Verifysoft TECHNOLOGY

Testwell CTC++

Testwell CTC++ Test Coverage Analyser

Was ist Testwell CTC++?

Testwell CTC++ ist das führende Code Coverage Tool um die Testabdeckung auf dem Host und allen Embedded Targets (selbst den kleinsten) zu messen. Das Tool kann verwendet werden, um die Code-Coverage-Anforderungen von Normen wie DO-178C, ISO 26262, EN 50128 und IEC 60880 zu erfüllen.
Hunderte von Firmen in über 30 Ländern weltweit nutzen Testwell CTC++ um die Qualität ihrer Software sicherzustellen. Testwell CTC++ ist die erste Wahl für Firmen in der Luftfahrt, der Automobilindustrie, der Medizintechnik sowie in vielen weiteren Industriezweigen, in denen Qualität wichtig ist und eine hohe Testabdeckung nachgewiesen werden muss.

Warum Code Coverage?

Code Coverage ist eine Messgröße, die angibt, zu wie viel Prozent der Quellcode eines Programms getestet ist. Diese kann auch als eine indirekte Messgröße der Softwarequalität betrachtet werden.
Teile des Programms, die noch nicht durch einen Testfall ausgeführt wurden, werden angezeigt. Auf diese Weise hilft Testwell CTC++ zusätzliche Testfälle zu kreieren um die Testabdeckung zu steigern. Gleichzeitig werden überflüssige Tests vermieden. Code Coverage ist in Modultestphase am nützlichsten. Es ist jedoch ebenfalls in der Integrationstestphase und in anderen Teststufen hilfreich.
Test Coverage ist bei der Entwicklung von sicherheitskritischer Software « dringend empfohlen » also quasi verbindlich. Standards wie **DO-178C** (Software Considerations in Airborne Systems and Equipment Certification), **IEC/EN 61508** (Functional safety of electrical/electronic, programmable electronic safety-related systems), **EN 50128** (Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems), **IEC 60880** (Nuclear Power) und **ISO 26262** (Functional safety of road vehicles) fordern je nach Sicherheitsstufe spezielle Coverage-Levels. Je höher das Risiko, desto höher ist die geforderte Testabdeckungsstufe.
Aber auch für die Entwicklung im nicht-sicherkeitskritischen Bereich ist Qualität wichtig. Die Testcoverage unterstützt Sie dabei, diese zu gewährleisten.

Erwiesener Erfolg von Testwell CTC++

Testwell CTC++ Test Coverage Analyser wird bereits seit 1989 erfolgreich in der Industrie eingesetzt. Die ersten Versionen wurden innerhalb der NOKIA Gruppe von den Firmen Oy Softalp Ab, Nokia Data Systems Oy and ICL Personal System Oy in Tampere (Finnland) entwickelt. 1992 wurde die Firma Testwell gegründet mit dem Ziel die „Testwell"-Tools weiter zu entwickeln, zu vermarkten und sich um den Support der Kunden zu kümmern.
Olavi Poutanen, Gründer der Firma Testwell, war bereits für die Entwicklung der Tools verantwortlich als er noch mit den oben genannten Firmen zusammenarbeitete.
Nach zehn Jahren voller erfolgreicher Geschäfte in Europa, erwirbt die Verifysoft Technology GmbH 2013 die Eigentumsrechte an der Software Testwell CTC++.
Heuten nutzen hunderte von Firmen in weltweit über 30 Ländern Testwell CTC++ mit großem Erfolg.

Warum ist Testwell CTC++ so erfolgreich?

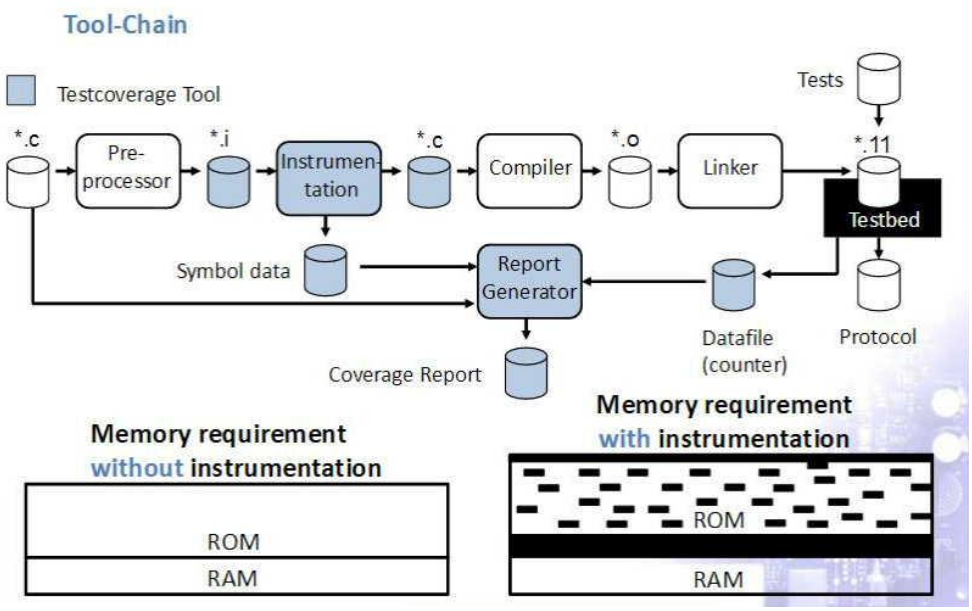
Testwell CTC++ ist die erste Wahl um die Testabdeckung zu analysieren.
Das Tool ist sehr einfach anzuwenden. Es unterstützt zahlreiche Sprachen (C,C++, Java und C#) und analysiert alle Testabdeckungsstufen bis hin zur Modified Condition/Decision Coverage (MC/DC) und Multicondition Coverage. Testwell CTC++ funktioniert mit allen Compilern und Cross-Compilern.
Das Tool hat einen sehr geringen Instrumentation-Overhead. Da es auf allen Embedded Targets funktioniert, ist Testwell CTC++ im Bereich der Softwareentwicklung auf den kleinsten Targets weit verbreitet. Das Tool ist in viele IDEs, Toolchains sowie in zahlreiche Testumgebungen integriert. Die Coverage-Reports sind deutlich und aussagekräftg.
Viele unserer Kunden vergleichen Testwell CTC++ mit einem Jeep der US-Army: Einfach zu handhabe, funktioniert und läuft in jedem „Gelände" (vor allem bei Nutzung des Host-Target-add-ons).
Sind Sie auf der Suche nach einem Coverage-Tool, das in jeder Situation funktioniert? **Testwell CTC++ ist die beste Wahl!**

Wie funktioniert Testwell CTC++?

