Experiment über den Einfluss des Fehlerpfades von CPAchecker beim Finden und Verbesseren von Fehler

Driemeyer Alexander

Buchecker Andreas

I. Einleitung

Software von hoher Komplexität ist fehlerbehaftet, selbst bei Anwendungen, in denen ein Fehler die Gefährdung von Menschenleben zur Folge hat. Das Finden und Beheben von Fehlern in Software stellt einen erheblichen Aufwand dar, und es wird an verschiedenen Techniken geforscht, Fehler ausfindig zu machen. Ein Bereich, in den diese Techniken fallen, ist die Softwareverifikation, welche die Abwesenheit von Fehlern in Software zu beweisen versucht. CPAchecker ist ein Programm, dass verschiedene Techniken der Softwareverifikation implementiert. Falls CPAchecker einen konkreten Fehler in der Software findet, wird von ihr ein Fehlerpfad ausgegeben. Unser Ziel ist es zu bestimmen, ob und in wie weit der Fehlerpfad von CPAchecker hilft, den Fehler in einem Programm zu finden und zu verbessern.

II. Ziel

Die konkrete Forschungsfrage, die unser Experiment adressieren soll, lautet:
Hilft der Fehlerpfad von CPAchecker, den Fehler in einem C-Programm zu finden und zu verbessern?
Unter Fehler in einem Programm verstehen wir in diesem Zusammenhang die Verletzung der Spezifikation des Programms. Die Spezifikation eines Programms definiert eine Menge von unsicheren Zuständen, in denen sich ein Programm nicht befinden darf. Ein Fehlerpfad von CPAchecker repräsentiert also einen Pfad des originalen Programms, der in einen unsicheren Zustand führt.
Wir betrachten ein Programm als verbessert, wenn sich ein Programm bei der Ausführung niemals in einem unsicheren Zustand befindet. Fehler, die dazu führen, dass ein Programm nicht ein gewünschtes Verhalten zeigt, werden nicht betrachtet, da solche Fehler nicht durch einen endlichen Fehlerpfad demonstriert werden können, und die Erstellung eines Fehlerpfades somit nicht möglich ist.

Wenn wir vom Fehlerpfad von CPAchecker sprechen, meinen wir nicht nur den reinen Fehlerpfad, welcher aus dem Pfad der CFA-Kanten besteht, der den Pfad des Programms repräsentiert, den das Programm beim Auftreten des Fehlers genommen hat. Wir verstehen unter dem Fehlerpfad von CPAchecker auch den Bericht, den CPAchecker im HTML-Format generiert. Dieser visualisiert den eigentlichen Fehlerpfad im CFA, ARG und im Quelltext. Zudem enthält er zusätzliche Informationen, wie konkrete Werte für Variablen, die zur Interpretation des Fehlerpfades nützlich sind.

Augenscheinlich sollten die zusätzlichen Informationen, die der Fehlerpfad von CPAchecker bereitstellt, bei der Identifizierung des Fehlers helfen, indem die Statements und die relevanten Belegungen der Variablen angezeigt werden, die den Fehler ausgelöst haben. Dies sollte sowohl zu einer korrekteren, wie schnelleren Identifizierung und Verbesserung eines Fehlers führen, als wenn nur der Quelltext des Programms benutzt wird.

Die konkrete Hypothese, die wir in diesem Experiment untersuchen, lautet: Durch die Benutzung des Fehlerpfades von CPAchecker, ist es möglich, korrekter einen Fehler in einem Programm zu finden und zu verbessern, als lediglich den Quelltext zu benutzen.

Zudem wird folgende Hypothese untersucht:

Durch die Benutzung des Fehlerpfades von CPAchecker ist es möglich, einen Fehler schneller in einem Programm zu finden und zu verbessern, als lediglich den Quelltext zu benutzen.

Diese Hypothesen spiegeln das erwartete Ergebnis wieder.

III. Experiment

A. Entwurf

Im Experiment mussten 6 Probanden 3 Aufgaben lösen. Alle Probanden hatten ein abgeschlossenes Informatikstudium und grundlegende Erfahrung mit den Programmiersprachen Java und/oder C. Der Grund, warum wir gerade diese Probanden ausgewählt haben, ist, dass sie verfügbar waren, und es sich zudem gezeigt hat, dass die Ergebnisse von Studenten für Experimente im Zusammenhang mit Programmverständnis genau so gut verwertet werden können wie für Experten.
Bei den Aufgaben ging es darum einen Fehler in einem Quelltext zu finden und dazu folgende Fragen zu beantworten: ”Wo tritt der Fehler auf?”, ”Unter welchen Umständen tritt der Fehler auf?” und ”Wie kann man den Fehler korrigieren?”.
Dabei wurden die Probanden in 2 Gruppen aufgeteilt. Der Unterschied zwischen den 2 Gruppen war, dass eine dieser Gruppen zusätzlich zum Quelltext, auch den Fehlerpfad des Verifikationstools CPAchecker erhalten hat, während die andere Gruppe nur den Quelltext zur Verfügung hatte um die Aufgabe zu lösen. Dies ist unsere unabhängige Variable. Diese hat zwei Stufen. Die erste Stufe ist mit Fehlerpfad, die zweite ist ohne Fehlerpfad.
Der Quelltext der Aufgaben ist in der Programmiersprache C geschrieben, da die Features, die wir von CPAchecker für die Generierung des Fehlerpfades benutzt haben, nur für C-Code verfügbar ist. Allerdings wurden keine C-spezifischen

