die nächsten Artefakte unter einer neuen GroupId zu finden sind.

Sandra: Am Anfang des Jahres half ich bei einer Java-8- auf Java-11-Migration in einem Eclipse-RCP-Projekt. Unter anderem wird das P2-Maven-Plug-in *[20]* eingesetzt, um Maven-Artefakte in ein P2-Repository bereitzustellen. P2 ist ein Repository-Format aus der Eclipse-RCP-Welt, um Dependencies in RCP-Anwendungen bereitzustellen. Bei der Java-11-Umstellung stellte sich heraus, dass das P2-Maven-Plug-in mit einigen JAR-Features nicht zurechtkam, die nach Java 8 dazu kamen, wie etwa Multi-Release JARs. Dazu kam, dass das P2-Maven-Plug-in-Projekt den Eindruck machte, dass es nicht mehr aktiv weiterentwickelt wurde. Da die Stellen, die ein Update benötigten, schnell im Code zu identifizieren waren, schrieb ich den Maintainer an und fragte nach, wie es um die Zukunft des Plug-ins aussah und wie man am besten im Projekt unterstützen könnte, da ich einfach einen Fork zu machen als letzte Möglichkeit ansah.

Ein Fork hätte mehrere Nachteile nach sich gezogen. Einmal hätte es das Update bei den Benutzern erschwert. Zum einen hätten sie über andere Kanäle mitbekommen müssen, dass ein neuer aktiver Fork existiert, zum anderen müsste der Fork unter neuen Maven-Koordinaten veröffentlicht werden.

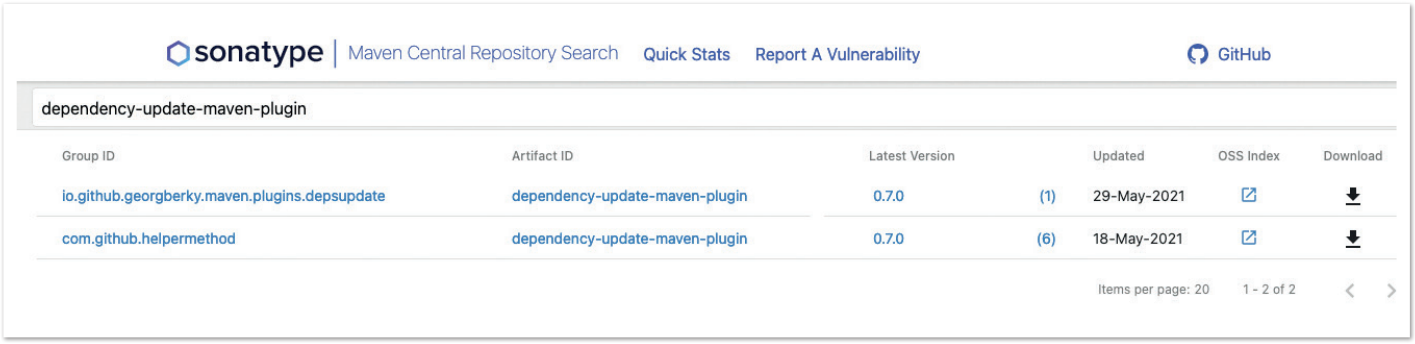

Abbildung 1: Dependency-Update-Maven-Plug-in – erstes Release in Maven Central

Der nächste Nachteil wäre die Trennung der Organisation der beiden Projekte gewesen. Beispielsweise wäre es schwer zu entscheiden, wo zukünftig Benutzer Bugreports melden sollen. Tomek Bujok *[21]* war erfreut, dass das Plug-in noch Verwendung fand, hat aber selbst wenig Zeit, sich weiter darum zu kümmern. Er bot mir daher an, gleich als Maintainer mitzumachen, und gab mir dafür die nötigen Rechte. Schnell stellte sich heraus, dass im Projekt erst mal Wartungsthemen (CI aufbauen, Abhängigkeiten aktualisieren, QA Checks einbauen etc.) bearbeitet werden mussten. Zum Glück hatte Benjamin Marwell *[22]*, ebenfalls ein Maven Committer, das zufällig mitbekommen und half mir mit den Wartungsthemen, sodass ich recht schnell auch die benötigten Fixes einbauen und releasen konnte. Dass im Repository wieder mehr Bewegung reinkam, bewegte auch wieder andere Contributor dazu, etwas beizutragen, sodass in kurzer Zeit ein weiteres Release entstand.

