Die Testabdeckung messen:

10 Kriterien zur Auswahl eines Code-Coverage-Tools

Um sichere und zuverlässige Software zu entwickeln, ist das Testen ein unverzichtbarer Bestandteil der Qualitätssicherung. Denn ohne ausreichende und dokumentierte Tests kann nicht festgestellt werden, ob eine Software sicher und funktional korrekt ist. Einen besonderen Stellenwert hat dabei die Messung der Code Coverage (Testabdeckung). Denn dadurch kann festgestellt werden, wie umfassend eine Software bereits getestet wurde. Die Code Coverage gibt das Verhältnis von getestetem Code und Gesamtcode an. Vereinfacht lässt sich beispielsweise sagen, dass die Code Coverage 75% beträgt, wenn beim Test drei von vier möglichen Optionen durchlaufen werden.

Speziell in der sicherheitskritischen Softwareentwicklung schreiben Branchennormen genaue Anforderungen für die Code Coverage vor, so dass Produkte hier ohne den Nachweis einer ausreichenden Testabdeckung nicht zertifiziert werden können. Aber auch bei anderen Entwicklungsprojekten legen Unternehmen zunehmend großen Wert auf Software-Qualität und messen die Code Coverage.

Zur Messung der Code Coverage sind am Markt unterschiedliche Code Coverage Analyser verfügbar. Die Lösungen unterscheiden sich im Handling und in der Qualität teils deutlich. Wir haben deshalb zehn grundsätzliche Kriterien für die Auswahl eines Code-Coverage-Tools zusammengestellt:

1 Unabhängigkeit vom eingesetzten Compiler

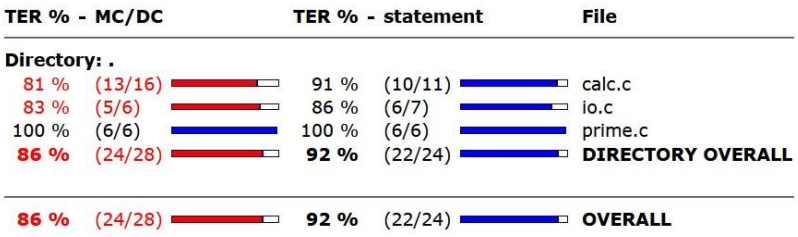
Selbstverständlich muss ein Code-Coverage-Tool mit dem im Projekt eingesetzten Compiler funktionieren. Allerdings ist es sehr sinnvoll, bereits von Beginn an auf ein Tool zu setzen, welches unabhängig vom Compiler eingesetzt werden kann. Solche Werkzeuge können dann in allen Projekten genutzt werden und auch im laufenden Projekt ist bei Bedarf der Wechsel des Compilers möglich. Ein Coverage-Tool, welches compilerunabhängig nutzbar ist, kann wesentlich vielfältiger verwendet werden und ist daher eine lohnende Investition.

