Tool Support for Cascading Style Sheets' Complexity Metrics

Adewole Adewumi[1], Onyeka Emebo[1], Sanjay Misra[2], and Luis Fernandez[3]

[1] Department of Computer and Information Sciences, Covenant University, Ota, Nigeria
[2] Department of Computer Engineering, Atilim University, Ankara, Turkey
[3] Department of Computer Science, University of Alcala, Madrid, Spain
{wole.adewumi, onye.emebo}@covenantuniversity.edu.ng
smisra@atilim.edu.tr
luis.fernandezs@uah.es

Abstract. Tools are the fundamental requirement for acceptability of any metrics programme in the software industry. It is observed that majority of the metrics proposed and are available in the literature lack tool support. This is one of the reasons why they are not widely accepted by the practitioners. In order to improve the acceptability of proposed metrics among software engineers that develop Web applications, there is need to automate the process. In this paper, we have developed a tool for computing metrics for Cascading Style Sheets (CSS) and named it as CSS Analyzer (CSSA). The tool is capable of measuring different metrics, which are the representation of different quality attributes: which include understandability, reliability and maintainability based on some previously proposed metrics. The tool was evaluated by comparing its result on 40 cascading style sheets with results gotten by the manual process of computing the complexities. The results show that the tool computes in far less time when compared to the manual process and is 51.25% accurate.

Keywords: Cognitive complexity, CSS Metrics, Tool Support.

1 Introduction

In recent times, Cascading Style Sheets (CSS) have become indispensable in the development of Web applications. They can be used for styling eXtensible Markup Language (XML) or HyperText Markup Language (HTML) documents. To style an HTML document, CSS can be applied in three ways namely: by placing them within the <head> tags of HTML; by placing them as tag attributes within the various other HTML tags that can be contained within the <body> tag of an HTML document; or by creating them as a separate document with extension (.css) and linking them to the <head> section of the HTML document. The latter of these procedures is a better practice as it separates content from presentation thereby promoting maintainability. Since CSS are an integral part of Web applications, they also add to the increasing complexity of such applications. In a previous work [1], we presented a suite of metrics to measure complexity in CSS at the International Conference on

Computational Science and Its Applications (ICCSA 2012). Prior to this, not much had been done in this regard. The metrics were based on cognitive complexity and were computed manually. This is a slow and tedious process that makes it of no practical use to software engineers and Web developers. Though we carried out a preliminary evaluation of the metrics and found them to be well structured, we proposed as future work to provide a tool to simplify the measurement process. This informs the motivation for this current paper.

Tool writing according to [2] is becoming a forgotten art. This is evidenced today by a growing body of literature that consists mainly of proposed and validated complexity metrics with no tools to measure them. Table I shows an in exhaustive list of such literature. In the XML schema document (XSD) domain, a metric has been proposed to measure complexity in XML schema documents (XSDs) [9]. The metric does this by considering the internal building blocks of XSD. The metric was demonstrated with examples and performed well in comparison to similar measures. To compute the metric however, no tool was provided. Similar to this, a design complexity metric was proposed for XSD in [17]. This metric covers all the major factors that affect the complexity of XSD. In addition, due to the diversity in structure of W3C XML schema languages, a metric based on the concept of entropy from information theory was proposed in [23] for assessing the structural complexity of XSDs. As for XSDs written in W3C Document Type Definition (DTD) language, a metric also based on the entropy concept from information theory was proposed in [14] to measure the structural complexity of XSDs written in DTD. The work was extended in [24] to include Distinct Structured Element Repetition Scale (DSERS) metric, which also measures the structural complexity of schemas in DTD language. This metric exploits a directed graph representation of a schema document and considers the complexity of schema due to its similar structured elements and the occurrences of these elements. In the XML/Web services domain, a data complexity metric for XML Web services was proposed in [16] which assesses the quality of Web services in terms of maintainability. Similarly, a suite of metrics for XML Web services was also proposed in [21] which includes: data weight of a web service description language, distinct message ratio metric, message entropy metric and message repetition scale metric. All the proposed metrics in the suite were evaluated theoretically and validated empirically. A comparative study with similar measures also proved the worth of the metric suite. In the coding language domain, a complexity metric was proposed in [20] for evaluating object-oriented code with emphasis on Python, Java and C++. The metric was validated empirically on real projects but no tool was developed for the metric. Similarly, a complexity metric was proposed in [19] called JavaScript Cognitive Complexity Measure (JCCM) for measuring the complexity of JavaScript code. Again the metric was evaluated theoretically and validated empirically but no tool support was made available. Other metrics based mostly based on cognitive informatics include: Modified Cognitive Complexity Measure [4], [5]; Complexity Measure based on Cognitive Weights [6]; Cognitive Program Complexity Measure [7]; Object Oriented Complexity Metric Based on Cognitive Weights [8]; a New Complexity Metric Based on Cognitive Informatics [10]; Object Oriented Programs Complexity Measure [11]; Unique Complexity Metric [12]; Weighted Class Complexity [13]; Unified Complexity

