XSL CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
7/27/2016	1.0	Original Release	Derek Lengenfelder

Table of Contents

No.	Contents	Page No.
1.0	Definitions	4
1.1	SLOC	4
1.2	Physical SLOC	4
1.3	Logical SLOC	4
1.4	Data declaration line	4
1.5	Compiler directive	4
1.6	Blank line	4
1.7	Comment line	4
1.8	Executable line of code	4
2.0	Checklist for source statement counts	5
3.0	Examples of logical SLOC counting	6
3.1	Executable Lines	6
3.2	Declaration lines	6
3.3	Element Lines	7
4.0	Complexity	8
5.0	Cyclomatic Complexity	9

1. Definitions

1.1. **SLOC –** Source Lines of Code is a unit used to measure the size of software program. SLOC counts the program source code based on a certain set of rules. SLOC is a key input for estimating project effort and is also used to calculate productivity and other measurements.

1.2. **Physical SLOC –** One physical SLOC is corresponding to one line starting with the first character and ending by a carriage return or an end-of-file marker of the same line, and which excludes the blank and comment line.

1.3. **Logical SLOC –** Lines of code intended to measure "statements", which normally terminate by a semicolon (C/C++, Java, C#) or a carriage return (VB, Assembly), etc. Logical SLOC are not sensitive to format and style conventions, but they are language-dependent.

1.4. **Data declaration line or data line –** A line that contains declaration of data and used by an assembler or compiler to interpret other elements of the program.

1.5. **Compiler Directives –** A statement that tells the compiler how to compile a program, but not what to compile.

1.6. **Blank Line –** A physical line of code, which contains any number of white space characters (spaces, tabs, form feed, carriage return, line feed, or their derivatives).

1.7. **Comment Line –** A comment is defined as a string of zero or more characters that follow language-specific comment delimiter.

Ada comment delimiters are "<!-- … -->". A whole comment line may span one line and does not contain any compliable source code. An embedded comment can co-exist with compliable source code on the same physical line. Banners and empty comments are treated as types of comments.

1.8. **Executable Line of code –** A line that contains software instruction executed during runtime and on which a breakpoint can be set in a debugging tool. An instruction can be stated in a simple or compound form.

- An executable line of code may contain the following program control statements:
  - XSL does not have any executable lines of code.
- An executable line of code may not contain the following statements:
  - Data declaration (data) lines
  - Whole line comments, including empty comments and banners
  - Blank lines

2. Checklist for source statement counts

PHYSICAL SLOC COUNTING RULES			
MEASUREMENT UNIT	ORDER OF PRECEDENCE	PHYSICAL SLOC	COMMENTS
Executable Lines	1	One per line	Defined in 1.8
Non-executable Lines			
Declaration (Data) Lines	2	One per line	Defined in 1.4
Compiler Directives	3	One per line	Defined in 1.5
Comments			Defined in 1.7
On their own lines	4	Not Included	
Embedded	5	Not Included	
Banners	6	Not included	
Empty comments	7	Not included	
Blank lines	6	Not Included	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	<name (s attribute)*>(data)*</name>	1	Count once	The contents of the tag and the end tag itself are not counted
R02	'<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'	2	Count once	

3. Examples

EXECUTABLE LINES

DECLARATION OR DATA LINES

DDL1 – processing instruction

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'	<?xml version="1.0"?>	1

DDL1 – document type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<!DOCTYPE' S Name (S ExternalID)? S? ('[' (markupdecl | DeclSep)* ']' S?)? '>'	<?xml version="1.0"?> <!DOCTYPE greeting SYSTEM "hello.dtd">	1 1

DDL1 – element type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<!ELEMENT' S Name S contentspec S? '>'	<!ELEMENT br EMPTY> <!ELEMENT p (#PCDATA|emph)* > <!ELEMENT %name.para; %content.para; > <!ELEMENT container ANY>	1 1 1 1

DDL1 – attribute type

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<!ATTLIST' S Name AttDef* S? '>'	<!ATTLIST termdef id ID #REQUIRED name CDATA #IMPLIED> <!ATTLIST form method CDATA #FIXED "POST">	1 1

DDL1 – entity

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<!ENTITY' S Name S EntityDef S? '>'	<!ENTITY % YN '"Yes"' > <!ENTITY WhatHeSaid "He said %YN;" >	1 1

ELEMENT LINES

EL1 – empty element

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<tag_name></tag_name>	<color></color> <shape> </shape>	1 0 1 0

EL2 – comment delimiter

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<!--this is a comment -->	<!--To check the lines--> <?xml version="1.0" encoding="ISO8859-1" ?> <message to="you@yourAddress.com" from="me@myAddress.com" subject="XML Is Really Cool"> <!-- This is a comment --> <text> How many ways is XML cool? Let me count the ways... </text> </message>	0 1 1 0 1 0 0

EL3 – string delimiter

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Escape Character – Both Entity and character references may be used to escape the delimiters. < > & ' "	<?xml version="1.0" encoding="utf-8"?> <string xmlns="http....."> < DataSet> <Order> <Customer >439 </Customer> </Order> </DataSet></string>	1 1 0

EL4 – tags with attributes

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name (s attribute)*>(data)*</name>;	<force units="Newtons">100</force>	1

EL5 – processing instruction

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
'<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'	<?xml version="1.0" encoding="UTF-8"?>	1

4. Complexity

Complexity measures the occurrences of different keywords in code baseline. Below table identifies the categories and their respective keywords that are counted as part of the complexity metrics.

MATH FUNCTIONS	TRIG	LOG	CALCULATIONS	CONDITIONALS	LOGIC	PRE-PROCESSOR	ASSIGNMENT
EXP	ARCCOS	LOG	+	IF	=	PRAGMA	:=
RANDOM	ARCCOSH		-	ELSIF	<		
SQRT	ARCCOT		*	ELSE	>		
	ARCCOTH		/	LOOP	/=		
	ARCSIN		MOD	WHEN	>=		
	ARCSINH		REM	CASE	<=		
	ARCTAN			FOR	&		
	ARCTANH			WHILE	AND		
	COS				OR		
	COTH				XOR		
	SIN				IN		
	SINH				NOT IN		
	TAN						
	TANH						

5. Cyclomatic Complexity

Cyclomatic complexity measures the number of linearly independent paths through a program. It is measured for each function, procedure, or method according to each specific program language. This metric indicates the risk of program complexity and also determines the number of independent test required to verify program coverage.

The cyclomatic complexity is computed by counting the number of decisions plus one for the linear path. Decisions are determined by the number of conditional statements in a function. A function without any decisions would have a cyclomatic complexity of one. Each decision such as an if condition or a for loop adds one to the cyclomatic complexity.

The cyclomatic complexity metric v(G) was defined by Thomas McCabe. Several variations are commonly used but are not included in the UCC. The modified cyclomatic complexity counts select blocks as a single decision rather than counting each case. The strict or extended cyclomatic complexity includes boolean operators within conditional statements as additional decisions.

XSL does not have cyclomatic complexity. Therefore, its cyclomatic complexity is always listed as zero (0).