Über die Effizienz des Automatischen Testens

Marcel Böhme
Lehrstuhl für Softwaretechnik
Universität des Saarlandes
boehme@st.cs.uni-saarland.de

Soumya Paul
School of Computing
National University of Singapore
pauls@comp.nus.edu.sg

Abstract: Wir analysieren die Effizienz des zufälligen und systematischen Ansatzes zum automatischen Programmtesten. Dabei nutzen wir wenige, einfache, jedoch realistische Annahmen um ein generelles probabilistisches Modell für das Softwaretesten zu entwickeln. Die zwei wichtigsten Ziele des automatischen Softwaretestens sind (i) in minimaler Zeit mit einem gewissen Zuversichtsgrad x festzustellen, daß das Programm korrekt funktioniert und (ii) eine maximale Anzahl an Fehlern innerhalb eines gegebenen Zeitbudgets $\hat{n}$ im Programm zu finden. Zu beiden Fällen berechnen wir eine obere Schranke für die Zeit eines systematischen Tests relativ zur Zeit eines zufälligen Tests. Im ersten Fall (i) zeigen wir auch, daß diese Schranke asymptotisch nur von x abhängt. Nehmen wir zum Beispiel an, daß ein Zufallstestgenerator $\mathcal{R}$ lediglich 10ms braucht um einen Test zu generieren und auszuführen. Um mit Zuversicht feststellen zu können, daß ein Programm P für 90% seiner Eingaben korrekt funktioniert, darf der *effektivste*, systematische Testgenerator $\mathcal{S}_0$ *nicht länger als 41ms brauchen* um einen Test zu generieren und auszuführen! Andernfalls kann $\mathcal{R}$ im Durchschnitt schneller etablieren, daß P für 90% seiner Eingaben korrekt funktioniert.

1 Vortragszusammenfassung

Effizienz ist eine wichtige Eigenschaft des automatischen Softwaretestens – möglicherweise sogar wichtiger als Effektivität. Erklärtes Ziel des automatischen Testens ist es, mit einer gewissen Zuversicht festzustellen, daß das getestete Programm korrekt funktioniert. Die effektivste Methode zu zeigen, daß ein Programm für *all seine Eingaben* korrekt funktioniert, nennen wir Programmverifikation. Allerdings beschränkt sich dessen Anwendung aufgrund kombinatorischer Explosion und anderer Probleme bisher nur auf kleine Programme mit wenigen, hundert Codezeilen. Das Softwaretesten erlaubt uns nun diese Effektivität gegen Effizienz einzutauschen. Unabhängig von der Größe des Programmes erhöht jeder ausgeführte Testfall die Zuversicht, daß es korrekt funktioniert. Damit ist das automatische Testen eine effiziente Methode um zu zeigen, daß ein Programm für *eine wachsende Menge von Eingaben* korrekt funktioniert.

Trotzdem richtet sich der größte Teil der Erforschung des automatischen Testens an dessen Effektivität: Das *effektivste Testverfahren* entdeckt die meisten Fehler und garantiert mit größter Zuversicht die Korrektheit des Programmes. Erst jetzt beginnen wir auch die Effizienz des automatischen Testens zu erforschen: Das *effizienteste Testverfahren* (i) generiert eine hinreichend effektive Testsuite in minimaler Zeit oder (ii) generiert die effektivste Testsuite in dem gegebenen Zeitbudget. Eine erweiterte Diskussion finden Sie in [BP14].

Wir modellieren das Testproblem als eine *Erkundung von fehlerbasierten Eingabepartitionen*. Wir nehmen also an, daß die Menge aller Eingaben eines Programmes in homogene Teilbereiche aufgeteilt werden kann. In jedem Teilbereich decken entweder alle Eingaben Fehler auf oder keine der Eingaben decken Fehler auf. Die Anzahl und Größe solcher fehlerbasierten Partitionen kann beliebig, aber muss begrenzt sein. Wenn wir annehmen, daß es *a-priori* nicht bekannt ist, ob eine Partition Fehler aufdeckt, ist das Testproblem nun systematisch Stichproben von diesen Partitionen zu entnehmen und damit die Zuversicht über die Korrektheit des Programmes zu erhöhen.

Ein Testverfahren entnimmt Stichproben vom Eingaberaum des Programmes. Wir sagen, daß eine Partition D_i *entdeckt* wird, wenn von D_i zum ersten Mal entnommen wird. Die entnommene Eingabe zeigt, ob D_i nun Fehler aufdeckt oder nicht. Ein Testverfahren erreicht einen Zuversichtsgrad x wenn wenigstens $x\%$ aller Programmeingaben in entdeckten Partitionen liegen. Wenn keine der entdeckten Partitionen Fehler aufdecken, kann man sicher sein, daß das Programm wenigstens für $x\%$ seiner Eingaben korrekt funktioniert.

Wir untersuchen zwei Testverfahren: Das *Zufallsverfahren* $\mathcal{R}$ entimmt Eingaben zufällig mit gleicher Wahrscheinlichkeit und dürfte manche Partitionen mehrmals und andere gar nicht entdecken. Wir zeigen, daß der erreichte Zuversichtsgrad mit der Zeit exponentiell abnimmt. Das *systematische Testverfahren* $\mathcal{S}_0$ entdeckt eine neue Partition mit jeder entnommenen Eingabe. Wir zeigen, daß der erreichte Zuversichtsgrad mit der Zeit linear wächst. Da $\mathcal{S}_0$ schlussendlich alle Partitionen entdeckt, ist es das *effektivste* Testverfahren.

Um die Effizienz beider Verfahren zu analysieren, weisen wir der Entnahme Kosten zu. Im Gegensatz zum Zufallstesten benötigt jedes systematische Testverfahren grundsätzlich etwas Zeit für die Analyse, beispielsweise einer Spezifikation oder des Programmtextes. Wir sagen, daß $\mathcal{R}$ genau 1 Zeiteinheit, während $\mathcal{S}_0$ genau c Zeiteinheiten für jede Entnahme benötigt.

Nun beobachten wir, daß die Effizienz systematischen Testens abnimmt, wenn mehr Zeit für Analyse genutzt wird, während die Effizienz von Zufallstesten unverändert bleibt. Das heißt umso höher c, desto mehr Zeit benötigt $\mathcal{S}_0$ den selben Zuversichtsgrad zu erreichen. Damit $\mathcal{S}_0$ effizienter ist als $\mathcal{R}$, darf c einen gewissen Wert also nicht überschreiten!

In unserem Vortrag untersuchen wir die Kosten c von $\mathcal{S}_0$, wenn das Zufallstesten $\mathcal{R}$ im Durchschnitt effizienter ist als das effektivste, systematische Testverfahren $\mathcal{S}_0$. Wenn das Ziel beider Verfahren ist, in minimaler Zeit einen Zuversichtsgrad $x\%$ festzustellen, geben wir eine obere Schranke für c, die asymptotisch nur von x abhängt. Nehmen wir beispielsweise an, daß $\mathcal{R}$ für die Generierung und Ausführung eines Testfalls 10ms benötigt. Um einen Zuversichtsgrad von $x = 90\%$ zu etablieren, darf $\mathcal{S}_0$ *nicht länger als 41ms* pro Testfall benötigen. Andernfalls erreicht $\mathcal{R}$ x im Durchschnitt schneller als $\mathcal{S}_0$. Wir zeigen auch, daß $\mathcal{R}$ das *einzige effiziente Testverfahren* für Programme ab einer bestimmten Größe ist.

Literatur

[BP14] Marcel Böhme und Soumya Paul. On the Efficiency of Automated Testing. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, Seiten 632–642, 2014.