COMPLEXITY METRICS FOR STATECHART DIAGRAMS

Ann Wambui King'ori[1, 2], Geoffrey Muchiri Muketha[3]
and Elyjoy Muthoni Micheni[4]

[1]Department of Information Technology, Murang'a University, Kenya
[2]Department of Information Technology, Nkabune Technical Training Institute, Kenya
[3]Department of Computer Science, Murang'a University, Kenya
[4]Department of Management Sciences & Technology,
Technical University of Kenya

ABSTRACT

Model-Driven Development and the Model-Driven Architecture paradigm have in the recent past been emphasizing on the importance of good models. In the Object-Oriented paradigm one of the key artefacts are the Statechart diagrams. Statechart diagrams have inherent complexity which keeps increasing every time the diagrams are modified, and this complexity poses problems when it comes to comprehending the diagrams. Statechart diagrams provide a foundation for analysing the dynamic behaviour of systems, and therefore, their quality should be maintained. The aim of this study is to develop and validate metrics for measuring the complexity of UML Statechart diagrams. This study used design science which involved the definition of metrics, development of a metrics tool, and theoretical and empirical validation of the metrics. For the measurement of the cognitive complexity of statechart diagrams, this study proposes three metrics. The defined metrics were further used to calculate the complexity of two sample statechart diagrams and found relevant. Also, theoretical validation of the defined metrics was done using the Weyuker's nine properties and revealed they are mathematically sound. Empirical validations were performed on the metrics and results indicate that all the three metrics are good for the measurement of the cognitive complexity of statecharts.

KEYWORDS

UML Diagrams, Statechart diagrams, Software metrics, complexity measures, Theoretical validations, Empirical validations.

1. INTRODUCTION

Software quality is defined as the magnitude to which a given software adheres to the required specifications [5, 18]. Software quality evaluation should be carried out to reduce the effort, time, and cost put in a software product [3, 2]. Software quality is controlled by use of metrics. These metrics assess various aspects of software complexity and therefore help in improving software quality. Additionally, software metrics provide a way of estimating efforts needed for testing.

The Unified Modeling Language (UML) is a software modelling language that is composed of both text and graphical elements [24, 30]. The language is in a broad sense used for modelling of systems using object-oriented technology [7]. UML diagrams have notations that give various context of the system being analysed [26], and they also convey the dynamic and the static views of a software structure [8, 14]. The Statechart diagram is one of the core diagrams of UML used to model the dynamic aspects of a system. Statecharts define various states of an object during its

lifespan [8,14]. Formally, a Statechart diagram S may be described as a 3-tuple <S, T, E>. S represents a state, T means a transition, and E is an event which triggers state changes.

The problem with Statechart diagrams is that they have inherent complexity that keeps growing with age whenever the diagrams are enhanced. The size and complexity of UML diagrams affect their cognition [6]. Some known causes of complexity include symbol redundancy, symbol overload and lack of utilization of dual coding [1]. Researchers have showed that inadequacy of UML diagrams, UML semantics that are not precisely defined and the large number of UML constructs affect the cognitive effectiveness of UML diagrams [28].

High complexity is undesirable because it affects the overall quality of software. Researchers have proposed metrics to ensure quality of statecharts. However, metrics that can assess the cognitive complexity of UML Statechart diagrams are lacking. This paper presents three metrics for measuring and controlling the cognitive complexity of statechart diagrams.

The rest of this paper is organised as follows; Section 2 presents related work; section 3 presents the measurable attributes; section 4 presents the newly defined metrics; section 5 illustrates computation of values from two real life scenarios. Section 6 explains the results and the paper ends with some conclusions and future works.

2. RELATED WORKS

Several cognitive complexity metrics for Object Oriented design have been proposed [17,20, 21, 27]. However, these metrics are not suitable for the measurement of cognitive complexity of statechart diagrams due to their limitations.

Shao & Wang [27] proposed Cognitive Functional Size (CFS). The measure was based on internal architectural control-flows, output data and input data. The CFS has drawbacks. For example, it does not consider essential factors such as inheritance and excludes the details found in operators and operands thus cannot be used to measure the cognitive complexity of UML behavioral diagrams [13, 16].

In [20] Misra & Akman proposed a type of metric using cognitive weights to determine the class complexity in object-oriented system. The measure functions by associating a particular weight with each method. After assigning weight it adds the weights of all methods. Thus, complexity of a whole class is determined. The calculation of class complexity is very easy; however, it excludes the internal structure of the Method [25].

