Arbeitsgewohnheiten und Expertise von Softwareentwicklern: Empirische Studien zur Verwendung von Skizzen, Implikationen von Code-Plagiaten und dem Aufbau von Expertise[1]

Sebastian Baltes[2]

Abstract: Eine wichtige Voraussetzung für die Optimierung von Arbeitsabläufen und die Entwicklung neuer Werkzeuge in Softwareprojekten ist es, die Arbeitsweise von Softwareentwicklern und die daraus resultierenden Bedürfnisse zu kennen und zu verstehen. Diese Dissertation beschreibt empirische Untersuchungen dreier noch wenig erforschter Bereiche der Softwareentwicklung, stellt neue Werkzeuge vor und gibt Handlungsempfehlungen basierend auf den durchgeführten Untersuchungen. Zunächst wird illustriert wie Softwareentwickler Skizzen und Diagramme in ihrer täglichen Arbeit einsetzen. Anschließend wird der Umgang von Softwareentwicklern mit Code-Fragmenten beschrieben, die auf der populären Onlineplattform Stack Overflow bereitgestellt werden. Nach einer detaillierten Erläuterung dieser beiden Aspekte der täglichen Arbeit von Softwareentwicklern stellen wir ein erstes Modell vor, das empirisch fundiert wichtige Eigenschaften eines Experten in der Softwareentwicklung darstellt und Faktoren auflistet, die die kontinuierliche Weiterbildung von Softwareentwicklern unterstützen. Neben diesen drei Forschungsrichtungen gehen wir auf methodische Erkenntnisse ein und stellen den offenen Datensatz SOTorrent vor, der im Rahmen dieser Dissertation entstand.

1 Einleitung

Ein "work habit", also eine Arbeitsgewohnheit, kann die tägliche Arbeit von Softwareentwicklern positiv wie negativ beeinflussen. Diese Gewohnheiten und die daraus resultierenden Bedürfnisse zu kennen und zu verstehen ist eine notwendige Voraussetzung dafür, bestehende Softwareentwicklungsprozesse und -werkzeuge zu verbessern. Die Software-Engineering-Forschungscommunity wird allerdings oft dafür kritisiert, nicht die Probleme zu adressieren, auf die Entwickler in der Praxis tatsächlich stoßen. Gleichzeitig treffen Entwickler Entscheidungen eher auf Grundlage ihrer persönlichen Erfahrung als auf Grundlage von empirisch fundierten Erkenntnissen. Um diese Lücke zwischen der akademischen Welt und der Praxis zu schließen – oder zumindest zu verkleinern – präsentieren wir in dieser Arbeit [Ba19] empirisch fundierte Ergebnisse und Werkzeuge, die sich mit verschiedenen Arbeitsgewohnheit von Softwareentwicklern beschäftigen. Zusätzlich beleuchten wir was

[1] Englischer Titel der Dissertation: "Software Developers' Work Habits and Expertise: Empirical Studies on Sketching, Code Plagiarism, and Expertise Development"

[2] QAware GmbH, Deutschland, und University Adelaide, Australien, sebastian.baltes@adelaide.edu.au

Experten in der Softwareentwicklung auszeichnet und welche Faktoren die Entstehung einer solchen Expertise begünstigen oder behindern können. Im Folgenden wird zunächst ein Überblick über die zentralen Kapitel der Dissertation gegeben, bevor die Ergebnisse zusammengefasst und ein Ausblick auf zukünftige Forschungsrichtungen gegeben wird.

