Die Verbesserung des Sicherheitsnachweises durch Induktion

Kann ein Denkbegriff aus dem 16. Jahrhundert nützlich sein?

Chris Hobbs, QNX Software Systems

Die verschiedenen Sicherheitsnormen (ISO 26262, IEC 61508 usw.) verlangen die Erstellung eines Sicherheitsnachweises. Diverse Untersuchungen haben belegt, dass eine solche Aufgabe oft zu Bestätigungsfehlern führt.

Diese Abhandlung beschreibt eine praktische Anwendung der eliminativen Induktion, um das Phänomen des Confirmation Bias positiv zu nutzen. Im Verlauf einer kürzlich erfolgten ISO 26262/IEC 61508 Zertifizierung hat dieses Vorgehen mehrere Sicherheitslücken aufgedeckt, die bis dahin unbemerkt geblieben waren.

Einführung

QNX Software Systems hat seit 2010 diverse Produkte erfolgreich durch Prüfungen gemäß ISO 26262, IEC 61508 und IEC 62304 zertifizieren lassen. Für jede Prüfung war von QNX ein Sicherheitsnachweis (sog. »Safety Case«) vorzubereiten. Gemäß den Normen besteht ein Sicherheitsnachweis aus drei Teilen: die Behauptung (was leistet das System, und unter welchen Bedingungen), das Argument (wie bestätigen wir, dass das System sicher ist) und die Beweismittel (die das Argument unterstützen). Damit soll die Sicherheit des Systems bewiesen werden können.

Wie die Referenzen [1] und [2] zeigen, ist es für Menschen sehr schwierig, fast unmöglich, Bestätigungsfehler (sog. »Confirmation Bias«) zu vermeiden. Wenn man die Aufgabe hat, zu beweisen, dass das System sicher ist, bemerkt man nur die Nachweise, die zeigen, dass das System sicher ist!

In seinem Bestseller (Referenz [5]) hat Rolf Dobelli den Confirmation Bias so beschrieben: »Der Confirmation Bias ist der Vater aller Denkfehler – die Tendenz, neue Informationen so zu interpretieren, dass sie mit unseren bestehenden Theorien, Weltanschauungen und Überzeugungen kompatibel sind. … Neue Informationen, die im Widerspruch zu unseren bestehenden Ansichten stehen … filtern wir aus. Das ist gefährlich.«

Dieser Vortrag beschreibt, wie QNX den Confirmation Bias positiv genutzt hat, um einen vertrauenswürdigeren Sicherheitsnachweis vorzubereiten.

Das Falsifikationsprinzip

Obwohl das Falsifikationsprinzip normalerweise im Zusammenhang mit Karl Popper im 20. Jahrhundert betrachtet wird, hatte Francis Bacon es schon 1597 in seinem *Novum Organum* vorhergesehen (Referenz [6]).

2013 hat John Goodenough (et al, Referenz [4]) „Eliminative Induction“ veröffentlicht. Diese beschreibt, wie Bacons Ideen über »Eliminative Induction« (auch als

»Eliminative Argumentation« bekannt) genutzt werden können, um Sicherheitsnachweise zu verbessern. QNX hat diese Ideen während der Zertifizierung des QOS 2.0-Systems getestet.

Die Grundidee ist einfach: Anstatt Nachweise zu suchen, die das Argument positiv bestätigen, sucht man Beispiele, die das Argument in Zweifel ziehen. Danach muss jeder dieser Zweifel ausgeschlossen werden – dadurch nutzt man die unvermeidbaren Bestätigungsfehler zum Vorteil.

Im Wesentlichen sagt man: »Ich habe alle Zweifel an der Sicherheit meines Systems, die ich mir vorstellen kann, identifiziert. Und jetzt kann ich belegen, dass jeder dieser Zweifel unbegründet ist. Deshalb bin ich der Meinung, dass das System sicher ist«.

Ein Beispiel

Tabelle 1 der ISO 26262-6:2011 empfiehlt “Use of Language Subsets” und Tabelle C.1 der IEC 61508-7:2010 empfiehlt die C-Programmiersprache gar nicht, außer wenn “C with subset and coding standard, and use of static analysis tools” benutzt wird.