Measure [18]; and Inheritance Complexity Metric for Object-Oriented Code [22]. In all, these measures have no tool support.

Since tools are a fundamental requirement for the acceptance of any metric in the software industry, we are extending our previous work and developing a tool for computing metrics of CSS. The tool is named CSS Analyzer (CSSA). The rest of the paper is organized as follows: Section 2 gives a description of the tool developed. Section 3 evaluates the tool by comparing its result with the manual approach to measurement. Section 4 discusses the results of the comparison while Section 5 concludes the paper.

2 Description of CSSA

CSSA was developed using the Java programming language. In this section, we describe CSSA based on the metrics that it measures.

2.1 Rule Length (RL)

CSS is made up of rules. This metric counts the number of lines of rules in a CSS without taking into account white spaces or comment lines [1]. The pseudo code used to implement this functionality in CSSA is given as:

```
Read a CSS file;

Initialize a line counter variable to zero;

While not End of CSS File

          If a line is not empty and is not a comment;

            Increment line counter variable by 1;
```

Table 1. Metrics proposed having no tool support

S/N	Metrics name	Reference
1	Cognitive Complexity Measure	[3]
2	Modified Cognitive Complexity Measure	[4], [5]
3	Complexity Measure Based on Cognitive Weights	[6]
4	Cognitive Program Complexity Measure	[7]
5	Object Oriented Complexity Metric Based on Cognitive Weights	[8]
6	Complexity Metric for XML Schema Documents	[9]
7	A New Complexity Metric Based on Cognitive Informatics	[10]
8	Object Oriented Programs Complexity Measure	[11]
9	Unique Complexity Metric	[12]

10	Weighted Class Complexity	[13]
11	Entropy Metric for XML DTD Documents	[14]
12	Cognitive Functional Sizes	[15]
13	Data Complexity Metrics for XML Web Services	[16]
14	Design Complexity Metric for XML Schema Documents	[17]
15	Unified Complexity Measure	[18]
16	Complexity Metric for JavaScript	[19]
17	Python Language Complexity Metric	[20]
18	Metrics suite for maintainability of XML Web Services	[21]
19	Inheritance Complexity Metric for Object-Oriented Code	[22]
20	Entropy of XML Schema Document	[23]
21	DTD Metrics	[24]

2.2 Number of Rule Blocks

A rule block in CSS refers to a selector and its attributes depicted as:

```
/* Syntax of a rule block */

Selector [, selector2, ...] [:pseudo-class] {

          Property: value;

          [Property2: value2;

...]

}
```

The pseudo code used to implement this functionality in CSSA is given as:

```
Read a CSS file;

Initialize a brace counter variable to zero;

While not End of CSS File

          Increment brace counter variable by 1 every
time an open brace is read
```

2.3 Number of Attributes Defined per Rule Block (NADRB)

NADRB as defined in [1] determines the average number of attributes defined in the rule blocks of a CSS file. The formula for calculating it is given in (1):

$$NADRB = \Sigma \text{ rule\_block\_attributes} / \Sigma \text{ rule\_blocks} \quad (\mathbf{1})$$

