Objective C CodeCount™ Counting Standard

University of Southern California

Center for Systems and Software Engineering

September ,2010

Revision Sheet

Date	Version	Revision Description	Author
9/24/10	1.0	Original Release	Group Objective C
12/1/10	1.1	Final Release	Group Objective C

~~1.~~ checklist for source statement counts

PHYSICAL AND LOGICAL SLOC COUNTING RULES

Measurement Unit	Order of Precedence	Physical SLOC	Logical SLOC	Comments
Executable lines	1	One per line	See table below	Defined in 2.9
Non-executable lines				
Declaration (Data) lines	2	One per line	See table below	Defined in 2.4
Compiler directives	3	One per line	See table below	Defined in 2.5
Comments				Defined in 2.8
On their own lines	4	Not included (NI)	NI	
Embedded	5	NI	NI	
Banners	6	NI	NI	
Empty comments	7	NI	NI	
Blank lines	8	NI	NI	Defined in 2.7

Table 1 Physical and Logical SLOC Counting Counts

LOGICAL SLOC COUNTING RULES

No.	Structure	Order of Precedence	Logical SLOC Rules	Comments
R01	"*for*", "*while*" or "*if*" statement	1	Count once.	"*while*" is an independent statement.
R02	*do* {…} *while* (…); statement	2	Count once.	Braces *{…}* and semicolon *;* used with this statement are not counted.
R03	Statements ending by a semicolon	3	Count once per statement, including empty statement.	Semicolons within "*for*" statement are not counted. Semicolons used with R01 and R02 are not counted.
R04	Block delimiters, braces {…}	4	Count once per pair of braces *{..}*, except where a closing brace is followed by a	Braces used with R01 and R02 are not counted. Function definition is

			semicolon, i.e. *};* or an opening brace comes after a keyword "*else*".	counted once since it is followed by *{…}*.
R05	Compiler directive	5	Count once per directive.	

Table 2 Logical SLOC Counting Rules

2. definitions

2.1 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.

2.2 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.

2.3 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.

2.4 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 Objective C keywords that denote data declaration lines:

Simple Data Types	Compound and User Defined Data Types	Access Specifiers	Type Qualifiers
	@class	@private	const
char	struct	@protected	volatile
double	union	@public	restrict
float	enum		
int	typedef	**Storage Class Specifiers**	**Miscellaneous**
long		auto	asm
short		extern	explicit
signed	@selector(method_name)	mutable	inline
unsigned	@protocol(protocol_name)	register	namespace
long long	@encode(type_spec	static	using
IOD	@synchronized()	_Complex	

SEL		_Bool	
IMP	@interface	_Imaginary	
STR	@implementation	void	
BOOL	@end	omitted	
id			

Table 3 Data Declaration Types

NOTE: See Section 3 of this document for examples of data declaration lines.

2.5 Compiler directive - A statement that tells the compiler how to compile a program, but not what to compile.

A list of common objective-C directives is presented in the table below:

@interface	@private	@private	@public	@public	@public	@selector	@synchronized	@
@catch	@finally	@throw	@”chars”	@class c1, c2,...	@defs	@dynamic names	@encode	z @

#define	#ifndef	#include	#import
#undef	#else	#line	
#if	#elif	#pragma	
#ifdef	#endif	#error	

Table 4 Compiler Directives

NOTE: See Section 3 of this document for examples of compile directive lines.

Executive Directives

@throw	@catch	@finally	@try

2.7 Preprocessor Directive- Preprocessor directives are special notations. Some keywords of Objective-C are not reserved outside. These are

in	Out	inout	oneway	byref	bycopy

Keyword for memory management in Objective-C

These are looking as keywords but infact these are methods of root class NSObject.

alloc	retain	release	nsautorelease

Some other keywords:

1. bool is a keyword used in objective-C but its value is YES or NO. In C and C++ it has value either TRUE or FALSE.
2. 'super' and 'self' can be treated as keywords but self is a hidden

parameter to each method and super gives the instructions to the compiler that how to use self differently.

2.6 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).

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

Objective C comment delimiters are "//" and "/*". A whole comment line may span one or more lines and does not contain any compilable source code. An embedded comment can co-exist with compilable source code on the same physical line. Banners and empty comments are treated as types of comments.

2.8 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

NOTE: See Section 3 of this document for examples of control 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

Examples Of LOGICAL SLOC Counting

Executable lines				
SELECTION STATEMENTS				
ID	Statement Description	General Form	Specific Example	SLOC Count

ESS1	if, else if, else and nested if statements	if (<boolean expression>)
		  <statements>;
		
		if (<boolean expression>)
		<statement>;
		else <statement>;
		
		if (<boolean expression>)
		  <statements>;
		else if (<boolean expression>)
		  <statements>;.
		.
		.
		else <statements>;
		
		if (<boolean expression>)
		{
		  <statements>;
		}
		else
		{
		  <statements>;
		}
		
		NOTE: complexity is not considered, i.e. multiple "&&" or "||" as part of the expression.
			if (x != 0)
			  NSLog(@"non-zero");
			
			if (x > 0)
			NSLog(@"positive");
			else
			NSLog(@"negative");
			
			if (x == 0)
			  NSLog(@"zero");
			else if (x > 0)
			  NSLog(@"positive");
			else
			  NSLog(@"negative");
			
			if ((x != 0) && (x > 0))
			  NSLog(@"%i", x);
			
			if (x != 0)
			{
			  NSLog(@"non-zero");
			}
			else
			{
			  NSLog(@"zero");
			}
				1
				1
				
