



# Testwell CTC++ Test Coverage Analyser for C and C++

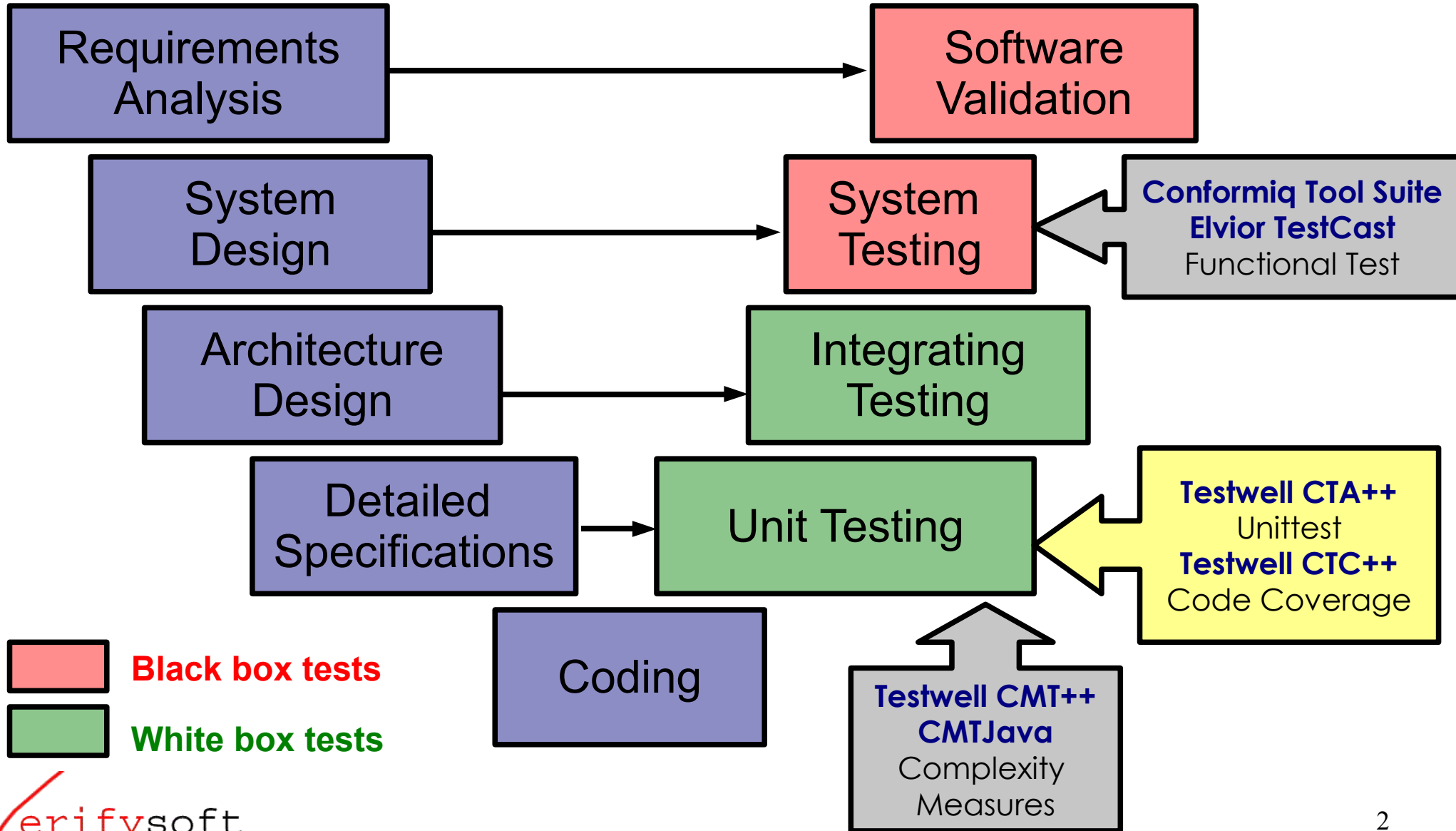
## CTC for Java and C#

© 2012 Verifysoft Technology GmbH

12 Jan 2012

  
**TECHNOLOGY**

# Software development process



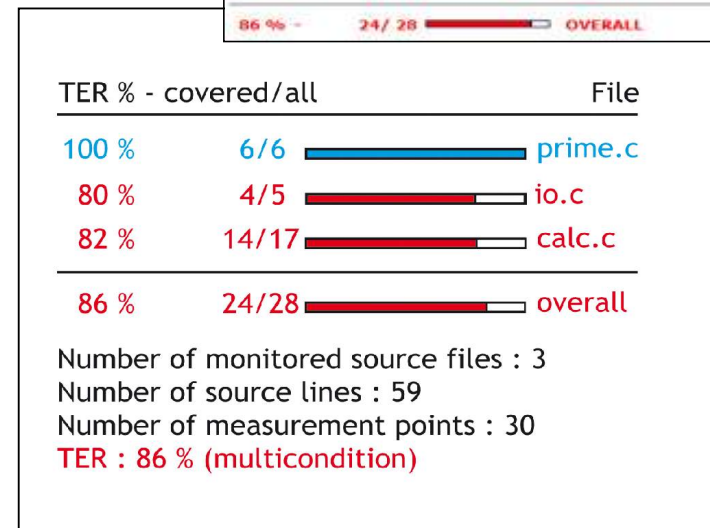
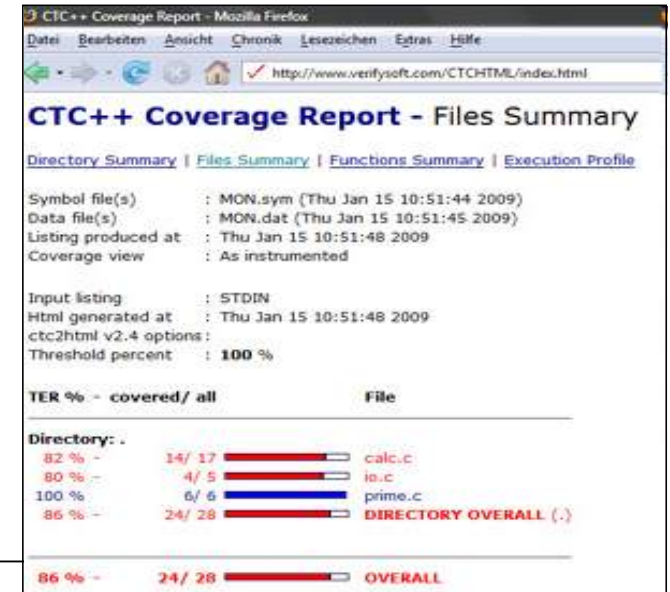
# Testwell CTC++ Test Coverage Analyser

## Code coverage

shows the parts of the code

- executed / not executed
- tested / not tested

**CTC++ development  
for more than 20 years**



# Testwell CTC++ Test Coverage Analyser

## Why measuring the code coverage?

- helps to write better (more adapted) tests/test cases
- helps to avoid that you spend time on writing redundant test cases
- you know when you can stop testing
- you can proof to your customers that your code is tested according to their requirements
- you can be sure that your outsourcing/development partner delivers quality according to your requirements
- ensures high quality with high code coverage
- helps to find "dead code"
- required to obtain certifications.