The pseudo code for computing this metric in CSSA is given as:

```
Read a CSS file;

Initialize a semi colon counter variable to zero;

Initialize a close brace counter variable to zero;

While not End of CSS File

          Increment close brace counter variable by 1
every time an open brace is encountered

          Increment semi colon counter variable by 1
every time a semi colon is encountered

Divide the semi colon counter by the close brace
counter to get the NADRB value
```

2.4 Number of Cohesive Rule Blocks (NCRB)

NCRB measures the number of rule blocks in a CSS file possessing a single attribute. The pseudo code for computing this metric in CSSA is given as:

```
Read a CSS file;

Initialize counter variable to zero;

While not End of CSS File

          If the number of semi colons within a rule
block is one

            Increment counter variable by 1;

          Else

            Do nothing;
```

3 Evaluation of the Tool

In this section, we present the evaluation of the tool by applying it on forty real CSS files downloaded from the Internet. We compare the results obtained for each metric with the results gotten by manual computation of the metrics. Table II presents the CSS Ids and the web links from which they were downloaded. Table III shows a comparison of the results obtained by applying CSSA to the CSS files as well as the results obtained by manual computation.

Table 2. CSS IDs and Web Links

CSS ID	Web Link
1	http://www.freecsstemplates.org/download/zip/argon
2	http://www.freecsstemplates.org/download/zip/boldness
3	http://www.freecsstemplates.org/download/zip/bolness2
4	http://www.freecsstemplates.org/download/zip/classifieds
5	http://www.freecsstemplates.org/download/zip/combinations
6	http://www.freecsstemplates.org/download/zip/compass
7	http://www.freecsstemplates.org/download/zip/consistent
8	http://www.freecsstemplates.org/download/zip/corporatestuff
9	http://www.freecsstemplates.org/download/zip/estatebroker
10	http://www.freecsstemplates.org/download/zip/flamingo
11	http://www.freecsstemplates.org/download/zip/flowering
12	http://www.freecsstemplates.org/download/zip/fotofolium
13	http://www.freecsstemplates.org/download/zip/fruityblue
14	http://www.freecsstemplates.org/download/zip/handcrafted
15	http://www.freecsstemplates.org/download/zip/igunalounge
16	http://www.freecsstemplates.org/download/zip/infrastructure
17	http://www.freecsstemplates.org/download/zip/inwild
18	http://www.freecsstemplates.org/download/zip/islandpalm
19	http://www.freecsstemplates.org/download/zip/lettering
20	http://www.freecsstemplates.org/download/zip/limitless
21	http://www.freecsstemplates.org/download/zip/networked
22	http://www.freecsstemplates.org/download/zip/officememo
23	http://www.freecsstemplates.org/download/zip/outdoor
24	http://www.freecsstemplates.org/download/zip/petalsandflowers
25	http://www.freecsstemplates.org/download/zip/redallover
26	http://www.freecsstemplates.org/download/zip/redandblack
27	http://www.freecsstemplates.org/download/zip/reinstated
28	http://www.freecsstemplates.org/download/zip/rifle
29	http://www.freecsstemplates.org/download/zip/simplified
30	http://www.freecsstemplates.org/download/zip/woodcrafting
31	http://www.freecsstemplates.org/download/zip/stampalike
32	http://www.freecsstemplates.org/download/zip/naturalprime
33	http://www.freecsstemplates.org/download/zip/grasstown
34	http://www.freecsstemplates.org/download/zip/fullycharge
35	http://www.freecsstemplates.org/download/zip/spikyflower
36	http://www.freecsstemplates.org/download/zip/simpledisplay
37	http://www.freecsstemplates.org/download/zip/modelling
38	http://www.freecsstemplates.org/download/zip/surround
39	http://www.freecsstemplates.org/download/zip/halcyonic

Figs. 1 - 4 give a graphical comparison of the results gotten from the tool as well as that computed by hand. The next section discusses this in detail.