				2
				1
				
				1
				1
				1
				1
				0
				1
				
				1
				1
				
				1
				0
				1
				0
				0
				0
				1
				0

ESS2	? operator	Exp1?Exp2:Exp3	x > 0 ? NSLog(@"+") : NSLog(@"-");	1

ESS3	switch and nested switch statements	switch (<expression>)
		{
		  case <constant 1> :
		  <statements>;
		  break;
		  case <constant 2> :
		  <statements>;
		  break;
		  case <constant 3> :
		  <statements>;
		  break;
		  default
			switch (number)
			{
			  case 1:
			  case 11:
			  foo1();
			  break;
			  case 2:
			  foo2();
			  break;
			  case 3:
			  foo3();
			  break;
			  default
			  NSLog(@"invalid case");
				1
				0
				0
				0
				1
				1
				0
				1
				1
				0
				1
				1
				0
				1
				0

		<statements>; }	}	

ESS4	@try-@catch	@try
		{
		// code that could
		@throw
		// an NSexception
		}
		@catch (NSexception-declaration)
		{
		// code that executes when
		// exception-declaration is thrown
		// in the try block
		}
			@try
			{
			NSLog(@ "Calling func \n");
			MyFunc();
			}
			catch (NSException e)
			{
			NSLog(@ " "Error: "e;
			)}
				1
				0
				1
				1
				0
				1
				0
				1
				0

ITERATIONS STATEMENTS				
ID	Statement Description	General Form	Specific Example	SLOC Count

EIS1	for	for (initialization; condition; increment)
		statement;
		
		NOTE: "for" statement counts as one, no matter how many optional expressions it contains, i.e.
		for (i = 0, j = 0; i < 5, j < 10; i++, ,j++)
			for (i = 0; i < 10; i++)
			NSLog(@"%i", i);
			
			for (i = 0; i < 10; i++)
			{
			NSLog(@"%i", i);
			}
				1
				1
				
				1
				0
				1
				0

EIS2	empty statements (could be used for time delays)	for (i = 0; i < SOME_VALUE; i++) ;	for (i = 0; i < 10; i++) ;	2

EIS3	while	while (<boolean expression>) <statement>;	while (i < 10)
			{
			NSLog(@"%i", i);
			i++;
			}
				1
				0
				1
				1
				0

EIS4	do-while	Do
		{
		<statements>;
			Do
			{
			ch = getchar();
				0
				0
				1

		} while (<boolean expression>);	} while (ch != '\n');	1

JUMP STATEMENTS (are counted as they invoke action – pass to the next statement)				
ID	Statement Description	General Form	Specific Example	SLOC Count
EJS1	return	return *expression*;	if (i == 0) return;	2

EJS2	goto, label	goto *label*;
		.
		.
		label:
			loop1:
			x++;
			if (x < y) goto loop1;
				0
				1
				2

EJS3	break	break;	if (i > 10) break;	2
EJS4	exit function	void exit (int return_code);	if (x < 0) exit (1);	2

EJS5	continue	continue;	while (!done)
			{
			ch = getchar();
			if (char == '\n')
			{
			done = true;
			continue;
			}
			}
				1
				0
				1
				1
				0
				1
				1
				0
				0

EXPRESSION STATEMENTS				
ID	Statement Description	General Form	Specific Example	SLOC Count
EES1	function call	[<function_name> <parameters>];	[read_file name];	1

EES2	assignment statement	<name> = <value>;	x = y;
			char name[6] = "file1";
			a = 1; b = 2; c = 3;
				1
				1
				3

EES3	empty statement (is counted as it is considered	one or more ";" in succession	;	1 per each

	to be a placeholder for something to call attention)			
BLOCK STATEMENTS				
ID	Statement Description	General Form	Specific Example	SLOC Count

EBS1	block = related statements treated as a unit	/* start of block */
		{
		<definitions>
		<statement>
		}
		/* end of block */
			/* start of block */
			{
			i = 0;
			NSLog(@"%i", i);
			}
			/* end of block */
				0
				0
				1
				1
				1
				0

declaration (data) lines				
ID	Statement Description	General Form	Specific Example	SLOC Count
DDL1	function prototype, variable declaration,	<type> <name> (< parameter_list>); <type> <name>;	void foo (int param); double amount, price;	1 1 1

	struct declaration	struct <name>
		{
		<type> <name>;
		<type> <name>;
		}
			int index;
			
			struct S
			{
			int x;
			int y;
			};
				0
				0
				1
				1
				1
				
				0

		struct { <type> <name>; <type> <name>; } <name>;	 struct { int x; int y; } S;	0 0 1 1 2 1
	typedef	typedef <type> <name>; typedef struct <name> { <type> <name>; … } <struct_name>; <type> <name> (<	typedef int MY_INT; typedef struct S { int i; char ch; } <struct_name>;	0 0 1 1 2

	interface
	
	implementation
		parameter_list>)
		{
		…
		}
		
		@interface <name>:
		<super>{
		<type> <name>;
		<type> <name>;
		}
		
		@implementation <name>
		-<type> name{
		...
		};
		}
			void main()
			{
			NSLog(@“hello”
			);
			}
			
			@interface
			Fraction: NSObject
			{
			int numerator;
			int denominator;
			}
			
			@implementation
			Fraction
			-(void) print {
			NSLog(@ "hello”
			);
			}
				0
				0
				1
				1

compiler directives				
ID	Statement Description	General Form	Specific Example	SLOC Count
CDL1	directive types	#define <name> <value> #import <library_name>	#define MAX_SIZE 100 #import <NSString>	1 1