SLOC Metric in RNG Schema Documents

Sotonwa K. A[1], Olabiyisi S. O[2], Omidiora E. O[3], Oyeleye C. A[4]
[1]*Department of Computer Science and Information Technology, Bells University of Technology, Ota, Ogun State, Nigeria*
[2,3,4]*Department of Computer Science and Engineering, Ladoke Ajintola University of Technology, Ogbomoso, Oyo State, Nigeria*

Abstract: **Source Lines of Code (SLOC) is a quantitative measurement in computer programming for files that contain codes from a computer programming language, in text form. SLOC is used to predict the amount of effort that will be required to develop a program. Different types of SLOC were considered for 40 different schema files acquired in Web Service Description Language (WSDL) and implemented in Relax-NG eXtensible Schema Language (XML) in order to estimate schemas productivity and maintainability.**

Keywords: **TSLOC. BSLOC, CSLOC, NCSLOC, ESLOC**

I. INTRODUCTION

SLOC is generally considered as the count of the lines in the source code of the software. Usually, it only considers the executable sentence [1]. eXtensible Markup Language (XML) is a mark-up language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable XML[2][3] serves very well as a ubiquitous, platform-independent data representation and transport format which is accepted in diverse fields.. The data representations are made by designing the schema, which can be written by a series of XML schema languages.

Today, the literature in this regard provides a variety of XML schema languages; amongst which are Document Type Definition (DTD) [4] World Wide Web Consortium XML Schema (WXS)/XML Schema Definition (XSD) [5] [6] [7] [8]. Schematron and Regular [9] [10] Language for Next Generation (Relax NG). SLOC count the line of schema files implemented in RNG and it is independent of what program language used. The SLOC evaluates the complexity of the software via the physical length and it is based upon two rules: the relationship between the count of code lines and the bug density and the independence between the bug density and the program language[1][11]. The XML documents used in this paper are acquired from WSDL and implemented in RNG[12][13][14][15[[16[17][18].

II. PROBLEM STATEMENT

Metrics are indispensable from several aspects such as measuring the comprehension of a code, testability of the software, maintainability and development processes. Various popular metrics for structured programs are under criticized. These criticisms are mainly based on lacking desirable measurement properties, being too labour intensive to collect and confine to the features of structured programs. Most of the available metrics cover only certain features of a language. For example SLOC metric considered size of a program and neglect all other factors that affect the complexity of software, there are various SLOC counting applications, each producing different logical and physical SLOC count results. This peculiarity demonstrates the deficiencies in current methods and techniques, and suggests better tools are required to satisfy the needs of the software industry.

III. RELATED WORK

Harrison *et. al.* [19] measured Line of Codes (LOC) metric, a code written in one programming language may be much more effective than another therefore two programs that give the same functionalities written in two different languages may have very different LOC values because of this it neglects all other factors that affect the complexity of software.

Vu Nguyen *et. al.*[20] presented a set of counting standards that defines what and how to count SLOC with the USC Code Count toolset, a. It was suggested that this problem can be alleviated by the use of a reasonable and unambiguous counting standard guide and with the support of a configurable counting tool.

Kaushal Bhatt *et. al.* [21] evaluate some drawbacks in SLOC metrics that affect the quality of software because SLOC metric output is used as a input in other Software Estimation methods Like COCOMO Model. Armit and Kumar [22] formulated a metric that counts the number of lines of codes but neglected the intelligence content, layout and other factors that affect the complexity of the code. Sotonwa *et. al*. [23] proposed a metric that count the number of line of codes, commented lines and non commented lines for various object oriented programming languages.

IV. METHDOLOGY

A. The SLOC Metric

The metric is applied on 40 different schema files acquired from WSDL and implemented in Relax NG. The RNG codes were different from each other in their architecture and different types of SLOCs are considered for each schema. The following approaches were adopted:

(1) Total Sources Lines of Code (TSLOC): It is obvious from its name that it counts the number of lines in source code. It counts every line including comments and blank lines.

(2) Blank Source Line of Code (BSLOC): this counts only the space line in the source codes. These lines of code only make the codes looked spacious enough and easy to comprehend, with or without the code will still execute.

(3) Non Commented Source Line of Codes (NCSLOC): counts line of codes that do not contain comments.

(4) Commented Source Line of Code (CSLOC): counts line of codes that contain comments.