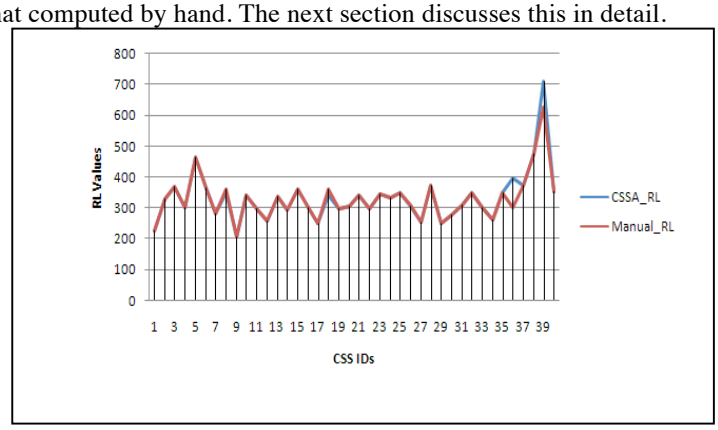

Fig. 1. This figure shows the correlation between Rule Length values calculated by hand and using the tool.

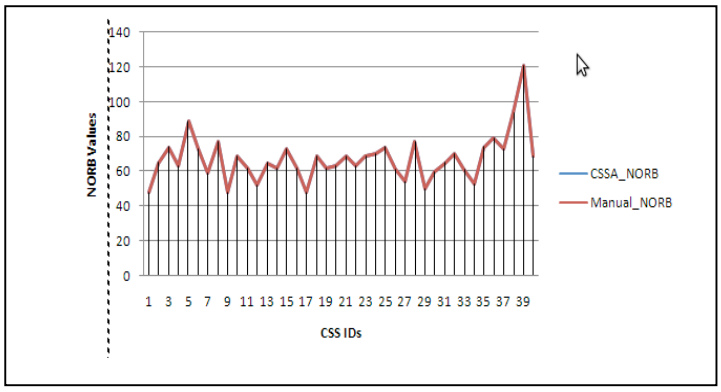

Fig. 2. This figure shows the correlation between NORB values calculated by hand and using the tool.

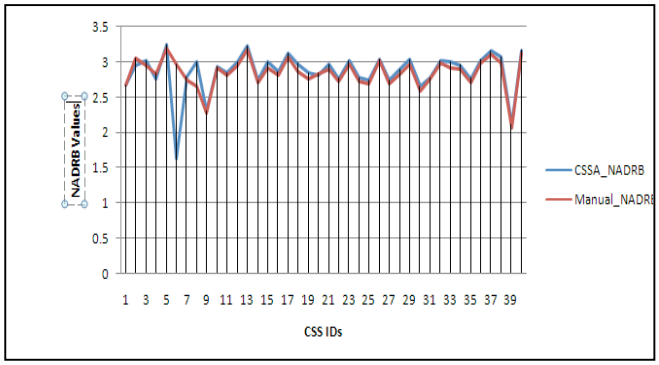

Fig. 3. This figure shows the correlation between NADRB values calculated by hand and using the tool.

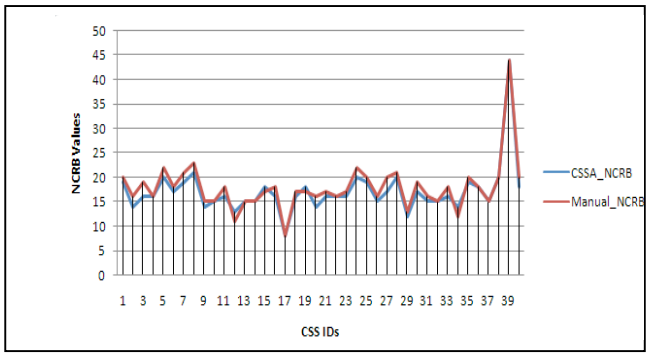

Fig. 4. This figure shows the correlation between NCRB values calculated by hand and using the tool.