# Testwell CTC++ Test Coverage Analyser

Testwell CTC++ for **all coverage levels**:

- Function Coverage
- Decision Coverage / Branch Coverage
- Statement Coverage
- Condition Coverage
- Modified Condition/Decision Coverage (MC/DC)
- Multicondition Coverage (MCC)

# Testwell CTC++ Test Coverage Analyser



Testwell CTC++ can be used to obtain certifications in automotive, avionics and medical industries

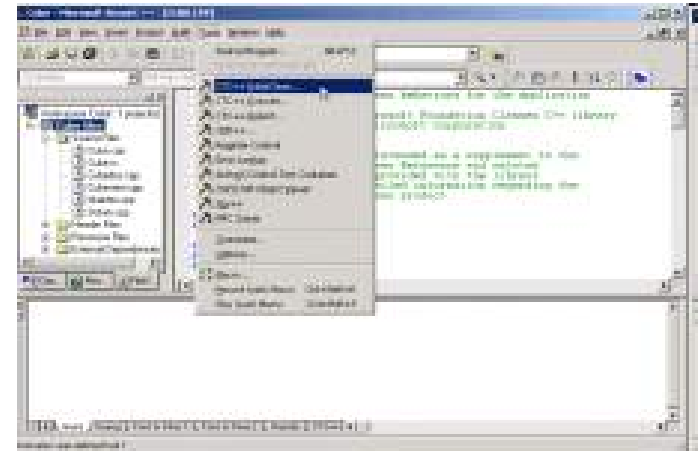
i.e. DO-178B - all levels: A, B, C - of the Federal Aviation Administration, FAA or for EN 61508



# Testwell CTC++ Test Coverage Analyser

Testwell CTC++ is very easy to use

- No need of code modifications
- Works with makefiles
- Works with command line
- GUI integration in several IDEs
  - Microsoft Visual Studio
  - Eclipse
  - WindRiver Tornado 2
  - IAR
  - Borland C++ 5.02
  - Fujitsu Softune



# Testwell CTC++ Test Coverage Analyser

## Testwell CTC++

### ideal for embedded targets

- very low instrumentation overhead
- performs code coverage in **all targets**
  - “host target add-on” is provided in source code
  - can be easily adapted to new targets
- works even with smallest targets and microcontrollers
- works with **all compilers** / cross-compilers



# Testwell CTC++ Test Coverage Analyser

„CTC++ Add-on for Java and C#“

extension of Testwell CTC++ for Java and C#


→ You only need one code coverage tool for C, C++, Java, C#, ...



# Testwell CTC++ Test Coverage Analyser

## Reports in Text, XML or HTML

- shows the untested code parts
- shows how many time each code part has been executed
- different coverage reports
  - Summary-Levels
  - Untested Code
  - Execution Profile Listing



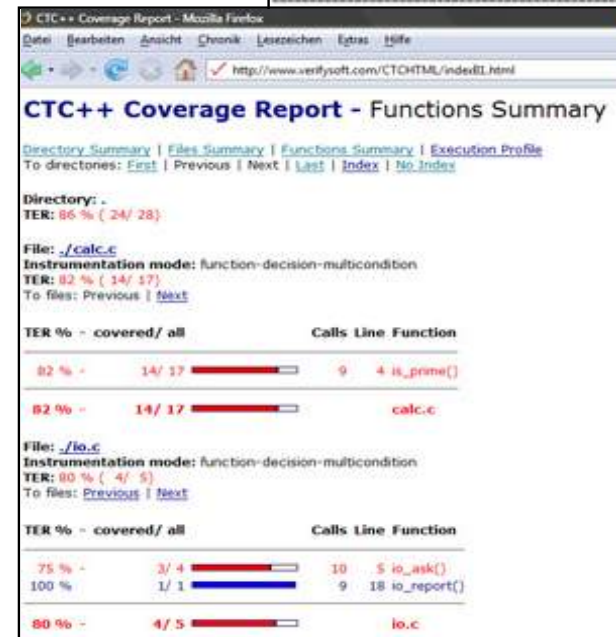
Example of CTC++ Untested Code Listing

```
CTC++ Test Coverage Analyser for C/C++, Version 4.5
UNTESTED CODE LISTING
Copyright (c) 1993-2008 Testwell Oy

Symbol file(s) used : MON.sym (Thu Feb 07 13:59:53 2008)
Data file(s) used   : MON.dat (Thu Feb 07 14:03:02 2008)
Listing produced at : Thu Feb 07 14:03:52 2008
Coverage view       : As Instrumented

MONITORED SOURCE FILE : calc.c
INSTRUMENTATION MODE  : function-decision-multicondition-timing

START/ END/ LINE DESCRIPTION
TRUE  FALSE
```



