Priorisierung von Quelltextduplikaten in Testcode durch die Kombination von Clone-Detection und testspezifischer Coverage*

Stefan Knilling♠, Jakob Rott♣, Roman Haas♣

♠stefan.knilling@tum.de – Technische Universität München, Fakultät für Informatik, Garching b. München
♣{rott, haas}@cqse.eu – CQSE GmbH, München

Zusammenfassung

Durch Clone-Detection aufgedeckte Codeduplikate in Testcode sind schwieriger zu bewerten als solche in Produktivcode und in der Praxis werden sie häufig ignoriert. In der Folge bleiben redundante Codeabschnitte in Testcode erhalten. Diese führen zu Wartungsproblemen und es besteht die Gefahr ungewollt inkonsistenter Änderungen. In diesem Artikel wird hinterfragt, ob die Kombination von Ergebnissen einer Clone-Detection mit testspezifischer Code Coverage helfen kann, Klonfunde in Testcode automatisch zu bewerten. Tests, die zwar von der Clone-Detection als redundant erkannt werden, aber stark unterschiedliche Teile im Code ausführen, werden von Entwicklern aufgrund der so anzunehmenden semantischen Entfernung möglicherweise weniger oft refaktoriert. Entsprechend sollten Klone von Tests mit einer hohen Ausführungsüberdeckung als wichtiger eingestuft werden. Durch die erhöhte Treffsicherheit soll die Clone-Detection von Entwicklern als nützlicheres Werkzeug anerkannt werden.

Es wurden Codeduplikate im Testcode von neun Open-Source-Projekten untersucht und die Ergebnisse von Entwicklern evaluiert. Dabei zeigte sich eine moderat positive Korrelation zwischen automatischer Bewertung und Entwicklerstimmen.

1 Einleitung

Clone-Detection ist ein probates Mittel in der Softwareentwicklung, um Quelltextduplikate aufzudecken. Werden dadurch Klone vermieden, bleiben deren negative Effekte auf die Wartbarkeit und Softwarequalität aus. Anderenfalls erhöht sich durch Codeklone im Softwaresystem der Wartungsaufwand [1], da Änderungen nicht nur an einem, sondern an allen Vorkommen des Klons durchgeführt werden müssen. Werden Änderungen im Quelltext nicht konsistent durchgeführt, verbleiben Fehler im Programm oder neue Fehler entstehen.

Durch den Einsatz von Clone-Detection werden Codeklone in Test- und Produktivcode aufgedeckt. In der Praxis kann man jedoch beobachten, dass von Clone-Detection aufgedeckte Klone in Testcode von Entwicklern häufiger nicht beachtet werden. Schon eine kleine Abweichung im Quellcode eines Testfalls kann dazu führen, dass eine andere Programmfunktionalität getestet wird. Durch die semantischen Unterschiede kann die Redundanz im Testfall dann als tolerabel wahrgenommen werden. Die aus Sicht einiger Entwickler als höher wahrgenommene Anzahl unnötiger Klonfunde kann dazu führen, dass Testcode von der Clone-Detection ausgenommen wird und Klone in Testcode unbeachtet bleiben.

Um Entwicklern besonders relevante Klonfunde in Testcode zeigen zu können, wird in diesem Artikel eine zusätzliche Bewertung syntaktisch ähnlicher Testfälle vorgeschlagen. Dies soll anhand der ausgeführten Funktionalität geklonter Tests geschehen. Aus der testspezifischen Code Coverage wird der Execution Overlap berechnet, d. h. wie viele gleiche Methoden von verschiedenen Instanzen eines Klons ausgeführt werden. Dieser Wert soll zur Priorisierung der gefundenen Codeduplikate in Testcode dienen. Zusätzlich wurde diese Möglichkeit zur automatischen Priorisierung in einer Entwicklerbefragung evaluiert.

2 Grundlagen

Für die **testspezifische Code Coverage** werden während der Ausführung eines einzelnen Testfalls alle aufgerufenen Methoden aufgezeichnet.