Als Beispiel (nicht aus QNXs Erfahrung genommen!) zeigt Bild 1 einen kleinen Teil des Arguments eines Sicherheitsnachweises eines imaginären Produkts. Die Darstellung folgt der Goal Structuring Notation (Referenz [3]) und enthält nur zwei Symbole:

1. Ein “Goal” (G36). Dieses behauptet, dass die Programmierer während der Produktentwicklung gemäß den Normen nur eine Teilmenge von C benutzt haben.
2. Eine “Solution” (Sn36). Das Beweismittel, das G36 unterstützt, ist das Dokument ABC123. Dieses Dokument enthält vermutlich den Coding-Standard, der die Teilmenge definiert.

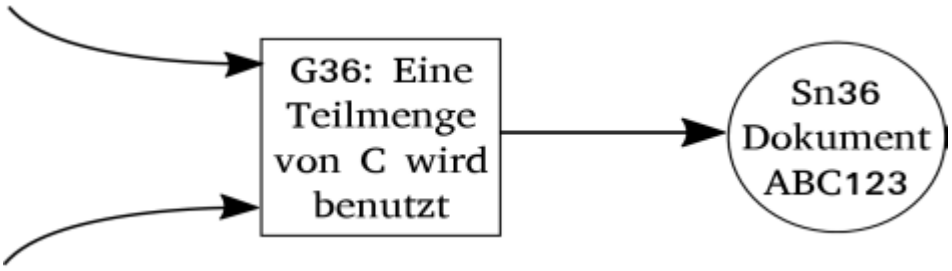

Bild 1: Ein Teil des Arguments

Hat man die Aufgabe, die Sicherheit des Systems zu demonstrieren, um dieses Argument zu prüfen, braucht man nur das Dokument ABC123 zu finden, und zu verifizieren, dass es bloß eine Teilmenge von C erlaubt.

Allerdings ist es möglich, drei Arten von Zweifeln hervorzubringen. In Referenz [4] sind diese wie folgt genannt:

1. Rebutting (widerlegend). Kann man Gegenbeispiele finden?
2. Undermining (zersetzend). Könnte das Beweismittel ungültig sein?
3. Undercutting (unterhöhlend). Kann das Beweismittel die Behauptung nicht unterstützen? Vielleicht ist das Beweismittel schon wahr, aber trotzdem irrelevant.

Bild 2 weitet Bild 1 mit möglichen “undermining” und “undercutting” Zweifeln aus. IR36a macht die zugrunde liegende Annahme deutlich: “Wenn ein Dokument existiert, das die Teilmenge definiert, so muss die Teilmenge verwendet werden”. Wenn es so erläutert wird, ist es einfach, Zweifel daran aufzuwerfen – kann man einen Programmierer finden, der dieses Dokument *nicht* anwendet oder *nicht* versteht (UC36a)? Man könnte auch einen “undercutting” Zweifel finden: Was enthält das Dokument ABC123? Die Normen verlangen nur eine Teilmenge der C-Sprache, aber es ist implizit, dass die Teilmengen nützlich sein müssen: es genügt nicht nur »GOTO darf nicht benutzt werden« zu spezifizieren. Wurde der Inhalt von Dokument ABC123 geprüft? Von wem? War er kompetent?