Kushwaha & Misra, [17] proposed Cognitive Information Complexity Measure (CICM) to aid in understanding the cognitive information complexity and the information coding efficiency of a program. The CICM relies on the cognitive weight of internal BCSs, identifiers, operators and lines of code (LOC) [17]. The measure has been criticized for being difficult to calculate if a software contains many lines of code [16, 19].

Misra [21] proposed the modified cognitive complexity measure (MCCM) which was a modification of cognitive functional size (CFS). The measure was based on the cognitive weights caused by the basic control structures, number of operators and operands. The measure is practicable as stated by the cognitive informatics works. However, the measure gives high complexity values [13,19].

3. IDENTIFICATION OF MEASUREMENT ATTRIBUTES

Statechart diagram complexity is caused by its building components such as states, transitions, events. Therefore, different metrics are required to assess each of its component [9]. Based on this argument, three base metrics will be defined for these identified attributes states, transitions and events.

3.1. States

A state models the circumstance when the object's behaviour will be stable. The object remains in a state until it is provoked to change by an event. A state is the environment during which an object fulfils some condition [26]. Also, a state model dynamic circumstances such as the process of executing some activity

3.2. Events

An event is a trigger that cause change of state. Events represents demands from the objects. An event can be defined as the conditions of noteworthy occurrence that has time and space [14, 26]. Events activate transitions in state machines.

3.3. Transitions

Transition is a directed connection joining a source state vertex and a target state vertex. It may take the state machine from one state to another, representing the complete reaction of the state machine to a particular Stimuli [14, 26]. A transition is composed of a trigger, a guard and actions to be executed upon entry into a state [14, 26]

4. METRICS DEFINITION

4.1. Base Metrics

The base metrics for UML statechart diagrams include: Number of States (NS) which is a count of the total number of states in a statechart diagram taking into considerations the simple states, composite state, orthogonal state, submachine state, initial state and final state. Number of Entry Actions (NEA) which is a count of the total number of entry actions performed upon entry to the state (entry actions are tasks that occur when an object is in a particular state), Number of Activities (NA) which is a count of the total number of do activities that is performed while the element is in this state (activities are behaviours performed while the element is in that state) and Number of Transitions (NT) a size attribute that counts the total number of transitions in the statechart diagram. In addition, Number of Guard (NG) which is the count of the total number of guard conditions in the statechart diagram (a guard is a Boolean condition that must be true for a transition to occur) and Number of Events (NE) which is the count of the total number of events in the statechart diagram. An event is a noteworthy occurrence that cause the transition from one state to another.

4.2. Derived Metrics

4.2.1. Weighted Number of States (WNS)

Weighted Number of States (WNS) is a function of a type of state and the weight assigned to the state as shown in Table 1. To calculate WNS, a weighted sum is obtained from the product of

each type of state by its corresponding weight. Therefore, Weighted Number of State (WNS) is calculated as follows:

$$WNS = \sum_{i=1}^{n} (W_i X_i)$$

Where X represents number of types of states, and i means the start of the first x value.

W is the weighted assigned to a state and n is the number of states in the statechart diagram.

$\sum W_i X_i$ is the summation of the products of states and their corresponding weight.

Table 1. Weights to categories of states

Category	**Weight (W_i)**
Simple state	1.5
Composite state	2.5
Orthogonal state	3
Initial state	1
Final state	1
Submachine state	2.5
History	2

The initial and final state are assigned a weight of 1 because they have no objects. A simple state is assigned a weight of 1.5 because it has no substructure. A composite and orthogonal states are assigned a weight of 2.5 because they are semantically equivalent. A weight of 2 is assigned to a history state and an orthogonal state is assigned a weight of 3 because it has more than one region.

4.2.2. Weighted Number of Transitions (WNT)

Weighted Number of Transitions (WNT) metric is a statechart adaptation of the Cognitive Functional Size (CFS) measure [31]. WNT is the function of types of transitions in the statechart diagram and the cognitive weights (W_j) shown in Table 2. To calculate WNT, a weighted sum is obtained from the product of each type of transition by its corresponding weight. Therefore, Weighted Number of Transitions (WNT) is calculated as follows:

$$WNT = \sum_{j=1}^{n} (W_j T_j)$$

Where T represents number of types of transitions, and j means the start of the first T value.