Konstrukte verwendet, wodurch der entsprechende Java-Code fast genauso aussehen würde. Damit wollten wir erreichen, dass alle Probanden den Code verstehen können, egal ob sie Erfahrung mit Java oder C haben.
Die erste Aufgabe war ein Bubblesort-Algorithmus, die zweite eine Implementierung einer queue und die dritte hat einen Algorithmus zur Berechnung eines minimalen Spannbaums beschrieben. Den Quelltext der drei Aufgaben haben wir aus dem Internet. Dabei haben wir zuerst eine Menge von in Frage kommenden Quelltexten, die einen Algorithmus implementiert haben, und frei im Internet verfügbar waren, heruntergeladen. Da wir das Experiment innerhalb eines Zeitrahmens von maximal 2 Stunden mit Einführung und Organisation planten, haben wir aus dieser Menge Quelltexte ausgewählt, von dem wir glaubten, dass sie innerhalb von 10, 20 oder 30 Minuten bearbeitet werden können. Wir haben uns jeweils eine Spezifikation für die Algorithmen, die von den Quelltexten implementiert wurden, überlegt, und jeweils einen Fehler in die Quelltexte eingebaut. Wir haben außerdem die Spezifikation als Teil des Programms durch Konditionen und Schleifen abgefragt. Somit mussten die Probanden nur entscheiden, ob eine der mehrmaligen Aufrufe der Funktion __VERIFIER_ERROR() im Programm ausgeführt wird, um den Fehler zu finden. Wir haben anhand von mehreren Versuchspersonen die Zeit gemessen, die sie für das Finden und Verbessern des Fehlers benötigt haben. Dadurch haben wir 3 passenden Aufgaben aussortiert. Erst dann haben wir die Aufgaben durch CPAchecker verifizieren und den Fehlerpfad generieren lassen. Manche Aufgaben mussten im Nachhinein noch leicht modifiziert werden, damit CPAchecker sie erfolgreich verifizieren konnte. Da wir den konkreten Fehlerpfad für eine Aufgabe erst gesehen haben, nachdem wir die Aufgabe erstellt und ausgewählt haben, konnten wir vermeiden, bei der Auswahl der Aufgaben vom Aussehen des Fehlerpfades beeinflusst zu werden. Die Aufgaben sind insofern repräsentativ , dass die Quelltexte ursprünglich für die Implementierung eines der oben genannten Algorithmen entwickelt wurde, und nicht speziell für unser Experiment.
Der Fehlerpfad selbst ist eine von CPAchecker generierte Datei im HTML-Format. Dieser wird durch einen Browser angezeigt, und enthält neben dem eigentlichen Fehlerpfad auch den Quelltext des Programms, sowie den CFA, ARG und weitere Statistiken.
Um Zeitdruck zu vermeiden, haben wir den Probanden insgesamt eine Bearbeitungszeit von 90 Minuten gegeben.
Da die Programmiererfahrung beim Lösen der Aufgabe einen großen Einfluss hat, wurde die Programmiererfahrung der Probanden vor dem Experiment mit einem Fragebogen gemessen. Dadurch konnten wir die Gruppen nach dem Kriterium der Programmiererfahrung ausbalancieren. Der Fragebogen zur Messung der Programmiererfahrung der Probanden wurde dabei nicht von uns selbst erstellt, sonder war bereits vorgefertigt, und hat sich unter anderen Umständen bewährt. Dadurch wurde gewährleistet, dass wir uns auf das Ergebnis des Fragebogens verlassen können. Ebenfalls haben wir mit dem Fragebogen die Erfahrung mit CPAchecker gemessen. Dadurch können wir sicherstellen, dass in der Gruppe, die den Fehlerpfad verwenden durfte nicht nur Probanden waren, die bereits Erfahrung mit CPAchecker haben.
Beim Experiment-Design haben wir uns für Between-Subjekt entschieden. Die Gründe warum wir weder Within-Subject noch Cross-Over verwendet haben, sind folgende. Da man bei Within-Subject nur eine Gruppe hat, ist der Störfaktor des Lerneffekts und des Reihenfolgeeffekts hier am größten. Diese Störfaktoren könnte man zwar am besten mit Cross-Over mildern, allerdings hatten wir mit Cross-Over ein anders Problem. Da wir bei Cross-Over zwei Gruppen gebraucht hätten, die jeweils einmal mit Fehlerpfad und einmal ohne Fehlerpfad Aufgaben lösen hätten müssen, wär die Dauer des Experiments im Vergleich zu Between-Subject mindestens verdoppelt worden. Durch die längere Dauer des Experiment schien die Motivation der möglichen Probanden beim Experiment teilzunehmen stark zu sinken, weshalb wir nicht genug Probanden finden konnten. Das selbe Problem wäre auch bei Within-Subject aufgetreten, da auch hier die Experimentdauer verdoppelt worden wäre. Wir haben uns schließlich für Between-Subject entschieden, da wir bei diesem Design keinen Lern- und Reihenfolgeeffekt haben. Wir haben gleichzeitig die Dauer des Experiments halbieren können, und so genug Probanden finden können.