CTC++ Coverage Report - Functions Summary

Directory: .  
TER: 85 % ( 24 / 28 )

File: ./calc.c  
Instrumentation mode: function-decision-multicondition  
TER: 82 % ( 14 / 17 )

TER % - covered / all	Calls	Line	Function
82 % - 14 / 17	9	4	is_prime()
82 % - 14 / 17			calc.c

File: ./io.c  
Instrumentation mode: function-decision-multicondition  
TER: 80 % ( 4 / 5 )

TER % - covered / all	Calls	Line	Function
75 % - 3 / 4	10	5	io_ask()
100 % - 1 / 1	9	18	io_report()
80 % - 4 / 5			io.c

# Testwell CTC++ Test Coverage Analyser

## CTC++ Coverage Report - Directory Summary

[Directory Summary](#) | [Files Summary](#) | [Functions Summary](#) | [Execution Profile](#)

Symbol file(s) : C:\PROGRA~1\Testwell\CTC\examples\prime\mon.sym (Thu Feb 07 13:59:53 2008)  
: C:\Symbian\9.1\S60\_3rd\S60Ex\helloworldbasic\group\mon.sym (Tue Feb 13 14:07:43 2007)

(Click on header to sort)

TER % -	Covered	Not-covered	All	Directory
75 %	21	7	28	.
81 %	34	8	42	\Symbian\9.1\S60_3rd\S60Ex\helloworldbasic\src
56 % -	94	73	167	c:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\Samples\cube
<b>63 % -</b>	<b>149</b>	<b>88</b>	<b>237</b>	<b>OVERALL</b>

# Testwell CTC++ Test Coverage Analyser

## CTC++ Coverage Report - Files Summary

[Directory Summary](#) | [Files Summary](#) | [Functions Summary](#) | [Execution Profile](#)

Symbol file(s) : C:\PROGRA~1\Testwell\CTC\examples\prime\mon.sym (Thu Feb 07 13:59:53 2008)  
: C:\Symbian\9.1\S60\_3rd\S60Ex\helloworldbasic\group\mon.sym (Tue Feb 13 14:07:43 2007)  
Data file(s) : C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\Samples\cube\mon.sym (Thu Feb 07 15:34:43 2008)  
: C:\PROGRA~1\Testwell\CTC\examples\prime\mon.dat (Thu Feb 07 14:03:02 2008)  
: C:\Symbian\9.1\S60\_3rd\S60Ex\helloworldbasic\group\mon.dat (Tue Feb 13 14:09:47 2007)  
: C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\Samples\cube\mon.dat (Thu Feb 07 15:36:05 2008)  
Listing produced at : Thu Feb 07 15:36:22 2008  
Coverage view : As instrumented  
Input listing : STDIN  
Html generated at : Thu Feb 7 17:36:23 2008  
ctc2html v2.3 options: -t 75 -s C: -s C:\PROGRA~1\Testwell\CTC\examples\prime  
Threshold percent : 75 %

TER % - covered/ all	File
<b>Directory: .</b>	
65 % - 11/ 17	calc.c
80 % 4/ 5	io.c
100 % 6/ 6	prime.c
75 % 21/ 28	<b>DIRECTORY OVERALL (.)</b>

# Testwell CTC++ Test Coverage Analyser

## CTC++ Coverage Report - Functions Summary #1/3

[Directory Summary](#) | [Files Summary](#) | [Functions Summary](#) | [Execution Profile](#)

To directories: [First](#) | [Previous](#) | [Next](#) | [Last](#) | [Index](#) | [No Index](#)

Directory: .