W is the weighted assigned to a transition and n is the number of transitions in the statechart diagram.

$\sum W_j T_j$ is the summation of the product of transitions and their corresponding weight.

Table 2. Weights to categories of events

Category	**Activity**	**W_j**
Sequence	Sequence	1
Branch	If, Pick	2
Loop	While, for each, report until	3

A transition following one path in a statechart diagram is represented by a sequence with a weight of 1 while a choice in a statechart diagram is represented by a branch with a weight of 2. Transitions that link one state to another and back to the prior state is represented by a loop with a weight of 3.

4.2.3. Weighted Number of Events (WNE)

Weighted Number of Events (WNE) is a function types of events in the statechart diagram and the weight assigned to the type of event (W_e). An active event (AE) that triggers any transition in the current state is given a weight of 2. A deferred event (DE) that does not trigger any transition in the current state is given a weight of 1. To calculate WNE, a weighted sum is obtained from the product of each type of event by its corresponding weight. Therefore, Weighted Number of Event (WNE) is calculated as follows:

$$WNE = \sum_{k=1}^{n} (W_k E_k)$$

Where E represents number of types of events, and k means the start of the first E value.

W is the weighted assigned to an event and n is the number of events in the statechart diagram.

$\sum W_k E_k$ is the summation of the products of transitions and their corresponding weight.

5. REAL LIFE SCENARIOS

5.1. Scenario I: Statechart Diagram for Interrupt Handling

This state diagram in Figure 1 shows the process of resuming to an activity that was suspended due to an occurrence of an interrupt. The composite state is made up of substates and a history state that will lead to resume of the activity. The computed values for WNS, WNT and WNE are shown in Table 3, 4 and 5 respectively.

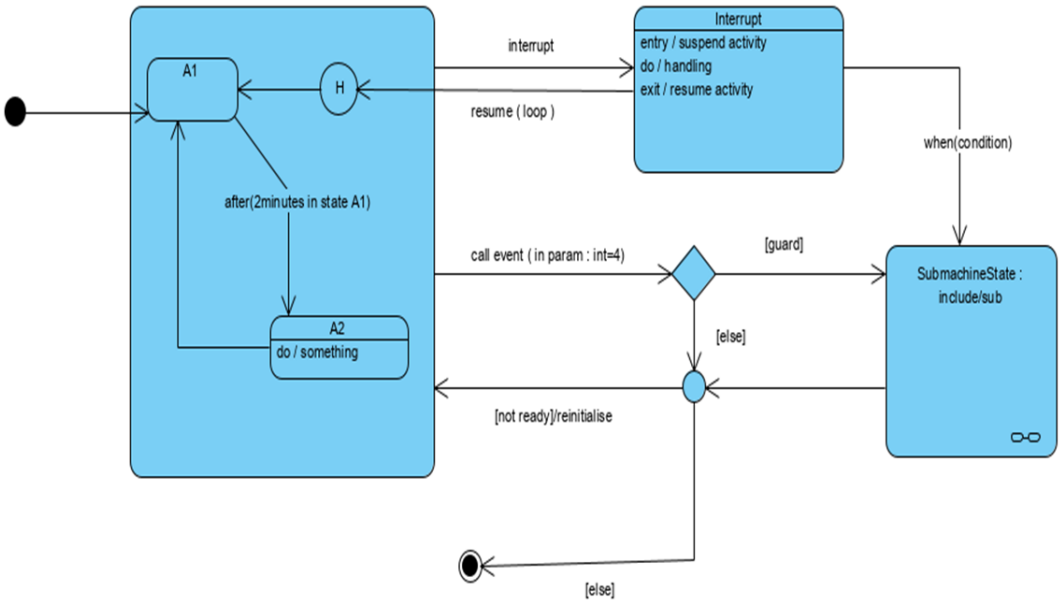

Figure 1. Interrupt handling Statechart diagram

Table 3. WNS Values

Category	**Weight (W_i)**	**X_i**	**$W_i X_i$**
Simple state	1.5	4	6
Composite state	2.5	1	2.5
Orthogonal state	3	0	0
Initial state	1	1	1
Final state	1	1	1
Submachine state	2.5	1	2.5
History State	2	1	2

WNS=6+2.5+0+1+1+2.5+2=15

The state diagram has 4 simple states which is multiplied by its corresponding weight of 1.5, the initial state is one is multiplied by 1, the final state is 1 is multiplied by its corresponding weight of 1, the history state is 1 is multiplied by 2 and the submachine state is 1 which is multiplied by 2.5. The summation of the product is 15.