Table 3. Comparison between results obtained using CSSA and manual computation

ID	CSSA				MANUALLY			
	RL	NORB	NADRB	NCRB	RL	NORB	NADRB	NCRB
1	224	48	2.68	19	224	48	2.67	20
2	329	65	2.95	14	329	65	3.06	16
3	368	74	3.02	16	368	74	2.96	19
4	300	63	2.76	16	300	63	2.84	16
5	463	89	3.25	20	463	89	3.20	22
6	363	73	1.63	17	364	73	2.97	18
7	280	59	2.79	19	280	59	2.75	21
8	349	77	3.00	21	362	77	2.66	23
9	205	48	2.31	14	205	48	2.27	15
10	340	69	2.94	15	340	69	2.93	15

11	298	62	2.85	16	298	62	2.81	18
12	258	52	3.01	13	258	52	2.96	11
13	336	65	3.23	15	336	65	3.19	15
14	292	62	2.75	15	292	62	2.71	15
15	359	73	3.00	18	359	73	2.92	17
16	299	62	2.87	16	300	62	2.82	18
17	248	48	3.12	8	248	48	3.08	8
18	340	69	2.98	16	360	69	2.87	17
19	296	62	2.85	18	296	62	2.77	17
20	305	63	2.82	14	305	63	2.84	16
21	339	69	2.97	16	339	69	2.91	17
22	298	63	2.77	16	298	63	2.73	16
23	343	69	3.02	16	343	69	2.97	17
24	332	70	2.78	20	332	70	2.73	22
25	348	74	2.75	19	348	74	2.70	20
26	306	61	3.04	15	307	61	3.02	16
27	253	54	2.74	17	253	54	2.69	20
28	373	77	2.90	20	373	77	2.84	21
29	249	50	3.04	12	249	50	2.98	13
30	276	60	2.65	17	276	60	2.60	19
31	309	65	2.78	15	309	65	2.77	16
32	349	70	3.02	15	349	70	2.99	15
33	302	61	3.00	16	302	61	2.92	18
34	260	53	2.96	14	260	53	2.91	12
35	349	74	2.77	19	349	74	2.72	20
36	395	79	3.03	18	302	79	3.00	18
37	374	73	3.16	15	374	73	3.11	15
38	475	94	3.07	20	475	94	3.00	20
39	711	121	2.11	44	628	121	2.07	44
40	354	69	3.17	18	354	69	3.13	20

4 Discussion

The comparison in Table III shows that for the RL metric only 7 (17.5%) of the CSS files analyzed do not give same results for the manual and automated complexity computation process. In other words, 82.5% of the CSS files match when computed using both automated and manual process. It is easy to identify the files that do not match by looking at the chart in Fig. 1. For instance, CSS file with ID 36 has an RL value of 395 when the tool is used but when calculated by hand the value obtained is 302 a sharp difference.

For the NORB metric, we observe that the results obtained by the tool and by manual computation are exactly alike for all 40 (100%) CSS files analyzed. This explains why the graph in Fig. 2 seems to have only one (red) color. A direct opposite of this is the case of NADRB metric where none of the 40 (0%) CSS files analyzed

give same results for the manual and automated computation process. This is supported by the irregular lines seen in Fig. 3 notable among them is the sharp difference between the automated and manual computation for CSS ID 6 (1.63 and 2.97 respectively).

As for the NCRB metric, only 9 (22.5%) out of the 40 CSS files analyzed have the same results. The differences in value for both automated and manual computation is not significant as can be seen in Fig. 4. To this end, summing the percentage match for each metric and dividing the total by the number of metrics considered helps to determine the percentage accuracy of the tool given as:

$(82.5 + 100 + 0 + 22.5)/4 = 51.25\%$ accuracy

The tool takes an average of 11 seconds to compute all four metrics while a manual computation takes an average of 390 seconds. This implies that the tool is 35 times faster than the manual computation process.

5 Conclusion

This paper presented the description and evaluation of CSSA - a tool for computing RL, NORB, NADRB and NCRB metrics for a CSS file. CSSA was implemented to alleviate the cumbersome process of determining the complexity of CSS by hand. The result from the comparison to the manual computation approach shows that CSSA is 35 times faster with an accuracy of 51.25%. Web developers and engineers can utilize the tool in its current form. As future work, we intend to improve on the accuracy of CSSA and also extend it to compute entropy of CSS files.

