



Center for Systems And
Software Engineering

UCC v.2010.07

Release Notes

1. Introduction

This document provides the release notes for the UCC v.2010.07. Unified CodeCount (UCC) is a unified and enhanced version of the CodeCount toolset. It is a code counting and differencing tool that unifies the source counting capabilities of the previous CodeCount tools and source differencing capabilities of the DiffTool (which is now replaced by UCC). It allows the user to count, compare, and collect logical differentials between two versions of the source code of a software product. The differencing capabilities allow users to count the number of added/new, deleted, modified, and unmodified logical SLOC of the current version in comparison with the previous version. With the counting capabilities, users can generate the physical, logical SLOC counts, and other sizing information such as comment and keyword counts of the target program.

This release supports both counting and differencing for various languages including C/C++, C#, Java, SQL, Ada, Perl, ASP, ASP.NET, JSP, CSS, HTML, XML, JavaScript, VB, PHP, VBScript, NeXtMidas, XMidas, Python, Fortran, Bash, ColdFusion, and C Shell Script.

2. Compatibility Notes

UCC v.2010.07 is released in C++ source code, thus it allows users to compile and run on various platforms. This release has been tested on Windows using MS Visual Studio and on Unix/Linux using the g++ compiler.

The UCC v.2010.07 does not support Assembly, PL/1, COBOL, Pascal, and Jovial, although these may be included in future releases. For the need of counting of code in these languages, users may consider using the CodeCount Tools Release 2007.07 which do not provide the differencing capability but use the counting rules compatible to those of UCC v.2010.07.

3. Requirements

Minimum Software Requirements:

- **Compiler:** a compatible C++ compiler that can load common C++ libraries including IO and STL, such as MS Visual Studio 2008, g++, and Eclipse.
- **Operating systems:** any platforms that can compile and run a C++ application. The tool has been tested on Windows 9x/Me/XP/Vista, Unix, Linux, Solaris, and Mac OS X.

Minimum Hardware Requirements:

- **RAM:** 512 MB. Recommended: 1024 MB.
- **HDD:** 100 MB available. Recommended: 200 MB available.

4. Features

- 1) **Counting Capabilities.** UCC allows users to measure the size information of a baseline a source program by analyzing and producing the count for:
 - logical SLOC
 - physical SLOC
 - comment
 - executable, data declaration, compiler directive SLOC
 - keywords
 - *complexity measures*: mathematic functions, logarithms, calculations, assignments, cyclomatic complexity.
- 2) **Differencing Capabilities.** UCC allows users to compare and measure the differences between two baselines of source programs. These differences are measured in terms of the number of logical SLOC added/new, deleted, modified, and unmodified. These differencing results can be saved to either plain text .txt or .csv files. The default is .csv, but .txt can be specified by using the `-ascii` switch.
- 3) **Counting and Differencing Directories.** UCC allows users to count or compare source files by specifying the directories where the files are located. This capability eliminates difficulties in creating the file list that users may have encountered in the previous versions of the CodeCount toolset.
- 4) **Various Programming Languages Supported.** The counting and differencing capabilities accept the source code written in C/C++, C#, Java, SQL, Ada, Perl, ASP, ASP.NET, JSP, CSS, HTML, XML, JavaScript, VB, PHP, VbScript, Bash, C Shell Script, ColdFusion, Fortran, XMidas, NeXtMidas, PHP, and Python, . The tool detects the language of each file using its file extension (see Feature #10)
- 5) **Command Arguments.** The environment file containing user's settings (e.g., `c_env.dat` file) in the CodeCount tools is no longer used. Instead, the tool accepts user's settings via command arguments.
- 6) **Duplication.** For each baseline, two files are considered duplicates if they have same content or the difference is smaller than the threshold given through the command line switch `-tdup`. Two files may be identified as duplicates although they have different filenames.

For counting, duplicates in the input files are counted and their counting results are saved into a file named `Duplicates-<lang>_outfile.csv`. Duplicate file pairs are identified in a file named `DuplicatePairs.csv`, with matching pairs displayed in two columns. The complexity metrics of the duplicate files are reported in a file named `Duplicates-outfile_cplx.csv`.

For differencing, duplicates in each baseline are counted, and their counting results are saved into files named “`Duplicates-A-<LANG>-outfile.csv`” and “`Duplicates-B-<LANG>-outfile.csv`”, where `LANG` is the name of the programming language used. As such, one or more files are generated as a result of the duplication feature. Duplicate pairs are identified

in files Duplicates-A-DuplicatePairs.csv and Duplicates-B-DuplicatePairs.csv. The complexity metrics of the duplicate pairs are reported in a file named Duplicates-A-outfile_cplx.csv and Duplicates-B-outfile_cplx.csv. Note that duplicates are identified within baselines, and not across baselines.

