netidee

STIPENDIEN

Automated Diagnosis of Heisenbugs

Zwischenbericht | Call 17 | Stipendium ID 6428

Inhalt

1 Einleitung

Heisenbugs sind Computerfehler, die nur gelegentlich auftreten. Diese Unvorhersehbarkeit ist nicht nur für Systemnutzer:innen frustrierend, sondern stellt auch die verantwortlichen Entwickler:innen vor große Herausforderungen. Der erste Schritt bei der Fehlerbehebung ist es, herauszufinden, unter welchen Umständen der Fehler auftritt. Dieser Schritt ist bei Heisenbugs besonders schwierig. Ziel dieses Dissertationsprojekts ist es, formale Methoden zu entwickeln, die bei der Suche und dem Beheben von Heisenbugs unterstützen.

Im ersten Abschnitt meines Projekts habe ich mich mit verschiedenen Beispielen von Heisenbugs auseinandergesetzt. Besonders eingehend habe ich mich mit einer speziellen Klasse von Heisenbugs befasst, die auftreten, wenn Berechnungsergebnisse (z.B. durch Memoization) abgespeichert und in einem anderen Kontext wiederverwendet werden.

Als Basis für meine weitere Arbeit habe ich mich in meine jüngsten Bemühungen darauf konzentriert, formale Grundlagen zu schaffen, die es ermöglichen mathematisch präzise zu definieren, was Heisenbugs sind. Im Folgenden werden die neuesten Entwicklungen in diesen Projektschritten erläutert.

2 Status

2.1 Meilenstein 1 – Memoization Bugs in JavaScript Programmen

Bei dem ersten Meilenstein ging es darum, eine spezielle Art von Heisenbugs zu erforschen, die auftritt, wenn Berechnungsergebnisse abgespeichert und in einem anderen Kontext wiederverwendet werden. Dafür habe ich zuerst unterschiedliche Eigenschaften definiert, die beschreiben, wann ein Programm, das Berechnungsergebnisse wiederverwendet, korrekt ist. Weiters habe ich eine dynamische Analyse entwickelt, die diese Eigenschaften zur Laufzeit überprüfen kann und eine Prototyp für die Fehlersuche in JavaScript Programmen implementiert.

Bezüglich der Vorgehensweise, beruht unser Ansatz darauf, unterschiedliche Korrektheitseigenschaften zu definieren, die unterschiedliche Charakteristika bezüglich der Performance automatischer Analysen haben. In der Praxis werden dadurch Analysen ermöglicht, deren Präzision und Resourcennutzung auf den konkrteten Anwendungsfall angepasst werden können.
Die Implementierung des Prototyps basiert auf einer dynamischen Analyse von JavaScript Programmen, die zur Laufzeit mitläuft. Wir haben dazu zwei unterschiedliche Ansätze basierend auf Instrumentierung des Quellcodes bzw.

des Interpreters umgesetzt. Während für das konkrete Szenario die Quellcode Instrumentierung besser skaliert, ist die Instrumentierung im Interpreter leichter auf neue Analysen erweiterbar.

Das erste Kernergebnis dieser Arbeit ist die Definition von zwei Korrektheitseigenschaften. Die erste Eigenschaft besagt, dass Wiederverwendung korrekt ist, wenn eine Neuberechnung zum gleichen Ergebnis führen würde. Die zweite Eigenschaft besagt, dass zwei aufeinanderfolgende Berechnungen mit äquivalenten Inputs das gleiche Ergebnis produzieren müssen. Wir beweisen, dass die beiden Charakterisierungen gleichbedeutend sind. Besonders die zweite Korrektheitseigenschaft ähnelt in ihren Grundzügen der formalen Definition von Heisenbugs, die im nächsten Meilenstein entwickelt wurde (siehe unten) und hat einen Erkenntnisgewinn für die allgemeine Definition gebracht.

Das zweite Kernergebnis ist die Implementierung eines Prototyps für die automatische Analyse von JavaScript Programmen. Der Prototyp implementiert zwei unterschiedliche Analysen basierend auf den beiden Korrektheitseigenschaften. Eine Evaluierung auf Benchmarks, die auf realistischen Szenarien beruhen, zeigt, dass die Analysen erfolgreich Fehler finden.
Eine große Herausforderung bei der Entwicklung von dynamischen Analysen ist die Skalierbarkeit – während möglichst viele Fehler erkannt werden sollen, sollte die Laufzeit durch die Analyse nicht zu sehr verlängert werden. Initiale Ergebnisse zeigen, dass insbesondere die zweite Chararkterisierung und Analyse gute Skalierbarkeitseigenschaften aufweisen.