2 Skizzieren: Wie verwenden Softwareentwickler Skizzen und Diagramme?

Kommunikation ist allgegenwärtig in der Softwareentwicklung. Anforderungen werden von potentiellen Benutzern zu den Softwareentwicklern kommuniziert, die Architektur der Software wird innerhalb des Entwicklungsteams kommuniziert, Entwickler kommunizieren während Pair-Programming-Sitzungen miteinander, und nach der Bereitstellung der Software werden Probleme von Benutzern zu den Entwicklern kommuniziert. An solchen Informationsflüssen sind diverse Kommunikationskanäle beteiligt, so zum Beispiel "face-to-face"-Kommunikation, E-Mails, Videokonferenzen, oder andere Kollaborationswerkzeuge. Insbesondere wenn alle Entwickler gemeinsam an einem Ort "colocated" arbeiten, verwenden Sie – oft informelle – Skizzen und Diagramme zur Kommunikation. Diese visuellen Artefakte umfassen verschiedene Medienarten wie Papier, Whiteboards oder Computerwerkzeuge und unterstützen Entwickler beim Entwerfen neuer und beim Verstehen bestehender Softwaresysteme. Nichtsdestotrotz fehlte zu Beginn unserer Forschung zum Thema Skizzen und Diagramme in der Softwareentwicklung ein umfassendes Bild darüber, wie und warum Entwickler diese visuellen Artefakte verwenden. Deshalb befasst sich der erste Teil dieser Dissertation zunächst damit, verschiedene Dimensionen von Skizzen und Diagrammen in der Softwareentwicklung aus verwandten Arbeiten und eigenen empirischen Studien abzuleiten. Auf Grundlage dieser Dimensionen entwarfen wir dann ein Studiendesign, das es uns erlaubte zu erforschen, wie Entwickler Skizzen und Diagramme in ihrer täglichen Arbeit verwenden. Unsere Forschung beinhaltete eine explorative Feldstudie in drei Softwareunternehmen, eine Onlineumfrage unter 394 Softwareentwicklern und eine Beobachtungsstudie in einer kontrollierten Umgebung mit sechs Pair-Programming-Teams. Nachdem wir unsere Ergebnisse und damit den "state of practice" beschreiben, leiten wir daraus Bedürfnisse von Entwicklern im Umgang mit Skizzen und Diagrammen ab und präsentieren weiterhin zwei Werkzeugprototypen, die wir auf Grundlage unserer empirischen Ergebnisse entwickelt haben. Der Inhalt des entsprechenden Kapitels dieser Dissertation basiert auf vier wissenschaftlichen Publikationen, die alle einen Peer-Review-Begutachtungsprozess durchlaufen haben [BD14, BSD14, BHD17, Ba15].

Forschungsbeiträge:

- Eine **Klassifizierung** von Skizzen und Diagrammen in der Softwareentwicklungspraxis, die auf verwandten Arbeiten und einer eigenen Feldstudie in drei Softwareunternehmen beruht [BD14].
- Eine empirische Untersuchung der 11 identifizierten Dimensionen von Skizzen und Diagrammen in der Softwareentwicklung, basierend auf einer **Onlineumfrage** mit 394 Softwareentwicklern [BD14].

- Eine Analyse darüber, wie Entwickler in einer **Pair-Programming**-Umgebung miteinander kommunizieren, wenn sie **Performanzprobleme** beheben, und insbesondere eine Analyse der **Rolle von Skizzen** in diesem Szenario [Ba15].
- Eine Beschreibung von zwei **Werkzeugprototypen**, die die tatsächlichen Arbeitsabläufe von Entwicklern beim Erstellen und Verwenden von Skizzen und Diagrammen besser unterstützten [BSD14, BHD17].