Comments and blank lines are not considered during duplication processing.

- 7) **Matching.** Two files are matched if they have the same filename regardless of which directories they belong to. Two files that have the same filename are matched if they have the least uncommon characters in their directory names. This feature allows users to handle to the situation where files are moved from one directory to another or the directory structure is changed. The remaining files are matched according to an algorithm that makes the most likely match.
- 8) **Complexity Count.** UCC produces complexity counts for all source code files. The complexity counts include the number of math, trig, logarithm functions, calculations, conditionals, logicals, preprocessors, assignments, pointers, and cyclomatic complexity. When counting, the complexity results are saved to the file “outfile_cplx.csv”, and when differencing the results are saved to the files “Baseline-A-outfile_cplx.csv” and “Baseline-B-outfile_cplx.csv”.
- 9) Under Unix/Linux when using the `-dir` option, any wildcards must be enclosed within quotes. Otherwise, the wildcards will be expanded on the command line and erroneous results will be produced. For example: `ucc -d -dir baseA baseB *.cpp` should be written as `ucc -d -dir baseA baseB “*.cpp”`.
- 10) **File Extensions.** The tool determines the language used in a source file using file extension. This release supports the following languages and file extensions:

Languages	File Extensions
Ada	.ada, .a, .adb, .ads
ASP, ASP.NET	.asp, .aspx
Bash	.sh, .ksh
C Shell Script	.csh, .tcsh
C#	.cs
C/C++	.cpp, .c, .h, .hpp, .cc, .hh
ColdFusion	*.cfm, .cfml, .cfc
CSS	.css
Fortran	.f, .for, .f77, .f90, .f95, .f03, .hpf
HTML	.htm, .html, .shtml, .stm, .sht, .oth, .xhtml
Java	.java
JavaScript	.js
JSP	.jsp
NeXtMidas	.mm
Perl	.pl, .pm
PhP	.php

Python	.py
SQL	.sql
VB	.vb, .frm, .mod, .cls, .bas
VbScript	.vbs
XMidas	.txt
XML	.xml

5. Changes and Upgrades

This section describes changes and upgrades to the tool since the Release 2009.10.

- 1) Newly supported languages include NeXtMidas, XMidas, Python, Fortran, Bash, ColdFusion, and C Shell Script.
- 2) Supported the .csv format for all output files. This is the default format.
- 3) Improved the accuracy of the counting and differencing functions for PHP and SQL.
- 4) The counts and complexity of duplicated files are printed separately for Baseline A and Baseline B.
- 5) The complexity report will now include a cyclomatic complexity count, which measures the number of individual paths through the code, and Nested Loops count, which indicates the depth of nested loops and the number of loops at each depth.
- 6) The flag *-nocomplex* can be added to the command line to inhibit complexity processing and reporting.
- 7) Improved the accuracy of the file matching and duplicate detecting function.
- 8) An error log file is generated to report all errors occurred. The file is named *error_log_date_time.txt*. For example, a file named *error_log_06072010_111806.txt* would be created when UCC is run on June 7, 2010, at 11:18:06 am.
- 9) Fixed the compiling error caused by compiler incompatibility issues on Ubuntu 10.04 LTS.
- 10) The Release 2009.10 generates unexpected differencing results if the threshold parameter *-t* is set to zero. This defect was fixed in this release.
- 11) If embedded comments are added into logical statements in *Baseline B*, the Release 2009.10 incorrectly counts them as modified. This defect was fixed in this release by considering that only adding embedded comments does not change the differencing results.

As a result of defect fixes and upgrades in the Release 2009.10 and this release, the logical SLOC count produced by UCC is generally **1% ± 0.2%** less than the logical SLOC count produced by the **CodeCount** toolset that was published prior to these releases. This value was determined by performing an experiment on open source applications that range from 40 KSLOC to 60 KSLOC.

6. Known Issues and Limitations

No	Issues
1	For JavaScript code, the tool does not count the statement that is not terminated by a semicolon.
2	<p>When the command option <code>-dir</code> is provided along with a file spec and there is no file found for some of the types specified by the file spec, the tool shows a file-not-found message. This does not affect the final result.</p> <p>For example, if the following command is run</p> <pre>UCC -dir .\proj1 *.cpp *.h</pre> <p>and if the directory <code>proj1</code> contains some <code>.cpp</code> files but it does not have any <code>.h</code> files, the tool shows a file-not-found message.</p>
3	The tool only detects and handles C# and VB as code-behind languages for the ASP.NET.
4	The tool would take several hours to count and compare the source code of several million lines of code in thousands of files. This slow performance seems to occur more often in the source code that has large files. To alleviate this issue, it is recommended that the users compile the tool in a performance-optimized mode and divide the source code into multiple packages and count them individually.