Testwell CTC++ Test Coverage Analyser ist sehr einfach zu nutzen. Es sind keine Veränderungen im Produktivcode erforderlich. Die Testabdeckung wird durch die Instrumentierung des Codes durchgeführt. Dem Sourcecode werden automatisch Zähler hinzugefügt, die verfolgen, wie oft die verschiedenen Teile des Codes ausgeführt (getestet) wurden.
Die Makefiles dazu zu bringen, den Code zu instrumentieren ist sehr einfach. Hierfür sind keinerlei Änderungen des Makefiles notwendig. Die Instrumentierung wird durchgeführt, in dem einfach ´ctc´ vor dem Aufruf des Compilers/Linkers gesetzt wird.

Testwell CTC++ kann über die **Kommandozeile** oder auch direkt aus vielen Entwicklungsumgebungen heraus genutzt werden.
Bei der Nutzung im Kommandozeilenmodus (über Makefiles oder andere Buildskripts) ist die Instrumentierung lediglich dem Kompilier-/Linkkommando vorangestellt. **Es sind keinerlei Änderungen der Quelldateien oder der Buildskripts erforderlich.** Testläufe werden mit der instrumentierten Version des Programms in der gleichen Weise wie im originalen Programm durchgeführt.
Der **Instrumentierungs-Overhead bzgl. Speicherplatz und Ausführungszeit ist mit Testwell CTC++ äußerst gering.** Dank besonderer technischer Konzepte für das On-Target-Test-Coverage kann Testwell CTC++ zur Messung der Testabdeckung in allen (selbst kleinsten) embedded Targets und Microcontrollern genutzt werden.
Die Berichte für Coverage und Execution-Profiling werden als Text, HTML und XML sowie in einem Excel-Input-Format ausgegeben. Der Report ist informativ und übersichtlich. Die Coverage-Reporte werden als Übersicht (Summary) mit Prozentzahlen und als Detailansichten auf verschiedenen Niveaus angegeben.

Das Browsen innerhalb der HTML-Ausgabe ist sehr einfach. Als Übersicht stehen farbige Histogramme (Farbe abhängig von der Testabdeckung – nicht ausreichend getestete Teile werden rot angezeigt) mit prozentualen Angaben der Testabdeckung zur Verfügung. Mit wenigen Mausklicks kann in detaillierte Informationen gezoomt werden. Testwell CTC++ kann mit allen Unit-Test-Tools sowie vielen Testumgebungen und GUIs eingesetzt werden.



Unterstützung aller Compiler und Cross-Compiler

Jede einzelne Lizenz des Testwell CTC++ Code Coverage Analysers unterstützt **alle Compiler** und alle Cross-Compiler.
Auch neue bzw. zusätzliche Compiler, die Sie eventuell in Zukunft nutzen, werden von der Lizenz abgedeckt. Es sind keine zukünftigen Ausgaben nötig, um Testwell CTC++ an neue Compiler anzupassen.

Testwell CTC++ hat aktuell fertige Settings für viele Compiler.
Die aktuelle Liste (Juni 2016) umfasst folgende Compiler:

Altium Tasking: cc166, ccm16c, cc51, ccarm, ccmcs, ccpcp, cctc, ccm16c
Borland/Inprise/Paradigm/Codegear compilers: bcc, bcc32, pcc, pcc32 (Paradigm)
Cosmic compilers: cx6805, cx6808, cx6812, cxs12x, cxxgate, cx6811, cx6816, cx332, cxst10, cxstm8, cxst7
gcc and all gcc based cross-compilers: i586-mingw32msvc-gcc, x86_64-linux-gnu-gcc, m68k-palmos-coff-gcc, tricore-gcc, arm-linux-gnueabi-gcc., arm-none-eabi-gcc, arm-none-linux-gnueabi-gcc, arm-elf-gcc, arm-montavista-linux-gnueabi-gcc, pic30-gcc, pic32-gcc, avr-gcc, xc16-gcc, mlx16-gcc, thumb-epoc-pe-gcc, arm4-epoc-pe-gcc, armv-epoc-pe-gcc, powerpc-wrs-linux-gnu-e500v2-glibc_small-gcc, *-gcc, *-*-gcc, *-*-*-gcc, HPUX CC, HP C++, aCC
IAR compilers and toolchains: iccm16c, icc430, icc8051, iccarm, iccavr, iccavr32, icccf, icchcs12, iccmaxq, iccdspic, iccpic18, icccr16c, icc78k, icc78k0r, iccv850, icch8, iccm32c, iccr32c, iccsam8
Fujitsu/Softune: fcc907s, fcc911s
GHS/GreenHills/Multi: ccv850, cxv850, ccmips, cxmips, ccarm, cxarm, ccppc, cxppc, gcc (GreenHill, not GNU)
Hitachi: shc, shcpp, ch38, ccrx
HI-Tech PICC compilers (Windows and Linux): picc, picc18, picc32, dspicc
VisualDSP++: ccblkfn, cc21k, ccts
Intel compilers (all platforms): icc, ic86, ic96
Java compilers: javac, jikes, ecj, gcj, kaffe
Keil compiler: c51, c166, c251, ca, armcc
Matlab/Simulink: lcc
Metaware: hcarm and others
Microsoft compiler: cl on host both 32 and 64 bit, cl for Smartphones and PocketPC, csc C# compiler, vjc J# compiler
Mitsubishi: nc30, nc308, nc77, nc79
Mono compilers: mcs, gmcs, smcs, dmcs
Motorola: chc12
NEC: ca830, ca850
Pathscale pathcc/pathCC
Renesas: shc, shcpp, ch38, ccrx, nc, nc308, nc77, nc79A, cc32R, CS+/CubeSuite+ cc78k0, cc78k0r, cx, ca850
Sun compilers: WorkShop compilers, javac
Symbian: various compilers
TI Code Composer Studio: cl2000
Trimedia: tmcc
Windriver: ccarm, ccsimpc, g++simpc, g++arm, cchppa, ccsimso, ccsparc, cc68k, cc386, cc960, ccmips, ccppc

Unsere Kunden nutzen Testwell CTC++ auch mit anderen Compilern. Die Anpassung an weitere Compiler ist einfach und kostenlos. Falls erforderlich, können die Anpassungen sogar durch den Kunden selbst vorgenommen werden.

Weitere Anpassungen an Compiler werden hinzugefügt, sobald ein Kunde dies benötigt. Bitte klicken sie auf http://verifysoft.com/de_code_coverage_all_compilers.html für eine aktualisierte Auflistung.