B. Messung

Bei unserem Experiment haben wir 2 Dinge gemessen. Diese stellen unsere abhängigen Variablen dar. Zum einen haben wir gemessen, wie lange die Probanden zum Lösen der einzelnen Aufgaben brauchen. Zum anderen haben wir die Korrektheit der Aufgabenbearbeitung gemessen. Für die Korrektheit haben wir den Probanden pro Aufgabe 3 Fragen gestellt, die beantwortet werden mussten. Wir haben die Antworten der Probanden in zwei Kategorien eingeteilt. Wenn die Frage zu unserer Zufriedenheit richtig beantwortet wurde, haben wir die Antwort in die Kategorie korrekt eingeordnet. Wenn die Frage falsch beantwortet wurde, haben wir die Antwort in die Kategorie falsch eingeordnet. In den folgenden Tabellen steht ’1’ für korrekt und ’0’ für falsch. Es gab keine Antwort, bei der wir uns uneinig waren, ob sie als korrekt oder falsch zu interpretieren ist. Bei der nachträglichen Analyse hat sich zudem herausgestellt, dass beide Gruppen von Probanden ihre Fragen mit ähnlicher Ausführlichkeit beantwortet haben, weshalb eine simple Projektion in korrekt und falsch möglich war.
Alle Messungen und Antworten wurden durch das Tool durchgeführt und gespeichert, das die Probanden zum Bearbeiten der Fragen verwenden mussten. Dieses Tool wurde von uns programmiert. Es ist dabei sehr einfach und intuitive gehalten. Es zeigt den Pfad zur Aufgabe an, die gerade bearbeitet wird. Für die Gruppe mit Fehlerpfad wird außerdem noch angezeigt, wo der Fehlerpfad gefunden werden kann. Außerdem wird noch abgefragt ob der Fehlerpfad beim Lösen der Aufgabe verwendet wurde. Dadurch wollten wir verhindern, dass die Werte derjenigen, die den Fehlerpfad zwar vor sich haben, ihn jedoch nicht benutzt haben, da sie beispielsweise die richtige Antwort auf den ersten Blick gesehen haben, das Ergebnis beeinflussen. Des Weiteren misst das Tool die Zeit der Aufgabe, die aktuell bearbeitet wird. Da die Zeit der einzelnen Aufgaben direkt vom Programm gemessen wurde, war sie für die Zwecke dieses Experiments präzise genug. Durch vorläufige Tests hat sich gezeigt, dass der Unterschied zwischen gemessener und tatsächlich vergangener Zeit nicht über 1 Sekunde hinausgeht. Bei beiden Teams wurde die Zeit, die sie zum Öffnen der Quelltexte und des Fehlerpfades einer Aufgabe benötigt haben, bis zur Absendung der Antwort durch einen Knopf unseres Programms, gemessen.

	Fehlerpfad	kein Fehlerpfad	Summe
richtiger Antworten	19	17	36
falscher Antworten	8	10	18
Summe	27	27	54

TABLE I: Korrektheit über alle Fragen

Die Zeit, die man bereits mit der Aufgabe verbracht hat, wird nicht angezeigt, um die Probanden nicht unter Zeitdruck zu setzen. Außerdem werden die Antworten zu den Fragen der einzelnen Aufgaben, sowie die Zeit in einer Datei gespeichert.

C. Durchführung

6 Probanden haben an dem Experiment teilgenommen. Die Probanden wurden in 2 Gruppen von je 3 Personen aufgeteilt. Dazu haben alle Probanden ein paar Tage vor dem Experiment einen Fragebogen ausgefüllt, der ihre allgemeine Programmiererfahrung, ihre Erfahrung mit verschiedenen Programmiersprachen, sowie ihre Erfahrung mit CPAchecker gemessen hat. Nach diesen Kriterien wurden die 2 Gruppen ausbalanciert. Die beiden Gruppen wurden in 2 Räume aufgeteilt, da jede Gruppe eine eigene Einführung vor dem Experiment erhalten hat.
Die Gruppe die keinen Fehlerpfad erhalten hat, bekam eine Einführung, was sie bei den Aufgaben machen mussten und wie das Tool, dass sie zum Beantworten der Aufgaben verwenden mussten, funktioniert. Dazu wurde eine kleine Einführungsaufgabe vorgestellt. Gleichzeitig wurde auch überprüft ob das Tool bei allen funktioniert hat. Das Ganze hat ungefähr 10 Minuten gedauert. Bei der Einführung der Gruppe mit dem Fehlerpfad wurde zusätzlich noch an der Einführungsaufgabe erklärt, wie der Fehlerpfad von CPAchecker funktioniert. Damit haben wir erreicht, dass auch die Probanden, die keine Erfahrung mit CPAchecker hatten, wussten wie dieser zu verstehen ist. Diese Einführung hat ca. 25 Minuten in Anspruch genommen.
Nach der Einführung hatte jeder Proband 90 Minuten Zeit 3 Aufgaben zu bearbeiten. Um Zeitdruck zu vermeiden gab es für die einzelnen Aufgaben kein Zeitlimit.
Bei der Durchführung des Experiments sind kleiner Abweichungen aufgetreten. Die Probanden des Experiments mit dem Fehlerpfad haben sich während des Experiments gegen Anfang und Mitte kurz über CPAchecker unterhalten. Dies hat die Zeitmessung minimal beeinflusst. Zudem fehlte die Farbmarkierung des Fehlerpfades im CFA bei einer Person, was an einem unbekannten Bug im Browser gelegen haben könnte.