3 Code-Plagiate: Stack-Overflow-Codefragmente in GitHub-Projekten

Stack Overflow ist heutzutage das populärste Onlineforum für Softwareentwickler. Viele der dort publizierten Beiträge enthalten Codefragmente, die Entwickler in ihre eigenen Projekte übertragen können. Die Verwendung dieser Codefragmente wirft aber Fragen bezüglich deren Wartbarkeit und möglichen Lizenzproblemen auf. Die Creative-Commons-Lizenz, die Stack Overflow verwendet (CC BY-SA), verlangt eine Zuschreibung des kopierten Inhalts ("attribution"), also das Hinzufügen einer Referenz auf die ursprüngliche Frage oder Antwort auf der Plattform, und verlangt die Verwendung einer kompatiblen Lizenz für Werke, die auf dem kopierten Inhalt aufbauen ("share-alike"). Während es eine hitzige Diskussion zum Lizenzmodell von Stack Overflow gab, insbesondere zur Lizenzierung von Quellcode auf der Plattform und der notwendigen Zuschreibung, gab es wenig empirische Evidenz darüber, wie häufig Code von Stack Overflow ohne die verlange Zuschreibung kopiert wird. Um diese Forschungslücke zu schließen, führten wir eine großangelegte empirische Studie durch, um die Verwendung und Zuschreibung von nicht-trivialen Java Codefragmenten aus Stack-Overflow-Antworten in öffentlichen GitHub-Projekten zu untersuchen. Mit Hilfe von drei verschiedenen Data-Mining-Ansätzen triangulierten wir eine Abschätzung für das Verhältnis zwischen zugeschriebenen und nicht zugeschriebenen Verwendungen solcher Codefragmente. Um diese quantitativen Abschätzungen zu komplementieren, befragten wir außerdem Softwareentwickler in zwei Onlineumfragen. In den drei untersuchten Stichproben von GitHub-Projekten lag der Anteil von Projekten mit Referenzen zu Stack-Overflow-Inhalten zwischen 3.3% und 11.9%. Unsere Untersuchungen zeigen, dass nur 1.8% aller untersuchten GitHub-Projekte, die Code von Stack Overflow enthalten, den Code entsprechend der Lizenzbedingungen von CC BY-SA verwenden. Unsere konservative Abschätzung des Anteils von Codefragmenten, die mit der verlangten Zuschreibung der Quelle verwendet werden, liegt bei 25%. Die Mehrheit nicht-trivialer Codefragmente von Stack Overflow werden also ohne Zuschreibung in öffentlichen GitHub-Projekten verwendet. Dies deckt sich auch mit den Ergebnissen unserer Onlineumfragen, in denen etwa die Hälfte der befragten Entwickler zugaben, Code von Stack Overflow ohne die notwendige Zuschreibung zu verwenden. Etwa zwei Drittel der Umfrageteilnehmer gaben allerdings auch an, dass ihnen die Lizenzsituation von Code auf der Plattform Stack Overflow und deren Implikationen nicht bewusst war. Der Inhalt des entsprechenden Kapitels dieser Dissertation basiert hauptsächlich auf einer wissenschaftlichen Zeitschriftenpublikation, die einen Peer-Review-Begutachtungsprozess durchlaufen hat [BD18b]. Außerdem wurden vorläufige Ergebnisse zuvor auf einer Konferenz präsentiert [BKD17].

Forschungsbeiträge:

- Eine fundierte Beschreibung der **Lizenzsituation** von Codefragmenten, die auf der Onlineplattform Stack Overflow veröffentlicht wurden.
- Eine **triangulierte Abschätzung** des **Anteils korrekt zugeschriebener** Codefragmente von Stack Overflow in öffentlichen GitHub-Projekten.
- Eine Analyse potentieller **Lizenzkonflikte** für GitHub-Projekte, die nicht-triviale Codefragmente von Stack Overflow enthalten.
- Eine qualitative Analyse darüber, **wie Softwareentwickler** auf Inhalte von Stack Overflow **verweisen**.
- Eine **Onlineumfrage** die suggeriert, dass vielen **Softwareentwicklern** die Lizenzierung von Inhalten auf Stack Overflow und deren Implikationen **nicht bewusst** ist.

4 Softwareentwicklungsexpertise: Der Weg zu einer ersten Theorie

Softwareentwicklung beinhaltet sehr unterschiedliche Aufgaben, angefangen von der Analyse von Anforderungen über die Implementierung neuer Funktionalitäten bis hin zum Beheben von Fehlern in existierender Software. Um ein Experte in all diesen Aufgaben zu sein, benötigt man verschiedene Fähigkeiten, Wissen und Erfahrung. Bestehende Studien beleuchteten individuelle Aspekte der Softwareentwicklungsexpertise, es fehlte allerdings eine umfassende und übergreifende Theorie zu diesem abstrakten Konzept. Im entsprechenden Kapitel der Dissertation beschreiben wir die erste konzeptionelle Theorie über Softwareentwicklungsexpertise. Unser Modell basiert auf den Ergebnissen von Onlineumfragen mit 335 Softwareentwicklern und verwandten Arbeiten aus Informatik und Psychologie. Wir konzentrierten uns zunächst primär auf Programmiertätigkeiten, unsere Arbeit führte aber bereits in diesem frühen Stadium zu wertvollen Einsichten für andere Forscher, Softwareentwickler und Arbeitgeber. Die Theorie beschreibt wichtige Eigenschaften, die maßgeblich für Expertise in der Softwareentwicklung sind, und Faktoren, die das entstehen einer solchen Expertise fördern oder behindern können. Außerdem zeigen die quantitativen Ergebnisse unserer Onlineumfrage, dass die Einschätzung der eigenen Expertise bei Softwareentwicklern stark kontextabhängig ist und dass Erfahrung nicht notwendigerweise mit Expertise korreliert. Der Inhalt des entsprechenden Kapitels dieser Dissertation basiert hauptsächlich auf einer wissenschaftlichen Publikation, die einen Peer-Review-Begutachtungsprozess durchlaufen hat [BD18a].