Aus der testspezifischen Coverage zweier Testfälle kann der **Execution Overlap (ExOlap)** berechnet werden. Diese für dieses Paper eingeführte Metrik beschreibt, wie viele der durch die beiden Testfälle ausgeführten Methoden identisch sind. Führten zwei Testfälle je zwei Methoden aus, wobei je eine der beiden Methoden identisch ist, betrüge der ExOlap 50%.

3 Fallstudie: Execution Overlap zur Bewertung von Klonen in Testcode

Ob sich ExOlap zur Bewertung von Klonen in Testcode eignet, wurde an Open-Source-Projekten getestet. Durch zusätzliche Daten aus einer Entwicklerbefragung wurde geklärt, ob der ExOlap als Indikator für die Relevanz eines Klons in Testcode dienen kann.

Idee Bei geklonten Testfällen, die im Testlauf überwiegend unterschiedlichen Code ausführen, kann von einer semantischen Entfernung ausgegangen werden. Aufgrund dieser werden solche Klone möglicherweise weniger oft refaktoriert. Zu überlegen ist, ob in einem Clone-Detection-Verfahren diese weniger interessanten Funde als nachrangig eingestuft werden können.

*Das diesem Artikel zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen **Sofie, 01IS18012A** gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

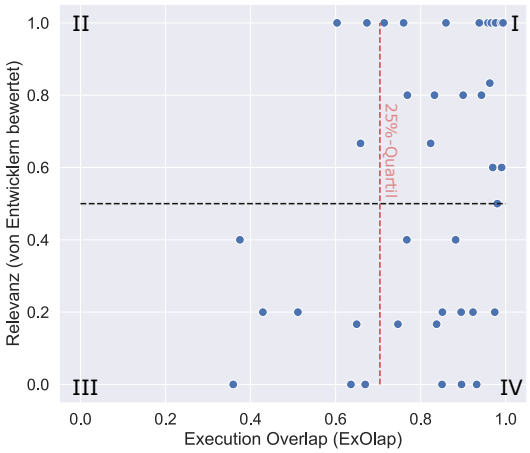

Abbildung 1: Streudiagramm untersuchter Klonpaare. Anordnung nach Execution Overlap (ExOlap) und nach manuell bewerteter Relevanz. In den Quadranten I+IV (rechts der gestrichelten roten Linie, die das 25%-Quartil der ExOlap-Verteilung zeigt) liegen Paare mit höherem ExOlap. In den Quadranten I+II liegen Paare, die Entwickler als relevant eingestuft haben.

Dafür wird in dieser Arbeit die Metrik **ExOlap** eingeführt: Geklonte Testfälle mit niedrigem **ExOlap** sollten in der Clone-Detection niedrige Prioritäten erhalten. Vice versa werden Testfällen mit hohem **ExOlap** hohe Prioritäten zugeordnet. Dabei ist das Ziel, dass Entwickler sich auf Klonfunde konzentrieren können, die sie wahrscheinlicher refaktorieren würden.

Durchführung Für aufgedeckte Klonpaare[1] in Testcode wurde der **ExOlap** berechnet. Die dafür notwendige testspezifische Coverage wurde in einer vorhergehenden Testausführung aufgezeichnet. Zur Durchführung der Clone-Detection sowie zur Verarbeitung der Testdaten wurde die Software-Intelligence-Plattform **Teamscale** verwendet.

Evaluation In einer Korrelationsanalyse wurden für Paare geklonter Testfälle der berechnete **ExOlap** und die von Entwicklern bewertete Relevanz gegenübergestellt. Ein Klon galt dabei als **relevant**, wenn die Mehrheit der befragten Entwickler angab, diesen Klon refaktorieren zu wollen und von einem automatischen Verfahren einen Hinweis auf diesen Klon zu erwarten. Weiterhin wurde in der Evaluation überprüft, ob es fehlerarm möglich ist, auf die Relevanz bzw. Nichtrelevanz durch eine Zweiteilung an einem bestimmten **ExOlap**-Wert zu schließen. Zur Dichotomisierung der automatischen Bewertung der Klonfunde wurde das 25%-Quartil der **ExOlap**-Verteilung verwendet, um zu vermeiden, dass relevante Klone mit niedrigem **ExOlap** falsch, also als nicht relevant, eingeordnet werden.