IV. Auswertung

In diesem Kapitel beschreiben wir die Ergebnisse des Experiments und werten diese aus. Dabei unterscheiden wir zwischen den Ergebnissen für die Korrektheit der Aufgaben, und den Ergebnissen für die Zeit, die gebraucht wurde, um die Fragen zu lösen. Die Daten, die wir hier verwenden, kommen von insgesamt 6 Probanden, die in 2 gleichgroße Gruppen aufgeteilt wurden.

A. Analyse der Korrektheit

Die Korrektheit haben wir wie folgt gemessen: Jede der 3 Fragen der 3 Aufgaben gibt einen Punkt bei korrekter Antwort.

	Fehlerpfad	kein Fehlerpfad	Summe
richtiger Antworten	8	4	12
falscher Antworten	1	5	6
Summe	9	9	18

TABLE II: Korrektheit über Frage 1

	Fehlerpfad	kein Fehlerpfad	Summe
richtiger Antworten	6	6	12
falscher Antworten	3	3	6
Summe	9	9	18

TABLE III: Korrektheit über Frage 2

Das heißt ein Proband kann maximal 9 Punkte erreichen. Diese Punkte werden dann zu der jeweiligen Gruppe des Probanden hinzugefügt. Außerdem haben wir auch die falsch beantworteten Ergebnisse gezählt. Diese Daten haben wir dann in einer 2x2 Matrix zusammengefasst, wobei die Spalten für die Gruppen stehen, und die Zeilen für die Anzahl der Richtig bzw. Falsch beantworteten Fragen stehen. Wir haben dabei verschiedene Aufteilungen der Daten vorgenommen. In Tabelle II werden nur die Antworten auf die erste Frage jeder Aufgabe betrachtet. In Tabelle III nur die der zweiten Frage, in Tabelle IV nur die der dritten Frage und in Tabelle I werden alle Fragen zusammengefasst. Außerdem haben wir zusätzlich noch eine weitere Variation der Korrektheit untersucht. Dabei werden nur die Aufgaben als korrekt angesehen, bei denen alle Fragen korrekt beantwortet worden sind. Das Ergebnis davon kann man in Tabelle V sehen.
Wenn wir die Daten von Tabelle III, IV und V ansehen, können wir keinen bis kaum einen Unterschiede zwischen der Gruppe mit Fehlerpfad (FPG) und der Gruppe ohne Fehlerpfad, auch Kontrollgruppe (KG) genannt, erkennen. Lediglich in Tabelle II , die die Korrektheit der ersten Frage jeder Aufgabe darstellt, zeigt auf den ersten Blick signifikante Unterschiede. Allerdings verschwinden diese Unterschiede bei der Zusammenfassung der Fragen, wie es in Tabelle I dargestellt ist, wieder.
Um zu Überprüfen ob ein signifikanter Unterschied zwischen beiden Gruppen besteht, führen wir einen entprechenden Signifikanztest durch. Da wir hier eine 2x2 Matrix, auch Vierertafel genannt über nominalen Daten haben, würde sich ein Chi-Quadrat Test anbieten. Aufgrund der wenigen Daten

	Fehlerpfad	kein Fehlerpfad	Summe
richtiger Antworten	5	6	11
falscher Antworten	4	3	7
Summe	9	9	18

TABLE IV: Korrektheit über Frage 3

	Fehlerpfad	kein Fehlerpfad	Summe
richtiger Antworten	4	5	9
falscher Antworten	5	4	9
Summe	9	9	18

TABLE V: Korrektheit über komplett gelöste Aufaben

Probanden ID	Gruppe	Zeit in s
11	1	535
13	1	1811
14	1	890
12	2	638
15	2	600
16	2	1016

TABLE VI: Zeiten von Aufgabe 1: Gruppe 1: FPG, Gruppe 2: KG

Gruppe	1	2
Max	1811	1016
Min	535	600
Mittel	1078	751
Mean	890	638

TABLE VII: Min,Max, Mean und Mittel von Aufgabe 1 : Gruppe 1: FPG, Gruppe 2: KG

und geringen erwarteten Häufigkeiten, greifen wir stattdessen auf die Variante Fisher's Exact Test zurück. Mit diesem Test überprüfen wir ob eine Nullhypothese abgelehnt wird. Dies ist bei einem p-Wert kleiner 0,05 der Fall. Da wir überprüfen wollen ob ein Unterschied zwischen den beiden Gruppen besteht, formulieren wir die Nullhypothese für den Fisher Exact Test wie folgt:
H0: Es besteht kein Unterschied zwischen der FPG-Gruppe und der KG-Gruppe