(5) Effective Sources Lines of Code (ESLOC): It only counts the lines that are not commented, blank, standalone braces or parenthesis. In a way this metric presents the actual work performed. It calculates all executed line of codes.

$$ESLOC = TSLOC - (NCSLOC + BSLOC) \quad (i)$$

B. *Demonstration of the Metric*

Demonstration sample of the proposed metric for Validate Card Number implemented in RNG is given in Figure 1 below:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <grammar
3.   xmlns="http://relaxng.org/ns/structure/1.0"
4.   xmlns:a=http://relaxng.org/ns/compatibility/annotations/1.0
5.   datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes">
6.   <start>
7.        <element name="ValidateCard">
8.             <element name="ValidateCardNumber">
9.
10.                 <zeroOrMore>
11.                      <element name="cardType">
12.                           <data type="string" />
13.                      </element>
14.                      <element name="cardNumber">
15.                           <data type="string" />
16.                      </element>
17.                 </zeroOrMore>
18.
19.            </element>
20.            <element name="ValidateCardNumberResponse">
21.
22.                 <zeroOrMore>
23.                      <element name="ValidateCardNumberResult" >
24.                           <data type="string" />
25.                      </element>
26.                 </zeroOrMore>
27.
28.            </element>
29.       </element>
30.  </start>
31. </grammar>
```

Figure 1- RNG Code for Validate Card Number

C. *Analysis of the Metric*

(1) TSLOC = 31
(2) TSLOC = 31
(3) BSLOC = 4
(4) CSLOC = 3
(5) NCSLOC = 28
(6) ESLOC = TSLOC – (BSLOC + CSCLO)
= 31 – (4 + 3) = 24

V. RESULT

Series of experiments were conducted to show the effectiveness of the proposed metric, analyses of all the implemented RNGs can be seen in Table 1 below:.

A. *Comparative Study of the SLOC Metric*

The Figure 1 show the difference between TSLOC and BSLOC, the graph indicated that TSLOC has higher complexity values than BSLOC because the absence of BSLOC has no effect on the code of RNG schema files since with or without BSLOC the code will still be executed, in fact it only increase the number of TSLOC. While a close inspection of Figure 2 and 3 show comparison between TSLOC with CSLOC and TSLOC with NCSLOC. Commented lines and non commented lines were indicated in the two graphs. There is a closely relation between TSLOC and NCSLOC than TSLOC and CSLOC because CSLOC has higher value. This is due to the fact NCSLOC has a great contribution to the code.

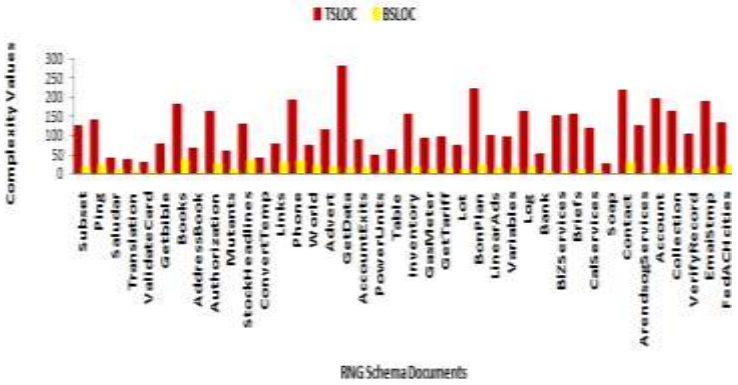

Figure 1: Comparison between TSLOC and BSLOC

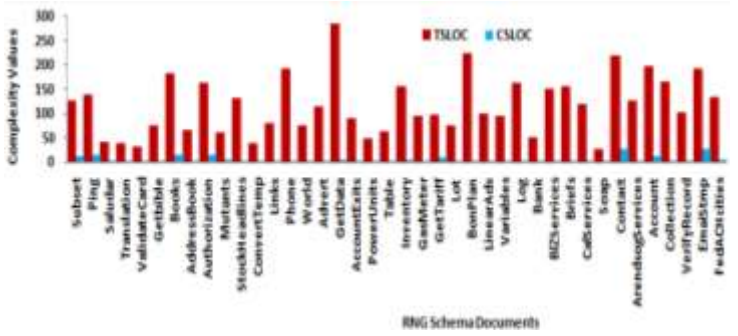

Figure 2: Comparison between TSLOC and CSLOC

Table 1: Complexity Measures for RNG Schema Documents

S/No	Schemas	TSLOC	BSLOC	CSLOC	NCSLOC	ESLOC
1	Subset	125	19	13	112	93
2	Ping	139	22	14	125	103
3	Saludar	40	9	3	37	28
4	Translation	38	6	3	35	29
5	ValidateCard	31	4	3	28	24