Table 4. WNT Values

Category	**Activity**	**W_j**	**T_j**	**$W_j T_j$**
Sequence	Sequence	1	9	9
Branch	If, Pick	2	1	2
Loop	While, for each, report until	3	4	12

WNT=9+2+12=23

The state diagram has 9 sequences (one path transition) which is multiplied by its corresponding weight of 1, 1 branch multiplied by its corresponding weight of 2 and 4 loop which is multiplied by its corresponding weight 3. The summation of the product is 23.

Table 5. WNE Values

Event type	**W_k**	**E_k**	**$W_k E_k$**
AE	2	9	18
DE	1	0	0

WNE=18+0=18

The toaster has 9 active triggers which is multiplied by its corresponding weight of 2 and 0 deferrable triggers. The summation of the product is 18.

5.2. Scenario II: Toaster Statechart Diagram

This state diagram shows the process of heating a sliced bread in a toaster. The simple state "power on" has two deferrable triggers i.e "upTemp" and "lwrTemp". At any given time, the toaster is either power on or power off. The computed values for WNS, WNT and WNE are shown in Table 6, 7 and 8 respectively.

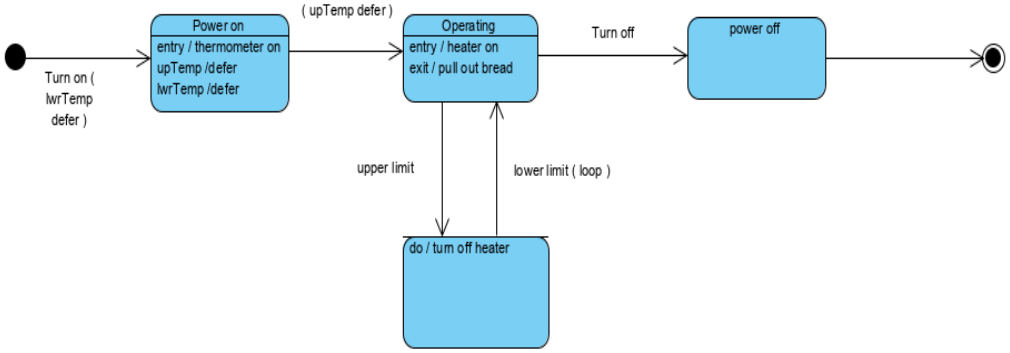

Figure 2. Toaster statechart diagram

Table 6. WNS Values

Category	**Weight (W_i)**	**X_i**	**$W_i X_i$**
Simple state	1.5	4	6
Composite state	2.5	0	0
Orthogonal state	3	0	0
Initial state	1	1	1
Final state	1	1	1
Submachine state	2.5	0	0
History State	2	0	0

WNS=6+1+1=8

The toaster has 4 simple states which is multiplied by its corresponding weight of 1.5, the initial state is one is multiplied by 1 and the final state is one is multiplied by its corresponding weight of 1. The summation of the product is obtained which is 8.

Table 7. WNT Values

Category	**Activity**	W_j	T_j	$W_j T_j$
Sequence	Sequence	1	6	6
Branch	If, Pick	2	0	0
Loop	While, for each, report until	3	1	3

WNT=6+0+3=9

The toaster has 6 sequences (one path transition) which is multiplied by its corresponding weight of 1and 1 loop which is multiplied by its corresponding weight 3. The summation of the product is obtained which is 9.

Table 8. WNE Values

Event type	W_k	E_k	$W_k E_k$
AE	2	6	12
DE	1	2	2

WNE=12+2=14

The toaster has 6 active triggers which is multiplied by its corresponding weight of 2 and 2 deferrable triggers which is multiplied by its corresponding weight 1. The summation of the product is obtained which is 14.