TER: 75 % ( 21/ 28)

File: [calc.c](#)

Instrumentation mode: function-decision-multicondition-timing

TER: 65 % ( 11/ 17)

To files: [Previous](#) | [Next](#)

TER % - covered/ all	Calls	Line	Function
65 % - 11/ 17 	3	4	is_prime()
65 % - 11/ 17 			calc.c

File: [io.c](#)

Instrumentation mode: function-decision-multicondition-timing

TER: 80 % ( 4/ 5)

To files: [Previous](#) | [Next](#)

TER % - covered/ all	Calls	Line	Function
75 % 3/ 4 	4	5	io_ask()

# Testwell CTC++ Test Coverage Analyser

## CTC++ Coverage Report - Execution Profile #1/12

[Directory Summary](#) | [Files Summary](#) | [Functions Summary](#) | [Execution Profile](#)  
To files: [First](#) | [Previous](#) | [Next](#) | [Last](#) | [Index](#) | [No Index](#)

**File:** calc.c

**Instrumentation mode:** function-decision-multicondition-timing

**TER:** 65 % ( 11/ 17)

**Start/ End/**

**True False - [Line](#) Source** (found by option -s source-dir)

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) */
		4	int is_prime(unsigned val)
		5	{
		6	unsigned divisor;
		7	
1	2	8	if (val == 1    val == 2    val == 3)
0	-	8	T    _    _
1	-	8	F    I    _
0	-	8	F    F    I
	2	8	F    F    F
1		9	return 1;
1	1	10	if (val % 2 == 0)
1		11	return 0;
0	1	12	for (divisor = 2; divisor <= val / 2; divisor++)

# Testwell CTC++ Test Coverage Analyser

## Example of CTC++ Execution Profile Listing

```
*****
*          CTC++, Test Coverage Analyzer for C/C++, Version 6.5          *
*                                                                           *
*          EXECUTION PROFILE LISTING                                     *
*                                                                           *
*          Copyright (c) 1993-2008 Testwell Oy                          *
*****
Symbol file(s) used      : MON.sym (Thu Feb 07 13:59:53 2008)
Data file(s) used       : MON.dat (Thu Feb 07 14:03:02 2008)
Listing produced at    : Thu Feb 07 14:03:34 2008
Coverage view         : As instrumented
MONITORED SOURCE FILE  : calc.c
INSTRUMENTATION MODE   : function-decision-multicondition-timing

  START/      END/
  TRUE        FALSE  LINE DESCRIPTION
-----
   3          0      4 FUNCTION is_prime()
   1          2      8 if (val == 1 || val == 2 || val == 3)
   0          -      8 T || _ || _
   1          -      8 F || T || _
   0          -      8 F || F || T
                   2      8 F || F || F
   1          -      9 return 1
   1          1     10 if (val % 2 == 0)
   1          -     11 return 0
   0          1 -    12 for (;divisor < val / 2;)
   0          0 -    14 if (val % divisor == 0)
   0          -     15 return 0
   1          -     17 return 1
***TER 65 % ( 11/ 17) of FUNCTION is prime()
```

# Testwell CTC++ Test Coverage Analyser

## Example of CTC++ Untested Code Listing

```
*****
*          CTC++, Test Coverage Analyzer for C/C++, Version 6.5          *
*                                                                           *
*          UNTESTED CODE LISTING                                         *
*                                                                           *
*          Copyright (c) 1993-2008 Testwell Oy                           *
*****
Symbol file(s) used      : MON.sym (Thu Feb 07 13:59:53 2008)
Data file(s) used       : MON.dat (Thu Feb 07 14:03:02 2008)
Listing produced at     : Thu Feb 07 14:03:52 2008
Coverage view           : As instrumented
MONITORED SOURCE FILE   : calc.c
INSTRUMENTATION MODE    : function-decision-multicondition-timing
  START/      END/
  TRUE        FALSE  LINE DESCRIPTION
-----
      3         0      4 FUNCTION is_prime()
      1         2      8 if (val == 1 || val == 2 || val == 3)
      0         -      8   T || _ || _
      0         -      8   F || F || T
      0         1 -    12 for (;divisor < val / 2;)
      0         0 -    14   if (val % divisor == 0)
      0         -    15   return 0
-----
MONITORED SOURCE FILE   : io.c
INSTRUMENTATION MODE    : function-decision-multicondition-timing
  START/      END/
  TRUE        FALSE  LINE DESCRIPTION
-----
      4         0      5 FUNCTION io_ask()
```