Forschungsbeiträge:

- Eine erste **konzeptionelle Theorie** über **Softwareentwicklungsexpertise** basierend auf einer Onlineumfrage mit 335 Softwareentwicklern sowie auf verwandten Arbeiten über Expertise aus Informatik und Psychologie.
- Quantitative Ergebnisse, die auf die **Kontextabhängigkeit** von **Expertise-Selbsteinschätzungen** von Softwareentwicklern hindeuten.
- Ein Ansatz zur **Theoriebildung**, der induktive und deduktive Schritte kombiniert und den andere Software-Engineering-Forscher adaptieren können.

5 Methodische Einsichten: Online-Umfragen mit Softwareentwicklern

Onlineumfragen so wie wir sie für diese Dissertation durchgeführt haben, gelten als wichtiges Werkzeug, um den aktuellen Stand der Praxis zu untersuchen. Insbesondere in der Software-Engineering-Forschung werden solche Umfragen als Werkzeug zum Erforschen und Beschreiben verschiedener Charakteristiken breiter Populationen verwendet. Nichtsdestotrotz stellt das Erreichen professioneller Softwareentwickler mit solchen Umfrage eine schwierige Aufgabe dar. Mit Ausnahme einzelner Unternehmen oder Institutionen, die es Forschern erlauben eine Liste ihrer Angestellten zum Ziehen einer Zufallsstichprobe zu verwenden, ist eine zufällige Auswahl von Umfrageteilnehmern in den meisten Fällen unmöglich. Forscher greifen daher oft auf verfügbare Teilnehmer zurück, was man als *convenience sampling* bezeichnet. Die Verwendung solcher nicht-zufälligen Methoden zur Auswahl von Umfrageteilnehmern führt häufig zu verzerrten Stichproben mit eingeschränkter externer Validität. Um diese Einschränkungen und potentiellen Verzerrungen abzumildern, benötigen Forscher detailliertes Wissen über die Population der Softwareentwickler, die sie untersuchen möchten – allerdings sind die dazu erforderlichen Daten oft nicht verfügbar. Hinzu kommt, dass manche der Methoden zur Auswahl von Umfrageteilnehmern, die Forscher in der Vergangenheit verwendeten, ethisch problematisch sind. Dazu zählt beispielsweise das Kontaktieren von Entwicklern mit Hilfe der E-Mail-Adresse, die sie auf GitHub hinterlegt haben – dies taten Sie allerdings nicht für den Zweck der Kontaktaufnahme durch Forscher. In dieser Dissertation fassen wir die methodischen und ethischen Erkenntnisse zusammen, die wir mit verschiedenen Onlineumfragen gewonnen haben. Der Inhalt des entsprechenden Kapitels dieser Dissertation basiert hauptsächlich auf einer wissenschaftlichen Publikation, die einen Peer-Review-Begutachtungsprozess durchlaufen hat [BD16].

6 Forschungsbeiträge:

- **Erfahrungsberichte** zu verschiedenen Methoden der Rekrutierung von Umfrageteilnehmern.
- Motivation dafür, eine globale **Datenbank mit demographischen Daten über Softwareentwickler** aus verschiedenen Studien aufzubauen, die zum Vergleich herangezogen werden kann, wenn die externe Validität von nicht-zufälligen Stichproben eingeschätzt werden soll.
- Erläuterung **ethischer Implikationen** verschiedener Ansätze zum Kontaktieren von Entwicklern, die von Forschern verwendet wurden und werden.

7 Offene Forschungsdaten: Aufbau und Pflege des SOTorrent-Datensatzes

Für alle Studien, die im Kontext dieser Dissertation durchgeführt wurden, stellt der Autor ergänzende Materialen zur Verfügung, die es anderen Forschern ermöglichen, die Ergebnisse nachzuprüfen. So publizierten wir die (anonymisierten) Daten unserer Studien im langfristig verfügbaren Onlinearchiv Zenodo. Außerdem veröffentlichten wir die Software und Skripte zum Abfragen und Auswerten dieser Daten auf GitHub und archivierten sie zusätzlich auf