2 Benutzerfreundlichkeit

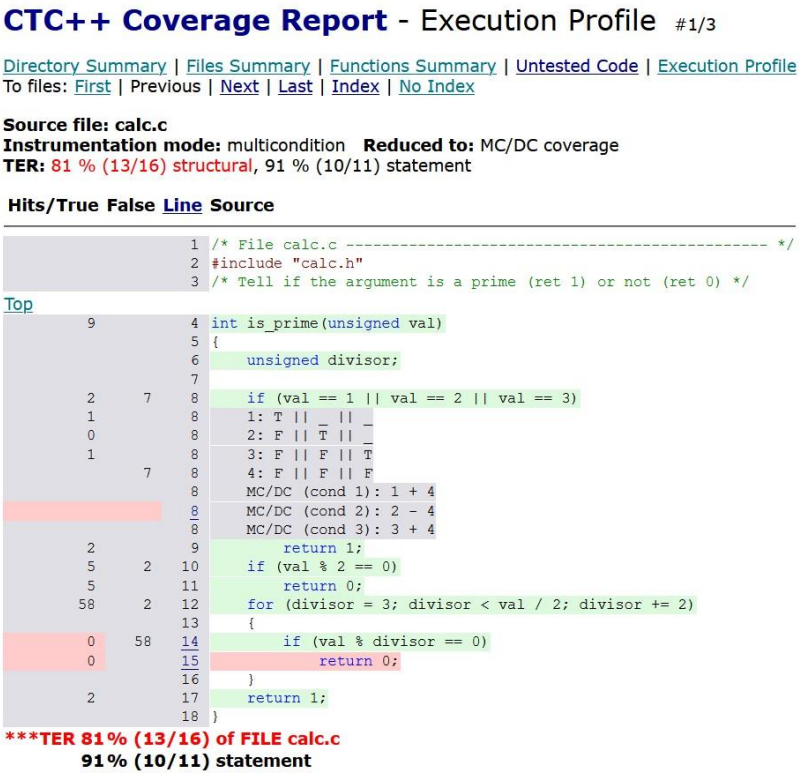
Auch für Code-Coverage-Tools gilt: Die beste Software wird ungern (und damit selten) genutzt, wenn die Bedienung unnötig kompliziert oder nicht durchdacht ist. Ein einfaches Handling hingegen kann die Akzeptanz für den Einsatz eines Testabdeckungstools beim Anwender deutlich erhöhen. Idealerweise läuft das Tool im Hintergrund und erzeugt für den Nutzer beim Test keine zusätzliche Arbeit.

3 Nachvollziehbarkeit der Coverage-Berichte

Bei der Auswertung der Coverage-Berichte sollte möglichst auf einen Blick klar werden, welche Teile des Codes bereits getestet sind und wo es noch an Coverage fehlt. Bei guten Coverage-Tools kann der Tester auf Quellcodeebene sehr einfach erkennen, welche Testfälle noch ausstehen. Durch Ausführen dieser fehlenden Tests kann die Code Coverage dann zielgerichtet erhöht werden. Dies vermeidet gleichzeitig unnötige Arbeit, die durch redundante Tests entstehen würde.



TER % - MC/DC		TER % - statement		File
Directory: .				
81 %	(13/16)	91 %	(10/11)	calc.c
83 %	(5/6)	86 %	(6/7)	io.c
100 %	(6/6)	100 %	(6/6)	prime.c
86 %	(24/28)	**92 %**	(22/24)	**DIRECTORY OVERALL**
86 %	(24/28)	**92 %**	(22/24)	**OVERALL**



CTC++ Coverage Report - Execution Profile #1/3

Directory Summary | Files Summary | Functions Summary | Untested Code | Execution Profile
To files: First | Previous | Next | Last | Index | No Index

Source file: calc.c
Instrumentation mode: multicondition **Reduced to:** MC/DC coverage
TER: 81 % (13/16) structural, 91 % (10/11) statement

Hits/True	False	Line	Source
		1	/* File calc.c ---------- */
		2	#include "calc.h"
		3	/* Tell if the argument is a prime (ret 1) or not (ret 0) */
Top			
9		4	int is_prime(unsigned val)
		5	{
		6	unsigned divisor;
		7	
2	7	8	if (val == 1 || val == 2 || val == 3)
1		8	1: T || _ || _
0		8	2: F || T || _
1		8	3: F || F || T
	7	8	4: F || F || F
		8	MC/DC (cond 1): 1 + 4
		8	MC/DC (cond 2): 2 - 4
		8	MC/DC (cond 3): 3 + 4
2		9	return 1;
5	2	10	if (val % 2 == 0)
5		11	return 0;
58	2	12	for (divisor = 3; divisor < val / 2; divisor += 2)
		13	{
0	58	14	if (val % divisor == 0)
0		15	return 0;
		16	}
2		17	return 1;
		18	}

***TER 81% (13/16) of FILE calc.c
91% (10/11) statement

Abb. 1 u. 2: Testwell CTC++ zeigt neben einer Übersicht über die Code Coverage der einzelnen Codeteile auch detaillierte Informationen, die auch für höchste Coveragestufen exakt zeigen, welche inwieweit der Quellcode durch Tests abgedeckt ist.