Unser Fazit

Open-Source-Arbeit hat viele Facetten. Es geht dabei nicht nur um die klassischen Code-Beiträge, sondern man kann auch durch Vorträge, Dokumentation, Hilfe in Foren und bei Treffen von User Groups der großen Community helfen. Gleichzeitig sind wir auch immer auf Leute getroffen, die uns geholfen haben. Von Karl Heinz und Oliver angefangen, über unsere Co-Organisatoren bei den User Groups bis hin zu den Organisatoren der Open-Source-Events, in deren Rahmen wir unsere Projekte weiterentwickelt haben. Ohne die Community im Rücken wäre es so viel schwerer gewesen. Darin liegt für uns auch der größte Reiz neben dem Schreiben von Code: Man ist mit einer großen Community verbunden, die gerne ihr Wissen teilt und hilft, wenn es Probleme gibt.

Bei den Hackergarten-Events haben wir gelernt, dass die Einstiegshürde in viele Projekte niedriger ist, als man denkt. Wir hatten ohne große Vorkenntnisse innerhalb weniger Stunden den ersten Beitrag zusammen. Danach mussten wir noch nacharbeiten, aber auch das ging schneller, als wir erwartet hatten, und das Gelernte half uns beim nächsten Beitrag. Viele Projekte freuen sich über Beiträge, die die Dokumentation verbessern oder Fehler melden.

Müssten wir den besten Zeitpunkt für einen Einstieg in Open-Source-Beiträge wählen, wäre es wohl das Hackergarten-Event kombiniert mit Pair-Programming. Dort bekommt man das Projekt vorgestellt und kann direkt mit einem der Maintainer sprechen, wenn es Fragen gibt. Außerdem ist man unter Gleichgesinnten und bekommt vielleicht sogar Inspiration und Hilfe aus anderen Projekten.

Referenzen

[1] *https://sourceforge.net/*
[2] Open Source Contribution via Patch-Dateien: *https://sourceforge.net/p/hsqldb/bugs/1186/*
[3] *https://de.wikipedia.org/wiki/Contributor_License_Agreement*
[4] *https://blog.sandra-parsick.de*
[5] *https://github.com/assertj*
[6] *https://github.com/assertj/assertj-core/issues/1319*
[7] *http://softwerkskammer.ruhr*
[8] *https://www.hackergarten.net/*
[9] *https://hacktoberfest.digitalocean.com*
[10] *http://www.groovy-lang.org/*
[11] *https://georg.berky.dev/publikationen/making-shell-scripts-groovy/*
[12] *https://georg.berky.dev*
[13] *https://github.com/helpermethod*
[14] *https://dependabot.com/*
[15] *https://www.soebes.de/*
[16] *https://github.com/khmarbaise/maven-it-extension*
[17] *https://github.com/georgberky/dependency-update-maven-plugin*
[18] *https://hack-commit-pu.sh/*
[19] *https://maven.apache.org/guides/mini/guide-relocation.html*
[20] *https://github.com/reficio/p2-maven-plugin*
[21] *https://twitter.com/tombujok*
[22] *https://twitter.com/bmarwell*

Georg Berky
hallo@berky.dev

Georgs Handwerk und Leidenschaft ist die Programmierung, meistens in JVM-Sprachen wie Java, Groovy, Kotlin oder Clojure. Dazu gehören für ihn auch Themen wie die Pflege von Legacy Code, Automatisierung von Builds und Deployments oder Agilität im Team. Seit einigen Jahren ist er Co-Organisator der Software-Craftsmanship-Communities im Ruhrgebiet und in Düsseldorf. Wenn er mal nicht programmiert, spielt er Trompete, pflegt seine Bonsai oder praktiziert Aikido.

Sandra Parsick
mail@sandra-parsick.de

Sandra Parsick ist freiberufliche Softwareentwicklerin im Java-Umfeld. Seit 2008 beschäftigt sie sich mit agiler Softwareentwicklung in verschiedenen Rollen. Ihre Schwerpunkte liegen im Bereich der Java-Enterprise-Anwendungen, agilen Methoden, Software Craftsmanship und in der Automatisierung von Softwareentwicklungsprozessen. Darüber schreibt sie gerne Artikel und spricht auf Konferenzen. In ihrer Freizeit engagiert sie sich in der Softwerkskammer Ruhrgebiet, einer Regionalgruppe der Software Craftmanship Community im deutschsprachigen Raum. Außerdem ist sie Java-Champion und Mitglied im Oracle-Groundbreaker-Ambassador-Programm.