Zenodo. Pre- oder Postprints aller Forschungspapiere sind online frei verfügbar. Neben diesen allgemeinen Bemühungen um offene Wissenschaft möchten wir an dieser Stelle den im Rahmen dieser Dissertation entstandenen Datensatz *SOTorrent* hervorheben, den wir entwickelt und veröffentlicht haben, um zukünftige Forschung über Codefragmente auf Stack Overflow zu erleichtern. Dieser Datensatz erlaubt es Forschern, die Evolution von Code auf Stack Overflow auf Ebene individueller Text- und Codeblöcke zu untersuchen, was mit dem offiziell von Stack Overflow zur Verfügung gestellten Datensatz nicht möglich ist. Neben der Verwendung des Datensatzes für unsere eigene Forschung haben wir ihn früh auch für andere Forscher zugänglich gemacht. Dies führte dazu, dass SOTorrent als offizielle *Mining Challenge* der *16th International Conference on Mining Software Repositories* (MSR 2019) ausgewählt wurde [BTD19]. Der Inhalt des entsprechenden Kapitels dieser Dissertation basiert auf zwei wissenschaftlichen Publikationen, die einen Peer-Review-Begutachtungsprozess durchlaufen haben. Ein Konferenzpapier, das die Architektur und Algorithmen hinter dem Datensatz zusammen mit ersten Ergebnissen vorstellt [Ba18] und ein Papier, das die MSR Mining Challenge beschreibt [BTD19]. Wir stellen außerdem zusätzliche Auswertungen vor, die wir für ein noch unveröffentlichtes Forschungspapier durchgeführt haben.

Forschungsbeiträge:

- Ein offener **Datensatz**, der es Forschern erlaubt die Evolution von Inhalten auf Stack Overflow zu untersuchen. Außerdem erleichtert der Datensatz die Analyse der Verbindung von Stack Overflow mit anderen Plattformen wie GitHub durch das Extrahieren von Hyperlinks.
- Eine fundierte **Evaluierung von 134 String-Ähnlichkeitsmetriken** hinsichtlich ihrer Anwendbarkeit für die Rekonstruierung der Versionshistorie von Stack Overflow Text- und Codeblöcken.
- Eine **erste Auswertung der Evolution** von Inhalten auf Stack Overflow, die einen Zusammenhang von Änderungen an Beiträgen und zugehörigen Kommentaren motiviert.
- Eine Auswertung von **Codeduplikaten innerhalb von Stack Overflow** mit einer Diskussion möglicher Lizenzkonflikte und Implikationen für die Wartbarkeit.

8 Zusammenfassung der Ergebnisse und Ausblick

In der hier vorgestellten Dissertation verwendete der Autor verschiedene Forschungsdesigns, um bisher wenig erforschte Aspekte der täglichen Arbeit von Softwareentwicklern zu untersuchen. Zunächst wurde erforscht, wie Softwareentwickler *Skizzen und Diagramme* in ihrer täglichen Arbeit verwenden, mit dem Ziel, Anforderungen für verbesserte Werkzeugunterstützung abzuleiten. Zwei Prototypen für Werkzeuge wurden implementiert und evaluiert. Im zweiten Forschungsprojekt analysierten wir die Angewohnheit von Entwicklern, *nicht-triviale Codefragmente* aus dem populären Onlineforum Stack Overflow in quelloffene GitHub-Projekte zu *kopieren*, ohne sich an die Bedingungen von Stack Overflows Lizenz zu halten. Zusätzlich untersuchten wir, inwiefern Entwicklern die Lizenzsituation und deren Implikationen bewusst ist. Während diese beiden Projekte Analysen von positiven beziehungsweise negativen Angewohnheiten von Entwicklern darstellen, fokussierte das dritte Projekt auf Verhaltensweisen, die Entwickler dabei unterstützen können, Experten in

bestimmten Softwareentwicklungsaufgaben zu werden. Neben Faktoren, die den Aufbau einer solchen Expertise über die Zeit beeinflussen, entwickelten wir eine erste konzeptionelle Theorie zum Strukturieren des abstrakten Konzepts der *Softwareentwicklungsexpertise*.