Integration in viele Entwicklungsumgebungen (IDEs)

Testwell CTC++ Test Coverage Analyser ist in viele Toolchains, Testumgebungen und Softwarequalität-Tools integriert.
Derzeit (Juni 2016) sind Integrationen in folgende IDEs verfügbar: Visual Studio, IAR-Workbench, Borland 5.02, BeckIPC, Eclipse, Fujitsu Softune, Renesas. Testwell CTC++ ist auch mit IDEs wie Keil µVision IDE (- c51, C166, c251, ca, - armcc) sowie mit ARM DS-5 armcc nutzbar.
Testwell CTC++ kann einfach an andere IDEs angepasst werden, auch wenn diese hier nicht aufgelistet sind.
Die Voraussetzungen für eine solche Integration sind: editierbare Kommandozeilenmuster, editierbare Makefile-Erstellung und modifizierbare Werkzeuge (wie Code::Blocks-Tools). Sobald eine dieser Voraussetzungen erfüllt ist, können Sie Testwell CTC++ sogar selbst integrieren. Ein Video, das erklärt wie Testwell CTC++ in Ihre Entwicklungsumgebung integriert werden kann finden Sie hier: https://www.youtube.com/watch?v=Pen_YNk_fQ0&feature=youtu.be

Integration in zahlreiche Werkzeugketten und Testumgebungen

Testwell CTC++ ist momentan (Juni 2016) in folgende Toolchains, Testumgebungen und Softwarequalität-Tools integriert:

- CATIA Systems – AUTOSAR Builder (Dassault Systemes)
- dSpace SystemDesk
- dSpace TargetLink
- Imagix 4D
- Jenkins
- Lauterbach
- MATLAB Simulink
- PikeTec Time Partition Testing (TPT)
- SonarQube
- QTronic TestWeaver
- QTronic Silver

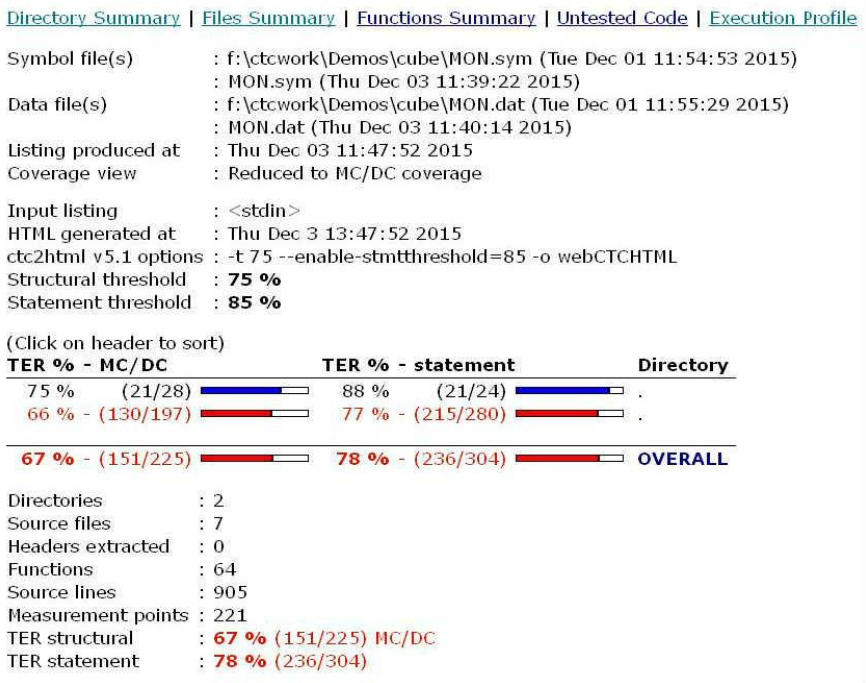
Bitte kontaktieren Sie uns, wenn Sie Testwell CTC++ in eine andere IDE integrieren möchten. Falls Sie als Toolhersteller an einer Integration mit Testwell CTC++ interessiert sind, freuen wir uns ebenfalls über Ihre Kontaktaufnahme.

Klare und aussagekräftige Reports

Testwell CTC++ analysiert die Testabdeckung für alle Coveragelevels bis hin zu MC/DC und Multicondition Coverage und stellt die Ergebnisse der Analyse in klaren und aussagekräftigen Reports dar. Die Ausgaben zeigen einerseits eine Top-Level-Ansicht, welche die prozentuale Testabdeckung auf verschiedenen Stufen zeigt, sowie detaillierte Ansichten. In den detaillierten Ansichten sind die Coverage-Aussagen mit dem zugehörigen Teilen des Quellcodes verlinkt.

CTC++ Coverage Report (HTML format, hierarchical with 4 levels)

- Directory Summary (Generelle Informationen)



Directory Summary | Files Summary | Functions Summary | Untested Code | Execution Profile

Symbol file(s) : f:\ctcwork\Demos\cube\MON.sym (Tue Dec 01 11:54:53 2015)
: MON.sym (Thu Dec 03 11:39:22 2015)
Data file(s) : f:\ctcwork\Demos\cube\MON.dat (Tue Dec 01 11:55:29 2015)
: MON.dat (Thu Dec 03 11:40:14 2015)
Listing produced at : Thu Dec 03 11:47:52 2015
Coverage view : Reduced to MC/DC coverage

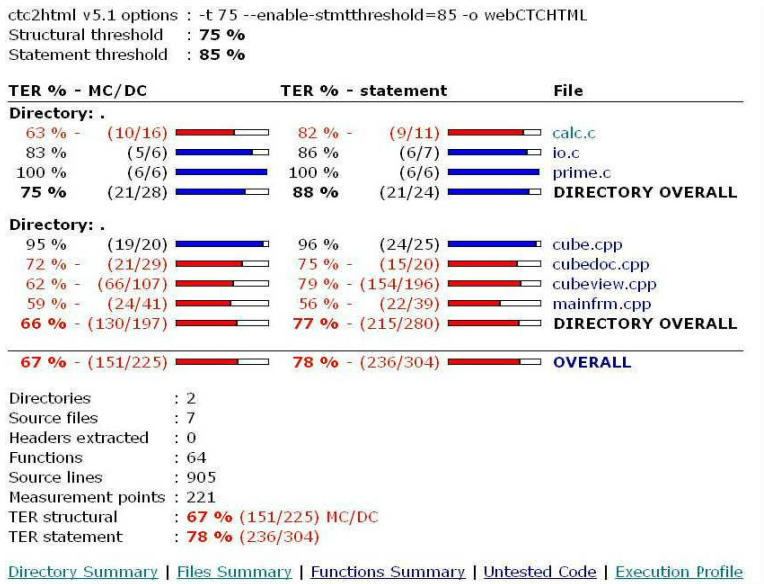
Input listing : <stdin>
HTML generated at : Thu Dec 3 13:47:52 2015
ctc2html v5.1 options : -t 75 --enable-stmtthreshold=85 -o webCTCHTML
Structural threshold : **75 %**
Statement threshold : **85 %**

(Click on header to sort)

TER % - MC/DC	TER % - statement	Directory
75 % (21/28)	88 % (21/24)	.
66 % - (130/197)	77 % - (215/280)	.
67 % - (151/225)	**78 %** - (236/304)	**OVERALL**

Directories : 2
Source files : 7
Headers extracted : 0
Functions : 64
Source lines : 905
Measurement points : 221
TER structural : **67 %** (151/225) MC/DC
TER statement : **78 %** (236/304)

- Files Summary (Zoom in die Dateien der Verzeichnisse)