# Testwell CTC++ Test Coverage Analyser

## Example of CTC++ Execution Time Listing

```
*****
*          CTC++, Test Coverage Analyzer for C/C++, Version 6.5          *
*                                                                           *
*                   EXECUTION TIME LISTING                               *
*                                                                           *
*                   Copyright (c) 1993-2008 Testwell Oy                 *
*****
Symbol file(s) used   : MON.sym (Thu Feb 07 13:59:53 2008)
Data file(s) used    : MON.dat (Thu Feb 07 14:03:02 2008)
Listing produced at  : Mon Feb 11 12:40:11 2008
Execution cost type  : Clock ticks
MONITORED SOURCE FILE : calc.c
INSTRUMENTATION MODE : function-decision-multiccondition-timing
EXECUTION          EXECUTION COST
COUNT            TOTAL          AVERAGE  LINE FUNCTION/TIMER
=====
          3          0             0.0       4 is_prime()
MONITORED SOURCE FILE : io.c
INSTRUMENTATION MODE : function-decision-multiccondition-timing
EXECUTION          EXECUTION COST
COUNT            TOTAL          AVERAGE  LINE FUNCTION/TIMER
=====
          4         16583          4145.8     5 io_ask()
          3          0             0.0       18 io_report()
MONITORED SOURCE FILE : prime.c
INSTRUMENTATION MODE : function-decision-multiccondition-timing
EXECUTION          EXECUTION COST
COUNT            TOTAL          AVERAGE  LINE FUNCTION/TIMER
=====
```

