Kann statische Analyse unnützen Code erkennen?

Roman Haas[1] Rainer Niedermayr[2] Tobias Roehm[3] Sven Apel[4]

Abstract: Dieser Beitrag fasst einen im ACM Journal Transactions on Software Engineering and Methodology veröffentlichten Artikel von Haas et al. [Ha20] zusammen. Darin geht es um die Identifikation von unnützem Code, also Code, der nicht mehr benötigt wird. In den meisten gewachsenen Softwaresystemen sind Teile des Codes im Laufe der Zeit unnütz geworden. Unnützer Code ist problematisch, weil dadurch Ressourcen bei der Entwicklung und Wartung verschwendet werden, beispielsweise bei der Vorbereitung für eine bevorstehende Migration oder Zertifizierung. Zwar kann mithilfe eines Profilers nicht mehr produktiv verwendeter Code aufgedeckt werden, aber oft ist es zu zeitaufwendig bis repräsentative Daten erhoben wurden. Im oben genannten Beitrag wird ein auf Codestabilität und -zentralität basierender statischer Analyseansatz vorgeschlagen, um unnützen Code zu identifizieren. Um die Nützlichkeit des Ansatzes zu untersuchen, wurde eine Studie mit 14 Open-Source- und Closed-Source Software-Systemen durchgeführt. Da es kein perfektes Orakel für unnützen Code gibt, wurden die Empfehlungen für unnützen Code auf Basis von Löschungen in der Codehistorie, Laufzeitnutzungsdaten und dem Feedback von 25 Entwicklern aus fünf Software-Projekten evaluiert. Die Studie zeigt, dass aus Stabilitäts- und Zentralitätsinformationen generierte Empfehlungen auf unnützen Code hinweisen. Insgesamt deuten die Ergebnisse darauf hin, dass die statische Analyse schnelles Feedback zu unnützem Code liefern kann und in der Praxis nützlich ist.

Keywords: Unnecessary code, code stability, code centrality

1 Zusammenfassung

Unnützer Code ist solcher, an dem kein Stakeholder Interesse hat. In den allermeisten großen Softwareprojekten werden Teile des Codes unnütz, beispielsweise weil Funktionalität erneut implementiert wird und die Originalimplementierung bestehen bleibt, oder weil sich die Interessen der Stakeholder ändern und so implementierte Funktionalität nicht mehr genutzt wird. Unnützer Code verschwendet Ressourcen unterschiedlichster Art, gleichzeitig lässt sich unnützer Code nur mit großem manuellen Aufwand erkennen. Im hier zusammengefassten Beitrag wird ein Ansatz zur statischen Identifizierung von unnützem Code vorgestellt. Dieser baut auf der Hypothese auf, dass der stabilste und gleichzeitig dezentralste Code in der Abhängigkeitsstruktur der Software wahrscheinlich unnütz ist. Zu diesem Zweck wurde eine Analyse implementiert, die Stabilitäts- und Zentralitätsmaße verwendet, um Dateien als unnützen Code zu empfehlen. Diese Empfehlungen sind als Ausgangspunkt für Praktiker gedacht, die unnützen Code aus ihrer Codebasis entfernen möchten.

[1] CQSE GmbH und Saarbrücken Graduate School of Computer Science
[2] Ergon Informatik AG
[3] CQSE GmbH
[4] Saarland University, Saarland Informatics Campus

Der auf statischer Analyse aufbauende Ansatz berechnet für jede Quelltextdatei des Softwaresystems einen Stabilitäts- und einen Zentralitätsscore. Diese spiegeln die Änderungshäufigkeit in der Historie des Repositories und die Zentralität der korrespondierenden Knoten im Abhängigkeitsgraph der Quelltextdateien wider. Als potentiell unnütz gelten Dateien, die besonders stabil und dezentral sind. Aus Gründen der Nutzerfreundlichkeit werden die Vorschläge für unnützen Code basierend auf der Dateisystemhierarchie gruppiert, sodass beispielsweise ganze Packages als potentiell unnütz vorgeschlagen werden können.

Anhand einer empirischen Studie wurde untersucht, ob mittels Stabilitäts- und Zentralitätsinformationen unnützer Code identifiziert werden kann. Da es keinen Goldstandard für die Frage nach unnützem Code gibt, wurden drei verschiedene Orakel für unnützen Code betrachtet: (1) Löschungen von als unnütz bezeichnetem Code in der Codehistorie, (2) Nutzungsdaten aus Laufzeitumgebungen und (3) Entwicklerbefragungen. Zunächst wurde in einer Untersuchung von in der Historie gelöschtem und als unnütz bezeichnetem Code festgestellt, dass dieser im Durchschnitt stabiler und dezentraler in der Abhängigkeitsstruktur der Studiensubjekte war. Das Kernergebnis der empirischen Studie ist, dass die Ergebnisse aus den drei Orakeln darauf hindeuten, dass Codestabilität und -zentralität zur Identifikation von unnutzem Code hilfreich sind. Einige der als unnütz erkannten Quelltextdateien wurden im Anschluss an die Befragungen sogar tatsächlich gelöscht, wodurch der Nutzen der vorgestellten Analyse unterstrichen wird. Nichtsdestotrotz muss mit falsch-positiven Empfehlungen gerechnet werden, insbesondere falls dynamische oder externe Codeabhängigkeiten bestehen.

Data Availability

In einem Archiv unter `https://figshare.com/s/8c3f63d2c620d1aae83a` sind die Konfigurationsdaten und Ergebnisauswertungen der Studie frei verfügbar.

Danksagung

Diese Arbeit wurde in Teilen aus Mitteln des Bundesministeriums für Bildung und Forschung (BMBF), Förderkennzeichen “SOFIE, 01IS18012A” finanziert. Apels Arbeit wurde durch die Deutsche Forschungsgemeinschaft (DFG) unterstützt (AP 206/11-1). Für diesen Beitrag sind die Autoren verantwortlich.

Literatur

[Ha20] Haas, R.; Niedermayr, R.; Roehm, T.; Apel, S.: Is Static Analysis Able to Identify Unnecessary Source Code? ACM Transactions on Software Engineering and Methodology 29/1, S. 1–23, 2020.