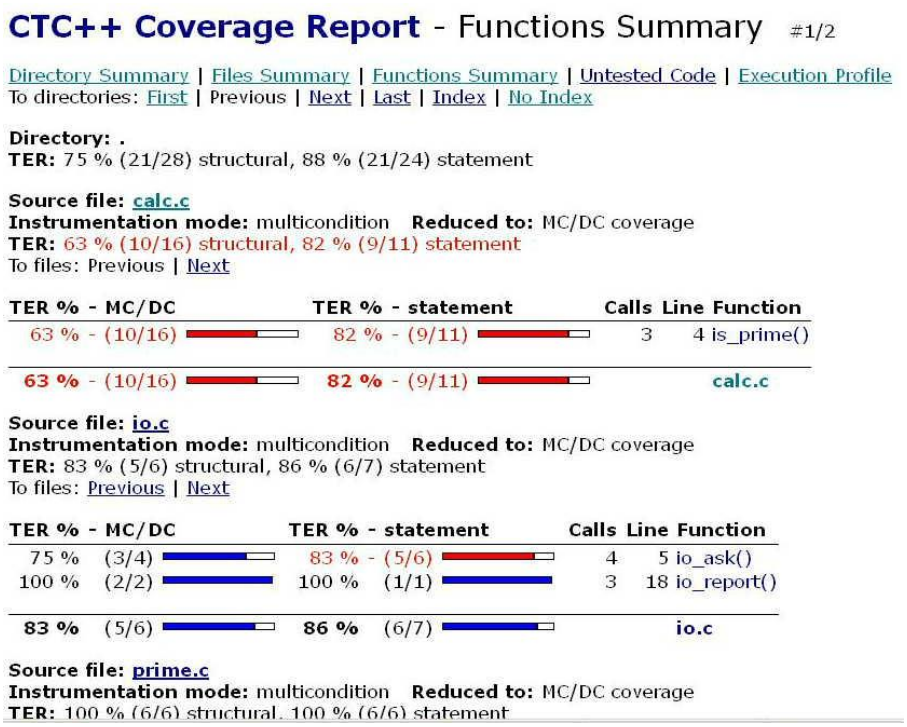
ctc2html v5.1 options : -t 75 --enable-stmtthreshold=85 -o webCTCHTML
Structural threshold : **75 %**
Statement threshold : **85 %**

TER % - MC/DC	TER % - statement	File
Directory: .		
63 % - (10/16)	82 % - (9/11)	calc.c
83 % (5/6)	86 % (6/7)	io.c
100 % (6/6)	100 % (6/6)	prime.c
75 % (21/28)	**88 %** (21/24)	**DIRECTORY OVERALL**
Directory: .		
95 % (19/20)	96 % (24/25)	cube.cpp
72 % - (21/29)	75 % - (15/20)	cubedoc.cpp
62 % - (66/107)	79 % - (154/196)	cubeview.cpp
59 % - (24/41)	56 % - (22/39)	mainfrm.cpp
66 % - (130/197)	**77 %** - (215/280)	**DIRECTORY OVERALL**
67 % - (151/225)	**78 %** - (236/304)	**OVERALL**

Directories : 2
Source files : 7
Headers extracted : 0
Functions : 64
Source lines : 905
Measurement points : 221
TER structural : **67 %** (151/225) MC/DC
TER statement : **78 %** (236/304)

Directory Summary | Files Summary | Functions Summary | Untested Code | Execution Profile

- Functions Summary
  (Zoom in die Methoden und Funktionen der Dateien)



CTC++ Coverage Report - Functions Summary #1/2

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

Directory: .
TER: 75 % (21/28) structural, 88 % (21/24) statement

Source file: calc.c
Instrumentation mode: multicondition **Reduced to:** MC/DC coverage
TER: 63 % (10/16) structural, 82 % (9/11) statement
To files: Previous | Next

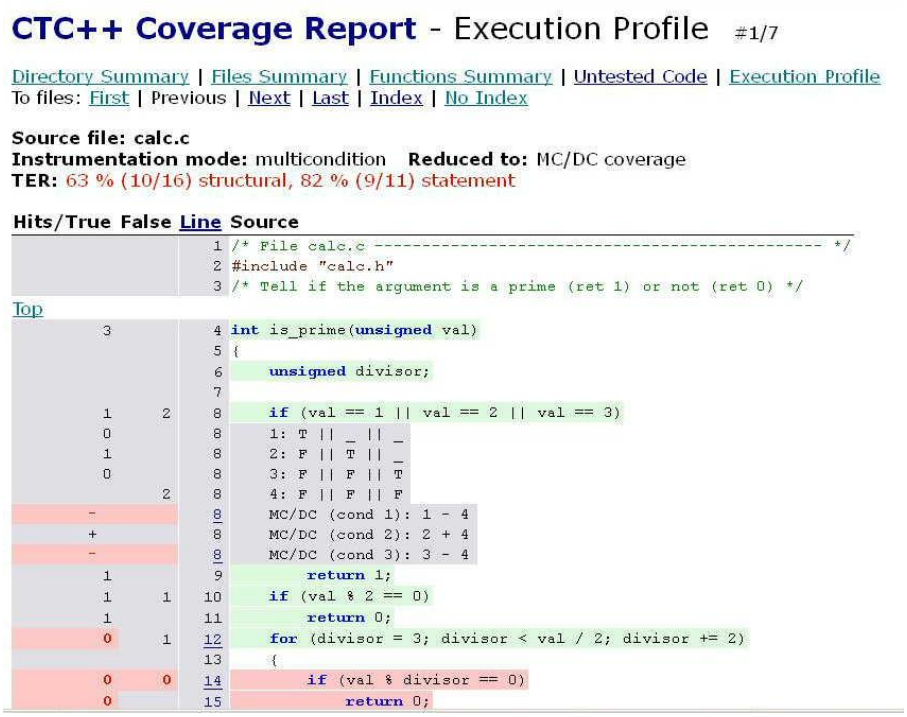
TER % - MC/DC	TER % - statement	Calls	Line	Function
63 % - (10/16)	82 % - (9/11)	3	4	is_prime()
63 % - (10/16)	**82 %** - (9/11)			**calc.c**

Source file: io.c
Instrumentation mode: multicondition **Reduced to:** MC/DC coverage
TER: 83 % (5/6) structural, 86 % (6/7) statement
To files: Previous | Next

TER % - MC/DC	TER % - statement	Calls	Line	Function
75 % (3/4)	83 % - (5/6)	4	5	io_ask()
100 % (2/2)	100 % (1/1)	3	18	io_report()
83 % (5/6)	**86 %** (6/7)			**io.c**

Source file: prime.c
Instrumentation mode: multicondition **Reduced to:** MC/DC coverage
TER: 100 % (6/6) structural, 100 % (6/6) statement

- Execution Profile
  Die detaillierte Ansicht zeigt die Ausführungszähler und den Quellcode.
  Unvollständig ausgeführte Codeabschnitte werden in rot dargestellt.