6	Getbible	76	6	3	73	67
7	Books	181	36	15	166	130
8	AddressBook	66	4	3	63	59
9	Authorization	163	26	15	148	122
10	Mutants	60	10	6	50	44
11	StockHeadlines	131	34	3	128	94
12	ConvertTemp	39	4	3	36	32
13	Links	79	30	3	76	46
14	Phone	192	34	3	189	155
15	World	74	21	3	71	50
16	Advert	114	17	3	111	94
17	GetData	282	15	4	178	263
18	AccountExits	90	15	3	87	72
19	PowerUnits	49	5	3	46	41
20	Table	63	9	3	60	51
21	Inventory	155	19	5	150	131
22	GasMeter	94	12	3	91	79
23	GetTariff	97	15	9	88	73
24	Lot	75	12	3	72	60
25	BonPlan	222	23	3	219	197
26	LinearAds	99	14	3	96	82
27	Variables	95	15	3	92	77
28	Log	163	18	3	160	142
29	Bank	51	5	3	48	43
30	BlZServices	151	3	3	148	145
31	Briefs	154	11	3	151	140
32	CalServices	119	7	3	116	109
33	Soap	27	0	3	24	24
34	Contact	217	28	28	189	161
35	ArendsogServices	126	4	3	123	119
36	Account	197	20	13	184	164
37	Collection	164	16	3	161	145
38	VerifyRecord	102	9	3	99	90
39	EmaiStmp	190	18	28	162	144
40	FedACHcities	132	23	4	128	105

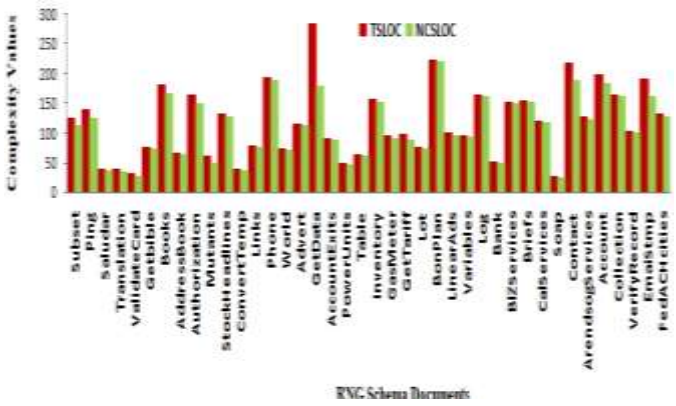

Figure 3: Comparison between TSLOC and NCSLOC

Figure 4 show relative graph of TSLCO and ESLOC with a close relation between the two measures except few schemas that show lower complexity values and the reason for this is that ESLOC is the real executed code for the schemas because the exception of ESLOC will automatically affect the code and rendered the code useless. Finally, is a relative graph of TSCLO, BSLOC, CSLOC, NCSLOC and ESLOC with different complexity values for each measure.

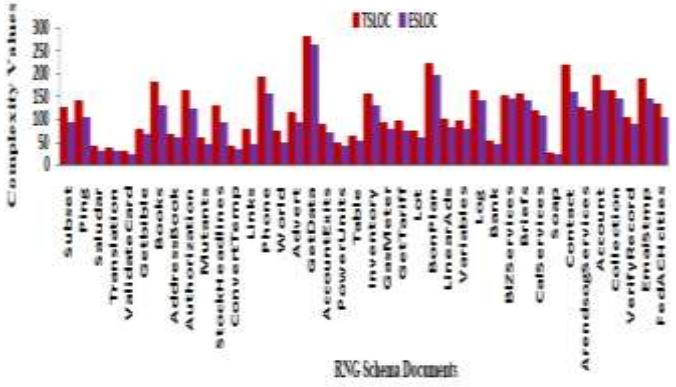

Figure 4: Comparison between TSLOC and ESLOC

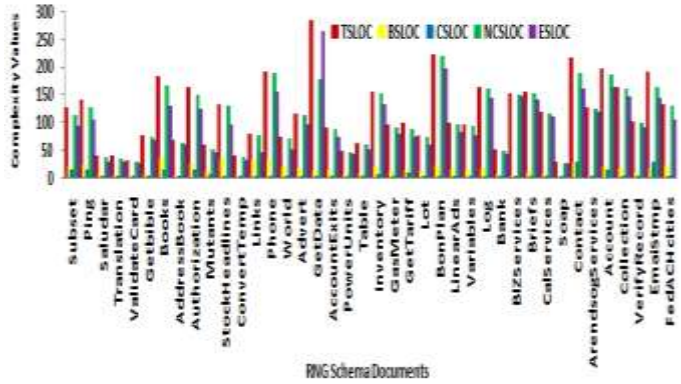

Figure 5: Reltive Graph between TSLOC, BSLOC, CSLOC, NCSLOC and ESLOC

Table 2: Pearson Correlation for SLOC Measures

		TSLOC	BSLOC	CSLOC	NCSLOC	ESLOC
TSLOC	Pearson Correlation	1	.587**	.455**	.966**	.981**
	Sig. (2-tailed)		.000	.003	.000	.000
	N	40	40	40	40	40
BSLOC	Pearson Correlation	.587**	1	.415**	.628**	.443**
	Sig. (2-tailed)	.000		.008	.000	.004
	N	40	40	40	40	40
CSLOC	Pearson Correlation	.455**	.415**	1	.419**	.326*
	Sig. (2-tailed)	.003	.008		.007	.040
	N	40	40	40	40	40
NCSLOC	Pearson Correlation	.966**	.628**	.419**	1	.939**
	Sig. (2-tailed)	.000	.000	.007		.000
	N	40	40	40	40	40
ESLOC	Pearson Correlation	.981**	.443**	.326*	.939**	1
	Sig. (2-tailed)	.000	.004	.040	.000	

.

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

The correlations coefficient is a statistical measure that measures the relationship between two variables. If one variable is changing its value then the value of second variable can be predicted. The positive correlation exists when a high value of one variable is associated with high value of another variable and the negative correlation exists when a high value is associated with low value. The value close to +1 means positive correlation exists, -1 means negative correlation exists and 0 means there is no correlation at all. Table 2 show the Person correlation coefficient of complexity value for TSLOC, BSLOC, CSLOC, NCSLOC and ESLOC for different measures in RNG schema documents.