Für die Fragen 2 und 3 der Aufgaben (siehe Tabelle III und IV erhalten wir einen p-Wert von 1. Da dieser Wert deutlich größer als 0,05 ist kann die Nullhypothese, dass kein Unterschied zwischen beiden Gruppen besteht nicht abgelehnt werden. Die gilt ebenso bei der Zusammenfassung der Aufgaben wie in Tabelle I zu sehen. Hier erhalten wir einen p-Wert von 0,77. Auch bei der alternativen Interpretation von Korrektheit (siehe Tabelle V) erhalten wir einen p-Wert von 1. Somit wird auch hier die Nullhypothese beibehalten. Einzig für Frage 1 (Tabelle II) erhalten wir eine p-Wert von 0,29. Es ist anzumerken das dieser p-Wert, im Vergleich zu den anderen deutlich näher an der Schranke von 0,05 ist. Allerdings ist der p-Wert trotzdem noch deutlich größer als 0,05, und somit wird auch hier die Nullhypothese beibehalten.
Zusammenfassend können wir bei keinen der Daten einen Unterschied zwischen der FPG-Gruppe und der KG-Gruppe feststellen. Daher wird unsere Hypothese, dass Fehler mit Hilfe des Fehlerpfads von CPAchecker korrekter gefunden und verbessert werden können, von unseren Daten abgelehnt.

B. Analyse der Zeit

Die Aufgaben 2 und 3 wurden von fast niemanden richtig gelöst. Außerdem gab es auch ein paar leere Antworten, was bedeutet, dass die Probanden aufgegeben haben diese Aufgabe zu lösen. Darum ist es nicht sinnvoll, die Daten von Aufgabe 2 und 3 im Hinblick auf Zeit auszuwerten.

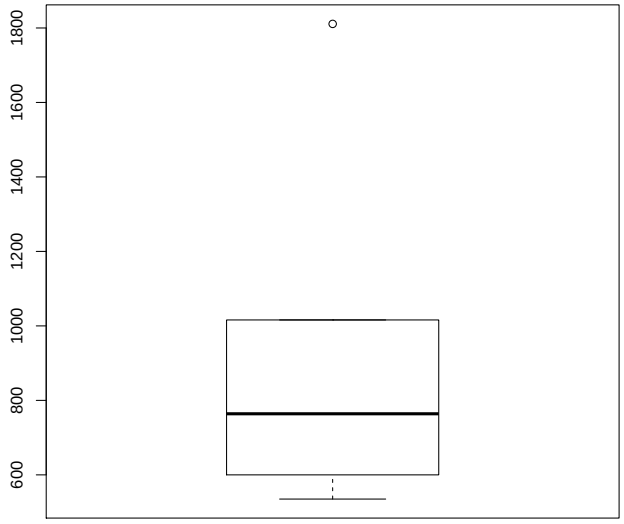

Fig. 1: Bloxplot der Zeit für Aufgabe 1

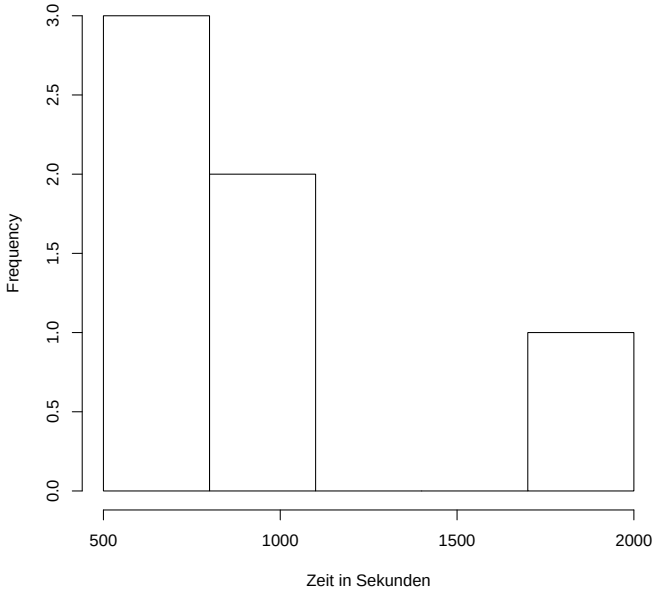

Fig. 2: Historamm der Zeit für Aufgabe 1

Probanden ID	Gruppe	Zeit A1 in s	Zeit A2 in s	Zeit A3 in s	Summe
11	1	535	2175	1299	4009
13	1	1811	1619	792	4222
14	1	890	1829	1524	4243
12	2	638	1750	982	3370
15	2	600	1971	1729	4300
16	2	1016	2825	676	4517