CTC++ Coverage Report - Execution Profile #1/7

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: 63 % (10/16) structural, 82 % (9/11) 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			
3		4	int is_prime(unsigned val)
		5	{
		6	unsigned divisor;
		7	
1	2	8	if (val == 1 || val == 2 || val == 3)
0		8	1: T || _ || _
1		8	2: F || T || _
0		8	3: F || F || T
	2	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
1		9	return 1;
1	1	10	if (val % 2 == 0)
1		11	return 0;
0	1	12	for (divisor = 3; divisor < val / 2; divisor += 2)
		13	{
0	0	14	if (val % divisor == 0)
0		15	return 0;

Execution Profile Listing

Zeigt, wie oft jeder Teil des Codes ausgeführt wurde
Zeigt Teile des Codes, die während der Tests noch nicht ausgeführt wurden (textueller Bericht)

Example of CTC++ Execution Profile Listing

```
*******************************************************************************
*           CTC++, Test Coverage Analyzer for C/C++, Version 8.0              *
*                                                                             *
*                          EXECUTION PROFILE LISTING                          *
*                                                                             *
*                    Copyright (c) 1993-2013 Testwell Oy                      *
*              Copyright (c) 2013-2015 Verifysoft Technology GmbH             *
*******************************************************************************

Symbol file(s) used   : MON.sym (Tue Nov 17 08:13:14 2015)
Data file(s) used     : MON.dat (Tue Nov 17 08:13:44 2015)
Listing produced at   : Tue Nov 17 08:14:16 2015
Coverage view         : As instrumented


MONITORED SOURCE FILE : prime.c
INSTRUMENTATION MODE  : multicondition

 HITS/TRUE      FALSE    LINE DESCRIPTION
================================================================================

         1                 8 FUNCTION main()
         3           1    12   while (( prime_candidate = io_ask ( ) ) > 0)
         2           1    14     if (is_prime ( prime_candidate ))
                          15     }+
                          16     else
                          17     }+
                          18   }+
         1                19   return 0
                          20 }

***TER 100 % (  6/  6) of FUNCTION main()
       100 % (  6/  6) statement
--------------------------------------------------------------------------------
```

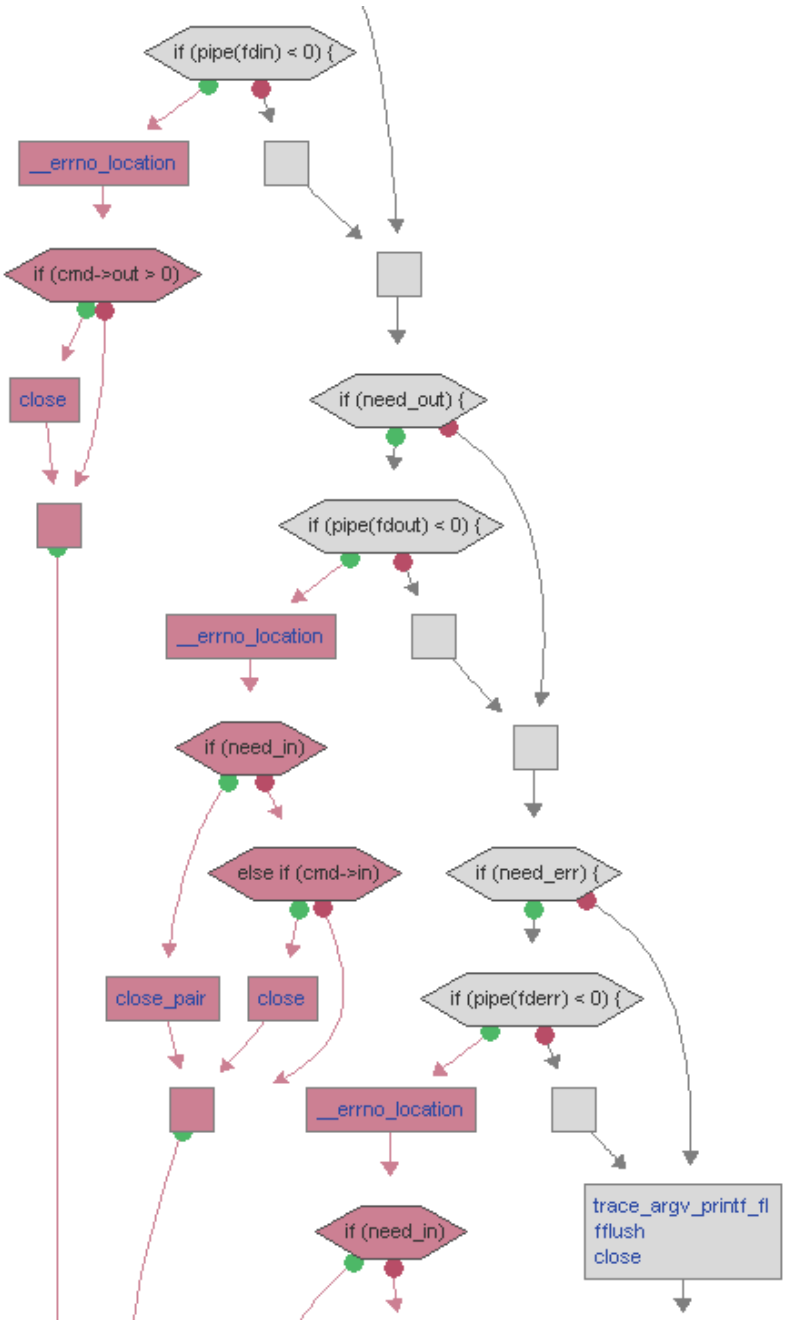
Untested Code Listing

Zeigt die Teile vom Code, die noch nicht getestet wurden (textueller Bericht)

```
 HITS/TRUE      FALSE    LINE DESCRIPTION
================================================================================

         4                 5 FUNCTION io_ask()
         0          4 -    11   if (( amount = scanf ( "%u" , & val ) ) <= 0)

--------------------------------------------------------------------------------


MONITORED SOURCE FILE : calc.c
INSTRUMENTATION MODE  : multicondition

 HITS/TRUE      FALSE    LINE DESCRIPTION
================================================================================

         3                 4 FUNCTION is_prime()
         1          2      8   if (val == 1 || val == 2 || val == 3)
         0            -    8     1: T || _ || _
         0            -    8     3: F || F || T
         0          1 -    12  for (;divisor < val / 2;)
         0          0 -    14    if (val % divisor == 0)
         0            -    15      return 0

--------------------------------------------------------------------------------


SUMMARY
=======

Source files       : 3
Headers extracted  : 0
Source lines       : 59
Measurement points : 27
TER                : 76 % (22/29) multicondition
TER                : 88 % (21/24) statement
```

Execution Time Listing
Zeigt die kommulative und durchschnittliche Ausführungszeit der Funktionen (textueller Bericht)