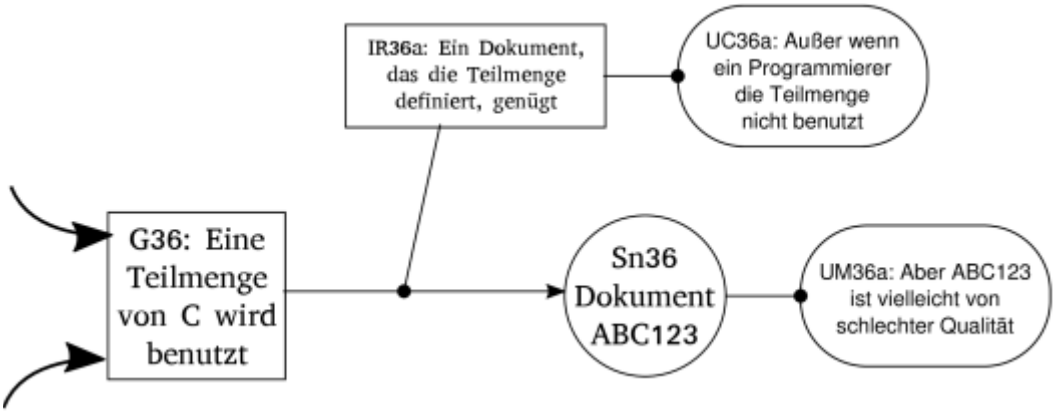

Bild 2: Zweifel über das Argument

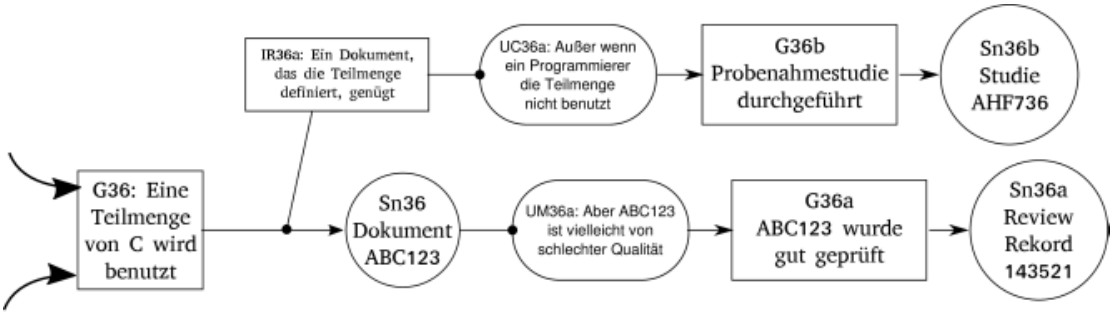

Bild 3: Zweifel Ausgeschlossen?

Die Zweifel müssen jetzt eliminiert werden. Vielleicht könnte man nachprüfen, ob Dokument ABC123, das die Coding Standards definiert, von qualifizierten Menschen überprüft wurde. Wenn ja, sollten Berichte zur Verfügung stehen, die die Überprüfung beschreiben. Bild 3 zeigt, wie diese Information einbezogen werden kann: Review Record 143521 zeigt, dass man ABC123 überprüft hat. Auf eine ähnliche Weise könnte man eine Probennahme des Codes mit einem Static-Analysis-Werkzeug durchführen, um herauszufinden, ob die Programmierer die Vorschriften beachtet haben: AHF736.

Nun könnte man, wie kleine Kinder es tun, weiter und weiter nachbohren: „Aber ist es nicht auch möglich, dass … ?“ Das würde aber zu abnehmenden Erträgen führen. Nach technischem Verständnis ist es nicht nützlich, immer tiefer einzusteigen.

Die Ergebnisse
QNX hat dieses Verfahren während der Vorbereitung seines aktuellen Sicherheitsnachweises (2018) des QOS 2.0 Betriebssystems angewendet. Dieses Betriebssystem war 2010 gemäß IEC 61508 (SIL3), 2012 gemäß IEC 62304 und 2015 und 2016 gemäß ISO 26262 (ASIL-D) und IEC 61508 (SIL3) schon zertifiziert worden.

Um Bestätigungsfehler zu vermeiden, hatten wir damals für diese früheren Produkte zwei strukurierte, semi-formale Argumente vorbereitet: mit Goal Structuring Notation bzw. Bayessche Netzwerke Notation.

Obwohl diese früheren Sicherheitsnachweise professionell vorbereitet wurden, und obwohl man die Gefahren von Bestätigungsfehlern zwar gut begriffen hatte, hat die neuste Analyse, die mit dem hier beschriebenen Verfahren durchgeführt wurde, etwa zwanzig Probleme gefunden, die während vorheriger Zerifizierungen nicht aufgedeckt worden waren.

Da sich diese neue Analyse als so erfolgreich erwiesen hat, haben wir entschieden, dass unser Bayessche-Netzwerk-Modell nicht länger erforderlich ist.