TABLE VIII: Zeiten aller Aufgaben: Gruppe 1: FPG, Gruppe 2: KG

Deshalb beschränken wir uns bei der Auswertung der Zeit auf Aufgabe 1, die von allen Probanden richtig gelöst wurde. Wenn wir die Daten in Tabelle VI betrachten, fällt vorallem der Ausreißer in der Fehlerpfadgruppe (FPG) auf. Dieser Ausreißer ist auch sehr gut im Boxplot von Abbildung 1 zu sehen. Desweiteren sieht man im Bloxplot das sich fast alle anderen Daten innerhalb der zwei Quantile des Boxplots befinden. Berechnen wir das Maximum, das Minimum, das arithmetrische Mittel und den Median für die EPG-Gruppe und für die Gruppe ohne Fehlerpfad (KG), erhalten wir die Daten, wie sie in Tabelle VII zu sehen sind. Bei Betrachtung der Daten fällt auf, dass die KG Gruppe ein deutlich niedrigeres Maximum hat. Die beiden Maxima unterscheiden sich um 13 Minuten. Das Minimum dagegen unterscheidet sich nur um ca. 1 Minute. Auch das arithmetrische Mittel fällt bei der KG-Gruppe niedriger aus. Allerdings, wegen dem Ausreißer in der FPG-Gruppe und der geringen Anzahl an Daten, ist es sinnvoller das Median zu betrachten. Aber auch dieses ist in der KG-Gruppe niedriger. Das Median über beide Gruppen, wie im Boxplot von Abbildung 1 zu sehen ist, teilt den Bereich zwischen den 2 Quantilen ca. in der Mitte. Nach einer ersten Analyse scheint es so, dass die KG-Gruppe die besseren Bearbeitungszeiten hat, was der Hypothese, dass der Fehlerpfad die Fehlersuche beschleunigt, widerspricht.
Um zu testen ob eine signifikanter Unterschied zwischen den Daten der 2 Gruppen besteht, brauchen wir einen passenden Signifikanztest. Zu aller erst haben wir untersucht, ob die Daten normalverteilt sind. Dazu haben wir den Shapiro-Willk-Test durchgeführt und zusätzlich ein Histogramm über die Zeiten der ersten Aufgabe erstellt. Das Histogramm sieht man in der Abbildung 2. Nach einen Blick auf das Histogramm fällt auf, dass die Daten nicht normalverteilt aussehen. Beim Shapiro-Willk-Test erhalten wir einen p-Wert von 0,07. Dieser liegt nur sehr knapp über dem Grenzwert von 0,05. Daher werden die Daten nur sehr knapp als normalverteilt eingestuft. Aus diesem Grund, und weil wir allgemein nur sehr wenige Daten haben, ist ein T-Test nicht sinnvoll.
Darum verwenden wir den Mann-Whitney-U-Test. Der Mann-Whitney-U-Test prüft, ob zwei Datenmengen aus der selben Population sind oder nicht. Dafür wird eine Null-Hypothese und eine dazugehörige Alternativ-Hypothese aufgestellt.
Die vorliegenden Hypothesen für unsere Experiment sind folgende:
H0: Die Daten für die Bearbeitungszeit der Aufgaben mit und ohne Fehlerpfad sind von der selben Population
H1: Die Daten kommen von verschiedenen Populationen

Normalerweise wird mit einem p-Wert von 0,01, 0,05 oder 0,1 verglichen. Dabei bedeutet eine niedrigerer p-Wert, dass die Nullhypothese stärker abgelehnt wird. Wir erhalten bei der Berechnung einen p-Wert von 1. Da dieser p-Wert alle üblichen Vergleichswerte bei weitem übersteigt, heißt das, dass die Nullhypothese nicht abgelehnt wird. Da wir einen starken Ausreißer in der FPG-Gruppe haben, haben wir den Whitney-U test noch einmal ohne den Ausreißer wiederholt. Dabei erhalten wir einen p-Wert von 0,8. Da auch dieser Wert weit über den Vergleichswerten liegt, wird auch hier die Nullhypothese beibehalten. Das Ergebnis aus den beiden Whitney-U-Tests ergibt, dass die Daten aus der selben Population kommen. Das wiederum bedeutet das mit diesen Daten, unsere Hypothese, dass das Finden und Verbessern von Fehlern mit dem Fehlerpfad von CPAchecker beschleunigt wird, abgelehnt wird.

C. Bedrohung der Validität

Für empirische Forschung ist es wichtig alle möglichen Elemente zu nennen, die eine Bedrohung der Validität darstellen könnten.
Das vermutlich stärkste Argument gegen die Validität unseres Experiments, ist die Auswahl der Aufgaben. Dabei lässt sich die Bedrohung in verschiedene Aspekte aufteilen. Der erste Aspekt ist, dass die Aufgaben so ausgewählt wurden, dass sie unsere Hypothese unterstützen. In unserem Fall wäre das der Fall, wenn der Fehlerpfad bei unseren Aufgaben besonders hilfreich zum Auffinden und Verbessern des Fehlers wäre. Dies haben wir verhindert, indem wir den Fehlerpfad erst nach Auswahl der Quelltexte und dem Einbauen des Fehlers, berechnet haben. Zudem haben wir statt nur einer Aufgabe, drei verschiedene Aufgaben ausgewählt. Damit wollten wir zusätzlich verhindern, dass der Fehlerpfad einer einzelnen Aufgabe, zufällig besonders gut oder besonders schlecht für die Bearbeitung der Aufgabe ist.
Ein weiterer Aspekt ist, dass unsere Aufgaben möglicherweise nicht ausreichend repräsentativ sind. Da die Quelltexte für unsere Aufgaben, ursprünglich für die in Section III-A genannten Algorithmen entwickelt wurden, sollten die Aufgaben zumindest für interne Validität ausreichend repräsentativ sein. Für externe Validität sind die Aufgaben mit 10, 20 und 30 Minuten Bearbeitungszeit, vermutlich zu klein gewählt. Dies konnten wir aber wegen dem bereits genannten Problems, der zu langen Experimentdauer nicht anderes lösen. Dessweiteren ist die externe Vaildität nicht gesichert, aufgrund zu weniger Probanden.
Auch die Motivation der Probanden kann das Ergebniss stören. Da alle Probanden freiwillig an dem Experiment teilgenommen haben, kann man davon ausgehen, dass eine grundlegende Motivation zu Beginn des Experiments vorhanden war. Ein weitere Faktor bei Motivation, der zu beachten ist, ist das diese auch während des Experiments absinken kann und so das Ergebnis beeinflusst. Bei unserem Experiment würde ein mögliches Absinken der Motivaton am wahrscheinlichsten an der Länge des Experiments liegen und die dazugehörige Ermüdigung. Aufgrund unserer kleinen Anzahl an Probanden, hätte diese Form der Motivationssenkung einen besonders starken Einfluss. Da die Länge des Experiments mit 1,5 Stunden ohne Pause, relativ klein ist, sollte das allerdings kein Problem sein.
Eine weitere Bedrohung ist die unterschiedliche Programmiererfahrung der Probanden. Da alle Probanden bereits ein Informatikstudium abgeschlossen haben, kann die Programmiererfahrung sehr stark variieren. Diese Bedrohung haben wir versucht durch ausbalancierte Gruppen zu kompensieren. Dazu