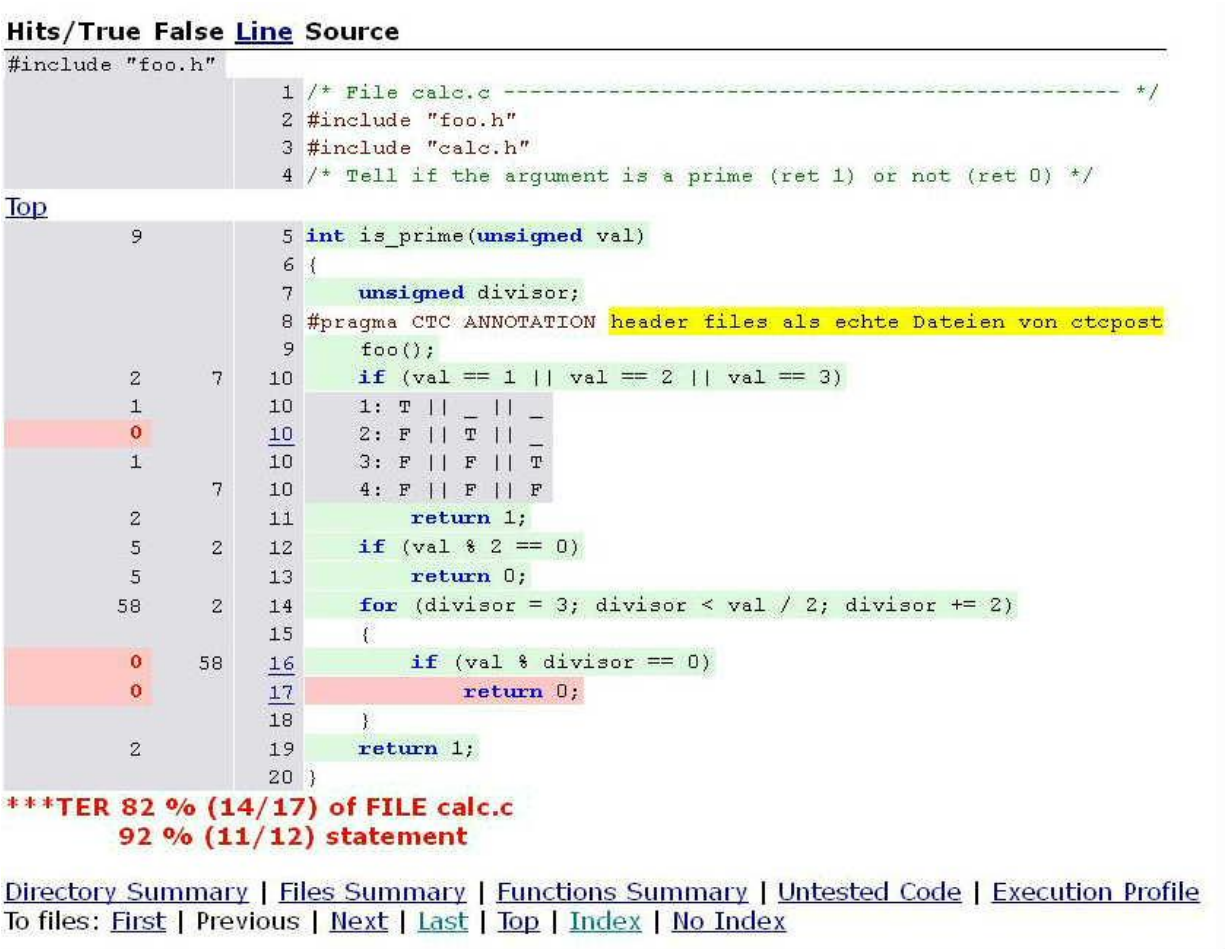
Control Flow Graph (Imagix 4D)
Indem Testwell CTC++ zusammen mit Imagix 4D angewendet wird, werden die Coverage-Daten in einem Kontrollflussdiagramm dargestellt.



In diesem logischen Layout repräsentieren die diamant-förmigen Symbole Entscheidungen. Ungetesteter Code wird mit roten Formen und Pfaden dargestellt.

Kommentare innerhalb des Coverage-Reports

Mit Testwell CTC++ können Kommentare in den Sourcecode einfügen werden, die im Prozess mit CTC++ nicht verloren gehen und auch im HTML-Report dargestellt werden. Kommentare können beispielsweise dazu verwendet werden um zu erklären, warum manche Teile des Codes noch nicht ausgeführt wurden (vgl. im Bild unten den gelben Text).



Hits/True	False	Line	Source
#include "foo.h"			
		1	/* File calc.c ---------- */
		2	#include "foo.h"
		3	#include "calc.h"
		4	/* Tell if the argument is a prime (ret 1) or not (ret 0) */
Top			
9		5	int is_prime(unsigned val)
		6	{
		7	unsigned divisor;
		8	#pragma CTC ANNOTATION header files als echte Dateien von ctcpost
		9	foo();
2	7	10	if (val == 1 || val == 2 || val == 3)
1		10	1: T || _ || _
0		10	2: F || T || _
1		10	3: F || F || T
	7	10	4: F || F || F
2		11	return 1;
5	2	12	if (val % 2 == 0)
5		13	return 0;
58	2	14	for (divisor = 3; divisor < val / 2; divisor += 2)
		15	{
0	58	16	if (val % divisor == 0)
0		17	return 0;
		18	}
2		19	return 1;
		20	}

***TER 82 % (14/17) of FILE calc.c
92 % (11/12) statement

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

Führt Kernelcoverage durch

Mit Testwell CTC++ können Sie die Testabdeckung im laufenden Linux-Kernel analysieren.

Die Messung der Testcoverage im Kernel ist eine Herausforderung für alle Coverage-Tools, die mit Code-Instrumentierung arbeiten. Bei der Analyse der Coverage des Kernel-Codes können die instrumentierten Sonden keine Bibliotheksfunktionen oder Systemaufrufe verwenden. Die Instrumentierung und die Laufzeitunterstützung von Testwell CTC++ benötigt lediglich den C-Code und keinerlei weitere Systemunterstützung. Aus diesem Grund kann Testwell CTC++ problemlos im Kernel ausgeführt werden.

Als "Proof of Concept" für die Messung der Testabdeckung im Kernel haben wir den kompletten Linuxkernel instrumentiert und eine Testsuite auf ihm ausgeführt.

Umfassende Unterstützung von Programmiersprachen

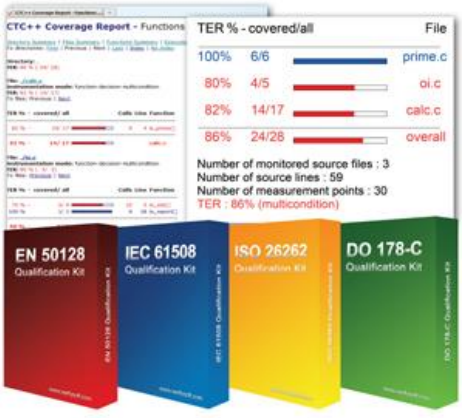
Testwell CTC++ unterstützt **C, C++, Java und C#.**
Ursprünglich ist Testwell CTC++ zur Messung der Testabdeckung für C und C++ entwickelt worden. Seit 2007 stehen Add-on-Packages zur Ausweitung auf Java und C# zur Verfügung.