4 Support von höheren Coverage-Maßen für sicherheitskritische Entwicklung

Für den Test von sicherheitskritischer Software schreiben die Normen (z.B. ISO 26262 im Automobilbereich, DO-178C in der Luftfahrt und EN-50128 im Schienenverkehr) hohe Coverage-Maße bis hin zur MC/DC-Coverage vor. Es ist deshalb zwingend darauf zu achten,

dass das auszuwählende Coverage Tool die jeweils geforderte Coverage-Stufe auch tatsächlich unterstützt. Um eine Lösung langfristig einsetzen zu können, sollten hierbei nicht nur aktuelle, sondern nach Möglichkeit auch zukünftige, bereits absehbare Anforderungen berücksichtigt werden. Wichtig zu wissen: Viele Coverage-Tools bieten maximal Decision- oder Branch-Coverage und sind daher für die sicherheitskritische Softwareentwicklung unzureichend.

5 Flexible Integration

Auch innerhalb eines Unternehmens sind Entwicklungsumgebungen und Toolketten oft sehr heterogen. Ein Coverage-Tool sollte problemlos mit all diesen verschiedenen Umgebungen zurechtkommen. Die Integration in den jeweiligen Build-Vorgang und in die Durchführung der Tests muss nahtlos und ohne großen Aufwand möglich sein. Sofern das Tool auch über die Kommandozeile genutzt werden kann, bieten sich Vorteile bei der Erstellung automatisierter Builds.

6 Geringer Instrumentation Overhead

Die meisten Coverage-Tools messen die Code-Coverage über die Instrumentierung des Quellcodes. Der Quellcode wird dabei durch das Coverage-Tool mit „Zählern“ angereichert, die mitzählen, wo und wie oft die betreffenden Codeteile beim Testen ausgeführt wurden. Hierdurch wird der originale Code allerdings vergrößert. Beim Test auf Embedded Targets, die über beschränkten Speicherplatz verfügen, ist daher darauf zu achten, dass dieser sogenannte Instrumentation Overhead möglichst gering bleibt. Die Unterschiede bzgl. des Speicherbedarfs sind bei den einzelnen Code-Coverage-Tools teilweise beträchtlich. Der Code Coverage Analyser Testwell CTC++ von Verifysoft Technology ist diesbezüglich beispielsweise sehr ressourcenschonend. Hier bietet sich unter anderem die Möglichkeit, durch Einsatz der Bit-Coverage-Option den erforderlichen Speicherplatz nochmals spürbar zu reduzieren (die Bit-Coverage misst dann nur, ob ein Codeteil getestet wurde, allerdings nicht mehr, wie oft dies geschehen ist).

7 Unterstützung verschiedener Programmiersprachen

Oft wird in Firmen mit verschiedenen Programmiersprachen gearbeitet oder die Einführung weiterer Sprachen ist künfig geplant. Es ist daher sinnvoll, von Anfang an ein Tool zu wählen, welches alle bzw. möglichst viele dieser Sprachen unterstützt.

8 Unterstützung von „kreativer“ Programmierung

Manche Coverage-Tools bekommen Probleme bei der Analyse von Sprachkonstrukten, die von üblichen Standards abweichen oder eine hohe Schachtelungstiefe haben. Eine gutes Tool für die Messung der Testabdeckung sollte jedoch auch mit einem „kreativen“ Programmierstil zurechtkommen.

Abb. 3: Unsere Kunden vergleichen Testwell CTC++ mit einem Geländewagen: das Tool arbeitet immer zuverlässig – auch in schwierigem „Gelände“…

9 Eignung für sicherheitskritische Softwareentwicklung

Bei der Entwicklung sicherheitskritischer Software fordern die entsprechenden Normen, dass die gesamte Werkzeugkette qualifiziert wird. Hier geht es darum, nachzuweisen, dass sowohl der Coverage Analyser als auch die anderen genutzten Werkzeuge innerhalb der gesamten Toolchain zuverlässig funktionieren. Hersteller professioneller Code-Coverage-Tools unterstützen Softwareprojekte durch Qualification-Kits und Beratung bei der Toolqualifikation. In diesem Zusammenhang sollte man auch darauf achten, ob das gewählte Coverage-Tool bereits erfolgreich in sicherheitskritischen Projekten eingesetzt wird.