Erläuterung
Natürlich haben wir uns die Frage gestellt: Warum hatten wir diese Sicherheitsprobleme nicht früher entdeckt? Insbesondere, warum hatte das Bayessche-Netzwerk-Modell sie nicht gefunden?
Um diese Frage beantworten zu können, haben wir verschiedene Programmierer befragt.

Als Beispiel untersuchten wir ein Problem, das sich mit Code Review beschäftigt und durch dieses neue Verfahren entdeckt wurde. Die QNX Code-Review-Überprüfungsvorschrift verlangt, dass die Ergebnisse der Static Analysis immer mit einem Code Review veröffentlicht werden müssen. Der Programmierer soll nicht nur die vorgeschlagenen Änderungen des Codes, sondern auch von den Static-Analysis-Warnungen mit dem Überprüfungsantrag berichten.

In der Vergangenheit hat der Auditor sowohl das Verfahren als auch die Prüfung des Verfahrens kontrolliert. Er hat dazu einige Reviews überprüft. Dabei wurde kein einziges Problem gefunden. Auf die Frage, ob der Vorgang vollständig durchgeführt worden ist, bestätigten die Programmierer: »Ja, natürlich folgen wir dem richtigen Verfahren«.

Stellten wir dir Frage umgekehrt: »Kannst Du dich an Beispiele erinnern, wo die Static Analysis Warnungen nicht mit der Überprüfungsvorschrift veröffentlicht wurden?«, bestätigten die gleichen Programmierer auch dies positiv. Obwohl die Warnungen beim ersten Schritt des Reviews immer gezeigt werden, wurden sie jedoch gelegentlich weggelassen, wenn der Code während späterer Schritte verändert werden musste.

Zusammenfassung
QNX hat die Richtlinien von Referenz [4] erweitert und genutzt, um einen Sicherheitsnachweis vorzubereiten.
Dieser Nachweis hat die Zertifizierung des Betriebssystems gemäß ISO 26262 (ASIL-D) und IEC 61508 (SIL3) von TÜV Rheinland unterstützt. Außerdem hat dieses neue Verfahren auch diverse sicherheitsrelevante Probleme gefunden, die während vorheriger Zertifizierungen nicht aufgedeckt wurden. Diese konnten wir dann vor der Produktfreigabe eliminieren.
QNX hat diese Ergebnisse mit der »Assurance Case Working Group« (https://scsc.uk/gc) geteilt.

Literatur
[1] „Die Bayessche Darstellung eines Sicherheitsnachweises“, Chris Hobbs, Embedded Systems Engineering Kongress, Sindelfingen, Dezember 2015

[2] “White Paper on the Use of Safety Cases in Certification and Regulation”, Nancy Leveson, 2012

[3] “GSN COMMUNITY STANDARD VERSION 1” , November 2011.

[4] “Eliminative Induction: A Basis for Arguing System Confidence”, John B Goodenough, Charles B Weinstock and Ari Z Klein, New Ideas and Emerging Results Workshop, International Conference on Software Engineering, 2013

[5] “Die Kunst des Klaren Denkens”, Rolf Dobelli, 2014, ISBN 978-3-423-34826-3

[6] “Novum Organum Scientiarum”, Francis Bacon, 1620,
http://www.thelatinlibrary.com/bacon.html

Autor

Chris ist Programmierer bei QNX Software Systems. Sein Spezialgebiet ist „ausreichend verfügbare” Software: Software, in die mindestens so viel Entwicklungsaufwand gesteckt wurde, dass sie den Ansprüchen bezüglich Ausfallsicherheit und Zuverlässigkeit des Kunden gerecht wird. Außerdem befasst er sich mit Sicherer Software (Konformität mit IEC 61508, ISO 26262 und IEC 62304). Neben seiner Arbeit im Bereich Software-Entwicklung ist Chris Hobbs außerdem als Fluglehrer tätig, singt gerne (vor allem Schubertlieder) und ist Autor einiger weiterer Bücher, z.B. *Flying Beyond: The Canadian Commercial Pilot Textbook* und *Embedded Software Development for Safety-Critical Systems*.

Kontakt

Email: chobbs@qnx.com