Tool Qualification-Kit verüfgbar

Das Tool-Qualification-Kit für Testwell CTC++ enthält Dokumentationen, Testfälle und Prozeduren, welche die Qualifizierung des Werkzeugs in Projekten, die auf den Normen **ISO 26262, IEC 61508, EN 50128, IEC 60880 und DO-178C** basieren untersützen.
Das Kit beinhaltet einen Tool-Qualification-Plan, Tool Operational Requirements und andere Inhalte, die benötigt werden, um Testwell CTC++ für die Nutzung bei sicherheitskritischen Projekten zu qualifizieren.
Durch die im Kit enthaltenen Testfälle können Nutzer die Abwesenheit von Fehlern bei der Anwendung in ihrer Entwicklungsumgebung nachzuweisen.
Das Qualification-Kit vereinfacht die Zertifizierung von embedded Software, deren Testabdeckung mit Testwell CTC++ analysiert wird. Die Benutzer können die Artefakte des Tool-Qualifikation-Kits selbst für konkrete Projekte anpassen. Das Qualification-Kit für Testwell CTC++ ist konform zu den Sicherheitsnormen ISO 26262, IEC 61508, EN-50128 und DO-178C.
Es besteht aus:

- Qualification-Support-Tool, welches durch die Qualifikation führt und folgende Dokumente generiert:
  - Tool-Classification-Report
  - Tool-Qualification-Plan/Report
  - Tool-Safety-Manual
  - Testplan
- Test-Automatisierungs-Einheit
- Test-Suite mit Testfällen.
- Benutzerhandbuch des Kits

Das Qualification-Kit für Testwell CTC++ ist erfolgreich bei vielen namhaften Firmen im Automotiv- und Luftfahrtbereich im Einsatz. Viele unserer Kunden bewerten das Testwell CTC++-Qualification-Kit als eines der besten aktuell im Markt verfügbare Kits.



Was sagen unsere Kunden über Testwell CTC++?

Wir sind stolz auf hunderte von Kunden, die Testwell CTC++ mit großer Zufriedenheit nutzen um höchste Qualität zu gewährleisten.
Hier sind einige Anwenderberichte:

"Testwell CTC++ wird in unseren embedded Projekten auf embedded Targets eingesetzt. Es war einfach auf den Target zu integrieren. Das Tool hat ein gut geschriebenes Benutzerhandbuch und Verifysoft überzeugt mit einem guten Kundensupport."
Anna Andgart, Software-Entwicklerin, ABB AB, Control Technologies, Schweden

„Wir nutzen Testwell CTC++ sehr intensiv in unserer Entwicklung und dem Testen von sicherheitskritischer Software für Kernkraftwerke.
Das Tool wird dabei nicht nur auf Host Systemen sondern auch auf anderen Embedded Target Plattformen genutzt. Ziel ist es, mühelos Statement/Decision oder MC/DC Coverage zu messen, um die Testanforderungen der für Kernkraftwerke gültigen Norm IEC608080 sowie die auf europäischer Ebene verabschiedeten und in den "Licensing of safty critical software for nuclear reactors" festgelegten Regularien zu erfüllen.
Der Support von Verifysoft ist sehr hilfreich, schnell, direkt und zielgerichtet."
Thorsten Oertel, Senior Firmware Engineer, AREVA GmbH, Deutschland

„Fehlende Abdeckung ist vor allem auf fehlende Anforderungen oder unvollständige Prüfvorschriften zurück zu führen. So verwenden wir Testwell CTC++ innerhalb der Entwicklung eines fortgeschrittenen Fahrerassistenzsystem, um fehlende Anforderungen zu finden und unsere Testspezifikationen zu vervollständigen. Das hilft uns, einen Entwicklungsprozess nach ISO 26262 aufzubauen. Es war leicht Testwell CTC++ in unsere bestehende Entwicklungstoolchain zu integrieren und es liefert sofortige Ergebnisse."
Michael Kalusche, Projektmanager, Bertrandt Ingenieurbüro GmbH, Deutschland

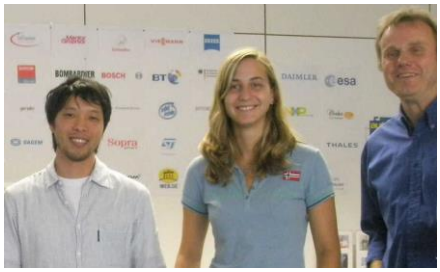



„Wir nutzen Testwell CTC++ für die Modultests der eingebetteten Software für Kombiinstrumente. Der große Vorteil ist die hohe Reproduzierbarkeit sowie die schnelle Ausführung. Mit diesem Werkzeug können wir das komplette Softwaremodul bei jeder kleinen Änderung testen. Nicht nur den geänderten Teil. Dies bringt uns sicherere Ergebnisse in kürzerer Zeit."
Iaran Gadotti, R&D Manager, Continental Brasil Indústria Automotiva Ltda., Brasilien

„IAV ist einer der größten Entwicklungsdienstleister in der Automobilbranche. In unserem Bereich entwickeln wir Serien-Software für die Domäne Karosserieelektronik. Hierzu müssen neben Anforderungen an die Entwicklungsprozesse nach Automotive SPICE auch Entwicklungsmethoden der ISO26262 u.a. bis ASIL B berücksichtigt werden. Testwell CTC++ kann durch seine Plattform-Unabhängigkeit einfach in die unterschiedlichen Tool- und HW-Umgebungen unserer Kunden integriert werden und bietet somit eine sehr gute Unterstützung unserer Testaktivitäten auf SW-Modulebene um die Testabdeckung parallel auf der Host- und Zielplattform zu messen. Um unsere Qualität und Testabdeckung kontinuierlich zu überwachen, haben wir Testwell CTC++ in unseren Continuous Integration Buildprozess integriert, um zeitnah Lücken in der Testabdeckung zu erkennen. Durch die hohe Etablierung von Testwell CTC++ im Markt, ist dieses Werkzeug ein fester Bestandteil der IAV Tool-Kette geworden."
Marko Meyer, Senior Projektmanager, IAV GmbH, Deutschland

"Wir setzen Testwell CTC++ in unseren Luftfahrtprojekten ein. Ziel ist eine "Requirements-based test coverage analysis" um den Test-Kriterien der DO178B zu genügen. Das Tool unterstützt uns in der Analyse unseres C-Quellcodes, um z.B. "Dead Code" zu lokalisieren. Wir konnten CTC++ problemlos nutzen."
Michael Görsdorf, Software-Entwicklung / R&D Software, Kappa optronics GmbH, Deutschland