10 Evaluierungsmöglichkeit, technischer Support und Kundenreferenzen

Die Eignung eines Coverage-Tools für die eigenen Projekte sollte auf jeden Fall im Rahmen einer Tool-Evaluation überprüft werden. Bereits hier wird man auch einen Eindruck von der Leistungsfähigkeit des technischen Supports bekommen. Ist der Support auch telefonisch oder ausschließlich per E-Mail erreichbar? Wie kompetent sind die Mitarbeiterinnen und Mitarbeiter im Support? Ist die Begleitung der Evaluation angemessen oder muss alles stets aufs Neue erklärt werden? Sind Anfragen an den Support auf Deutsch möglich? Wie verhält es sich mit den Reaktionszeiten? Wie gut und praxisnah ist das Benutzerhandbuch? Bietet der Hersteller darüber hinaus Schulungen an? Nicht zuletzt empfiehlt sich auch ein Blick auf die Kundenreferenzen des Herstellers. Dieser kann weitere Hinweise auf die Qualität des Coverage Analysers und die Leistungsfähigkeit des Anbieters geben.

„Code Coverage ist aus gutem Grund für die sicherheitskritische Softwareentwicklung vorgeschrieben. Aber auch für alle, die ihre Softwarequalität generell verbessern wollen, handelt es sich um eine gute Methode, um die Abdeckung und Aussagekraft der dynamischen Tests zu messen und zu erhöhen“, erklärt Klaus Lambertz, Geschäftsführer der Verifysoft Technology GmbH. „Bei der Auswahl eines Code Coverage Analysers muss darauf geachtet werden, dass das Werkzeug den gestellten Anforderungen entspricht. Zudem spielen Faktoren wie einfache Benutzung und professionelle Unterstützung bei Rückfragen und im Supportfall eine wichtige Rolle. Richtig eingesetzt, hilft ein gutes Testabdeckungstool dabei, die Qualität deutlich zu verbessern, die Motivation von Entwicklern und Testern zu erhöhen und Tests kostensparend durchzuführen.“

Code Coverage im Überblick: Unterschiedliche Stufen der Testabdeckung

Function Coverage
Die Function Coverage (Aufrufüberdeckung) misst, ob alle Funktionen des Programms aufgerufen wurden. Die Function Coverage ist die „schwächste“ der üblichen Testabdeckungsstufen.

Statement Coverage
Bei der Statement Coverage (Anweisungsüberdeckung) wird gemessen, wie hoch der Anteil der getesteten Anweisungen im Vergleich zu allen Anweisungen ist.

Decision Coverage / Branch Coverage
Bei dieser Coverage-Stufe muss jede Entscheidung mindestens einmal als „wahr“ und als „falsch“ getestet werden. Bei normalen „if-Statements“ entspricht dies der Branch Coverage, bei der jeder Zweig ausgeführt worden sein muss (Zweigabdeckung).

Condition Coverage
Bei der Condition Coverage (Bedingungsüberdeckung) werden zusammengesetzte Entscheidungen im Detail betrachtet. Bei Entscheidungen, die aus mehreren, über Boolesche Operatoren zusammengesetzten atomaren Bedingungen bestehen, muss jede dieser Bedingungen einzeln als „wahr“ und als „falsch“ getestet werden.

Multicondition Coverage und Modified Condition/Decision Coverage (MC/DC)
Bei der Multicondition Coverage (Mehrfachbedingungsüberdeckung) müssen bei zusammengesetzten Entscheidungen alle möglichen Wahr-Falsch-Kombinationen überprüft werden. Bei mehreren Bedingungen innerhalb einer Entscheidung erfordert dies eine zumeist unpraktikabel hohe Anzahl von Testfällen. In der Praxis und in den Normen ist daher die Modified Condition/Decision Coverage (MC/DC) relevant, bei der die Anzahl der Testfälle reduziert wird und die Aussagekraft der Testabdeckung ausreichend hoch bleibt.

Autor: Klaus Lambertz, Verifysoft Technology GmbH