Correlation coefficient between TSCLO, NCSLOC and ESLOC show value of 0.966 and 0.981 and between NCSLO and ESLOC show value of 0.939 indicate strong positive relation between the metrics and yielded the computed probability of values (p) of $0.000 < 0.05$ level of significant therefore this showed that a correlation exists between the metrics therefore H_0 rejects P-values. This implies a linear relationship and high degree of correlation between TSLOC, NCSLOC and ESLOC.

VI. CONCLUSION

SLOC Metric had being evaluated for software program or codebase according to its size because it serves as an intuitive metric for measuring the size of software which can be seen, visualized and most especially used in various fields of software engineering, to enable practitioners to carefully plan their measurement, cost estimation, effort reduction, usage of tools, comparison of data to many other projects, prevent lengthy code by organizing reviews, increase, proposes or introduce adjustment factors.. It gives reliable results, for projects for example TSLOC give details of all line

of codes; whether helping in the efficiency of the code or not, BSLOC showed a blank space to give the code good looking and give room for any adjustment by adding more to the codes, CSLOC indicates the lines with comment which show what is being done alongside or the title of the program itself and also shows its impact on the codes which does not affect its execution,. Finally, NCSLOC and ESLOC are the major useful programs in software that play very vital role for debugging or execution of codes. Future work may be toward evaluating and maintaining the quality of different XML schema languages of enabling database.

REFERENCES

[1]. Gill G.K. and Kemerer C.F. (1991): Cyclomatic Complexity Density and Software Maintenance, IEEE Trans. Software Engineering, 17: 1284-1288.

[2]. Basci D. and Misra S. (2010): Entropy as a Measure of Quality of XML Schema Document, The *International Arab Journal of Information Technology*, 8 (2011), 75-83.

[3]. Basci D., Misra S. (2011): Document Type Definition (DTD) Metrics, *Romanian Journal of Information Science and Technology* Vol. 14 no 1, pp 31-50.

[4]. Bray T., Jean P. and Sperberg-McQueen M.C. (2004): Extensible Markup Language (XML) 1.0 W3C Recommendation, *February 1998, World Wide Web Consortium (W3c) URL: http//www.w3.org/TR/1998/REC-xml-19980210.html.2012-02-02*.

[5]. Binstock C., Peterson D., Smith M., Wooding M., Dix C. and Galtenberg C. (2002): The XML Schema Complete Reference, Addison Wesley Professional Publishers.

[6]. Thompson H. S, Beech D., Muzmo. M. and Mendel- sohn N. (2004): *XML Schema Part 1: Structures Second Edition, W3C* Recommendation, 2004. http://www.w3.org/TR/xmlschema-1/

[7]. David C. F. (2001): XML Schema Part 0: Primer" [Online] 30 March 2001 (2001-03-30), W3C, XP002305611 *Retrieved from the Internet: URL:http://www.w3.org/TR/2001/PR-xmlschema-0-20010330/*.