"Testwell CTC++ ist ein IEC61508 T3-qualifiziertes Werkzeug, welches wir zum Testen von Embedded Anwendungen nutzen. Wir haben mit diesem Tool die Testabdeckung sowohl auf dem Target als auch auf dem Windows-Host überprüft. Das Werkzeug lässt sich einfach in die embedded Anwendugen einbinden und generiert gute Berichte."
S. Sánchez-Manjavacas, Senior Firmware/FPGA Architekt, Kongsberg Maritime, Norwegen



"Wir nutzen Testwell CTC++ in unserem Embedded-Projekt. Es hat uns dabei geholfen, jeden nicht abgedeckten Code und Kontrollpfad zu erkennen, sofern unvollständige Unit-Test-Spezifikationen die Ursache waren. Die Integration ist simpel und die Nutzerunterstützung hervorragend."
Srinivasulu, Projektmanager, Knorr-Bremse Technical Center, India

"Wir setzen CTC++ umfassend zur Begleitung von Entwicklung und Test sicherheitsgerichteter Software auf Mikrocontrollern ein. CTC++ ermöglicht es uns, die Überdeckung einzelner Software-Bestandteile zu verfolgen, um unsere Tests dementsprechend präzise darauf auszurichten, alle Funktionszweige zu erfassen. Dies dient gleichermaßen als Nachweis zur Zertifizierung bei einer Prüfstelle."
Thomas Bartzick, Abteilung Software-Test, ISH, Deutschland

"Wir nutzen Testwell CTC++ um die Testüberdeckung unserer Unit- und Systemtests unserer (Embedded) Targets zu ermitteln. Die Berichte sind klar, einfach und beinhalten was wie benötigen. Die Unterstützung durch Verifysoft ist ordentlich und schnell. Die CTC-Bearbeitung kann leicht in einfache Erstellungen eingefügt werden."
Kees Valkhof, Tester, Lely, Niederlande

" Wir nutzen CTC++ zur Ermittlung der Testabdeckung bei unseren Medizinprodukten im embedded Software Bereich auf Unit-Test Ebene.
CTC++ war gut in unsere Build Umgebung und auf unserem Target zu integrieren und liefert seitdem zuverlässig und schnell die gewünschten Aussagen. Der Support bei eventuellen Fragestellung war schnell und konnte jederzeit weiterhelfen. Wir können CTC++ ohne Einschränkungen weiterempfehlen."
Heiko Schmidt, Software Team Manager, MAQUET Cardiopulmonary AG, Deutschland

"REC Global ist Embedded Software Entwicklungspartner für mehrere große Automobilzulieferer. Testwell CTC ++ ist von unseren Kunden als Werkzeug der Wahl für die Software-Qualitätssicherung anerkannt und wir folgten ihrem Beispiel und setzten es auch für unsere Projekte ein. Wir verwenden es für den Test von Embedded-Anwendungen. Die CTC++-Reports dienen uns als objektives Maß für Testqualität und helfen uns unsere Entwicklungsprozess zu verbessern."
Borivoje Dermanovic, Projektmanager, REC Global, Kroatien

"Wir nutzen Testwell CTC++ um die Testabdeckung der Unit- und Systemtests der Embedded Software unseres Safety-Produkts, das nach IEC61508 erstellt wurde, zu ermitteln. Die Berichte, die Testwell CTC++ erstellt sind klar, einfach und beinhalten alles Notwendige. Testwell CTC++ war einfach in die bestehende Entwicklungstoolchain zu integrieren. Verifysoft überzeugt mit einem guten Kundensupport."
Thomas Schneider, Senior Software Engineer, Schneider Electric, Deutschland

"Wir setzen Testautomatisierung in Bezug auf ein Projekt um, dass 15 Jahre alten Quellcode enthält und die US FDA Software Standards erfüllen muss. Eine Beurteilung der Testabdeckung basierend auf den funktionalen Anforderungen ("requirements-based testing") ist vor diesem Hintergrund unzureichend. Testwell CTC++ ermöglicht uns eine Vielzahl von Testreihen ablaufen zu lassen, um damit nicht abgedeckte, logische Pfade zu identifizieren. Testwell CTC++ ist ein exzellentes Produkt."
Robert Evans, Software Development Engineer, Siemens Medical Diagnostics, USA

"Volvocars Powertrain nutzt Testwell CTC++, weil es die Norm ISO26262 und SPICE unterstüzt und gut mit unserer Modultestplattform hinsichtlich der Testüberdeckungsmessung zusammenarbeitet."
Johannes Foufas, Developer, Volvocars, Schweden

„Wir nutzen Testwell CTC++ regelmäßig in der Luftfahrtindustrie um die Testabdeckung auf Embedded Targets zu messen.
Das Tool hilft uns dabei, die DO-178C Test-Anforderungen zu erfüllen. Die Integration ist sehr einfach und der Support ist ausgezeichnet."
Dr. Martin Ettl, Software Entwicklung, Avionik Straubing, Deutschland

Zusammenfassung

Hunderte Kunden verlassen sich auf Testwell CTC++ Test Coverage Analyser aufgrund seiner Fähigkeit alle Embedded Targets, alle Compiler und alle Coverage Level zu untersützen.

Weitere Informationen

Weitere Informationen und Neuigkeiten über Testwell CTC++ erhalten Sie auf unserer Homepage http://www.verifysoft.com/de_ctcpp.html

Für Webinare und Videos über Testwell CTC++ und unterstützte Normen nutzen Sie bitte den folgenden Link: http://www.verifysoft.com/en_ctcpp_online_presentations.html

Momentan (Juni 2016) gibt es folgende Videos:

- Allgemeine Präsentationen:
  - Testwell CTC++ General Presentation
  - Code Coverage for Embedded Targets
  - Testwell CTC++ Short Introduction (Prime Example)
  - Safety Standards and related Code Coverage Levels
  - ISO 26262 and Code Coverage
- Nutzung mit speziellen IDEs / Umgebungen:
  - Usage of Testwell CTC++ with Microsoft Visual Studio 2008 IDE
  - Usage of Testwell CTC++ with IAR Embedded Workbench IDE
  - Usage of Testwell CTC++ for Embedded Targets (demo is based on an Atmel ATmega 328p µController)
  - How to integrate Testwell CTC++ in your IDE?
  - Integration of Testwell CTC++ in Eclipse
  - Testwell CTC++ and MATLAB/Simulink interface example
  - Testwell CTC++ Code Coverage Lauterbach Trace32
  - Usage of Testwell CTC++ with Renesas CS+ IDE
  - Usage of Testwell CTC++ with Microship MPLAB IDE
- Videos für CTC++ Nutzer
  - Code Coverate on Embedded Targets with HoTa for Testwell CTC++ Users

Haben Sie weitere Fragen oder Interesse an Testwell CTC++? Dann kontaktieren Sie uns bitte:

Verifysoft Technology GmbH
In der Spöck 10-12
77656 Offenburg (Deutschland)
qualify @ verifysoft.com
Telefon: +49 781 127 8118-0