Geschichte und Bedeutung des Hoare Kalküls

Maximilian Peintner

26. Mai 2015

Inhaltsverzeichnis

1 Einleitung

Heute gibt es kaum eine Wissenschaft, die solch einen massiven Einfluss auf den Alltag hat wie die Informatik. Unsere Umwelt ist von Informatiksystemen geprägt, und wir verlassen uns zunehmend auf die korrekte Funktionsweise der Software. Bereits seit den Anfängen der Informatik war die Programmierung ein kritischer Aspekt, und hundertprozentig korrekte Programme sind eher die Ausnahme als die Regel. Allerdings gab es in den vergangenen fünfzig Jahren einige Entwicklungen von Methoden zur Spezifikation, Verifikation und zum Test von Software, mit denen sich beachtliche Qualitätssteigerungen erzielen lassen. Diese Arbeit schildert einige historische Wurzeln auf dem Gebiet der Qualitätssicherung von Software und geht näher auf die geschichtlichen Aspekte des Hoare Kalküls ein.

2 Softwarefehler

Die täglichen Nachrichten sind voll von Berichten über Probleme, die auf fehlerhafte Software zurückzuführen sind. Dabei scheint es keinen Bereich zu geben der von Problemen verschont bleibt. Heute scheint sich die Ansicht zu verbreiten, dass hundertprozentig korrekte Software nicht realisierbar sei, und man daher gewisse Restrisiken akzeptieren muss. Das dabei häufigst gebrauchte Argument ist, dass es auf Grund der hohen Komplexität unmöglich sei, sämtliche Abläufe eines größeren Programms zu testen. Obwohl dies zweifelsfrei als richtig angesehen werden kann, gibt es jedoch noch keinen Beweis, dass Software notwendigerweise fehlerhaft sein muss. Bereits seit dem Anfängen der Informatik gibt es Versuche, nachweisbar korrekte Software mit formalen Methoden zu entwickeln.

3 Formale Methoden

Ein Fehler ist eine Abweichung vom tatsächlichen auf das gewünschte Verhalten eines Systems. Anhand einer guten Spezifikation lässt sich ein detaillierter Vergleich zwischen Soll- und Ist-Zustand vornehmen. Wenn die Implementierung nicht von der Spezifikation des Programms abweicht, wird das Programm als verifiziert bezeichnet. Die Fragestellung, wie eine solche Spezifikation dargestellt werden kann, geht auf die Wurzeln der verschiedenen Formalismen für die Programmverifikation zurück. Dabei stößt man immer wieder auf die Namen John von Neumann und Alan Turing. Sie beschrieben die Programmierung als sechsstufigen Prozess:[1, S. 10]

1. Konzeptualisierung des mathematischen und physikalischen Problems
2. Wahl des numerischen Algorithmus
3. Abschätzung der Rundungsfehler
4. Skalierung der Variablen auf die verfügbare Bitbreite
5. Beschreibung des Programmverhaltens als Ablaufplan
6. Codierung, d.h. Umsetzung des Plans für die entsprechende Maschine

Diese Liste beschreibt eine auch heute noch gängige Vorgehensweise bei der Entwicklung eingebetteter Algorithmen. Für die Beschreibung des Programmverhaltens (Schritt 5) wurde 1949 erstmals die Verwendung von Flussdiagrammen als Programmierkonzept eingeführt. Diese formale Beschreibungstechnik wird bis heute verwendet (UML). Erste Beweismethoden stammen aus einem Artikel von Alan Turing, welche leider für lange Zeit unbeachtet blieben, die aber wesentliche Erkenntnisse der folgenden 30 Jahre vorwegnehmen. Erst in den 1960-er Jahren tauchten diese in Arbeiten von Peter Naur und Robert Floyd wieder auf.
Zwanzig Jahre nach Turings Artikel erarbeitete Tony Hoare oder Charles Antony

Richard Hoare schließlich mit der sogenannten Hoare-Logik eine axiomatische Basis, auf der solche Beweise geführt werden konnten.

4 Das Hoare Kalkül

Hoare basierte sich bei seinen Arbeiten dabei auf Floyds „Assigning meaning to programs“ welche 1967 veröffentlicht wurde. Floyd hatte einen Weg beschrieben, den Flussdiagrammen Zusicherungen hinzuzufügen. Diese ermöglichten es zu zeigen, dass eine Spezifikation korrekt umgesetzt worden ist. Hoare machte dazu zwei weitere Schritte und seine Arbeit „An axiomatic basis for computer programming“ wurde eine der einflussreichsten Arbeiten der Theoretischen Informatik.
Zuerst verwarf er die Idee der Flussdiagramme und entwickelte ein System, das eine Menge von logischen Regeln lieferte, die es erlauben, Aussagen über die Korrektheit von imperativen Computer-Programmen zu treffen und sich dabei der mathematischen Logik zu bedienen. Später bekannt als Hoare Tripel. Zweitens argumentierte er, dass sein „axiomatisches“ System als Aufzeichnung der Semantik von Programmiersprachen angesehen werden konnte. Hoare veröffentlichte einige Versionen seiner Entwicklung und die Hoare Semantik hatte tiefgründige Auswirkungen auf die Programmanalyse und das Verständnis von Programmiersprachen

5 Fazit

Betrachtet man allerdings komplexere Programmkonstrukte, so ist das Hoare Kalkül ungeeignet. Diese Erkenntnis, geht auf Edmund M. Clarke zurück, der zusammen mit Allen Emerson das Model Checking [3] als formale Verifikationsmethode von endlichen Systemen vorgeschlagen hat.
Zu guter Letzt muss verdeutlicht werden, dass kein auf dem Programmcode aufsetzendes System, wie das Hoare-Kalkül, Fehler entdecken kann, die in der Spezifikation auftreten. Es wird lediglich gezeigt, dass die Spezifikation korrekt umgesetzt worden ist
Ein Programm mit korrekt umgesetzten Routinen, die jedoch wichtige Zustände der Wirklichkeit nicht situationsgerecht erfassen, hat in der richtigen Situation jedoch das Potential genauso gefährlich zu sein, wie eine schlechte Implementierung einer an sich sauberen Spezifikation.[2][S. 56]

> “I conclude there are two ways of constructing software design: one way is to make it so simple there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies.”
>
> — Tony Hoare

6 Schlusswort

Schon in den Anfängen der Programmierung kam das Bedürfnis auf, die Korrektheit eines Programmes formal verifizieren zu können. Zu dieser Fragestellung gab es erstmals 1949 Entwicklungen von Alan Turing. Auch Robert Floyd, Charles Antony Richard Hoare, Edmund M. Clarke und Allen Emerson sind bedeutende Persönlichkeiten, die sich der Fragestellung gewidmet haben. Alle samt sind mit dem Turingaward ausgezeichnet worden.

Literatur

[1] Bernd Holger Schlingloff *Softwarequalität - Geschichte und Trends*

[2] Michael Gellner. *Der Umgang mit dem Hoare Kalkül zur Programmverifikation*

[3] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking.* MIT Press, 2000.