4 Ergebnisse und Diskussion

Aus neun Open-Source-Projekten wurden 40 zufällig ausgewählte Paare geklonter Testfälle untersucht. Für jedes Paar wurde der **ExOlap** berechnet und außerdem die Relevanz durch mehrere Entwickler bewertet. Anhand der erhobenen Daten wurden zwei Forschungsfragen beantwortet:

– Sind Klone tatsächlich nicht nur in Produktivcode, sondern auch in Testcode präsent und somit dort ein Problem?
– Eignet sich Execution Overlap (**ExOlap**) als Indikator für Relevanz von Klonen in Testcode?

Clone Coverage in Test- und Produktivcode

Clone Coverage beschreibt den Anteil der Statements in Code, der Teil zumindest eines Klons ist. Die durchschnittlich festgestellte Clone Coverage der analysierten Projekte betrug im Produktivcode 15,2% und im Testcode 27,7%. Dies unterstützt die Annahme, dass **Klone in Testcode präsent sind** und im Testcode der Studienobjekte im Schnitt größere Anteile dupliziert wurden als in Produktivcode.

Zusammenhang von ExOlap und Relevanz

Der berechnete Rangkorrelationskoeffizient (Spearman) $r_s = 0,364$ lässt auf einen **geringen positiven Zusammenhang zwischen der bewerteten Relevanz und dem ExOlap** schließen. Im Streudiagramm (Abb. 1), das die Ergebnisse zeigt, trennt

– die vertikale rote Linie Paare mit höherem **ExOlap** beim 25% Quartil (Quadranten Q**I**+Q**IV**) und
– die horizontale Linie den Datensatz nach von Entwicklern bewerteter Relevanz (Q**I**+Q**II**).

Eine Zuordnung von **ExOlap** zu Relevanz war fehlerfrei für die 25 Klone (62,5%) in Q**I** und Q**III**. In diesen Quadranten liegen Testfälle mit gleichzeit hoher Relevanz und hohem **ExOlap** (Q**I**, 18 Klone) bzw. niedriger Relevanz und niedrigem **ExOlap** (Q**III**, 7 Klone).
Für die restlichen 15 Klone (37,5%) stimmte die automatische Bewertung nicht mit der Experteneinschätzung überein. Die drei Klone in Q**II** sind relevant, haben aber einen niedrigen **ExOlap**. Sie gelten als **False Negatives**. Zwölf Klone haben eine niedrige Relevanz bei hohem **ExOlap** und sind **False Positives**. Dabei sind vor allem die False Negatives problematisch, da diese aufgrund ihres niedrigen **ExOlap** automatisiert nicht als relevant eingestuft werden könnten, obwohl die Klonpaare laut den Entwicklern relevant sind.

5 Fazit

Eine durchschnittliche Clone Coverage der untersuchten Projekte von 27,7% zeigt, dass Codeduplikate in Testcode ein Problem sind. Indem zusätzlich zu den Kloninformationen die testspezifische Code Coverage der Testfälle analysiert wird, kann die Relevanz gefundener Klone automatisiert mit Hilfe des **ExOlap** bewertet werden, um Entwicklern relevante Klonfunde zum Refactoring vorzulegen. Bei der Bewertung der Relevanz der Klonpaare treten eine geringere Anzahl von False Positives und False Negatives auf.

Literatur

[1] E. Jürgens, F. Deißenböck, B. Hummel, and S. Wagner. Do code clones matter? In *IEEE 31st International Conference on Software Engineering*, 2009.
[2] Benjamin Hummel, Elmar Jürgens, Lars Heinemann, and Michael Conradt. Index-based code clone detection: Incremental, distributed, scalable. In *In IEEE 26th International Conference on Software Maintenance*, 2010.
[3] Rainer Koschke. Survey of Research on Software Clones. *Dagstuhl Seminar Proceedings*, 2007.

[1] Die verwendete Clone-Detection war indexbasiert [2], erkannte Klone der Typen I und II [3] und war auf eine Mindestlänge von zehn Statements konfiguriert.