References

1. Adewumi, A., Misra, S., Ikhu-Omoregbe, N.: Complexity metrics for cascading style sheets, Lecture Notes in Computer Science, vol. 7336, (2012) 248-257
2. Spinellis, D.: Tool writing: a forgotten art?, IEEE Software, vol. 22, (2005) 9-11
3. Misra, S., Misra, A. K.: Evaluating cognitive complexity measure with Weyuker Properties, in Proc. 3rd IEEE International Conference on Cognitive Informatics, Victoria, Canada, (2004) 103-108.
4. Misra, S.: Modified cognitive complexity measure, in 21st International Symposium on Computer and Information Sciences, (2006) 1050-1059.
5. Misra, S.: Validating modified cognitive complexity measure, ACM SIGSOFT Software Engineering Notes, vol. 32, (2007) 1-5.
6. Misra, S.: Complexity measure based on cognitive weights, International Journal of Theoretical and Applied Computer Sciences, vol. 1, (2006) 1-10
7. Misra, S.: Cognitive program complexity measure, in Proc. 6th IEEE Int. Conf. on Cognitive Informatics, Lake Tahoe, CA, (2007) 120-125.
8. Misra, S.: An object oriented complexity metric based on cognitive weights, in Proc. 6th IEEE Int. Conf. on Cognitive Informatics, Lake Tahoe, CA, (2007) 134-139.
9. Basci, D., Misra, S.: Complexity metric for XML schema documents, in Proc. International Conference on SOA and Web services, (2012) 1-14.
10. Misra, S., Akman, I.: A new complexity metric based on cognitive, Rough Sets and Knowledge Technology, vol. 5009, (2008) 620-627.

11. Misra, S., Akman, I.: Measuring complexity of object oriented programs, in Proc. Computational Sciences and Its Application – ICCSA '08, Perugia, Italy, 2008 652-667.
12. Misra, S., Akman, I.: An unique complexity metric, in Proc. Computational Sciences and Its Application – ICCSA '08, Perugia, Italy, (2008) 641-651.
13. Misra, S., Akman, I. K.: Weighted class complexity: a measure of complexity for object oriented system, Journal of Information Science and Engineering, vol. 24, (2008) 1689-1708
14. Basci, D., Misra, S.: Entropy metric for XML DTD documents, ACM SIGSOFT Software Engineering Notes, vol. 33, (2008) 1-6
15. Misra, S.: Cognitive functional sizes, Int. J. of Software Science and Computational Intelligence, IGI-Global, 2009.
16. Basci, D., Misra, S.: Data complexity metrics for XML Web Services, Advances in Electrical and Computer Engineering, vol. 9, (2009) 9-15
17. Basci, D., Misra, S.: Measuring and evaluating a design complexity metric for XML schema documents, Journal of Info. Sci. and Engineering, vol. 25, (2009) 1405-1425
18. Misra, S., Akman, I.: Unified complexity measure: a measure of complexity, in Proc. Nat. Acad. Sci., (2010) 167-176.
19. Misra, S., Cafer, F.: Estimating quality of JavaScript, The International Arab Journal of Information Technology, vol. 9, (2012) 535-543
20. Misra, S., Cafer, F.: Estimating complexity of programs in Python Language, Technical Gazette, vol. 18, (2011) 23-32
21. Basci, D., Misra, S.: Metrics suite for maintainability of eXtensible markup language Web services, IET Software, vol. 5, (2011) 1-22
22. Misra, S., Akman, I., Koyuncu, M.: An inheritance complexity metric for object-oriented code: A cognitive approach, Sadhana, vol. 36, (2011) 317-337
23. Basci, D., Misra, S.: Entropy as a measure of quality of XML schema document, The International Arab Journal of Information Technology, vol. 8, (2011) 75-83
24. Basci, D., Misra, S.: Document Type Definition (DTD) metrics, Romanian Journal of Info. Science and Technology, vol. 14, (2011) 31-50