6. RESULTS

6.1. Theoretical Validations using Weyukers' Properties

Weyuker [33] proposed nine properties for validating complexity metrics. Weyuker's second and eighth property are applicable to all object-oriented metrics. The seventh property is not required for object -oriented programming. Weyuker did not address non- complexity metrics such as size or lengthy metrics. However, Weyuker's properties are one of the most widely referred frameworks for theoretical validations of object-oriented metrics. Weyuker's properties can be useful for statechart diagrams metrics validations because they are complexity metrics. Table 9 shows a summary of Weyuker's properties of measures.

Property 1: This property states that a measure should not rank all statechart diagrams as equally complex if they are non-identical.

All the proposed metrics return a different complexity value for any two non-identical statechart diagrams. For example, all the three metrics return different complexity values for Scenario I and II since the diagrams are non-identical. Therefore, the metrics WNE, WNS and WNT satisfy this property.

Property 2: Let c be a non-negative number. Then there are only finite number of diagrams of complexity c. This property asserts that a changing diagram must also cause a change to its complexity.

The metrics WNS, WNE and WNT can detect changes in complexity when the number of types of states, events and transition is changed. Therefore, all these metrics satisfy property 2.

Property 3: There are distinct statechart diagrams P and Q for which |P| = |Q|. This property asserts that there exist two different statechart diagrams whose effect is identical i.e. two different statechart diagrams with identical values. The metrics WNS, WNE and WNT can detect changes in complexity when the number of types of states, events and transitions is changed. Therefore, WNS, WNE and WNT satisfy this property.

Property 4: $(\exists P)\ (\exists Q)\ (P \equiv Q\ \&\ |P| \neq |Q|)$. There exist statechart diagrams P and Q such that the external effect of P and Q are identical, but |P| is not equal to |Q|. This property asserts that two statechart diagrams P and Q could look identical in terms of the fact that they contain the same number of states, events and transitions, but could have different complexities if the types of these states, events and transitions are different.

The metrics WNS, WNE and WNT can detect changes in complexity when the types of states, events and transitions is changed. Therefore, all the proposed metrics satisfy this property.

Property 5: For all statechart diagrams P and Q, considering also the diagram P; Q obtained by combining P and Q, |P| + Q| is less than or equal to |P; Q|. Monotonicity asserts that if a compound statechart diagram is developed by merging two simple statechart diagrams, the complexity of the compound statechart diagram is greater than the complexities of the two-initial simple statechart diagrams calculated separately. The metrics WNE, WNS and WNT satisfy property 5 since they all return numeric results.

Property 6: There exist statechart diagrams P, Q and R such that |P| is equal to |Q| but |P; R| is not equal to |Q; R|. Nonequivalence of interaction asserts that if two equivalent statechart diagrams are combined with a separate but statechart diagram, then their new complexities will differ from the original complexities. This shows that there is some complexity that arises as a result of interaction which is distinct from the interacting diagrams.

The WNS, WNT and WNE allocated constant weights to each of their types thus concatenation of two statechart diagrams does not result to arise of external complexity. The WNS, WNT and WNE metrics do not satisfy Weyuker's property 6.

Property 7: The permutation property asserts that the order of elements within a statechart diagram can affect its complexity. The metrics WNS, WNT and WNE allocated constant weights to each of their types thus permutation does not cause change of their complexity values. Therefore, the metrics do not satisfy property 7.

Property 8: If two statechart diagrams P and Q differ only in the choice of names for different states, events and transitions, then |P| is equal to |Q|. This property asserts that two diagrams are equal if their only difference is the choice of names.

All proposed metrics return numeric values. This means that renaming a diagram cannot affect its cognitive complexities. Therefore, all proposed metrics satisfied this property.

Property 9: There exist statechart diagrams P and Q for which |P|+|Q| is less than |P; Q): This property implies that adding two statechart diagrams results in extra complexity. Since growth in diagram complexity occurs when new states, events and transitions are added, which have non negative values, then the complexity of the new diagram is always equal to or greater than the sum of the original diagrams. Consequently, WNS, WNT and WNE metrics satisfied property 9. A summary of these results is shown in Table 9.

Table 9. Theoretical validation results

Property	1	2	3	4	5	6	7	8	9
WNS	YES	YES	YES	YES	YES	No	No	YES	YES
WNT	YES	YES	YES	YES	YES	No	No	YES	YES
WNE	YES	YES	YES	YES	YES	No	No	YES	YES

Key: Yes, satisfies property; No, does not satisfy property.