# Testwell CTC++ Test Coverage Analyser

## Example of CTC++ Coverage XML Report

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ctc_xml_report>
  <header_info>
    <ctcpost_version>6.5</ctcpost_version>
    <copyright>Copyright (c) 1993-2008 Testwell Oy</copyright>
    <license_notes>
    </license_notes>
    <symbolfiles>
      <symbolfile>
        <name>MON.sym</name>
        <modified>Thu Feb 07 13:59:53 2008</modified>
      </symbolfile>
    </symbolfiles>
    <datafiles>
      <datafile>
        <name>MON.dat</name>
        <modified>Thu Feb 07 14:03:02 2008</modified>
      </datafile>
    </datafiles>
    <chosen_source_files>
    </chosen_source_files>
    <report_generated>Thu Feb 07 14:04:16 2008</report_generated>
    <requested_coverage_view>As instrumented</requested_coverage_view>
    <ctcpost_options>-x xmlreport.txt</ctcpost_options>
    <execution_cost_function>clock</execution_cost_function>
    <execution_cost_type>Clock ticks</execution_cost_type>
    <execution_cost_scaling>1</execution_cost_scaling>
  </header_info>
  <ctcpost_notices>
  </ctcpost_notices>

```

# Testwell CTC++ Test Coverage Analyser

## Testwell CTC++

### ideal for embedded targets

- Very low instrumentation overhead
- Works with **any embedded targets**
- Works even with smallest targets and microcontrollers



# Testwell CTC++ Test Coverage Analyser

Works with **all compilers**

Vendors of "competitor tools" shows on their web sites lists of compilers they support:

- Embedded Linux user and kernel mode ARM, Itanium, MIPS, PowerPC, SuperH, x64, x86
- GCC targeting 68K/ColdFire, Alpha, ARM, Itanium, MIPS, PARISC, Power/PowerPC, SPARC, SuperH, x64, x86
- [Analog Devices VisualDSP++](#) Blackfin, SHARC, TigerSHARC
- [ARM RealView Developer Suite](#)
- [Freescale CodeWarrior](#) (command line compiler only, no IDE) for ColdFire, Power Architecture, StarCore
- [IAR Systems C/C++ ARM, SuperH](#)
- [Mentor Graphics Nucleus OS and EDGE](#)
- [Microchip PIC32 MCUs](#) (MIPS core), MPLAB C Compiler for PIC32 MCUs
- [NEC V850 CA850 C Compiler](#)
- [QNX Neutrino](#)
- [Qualcomm BREW SDK 3.1 Simulator](#)
- [Renesas C/C++ for SuperH](#)
- Svmhian OS emulator

Testwell CTC++ works with all of this compilers and all other !

**No unsupported compilers!**

# Testwell CTC++ Test Coverage Analyser

List of the compilers we have addons or settings prepared (as of 05/2011):

**Altium Tasking** (classic toolsets, VX-toolset toolsets, c166, cc166, ccm16c, cc51), **Borland/Inprise/Paradigm/Codegear** compilers, **Cosmic** (cx6805, cx6808, cx6812, cxs12x, cxxgate, cx6811, cx6816, cx332, cxst10, cxstm8, cxst7, bcc, bcc32, pcc, pcc32 Paradigm), **Freescale/Metrowerks** (mwccmcf, mwccppc, mwccmcore, mwcc56800, mwcc56800e, chc12, chc08), **Fujitsu/Softune** (fcc907s, fcc911s, fcc896s), **gcc and all gcc based cross-compilers**, **GHS/GreenHills/Multi** (ccv850, cxv850, ccmips, cxmips, ccarm, cxarm, cctthumb, cxthumb, ccppc, cxppc, gcc GreenHill), **HI-Tech PICC** (picc, picc18, picc32, dspicc), **HP** (HPUX CC, HP C++, aCC), **IAR** (iccm16c, icc430, icc8051, iccarm, iccavr, iccavr32, icccf, icchcs12, iccmaxq, iccdspic, iccpic18, icccr16c, icc78k, icc78k0r, iccv850, icch8, iccm32c, iccr32c, iccsam8, iccstm8, iccrx, iccm8k), **Intel compilers** (icc, ic86, ic96), **Java compilers** (javac, jikes, ecj, gcj, kaffe), **Keil** (c51, c166, c251, ca, cx51, cx2, tcc, armcc), **Matlab/Simulink** (lcc), **Metaware** (hccarm and others), **Microchip MPLAB C** (pic30-gcc, pic32-gcc), **Microsoft** (cl on host, both 32 and 64 bit, cl for Smartphones and PocketPC, csc C#, vjc J#), **Mono** (mcs, gmcs, smcs), **Motorola** (chc12, chc08), **Pathscale** pathcc/pathCC, **Sun** (WorkShop compilers, javac), **Symbian**, **TI Code Composer Studio** (cl2000, cl16x, cl470, cl55, cl500, cl430), **Texas Instruments Linux compilers** (cl2000, cl16x, cl470, cl55, cl500, cl430), **Trimedia** (tmcc), **VisualDSP+** (ccblkfn, cc21k, ccts), **Windriver** (ccarm, ccsimpc, g++simpc, g++arm, cchppa, ccsimso, ccsparc, cc68k, cc386, cc960, ccmips, ccppc)

Our customers have run Testwell CTC++ also with other tool-chaines.

**Adaptations to other compilers is easy and can be done by us or even by the customer himself.**

# References



and many more...

# Testwell CTC++ Test Coverage Analyser

**What can we do for you?**

Tool Evaluation

Testwell CTC++ Training



# Testwell CTC++ Test Coverage Analyser



**Further information:**

[www.verifysoft.com](http://www.verifysoft.com)