Skizzieren Um einen Überblick über die Verwendung von *Skizzen und Diagrammen* in der Softwareentwicklung zu erhalten, führten wir mehrere qualitative und quantitative Studien durch, die zeigen, dass Softwareentwickler in ihrer täglichen Arbeit regelmäßig Skizzen und Diagramme erstellen und wiederverwenden. Unsere Onlineumfrage mit 394 Teilnehmern ergab, dass diese Skizzen und Diagramme oft informell sind, aber trotzdem als wertvolle Ressource angesehen werden, die verschiedene Aspekte des Softwareentwicklungsprozesses dokumentieren. Wir zeigen auf, wie diese Artefakte mit dem Quellcode auf verschiedenen Abstraktionsebenen zusammenhängen und dass etwa die Hälfte der erstellten Skizzen und Diagramme potentiell hilfreich zum Verstehen des zugehörigen Quellcodes sind. Da Softwaredokumentation oft nicht vorhanden oder nicht aktuell ist, können diese visuellen Artefakte Softwaredokumentation wie Quellcodekommentare oder Textdokumente ergänzen. Die meisten Skizzen und Diagramme werden jedoch auf analogen Medien wie Papier oder Whiteboards erstellt. Viele dieser analogen Skizzen werden archiviert, unsere Studienteilnehmer erwähnten aber auch technische Probleme wie fehlende Werkzeugunterstützung, um Skizzen zusammen mit dem zugehörigen Quellcode zu archivieren. Basierend auf diesen Ergebnissen entwickelten und evaluierten wir den Werkzeugprototypen *SketchLink*, der Entwickler beim Archivieren von und Zugreifen auf Skizzen, die mit Quellcode in Verbindung stehen, unterstützt. Bezüglich der Evolution von Skizzen deuten unsere qualitativen Ergebnisse darauf hin, dass es üblich ist, Skizzen zunächst auf analogen Medien wie Papier oder Whiteboards zu erstellen. Nach potentiell mehreren Iterationen werden die zunächst analogen Artefakte dann häufig digital archiviert. Um solche analog-digitalen Arbeitsabläufe zu unterstützten, entwickelten wir einen zweiten Werkzeugprototypen namens *LivelySketches*, der die Transitionen zwischen verschiedenen analogen oder digitalen Medien erleichtert. Ein möglicher Schritt für zukünftige Arbeiten ist es, die Funktionalitäten der beiden Prototypen zu vereinigen und das resultierende Werkzeug dann in einem größeren Softwareprojekt zu evaluieren. Außerdem beobachten wir, dass Ansätze wie *graphic facilitation* und *sketchnoting* immer beliebter werden. Potentielle Anwendungsfälle solche Techniken in Softwareprojekten zu studieren ist ein weitere Richtung für zukünftige Arbeiten. Tatsächlich haben wir bereits "graphic facilitators" mit Erfahrung in Softwareprojekten befragt und deren Aussagen mit denen von Softwareentwicklern und -architekten mit Skizziererfahrung verglichen. Basierend auf diesen Interviews werden wir Empfehlungen für die Anwendung von informellen Visualisierungs- und Skizziertechniken in verschiedenen Phasen der Softwareentwicklung ableiten.

Code-Plagiate Unsere Untersuchungen zu Kopien von *Stack-Overflow-Code-Fragmenten* in Java-Projekten auf GitHub ergab, dass höchstens ein Viertel der Fragmente wie von Stack Overflows Lizenz CC BY-SA gefordert die Quelle zuschreiben ("attribution"). Ein

weiteres Ergebnis ist, dass zwischen 3.3% und 11.9% der analysierten GitHub-Projekte in mindestens einer Datei auf Stack Overflow verwiesen. In unserer Studie erfüllten nur 1.8% der Projekte mit Kopien von Stack-Overflow-Codefragmenten beide Kriterien der Lizenz – verwiesen also auf die Quelle ("attribution") und verwendeten eine kompatible Lizenz ("share-alike"). Für die restlichen 98.2% der Projekte kann insbesondere die Verletzung des "share-alike"-Kriteriums zu Lizenzkonflikten mit den in der Dissertation diskutierten Konsequenzen – von möglichen Unterlassungsklagen bis hin zu Schadensersatzforderungen – führen. Die beiden Onlineumfragen, die wir in diesem Kontext durchgeführt haben, zeigen, dass viele Entwickler zugeben, Code von Stack Overflow ohne Zuschreibung zu kopieren. Vielen Umfrageteilnehmern war aber gleichzeitig nicht bewusst, wie Inhalte auf Stack Overflow lizenziert sind und welche Implikationen dies für die Verwendung in eigenen Projekten hat. Im Zuge dieses Forschungsprojekts entwickelten wir auch den offenen Datensatz *SOTorrent*, den wir seitdem regelmäßig aktualisieren. Neben eigenen Studien zur Wartung und Aktualisierung von Codefragmenten auf Stack Overflow verfolgen wir die Arbeiten anderer Forschergruppen, die unseren Datensatz verwenden. Eine zukünftige Forschungsrichtung, die nicht auf Code von Stack Overflow beschränkt ist, ist es, bessere Werkzeugunterstützung für Softwareentwickler zu schaffen, um ihnen die Lizenz- und Wartungsimplikationen von Codefragmenten aus dem Internet zu verdeutlichen. So könnten in Zukunft Continuous-Integration-Werkzeuge automatisch überprüfen, ob Commits nicht-triviale Codefragmente von Onlinequellen zu einem Projekt hinzufügen. Solche Werkzeuge können helfen, die notwendige Zuschreibung hinzuzufügen, sie können aber auch Entwickler über die Lizenzkompatibilität und die entsprechenden Anforderungen aufklären. Diese Ansätze können dabei helfen, lizenzrechtliche Probleme für Open-Source-Projekte, die durch die absichtliche oder unabsichtliche Verwendung von lizenziertem Onlinecode entstehen, zu verhindern.