6.2. Experiment Validations

Experimentation is usually used due to its formal qualities [22, 23, 32]. Theoretical validation is not sufficient to reveal the soundness of a metric [4, 10, 11, 15, 29], because by means of experimental validation reveal evidence on the usefulness of proposed metrics is confirmed.

In this section, empirical results are presented as evidence that UML statechart diagram complexity metrics are related to cognition of UML statechart diagram, and cognition time of Statechart diagram. The variables in the experiment were statechart metrics as the independent variable while subjects' rating of cognition of statechart diagram and cognition time of statechart diagram are the dependent variables. A static analyser tool was developed by the researcher to automatically compute metrics from statechart diagrams. A within-subject design was used for the experiment where each subject analysed 34 statechart diagrams independently.

The hypotheses under investigation in the empirical studies were for the purpose of establishing if there exist a relationship between statechart metrics and the subjects rating of cognition of statechart diagram, and subjects' cognition time of a statechart diagram. This hypothesis includes:

i. Null Hypothesis (H0-c): There exists no significant correlation between the statechart metrics and subjects rating of cognition of a statechart diagram.
ii. Alternative Hypothesis (H1-c): There exists significant correlation between the statechart metrics and subjects rating of cognition of statechart diagram.
iii. Null Hypothesis (H0-ct): There exists no significant correlation between the statechart metrics and cognition time of a statechart diagram.
iv. Alternative Hypothesis (H1-ct): There exists significant correlation between the statechart metric and cognition time of a statechart diagram.

The complexity metrics of each diagram computed by the static analyzer tool are shown in Table 10. The metrics were applied to 34 different statechart diagrams. Table 11 shows subjects' rating of statechart diagrams while Table 12 shows cognition time of statechart diagrams.

Table 10. Statechart Metric Values

STATECHART DIAGRAM NO	WNS	WNT	WNE
1	9	12	12
2	15	20	22
3	9	14	10
4	21	39	36
5	20.5	19	22
6	10.5	7	14
7	12.5	8	16
8	27	14	28
9	6.5	6	12
10	16.5	21	30
11	22	19	32
12	12.5	12	18
13	6.5	16	14
14	8	11	12
15	11	12	18
16	14	22	20
17	14	18	24
18	12.5	17	18
19	10.5	7	14
20	19	24	30
21	11	11	16
22	23	43	44
23	3.5	8	6
24	11	16	20
25	9	9	18
26	16.5	6	12
27	25	10	20
28	17.5	15	30
29	14	29	28
30	17	26	22
31	21.5	27	42
32	11	13	16
33	14	18	18
34	8	9	14

Table 11. Subjects' cognition rating

STATECHART DIAGRAM NO	**SUBJECTS' RATING**
1	2.49
2	3.35
3	2.39
4	3.70
5	3
6	1.42
7	2.56
8	3.46
9	2.18
10	3.16
11	4
12	2.81
13	2.46
14	1.46
15	2.78
16	3.42
17	2.95
18	4.14
19	1.76
20	3.37
21	1.71
22	3.53
23	2.25
24	2.71
25	2.46
26	2.45
27	2.73
28	3.67
29	3.02
30	3.76
31	3.40
32	2.66
33	3.18
34	1.85

Table 12. Subjects cognition time

STATECHART DIAGRAM NO	SUBJECTS' COGNITION TIME (SECS)
1	197.91
2	168.31
3	96.11
4	197.05
5	90.54
6	28.67
7	79.09
8	108.84
9	80.2
10	172.08
11	178.65
12	96
13	133.74
14	62.12
15	105.07
16	109.98
17	89.04
18	147.10
19	59.59
20	119.59
21	84.84
22	143.24
23	82.06
24	108.49
25	102.37
26	80.32
27	76.65
28	151.25
29	122.44
30	149.16
31	168.45
32	72.95
33	135.73
34	57.02

Spearman's correlation coefficient was used to correlate each of the defined metrics with subject's rating of cognition and cognition time of UML statechart diagrams. The correlation coefficients are shown in Table 13 and 14.

Table 13. Correlation for metrics and cognition of statechart diagram

Statechart Metrics	Correlation Coefficients	p-value (2-tailed)
WNS	0.771**	0.000
WNT	0.791**	0.000
WNE	0.825**	0.000
**=99% confidence		

The correlation coefficients in Table 13 show that there exists a high positive correlation between statechart metrics and subjects' rating of cognition of UML statechart diagram. This is due to the fact that all the coefficient values are greater than 0.7.

