Rainer Schmidberger: Wohldefinierte Überdeckungsmetriken für den Glass-Box-Test

1. Gutachter: Prof. Dr. Jochen Ludewig (Universität Stuttgart)

2. Gutachter: Prof. Dr. Martin Glinz (Universität Zürich)

Datum der Prüfung: 14. Juli 2014

Zusammenfassung:
Der Glass-Box-Test (GBT), der auch als White-Box- oder Strukturtest bezeichnet wird, zeigt den im Test ausgeführten – und viel wichtiger: den nicht ausgeführten – Programmcode an. Dieser Grad der Ausführung wird als Überdeckung bezeichnet. Empirische Untersuchungen zeigen eindeutig, dass eine hohe GBT-Überdeckung mit einer geringeren Restfehlerrate korreliert. Viele brauchbare Werkzeuge sind für den GBT für praktisch alle Programmiersprachen verfügbar, und wichtige Industriestandards zur Zertifizierung sicherheitskritischer Software verlangen den Nachweis einer vollständigen (oder sehr hohen) Überdeckung. Auf den ersten Blick erscheint uns damit der GBT als eine ausgereifte und etablierte Testtechnik, die auf standardisierten Metriken basiert.

Doch bei genauer Betrachtung der zugrundeliegenden Modelle und Metriken zeigen sich erhebliche Mängel, die zu unpräzisen und inkonsistenten Resultaten der verschiedenen GBT-Werkzeuge führen. Eine wesentliche Ursache hierfür bildet der Kontrollflussgraph (CFG), auf dessen Grundlage überwiegend die Definitionen der GBT-Metriken vorgenommen werden. Der CFG hat hierfür gravierende Nachteile: Die Transformation der realen Programme in den CFG ist nicht eindeutig definiert, und es fehlt im CFG eine Repräsentation für die Ausnahmebehandlung sowie die GBT-relevanten Ausdrücke wie z. B. bedingte Ausdrücke oder die Kurzschlusssemantik boolescher Operationen. Als logische Konsequenz folgen die Werkzeuge des GBT keinem gemeinsamen Standard und zeigen für die gleiche Programmausführung deutlich verschiedene Überdeckungswerte an.

In dieser Arbeit wird ein präzises Modell für den GBT präsentiert. Dieses Modell entsteht in zwei Schritten: Zunächst wird eine GBT-Modellsprache definiert (die Reduced Program Representation, RPR), die die GBT-relevanten Aspekte der realen Sprachen abbildet, die irrelevanten dagegen präteriert. Aus der RPR-Definition entstehen sogenannte Ausführungselemente, die entweder primitiv sind (wie z. B. primitive Anweisungen), oder die sogenannten Verbund-Ausführungselemente (wie z. B. if-Anweisungen), die als Teil der eigenen Struktur andere Ausführungselemente enthalten. RPR bildet dabei sowohl den Kontrollfluss als auch die GBT-relevanten Ausdrücke wie z. B. den bedingten Ausdruck oder zusammengesetzte boolesche Ausdrücke ab. Auch ist RPR so angelegt, dass Ausnahmen berücksichtigt werden und reale Programme systematisch und eindeutig nach RPR transformiert werden können.

Im zweiten Schritt wird die Ausführungssemantik der Ausführungselemente durch Petri-Netze, sogenannte Modellnetze, definiert. Primitive Ausführungselemente lassen sich direkt als Modellnetz beschreiben, die Verbund-Ausführungselemente wie z. B. die Entscheidungsanweisung oder der zusammengesetzte boolesche Ausdruck werden durch Komposition der Teilnetze beschrieben. Die Modellsprache RPR liefert hierzu die Kompositionsregeln. Ein Vorteil des formalen Modells ist, dass die Evaluation der Netze werkzeugunterstützt über die Erreichbarkeitsanalyse erfolgen kann. Ein besonderer Vorteil der Modellnetze ist auch, dass sich die zur Metrikenbestimmung wichtigen Ausführungszähler präzise im Modell platzieren lassen und so die Spezifikation für eine Werkzeugimplementierung liefern.

Auf Grundlage der Ausführungselemente und der Ausführungszähler der Modellnetze erfolgt eine präzise Definition der populären sowie weiterer GBT-Metriken. Zur formalen Prüfung der Plausibilität, Differenziertheit und Vergleichbarkeit dieser Metriken wird ein Regelsystem aufgestellt, anhand dessen die GBT-Metriken geprüft werden.

Da für den GBT ein Werkzeugeinsatz Voraussetzung ist, wird das Werkzeug CodeCover[1] präsentiert, das eine Referenzimplementierung der definierten Metriken liefert. Als eine Folge der programmiersprachenunabhängigen RPR-Darstellung unterstützt CodeCover mehrere Programmiersprachen: Java, C und COBOL.

CodeCover bietet auch eine neue Funktion, die den Tester durch sogenannte Testfall-Hinweise beim Entwurf von GBT-basierten Testfällen systematisch unterstützt. Diese Testfälle führen einerseits zu einer Erhöhung der Überdeckung. Durch eine gezielte Priorisierung der Testfall-Hinweise wird aber auch eine hohe Fehlersensitivität der neu entwickelten Testfälle angestrebt.

Veröffentlicht als: Rainer Schmidberger: Wohldefinierte Überdeckungsmetriken für den Glass-Box-Test, Dissertation, Universität Stuttgart, 2014 (Buchveröffentlichung folgt noch)
Online unter
http://www.iste.uni-stuttgart.de/fileadmin/user_upload/iste/se/people/schmidberger/Diss.pdf

[1] www.codecover.org