Eine initiale Version der Implementierung sowie Dokumentation dieser Arbeit in Form eines Papers, liegen vor. Allerdings wurde das Paper bisher nicht zur Veröffentlichung akzeptiert. Hilfreiches Feedback aus den Reviews hat ergeben, dass es wichtig ist, die Problemstellung etwas zu verallgemeinern. Die momentane Umsetzung konzentriert sich vor allem auf das Wiederverwenden von Ergebnissen in Form von Memoization. Die Ergebnisse sind aber auf weitere Anwendungen erweiterbar. Das erfordert einen Schritt der Überarbeitung, für die Anfang 2024 Zeit im neuen Projektplan vorgesehen ist.

2.2 Meilenstein 2 - Formale Definition von Heisenbugs

Ziel dieses Meilensteins war es, eine formal exakte Definition des Begriffs Heisenbug zu geben. Dazu habe ich zunächst eine Literaturrecherche gemacht, um eine Sammlung der verschiedenen Konzepte zu erstellen, die mit dem Begriff Heisenbug bezeichnet werden, und eine Reihe typischer Beispiele betrachtet. Im nächsten Schritt habe ich ein formales Model festgelegt, das aussagekräftig genug ist, um die identifizierten Konzepte zu beschreiben. Schließlich habe ich

einen passenden Formalismus für die Spezifikation ausgewählt und die formale Definition entworfen.

Bezüglich der Vorgehensweise, war es spannend zu sehen, dass der Begriff Heisenbug zwar in der wissenschaftlichen Literatur vorkommt, aber auch stark durch die Verwendung in nichtwissenschaftlichen Artikeln, Blogposts etc. zum Thema Software Engineering geprägt ist. Um dem Rechnung zu tragen, haben wir versucht eine Definition zu finden, die gängigen Beispiele aus verschiedenen Quellen abdeckt.
Die Definition des formalen Modells und der verwendete Spezifikationsformalismus hängen eng miteinander zusammen. Daher hat das Design mehrere Iterationen durchlaufen, bis eine passende Lösung gefunden wurde. So gibt es zum Beispiel unterschiedlichste Ansätze, wie man User Inputs modellieren kann und sie in der Definition behandelt.

Die Literaturrecherche hat ergeben, dass der Begriff Heisenbug in der Software Enigneering Community vor allem verwendet wird, um Fehler zu beschreiben, die schwer aufzuspüren sind, weil ihr Verhalten durch Mechanismen, die zur Fehlersuche eingesetzt werden, modifiziert wird. In der wissenschaftlichen Literatur zu formalen Methoden wird der Begriff verwendet, um Fehler zu beschreiben, die aufgrund von Nichtdeterminismus im System nur in manchen Ausführungen auftreten. Es ist uns gelungen eine formale Definition zu finden, die beide Konzepte auf ähnliche Weise behandelt, was unsere Arbeit deutlich von bisherigen Arbeiten abhebt und ein großer Erfolg ist.
Als formales Modell haben wir Transitionssysteme gewählt, die basierend auf Benutzerinneneingaben Zustandsübergänge machen. Benutzerinneneingaben spielen in vielen Beispielen von Heisenbugs, die in der Praxis beobachtet werden, eine Rolle.
Bezüglich der Spezifikation haben wir herausgefunden, dass Heisenbugs durch eine *Hyperproperty* beschrieben werden können. Anders als gängige *Properties*, die korrektes Systemverhalten als Eigenschaft einzelner Ausführungen des Systems definieren, basieren Hyperproperties auf einem Vergleich mehrerer Systemausführungen. Hyperproperty basierte Definitionen sind Gegenstand aktueller Forschung, eine passende Anwendung auf ein neues Gebiet ist daher besonders herausfordernd.

In einfachen Worten, besagt unsere Definition von Heisenbugs, dass es zwei Ausführungen im System gibt, die die gleichen Benutzerinneneingaben haben, aber unterschiedliche Ergebnisse produzieren.
Das Hauptergebnis dieses Meilensteins ist eine logische Formel, die diese Definition beschreibt.

Die Arbeit an diesem Meilenstein verlief weitgehend nach Plan. Die Ausarbeitung hat einen Monat länger gedauert als ursprünglich vorgesehen, was vor allem mit der Komplexität des Formalismus zusammenhängt, der nötig war, um das Phänomen Heisenbug in all seinen Facetten exakt zu beschreiben.

1 Zusammenfassung Planaktualisierung

Folgende Punkte haben sich am Zeitplan geändert:

- Die Arbeit am Milestone „Formale Definition von Heisenbugs“ wurde um einen Monat bis Juni 2023 verlängert.
- Daher wurde die Einreichung des Zwischenberichts von Juni 2023 auf Juli 2023 verschoben.
- Da ich im Herbst 2023 neben der Projekttätigkeit für 3 Monate ein Praktikum mache, verlängert sich der Zeitraum für den Milestone „Kausalität von Heisenbugs“ bis Dezember 2023.
- Wie oben erklärt, ist die Veröffentlichung für den Milestone „Memoization Bugs in JavaScript“ noch nicht abgeschlossen. Dafür wurde im Jänner 2024 zusätzlich Zeit eingeplant.
- Die restlichen Milestones verschieben sich dementsprechend nach hinten.