mussten die Probanden einen Fragebogen ausfüllen der ihre Programmiererfahrung misst. Allerdings besteht immer noch die Möglichkeit, dass sich Probanden im Fragebogen falsch eingeschätzt haben, was ihre Programmiererfahrung betrifft.
Neben der Programmiererfahrung ist auch die Toolerfahrung eine ernste Bedrohung für die Validität. Die Tools, die die Probanden in unserem Experiment verwendet, haben waren zu einem der Fehlerpfad, und zum anderen das Programm, dass die Zeiten und Antworten aufgezeichnet hat. Für beide Tools gab es eine Einführung vor dem Experiment. Da der Fehlerpfad aber als kompliziert eingestuft werden kann, kann man argumentieren, dass wir die Einführung für den Fehlerpfad nicht ausreichend lang und ausführlich gemacht haben. Aufgrund der eingeschränkten Zeit, die wir für das Experiment hatten, konnten wir die Einführung nicht länger gestalten. Für das Messprogramm ist eine solche Argumentation allerdings nicht richtig, da es ein sehr einfaches und intuitives Programm ist.
Weiter kann begründet werden, dass Probanden mit mehr Erfahrung in der Programmiersprache C einen Vorteil haben, da alle Quelltexte in C geschrieben sind. Da keine der Quelltexte C-spezifische Konstrukte verwenden, würde der entsprechende Java-Quelltext ähnlich aussehen. Da alle Probanden entweder mit C oder mit Java Erfahrung haben, hat kein Proband deswegen Vor- oder Nachteile.
Auch eine fremde Arbeitsumgebung kann das Ergebnis verfälschen. Bei unserem Experiment ist diese fremde Umgebung dadurch gegeben, dass die Probanden nur den Quelltext des Programms erhalten haben. Die Möglichkeit das Programm auszuführen und/oder zu debuggen wurden den Probanden nicht gegeben, obwohl das Standardwerkzeuge zum Finden von Fehlern sind. Wir haben diese Umgebung bewusst so gewählt, weil wir befürchteten, dass auch die Probanden, die den Fehlerpfad zur Verfügung hatten, in gewohnte Arbeitsweisen zurückfallen würden und den Fehlerpfad überhaupt nicht verwendet hätten. Dadurch wären alle unsere Ergebnisse nutzlos geworden. Aber auch trotz der vorhandenen Arbeitsumgebung war es vor dem Experiment nicht auszuschließen, dass die Probanden den Fehlerpfad nicht benutzen. Darum haben wir auch abgefagt ob sie den Fehlerpfad für eine Aufgabe verwendet haben oder nicht. Das Ergebnis hiervon war, dass alle Probanden der Fehlerpfadgruppe diesen auch verwendet haben.
Eine Bedrohung der Konstruktvalidität ist, dass wir bei der Wahl unserer Fragen zu einer Aufgabe, und bei der anschließenden simplen Kategorisierung, das Verständnis des Probanden bezüglich des Fehlers nicht präzise genug abbilden. So wäre es theoretisch möglich, dass einige Probanden eine wesentlich genauere Vorstellung haben, unter welchen Umständen der Fehler auftritt, als andere Probanden. Wir würden beide Probanden jedoch als gleich korrekt behandeln. Dies ist bei unserem Experiment jedoch nicht der Fall, da eine nachträgliche Analyse ergeben hat, dass alle Probanden, die korrekt waren, den Fehler ähnlich ausführlich beschrieben haben. Durch die genaue Definition, was ein Fehler ist, und wann ein Fehler als verbessert angesehen werden kann, ist unserer Messung der Korrektheit der Aufgabenbearbeitung anhand der Fragen präzise. Die genaue Definition von korrekt wurde den Probanden während der Einführung mitgeteilt.