Softwareentwicklungsexpertise In unserer Forschung zur *Softwareentwicklungsexpertise* identifizierten wir verschiedene Charakteristiken von Experten in der Softwareentwicklung, zeigten aber auch Faktoren auf, die den Aufbau einer solchen Expertise über die Zeit positiv wie negativ beeinflussen können. Neben der Entwicklung einer ersten konzeptionellen Theorie fanden wir heraus, dass die Expertise-Selbsteinschätzung von Entwicklern kontextabhängig ist und nicht unbedingt mit der Erfahrung in Jahren korreliert. Wir empfehlen Forschern, wenn sie Studienteilnehmer nach einer Selbsteinschätzung fragen, möglichst explizit zu beschreiben, was in ihrer spezifischen Studie Anfänger von Experten unterscheidet. Neben dem Entwurf neuer Studien erlaubt es unsere Theorie, Ergebnisse vergangener Studie neu einzuordnen, da verschiedener Einflussfaktoren auf Verhalten und Leistung aufgezeigt werden. Softwareentwickler können mit Hilfe unserer Ergebnisse lernen, was andere Entwickler von Experten erwarten und welche Verhaltensweisen für den gezielten Aufbau von Expertise hilfreich sind. So erläutern wir beispielsweise das Konzept der "deliberate practice" und weisen auf die Wichtigkeit anspruchsvoller Ziele, einer unterstützenden Arbeitsumgebung und des Feedbacks von Arbeitskollegen hin. Erfahrenere Entwickler können lernen, was von guten Mentoren erwartet wird. So werden sie als wichtige Quelle für Feedback

betrachtet und sollten vor allem geduldig und unvoreingenommen sein. Arbeitgeber können von unsere Studie lernen, was typische Faktoren der Demotivierung ihrer Mitarbeiter sind und wie sie eine Arbeitsumgebung schaffen können, die Lernen und Weiterentwicklung unter ihren Angestellten fördert. Neben offensichtlichen Strategien wie Fortbildungen und Konferenzbesuche suggerieren unsere Ergebnisse, dass der Informationsfluss zwischen verschiedenen Entwicklungsteams und auch zwischen Entwicklungsteams und anderen Teilen des Unternehmens ein wichtiger Faktor ist. Arbeitgeber sollten sich ebenfalls darum bemühen, eine gute Mischung aus Kontinuität und Wandel in ihren Softwareentwicklungsprozess zu integrieren, da wenig anspruchsvolle Aufgaben ein Hauptdemotivationsfaktor für Softwareentwickler sind. Eine wichtige zukünftige Forschungsrichtung ist die *Rolle älterer Softwareentwickler*. Insbesondere in Industrieländern führt der demographische Wandel zu einer alternden Erwerbsbevölkerung. Trotz dieser Tatsache sind die Hürden und Herausforderungen älterer Entwickler in einem kompetitiven Feld wie der Softwareentwicklung größtenteils unbekannt. Unsere Studienteilnehmer nannten bereits verschiedene altersbezogene Herausforderungen. Wir möchten diese weiter untersuchen, um sie wo möglich zu adressieren und abzumildern, damit erfahrende Entwickler sich nicht dazu entscheiden, ab einem gewissen Alter die Softwareindustrie zu verlassen.

