Visual Basic CodeCount™

Counting Standard

University of Southern California

Center for Systems and Software Engineering

December , 2016

Revision Sheet

Date	Version	Revision Description	Author
7/7/2016	1.0	Original Release	Matthew Swartz
7/12/2016	1.1	Updated complexity table	Matthew Swartz

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	7
3.1	Executable Lines	7
3.1.1	Selection Statements	7
3.1.2	Iteration Statements	8
3.1.3	Jump Statements	8
3.1.4	Expression Statements	9
3.2	Declaration lines	9
3.3	Compiler directives	9
4.0	Complexity	9
5.0	Cyclomatic Complexity	10

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.

The following table lists the Visual Basic keywords that denote data declaration lines:

User-defined	ValueType	DateTime	Uinteger	Boolean
Decimal	Integer	Double	Object	Single
String	UInt16	UInt32	UInt64	Ushort
Int16	Int32	Int64	Sbyte	Short
ULong	Byte	Char	Date	Long

Table 1 Data Declaration Types

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

The following table lists the Visual Basic keywords that denote data declaration lines:

#ExternalSource	#ElseIf	#Region	#Const
#Else	#End	#If	

Table 2 Compiler Directives

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.

Visual Basic comment delimiter is “.”. 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:
    - Selection statements (if, ? operator, switch)
    - Iteration statements (for, while, do-while)
    - Empty statements (one or more “;”)
    - Jump statements (return, goto, break, continue, exit function)
    - Expression statements (function calls, assignment statements, operations, etc.)
    - Block statements
- An executable line of code may not contain the following statements:
    - Compiler directives
    - 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	8	Not Included	Defined in 1.6

LOGICAL SLOC COUNTING RULES				
NO.	STRUCTURE	ORDER OF PRECEDENCE	LOGICAL SLOC RULES	COMMENTS
R01	If/Elseif condition Then statement Else statement Endif Select var Case cond:statement . Case Else :statement End Select	1	Count once	
R02	do while (...) statement loop for (...) statement next do...statements until(...) while(...) statements wend	2	Count once	
R03	Block delimiters Private Sub End Sub	3	Count once per private pair of Private Sub and End Sub	
R04	Compiler directive	4	Count once per directive	

3. Examples

EXECUTABLE LINES

SELECTION Statements

ESS1 – if-ElseIf-Else and nested if statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	If (total = firstnum + secondnum	1
	And Val(sum.Text) <> 0) Then	0
	correct.Visible = True	1
If conditionThen	wrong.Visible = False	1
<statement>	End If	0
End If		
	If (total = firstnum + secondnum	1
If condition Then	And Val(sum.Text) <> 0) Then	0
<statement>	correct.Visible = True	1
Else	wrong.Visible = False	1
<statement>	Else	0
End If	correct.Visible = False	1
	wrong.Visible = True	1
If condition1 Then	End If	0
<statement>		
Else If condition2 Then	If (total = firstnum + secondnum	1
<statement>	And Val(sum.Text) <> 0) Then	0
Else	correct.Visible = True	1
<statement>	wrong.Visible = False	1
End If	Else If (total = firstnum –	1
	secondnum) Then	0
NOTE: complexity is not	correct.Visible = False	1
considered, i.e. multiple “And” or	wrong.Visible = True	1
“Or” as part of the expression.	Else	0
	correct.Visible = False	1
	wrong.Visible = True	1
	End If	0

ESS2 – select case statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	Select Case Err.Num	1
Select Case	Case 53 'File not found	0
Case <constant 1> :	answer=MsgBox("File not	0
<statements>	found. Try again?",	0
Case Else	_ vbYesNo)	1
<statements>	Case 76 'Path not found	0
End Select	answer=MsgBox("Path not	0
	found. Try again?",	0

	_vbYesNo)	1
	Case Else 'unknown error	0
	MsgBox "Unknown error.	0
	Quitting now."	0
	'SHOULD LOG ERROR!	1
	Unload Me	1
	End Select	0

ESS3 – error handler statements

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	Private Sub	1
	CodeWithErrorHandler()	0
OnError Goto	On Error GoTo ErrHandler	1
	'...Procedure code ...	0
	'...	0

ITERATION Statements

EIS1 – For Loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
For num = 1 To 10	For num = 1 To 10	1
STATEMENTS	studentName(num)= 999	1
Next	Next	0

EIS2 – While Loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
	While Not IsEmpty(ActiveCell)	1
While condition	MsgBox	1
Statements	ActiveCell.Value	1
Wend	ActiveCell.Offset(1, 0).Select	1
	Wend	0

EIS3 – Do-until loop

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Do Statements Until condition	Do MsgBox ActiveCell.Value ActiveCell.Offset(1, 0).Select Until IsEmpty(ActiveCell)	0 1 1 1

JUMP Statements

(are counted as they invoke action-pass to the next statement)

EJS1 – exit statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Exit Sub/Function	Exit Sub	1

EXPRESSION Statements

EES1 – function and procedure call

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<function_name> (<parameters>)	read_file (name)	1

EES2 – assignment statement

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
<name> = <value>	x = y	1

Declaration or Data Lines

DDL1 – subroutine/function declaration, variable declaration

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
Private Sub Name(var_list) Statements End Sub Dim <var> As <type>	Private Sub Start_Click() Form1.Cls addName End Sub Dim var As String	1 1 1 0 1

Compiler Directives

CDL1 – directive types

GENERAL EXAMPLE	SPECIFIC EXAMPLE	SLOC COUNT
#Const Directive	#Const Directive	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
IEEERemainder	Atan2	Log10	<<=	Elseif	AndAlso	#ExternalSource	=
Truncate	Acos	Log	>>=	Select	Andalso	#ElseIf	
Ceiling	Asin		&=	While	Isfalse	#Region	

BigMul	Atan		*=	Case	Orelse	#Const	
DivRem	Cosh		/=	Else	Istrue	#Else	
Floor	Sinh		\=	For	And	#End	
Round	Tanh		^=	Do	Not	#If	
Sign	Cos		+=	If	Xor		
Sqrt	Sin		<<		<>		
Abs	Tan		-=		>=		
Exp			>>		<=		
Max			-		Or		
Min			&		>		
Pow			*		<		
			/				
			\				
			^				
			+				
			=				

5. Cyclomatic Complexity

Cyclomatic complexity measures the number of linearly independent paths through a program. It is measured for eachfunction, 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 UCC. The modified cyclomatic complexity counts select blocks as a single decision rather than counting each case. The strickt or extended cyclomatic complexity includes Boolean operators within conditional statements as additional decisions. Please see: cyclomatic_complexity_standard.pdf which has more details of different ways where soecific cyclomatic complexity metrics are found and presented.

Cyclomatic Complexity	**Risk Evaluation**
1-10	A simple program, without much risk
11-20	More complex, moderate risk
21-50	Complex, high risk program
>50	Untestable program, very high risk

For Visual Basic, the following table lists the conditional keywords used to compute cyclomatic complexity.

Scala statement	**CC count**	**Rationale**
If, iif	+1	if adds a decision
ElseIf	+1	else if adds a decision

Else	0	decision is at the if statement
Select case	+1 per case	each case adds a decision – not the select
Select Else	0	Decision is at the case statements
For	+1	For adds a decision at loop start
While	+1	While adds a decision at loop start
Until	+1	Until adds a decision at loop start
Do	0	Decision is at while/until statement – no deicision at unconditional loop
Catch	+1	Catch adds a decision
When	+1	When adds a decision