V. Interpretation

Die Resultate des Experiments stimmen nicht mit den erwarteten Ergebnissen überein. Die Probanden, die den Fehlerpfad benutzt haben, haben nur bei der Identifizierung des Fehlers besser abgeschnitten, als die Probanden, die den Fehlerpfad nicht benutzt haben. Sowohl bei der Frage, unter welchen Umständen der Fehler auftritt, als auch bei der Frage, wie der Fehler zu verbessern sei, haben die zusätzlichen Informationen aus dem Fehlerpfad nicht dazu geführt, dass die Probanden mit Fehlerpfad besser abgeschnitten haben als die Probanden ohne Fehlerpfad, wie aus Tabelle I zu beobachten. Die Fragen wurden zudem auch nicht von der Gruppe mit dem Fehlerpfad merklich ausführlicher beantwortet. Da das Experiment auf 90 Minuten beschränkt war, jedoch nach Tabelle VIII der Proband, der am längsten die Aufgaben bearbeitet hat, insgesamt 75 Minuten gebraucht hat, wurde das Ergebnis auch nicht durch Zeitdruck verfälscht. Da die Probanden keine Pause hatten und wir nur wenige Probanden finden konnten, könnten Ermüdungserscheinungen, Motivation sowie generelles Befinden das beobachtete Ergebnis verfälscht haben, speziell bei Aufgabe 3. Zudem sind Programmierer wesentlich gewöhnter, nur anhand des Quelltextes den Fehler zu finden, als den Fehlerpfad von CPAchecker und den Quelltext zu benutzen, und das Ergebnis würde sich verschieben, wenn alle Probanden ausführliche Erfahrung mit dem Fehlerpfad von CPAchecker haben.

Es gibt jedoch auch mehrere Möglichkeiten, die unsere Ergebnisse unter der Annahme, dass unser Experiment intern valide ist, erklären. Der Fehlerpfad ermöglicht zwar eine schnelle Identifizierung, welche Eigenschaft der Spezifikation des Programms verletzt ist, jedoch muss bei der Frage, unter welcher Situation der Fehler auftritt, und wie der Fehler zu verbessern ist, erst das Programm verstanden werden. In unserem Experiment hat der Fehlerpfad nicht signifikant dazu beigetragen, das Programm zu verstehen, da sich ansonsten die Korrektheit bei Frage 2 und 3 der Aufgaben merklich zugunsten der Probanden mit Fehlerpfad verschoben hätte. Dies könnte daran liegen, dass die Probanden die Informationen im Fehlerpfad schwer interpretieren konnten, oder dass diese Informationen wenig bei dem Verständnis von einem Programm beitragen. Eine weiterführende Frage ist, inwiefern es Sinn macht, einen Fehlerpfad lediglich für die Identifizierung von Fehlern in Programmen zu nutzen, und nicht für ihre Verbesserung. Durch den Umfang und die Art unseres Experimentes lassen sich diese Schlüsse jedoch nicht verallgemeinern. Augenscheinlich korreliert der Nutzen eines Fehlerpfades bei der Verbesserung und Identifizierung eines Programms mit der Größe des Programms, Komplexität des Programms und der Art des Fehlers. In weiteren Experimenten könnte diese Korrelation genauer untersucht werden, um herauszufinden,ob und ab wann die Benutzung eines Fehlerpfades zu signifikante Unterschiede bei der Verbesserung von Programmen führt. Zudem haben wir nicht getestet, inwiefern ein Fehlerpfad bei der Verbesserungen von Programmen hilft, wenn das Debugging und das Ausführung des Programms erlaubt sind.
Für die Aufgabe, die jeder Proband korrekt gelöst hat, hat sich auch kein signifikanter Unterschied zwischen den Zeiten der Probanden ergeben, wie aus Tabelle VI und dem Kapitel Auswertung ersichtlich ist. Da keine Informationen vorliegen, für welche Frage wie lange gebraucht wurde, lässt sich nicht

entscheiden, ob die Probanden mit Fehlerpfad bei der Identifizierung des Fehlers schneller waren, jedoch das Identifizieren des Fehlers insgesamt wenig Einfluss auf die Gesamtzeit der Aufgabe hatte, oder die Benutzung des Fehlerpfades keine signifikanten Auswirkung auf die Bearbeitungzeit hatte. Um weitere Schlüsse für die Bearbeitungszeit mit Fehlerpfad zu ziehen, muss ein präziseres Experiment entworfen werden.

VI. Fazit

In diesem Bericht präsentieren wir das Ergebnis unseres Experiments in Bezug auf den Einfluss der Benutzung des Fehlerpfades bei der Verbesserung von Programmen. Dazu haben wir die Ergebnisse bei der Verbesserung von Programmen zweier Gruppen von Probanden untersucht, wobei eine Gruppe den Fehlerpfad von CPAchecker zur Verfügung hatten. Wir haben die Zeit gemessen, die die Probanden für die einzelnen Aufgaben benötigt haben, sowie anhand ihrer Antworten bestimmt, ob sie den Fehler eines Programms korrekt identifiziert und verbessert haben. Es hat sich herausgestellt, dass weder bei der Korrektheit der Verbesserungen der Programme als bei der Schnelligkeit der Bearbeitung signifikante Unterschiede bei beiden Gruppen aufgetreten sind. Dieses Ergebnis widerspricht der augenscheinlichen Annahme, dass die Informationen im Fehlerpfad zur Korrektheit und Schnelligkeit bei der Verbesserung von Fehlern in Programmen beitragen. Es möglich, dass sich dieser Effekt nur bei größeren Programmen, oder einer anderen Art von Fehler zeigt. Durch den Umfang und der Art des Experiments lassen sich diese Ergebnisse, und die Schlüsse aus den Ergebnissen jedoch nicht verallgemeinern. Weiterführende Forschung sollte vor allem untersuchen, ob und unter welchen Umständen sich positive Effekte bei der Benutzung von Fehlerpfaden zeigen.