Im Verlauf dieser Dissertation beschrieben wir nicht nur unsere Studiendesigns und Ergebnisse, sondern versuchten immer auch aufzuzeigen wie unsere empirischen Ergebnisse dazu verwendet werden können, nächste Schritte in Forschung oder Praxis zu motivieren. So haben wir unsere Studienergebnisse beispielsweise dazu verwendet: (1) neuartige Werkzeuge zur Unterstützung der Arbeitsabläufe von Entwicklern im Zusammenhang mit Skizzen und Diagrammen zu entwerfen, (2) Open-Source-Entwickler über mögliche Lizenzkonflikte in ihren Projekten zu informieren, (3) eine erste konzeptionelle Theorie über Softwareentwicklungsexpertise abzuleiten, welche sowohl in der Forschung als auch in der Praxis angewandt werden kann, (4) auf altersbezogene Probleme von Softwareentwicklern hingewiesen, (5) Aufmerksamkeit unter Forschern über ethische Implikationen ihrer Befragungsstrategien zu schaffen, und (6) einen offenen Datensatz bereitzustellen, der für zukünftige Forschungsprojekte verwendet werden kann. Unsere Arbeiten unterstützen Forscher und Menschen aus der Praxis darin, Entscheidungen über die Verwendung bestimmter Werkzeuge oder Abläufe im Softwareentwicklungsprozess auf Grundlage von Daten und empirischen Ergebnissen – statt “aus dem Bauch heraus” – zu treffen.

Literaturverzeichnis

[Ba15] Baltes, Sebastian; Moseler, Oliver; Beck, Fabian; Diehl, Stephan: Navigate, Understand, Communicate: How Developers Locate Performance Bugs. In: ESEM 2015. 2015.

[Ba18] Baltes, Sebastian; Dumani, Lorik; Treude, Christoph; Diehl, Stephan: SOTorrent: Reconstructing and Analyzing the Evolution Stack Overflow Posts. In: 15th International Conference on Mining Software Repositories (MSR 2018). 2018.

[Ba19] Baltes, Sebastian: Software Developers' Work Habits and Expertise: Empirical Studies on Sketching, Code Plagiarism, and Expertise Development. Doctoral Dissertation. University of Trier, Germany, 2019.

[BD14] Baltes, Sebastian; Diehl, Stephan: Sketches and diagrams in practice. In: FSE 2014. 2014.

[BD16] Baltes, Sebastian; Diehl, Stephan: Worse Than Spam: Issues In Sampling Software Developers. In: ESEM 2016. 2016.

[BD18a] Baltes, Sebastian; Diehl, Stephan: Towards a Theory of Software Development Expertise. In: ESEC/FSE 2018. 2018.

[BD18b] Baltes, Sebastian; Diehl, Stephan: Usage and Attribtion of Stack Overflow Code Snippets in GitHub Projects. Empirical Software Engineering, Online First:1–37, 2018.

[BHD17] Baltes, Sebastian; Hollerich, Fabrice; Diehl, Stephan: Round-Trip Sketches: Supporting the Lifecycle of Software Development Sketches from Analog to Digital and Back. In: VISSOFT 2017. 2017.

[BKD17] Baltes, Sebastian; Kiefer, Richard; Diehl, Stephan: Attribution required: Stack overflow code snippets in GitHub projects. In: ICSE 2017 Companion Volume. 2017.

[BSD14] Baltes, Sebastian; Schmitz, Peter; Diehl, Stephan: Linking sketches and diagrams to source code artifacts. In: FSE 2014. 2014.

[BTD19] Baltes, Sebastian; Treude, Christoph; Diehl, Stephan: SOTorrent: Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets. In: MSR 2019. 2019.

Sebastian Baltes arbeitet als Senior Software Engineer bei der QAware GmbH in Deutschland und als Adjunct Lecturer in der School of Computer Science der University of Adelaide in Australien. Er wurde im April 2019 an der Universität Trier mit der Dissertation "Software Developers' Work Habits and Expertise" als Dr. rer. nat. promoviert. Zuvor studierte er an den Universitäten in Trier, Saarbrücken und Växjö, Schweden, Informatik (B.Sc. und M.Sc.). In seiner Forschung widmet er sich der empirischen Analyse der Arbeitsweise von Softwareentwicklern, um deren tägliche Arbeit durch verbesserte Werkzeuge und Handlungsempfehlungen produktiver zu gestalten. Die Hauptmotivation seiner Forschung ist, dass in der Softwareentwicklung immer noch zu viele Entscheidungen eher meinungs- als datenbasiert getroffen werden. Diese Lücke zwischen empirischer Forschung und Praxis gilt es zu schließen, sowohl durch das Erforschen relevanter Probleme als auch durch geeignete Wissenschaftskommunikation der Ergebnisse zurück zu den Entwicklern in der Praxis.