[8]. Lee D. and Chu W. (2000): Comparative Analysis of Six XML Schema Language, *ACM SIGMOD Record*, 29, 3, pp. 1 {12.

[9]. Makoto M. (2002): Relax (Regular Language Description for XML). *Retrieved* June 2010 from http://www.xml.gr.jp/relax/

[10]. ISO: (2002). ISO/IEC TR 22250-1:2002 - Information Technology -- Document Description and Processing Languages -- Regular Language Description for XML (RELAX) -- Part 1: RELAX Core". *ISO*. Retrieved 2009-12-26.

[11]. Fenton N. E. (1992): Software Metrics – A Rigorous Approach, Chapman & Hall, London Computer Journal 29(4), 330 - 340.

[12]. http://docbook.sourceforge.net/release/dsssl/current/dtds/

[13]. http:// java.sun.com/dtd/

[14]. http://struts.apache.org/dtds/

[15]. http://jonas.objectweb.org/dtds/

[16]. https://www.researchgate.net/publication/319877377
http://www.ncbi.nlm.nih.gov/dtd/
http://www.cs.helsinki.fi/group/doremi/publications/XMLSCA2000.html
http://www.pramati.com/dtd/
http://www.w3.org/TR/REC-xml-names/
http://www.omegahat.org/XML/DTDs/
http://www.openmobilealliance.org/Technical/dtd.aspx
http://www.python.org/topics/xml/dtds/
http://www.okiproject.org/polyphony/docs/raw/dtds/
http://www.w3.org/TR/wsdl, Last Visited 2008.
http://www.w3.org/XML/, Last Visited 2008.
http://www.xml.gr.jp/relax, Last Visited 2008.

http://www.w3. org/TR /2004/REC- xmlschema- 1-20041028/, Last Visited 2008.
http://www.w3. org/TR /2001/PR- xmlschema-0-20010330/, Last Visited 2008.
http://www.w3. org/TR /2004/REC- xmlschema-2-20041028/, Last Visited 2008.
http://www.w3.org/TR/1998/REC-xml-19980210, Last Visited 2008.
http://www.xfront.com/GlobalVersusLocal. html, Last Visited 2008.

[17]. http:// ivs. cs. uni- magdeburg. de/sw- eng/ us/ metclas /index.shtml, Last Visited 2008.

[18]. http://fisheye5.cenqua.com/browse/glassfish/update-center/dtds/

[19]. Harrison, W., K. Magel, R. Kluczny, and A. DeKock (1982): Applying Software Complexity Metrics to Program Maintenance. Computer 1982 Published in: Journal Computer archive Volume 15 Issue 9, September 1982 Pages 65-79 IEEE Computer Society Press Los Alamitos, CA, USA

[20]. Vu Nguyen; Sophia Deeds-Rubin; Thomas Tan; Barry Boehm (2007): *A SLOC Counting Standard (PDF)*, Center for Systems and Software Engineering University of Southern California.

[21]. Kaushal Bhatt, Vinit Tarey and Pushpraj Patel (2012): Analysis Of Source Lines Of Code(SLOC) Metric *International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 5, pp 150-154 May 2012)*

[22]. Amit Kumar Jakhar and Kumar Rajnish (2014): A New Cognitive Approach to Measure the Complexity of *Software's International Journal of Software Engineering and Its Applications Vol.8, No.7 (2014), pp.185-198 http://dx.doi.org/10.14257/ijseia.2014.8.7,15 ISSN: 1738-9984 IJSEIA Copyright ⓒ2014 SERSC*

[23]. Sotonwa K. A, Olabiyisi T. O. and Omidiora E. A. (2014): Comparative Analysis of Software Complexity of Searching Algorithm using code Base Metric *International Journal of Scientific & Engineering Research, Volume 4, Issue 6, ISSN 2229-5518 June-2013 pp 2983-2993*