From the results, the null hypothesis that there exists no significant correlation between the metrics and subjects' cognition of a statechart diagram is rejected and the alternative hypothesis accepted. These results indicate that the proposed metrics are indicators of the ease with which the subject comprehends a UML statechart diagram.

Table 14. Correlation for metrics and cognition time

Statechart Metrics	**Correlation Coefficients**	**p-value (2-tailed)**
WNS	0.463**	0.006
WNT	0.750**	0.000
WNE	0.617**	0.000
**=99% confidence		

Analysis of the spearman's correlation in Table 14, leads to conclusion that there exists a positive correlation between the statechart metrics and cognition time of statechart diagram. This leads to the rejection of the null Hypothesis there exists no significant correlation between the statechart metrics and cognition time of a statechart diagram and the alternative Hypothesis there exists significant correlation between the statechart metric and cognition time of a statechart diagram is accepted.

Regression model strengthens the correlation results. Regression helps to understand the relationship between the dependent variable and independent variable. Table 15 shows the p-values of the regression model based on metric values and cognition while Table 16 displays the p-values based on metric values and cognition time.

Table 15. Regression model based on metric values and cognition

Metric	**R-square**	**P-Value**
WNS	0.432	0.000
WNT	0.445	0.000
WNE	0.480	0.000
*P< 0.05		

From the regression model, the p-values are less than 0.05 indicating that the linear regression model can be used to predict the subjects' cognition of statechart diagram.

Table 16. Regression model based on metric values and cognition time

Metric	R-square	P-value
WNS	0.151	0.023
WNT	0.413	0.000
WNE	0.350	0.000
*P< 0.05		

The p-values of the regression model based on metric values and cognition time are less than 0.05. This indicates that the proposed metrics can be used to predict the cognition time of a statechart diagram.

7. CONCLUSIONS AND FUTURE WORKS

In this paper, three metrics namely, WNS, WNT and WNE were proposed in a methodological way. The metrics assess the cognitive complexity of UML statechart diagram. The theoretical validity of the defined metrics which indicate that the proposed metrics measure the characteristic they intend to measure was demonstrated through the validation through the Weyuker's properties. With the objective of confirming that there exists a great correlation between the metric values and the subject's cognition and cognition time of a statechart diagram, a controlled experiment was carried out. Spearman's correlation results led to the conclusion that the statechart metrics were directly related with cognition and cognition time of Statechart diagrams. In addition, the linear regression models imply that the proposed metrics can be used to predict the cognition of a statechart diagram and cognition time of a statechart diagram.

The literature conducted during the period of this study reveal that metrics for the measurement of statechart diagrams are scarce. Thus, one future work is to further investigate all factors that could possibly affect the structural complexity of UML dynamic models and then come up with new ways of measuring them. Another future work would be to validate the metrics using the DISTANCE framework as proposed by Poels and Dedene and conduct replica experiments with industry experts for further validation of the presented metrics.

REFERENCES

[1] Anwer, S., & El-Attar, M. (2014). An evaluation of the statechart diagrams visual syntax. *In Information Science and Applications (ICISA), 2014 International Conference on (pp. 1-4).* IEEE.

[2] Al Obisat, F. M., Alhalhouli, Z. T., Alrawashdeh, T. I., & Alshabatat, T. E. (2018). Review of Literature on Software Quality. *World Comput. Sci. Inf. Technol. J*, 8(5), 32-42.

[3] Azar, D., Harmanani, H., & Korkmaz, R. (2009). A hybrid heuristic approach to optimize rule-based software quality estimation models. *Information and Software Technology*, 51(9), 1365-1376.

[4] Basili V., Shull F. and Lanubille F. (1999). Building Knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4), 456-473

[5] Chandrasekar, A., Rajesh, S., & Rajesh, P. (2014). A research study on software quality attributes. *International Journal of Scientific and Research Publications*, 4(1), 14-19.

[6] Dori, D., Wengrowicz, N., & Dori, Y. J. (2014). A comparative study of languages for model-based systems-of-systems engineering (MBSSE). In *2014 World Automation Congress (WAC)* (pp. 790-796). IEEE.

[7] Fahad A. (2012). State Based Static and Dynamic Formal Analysis of UML State Diagrams. *Journal of Software Engineering and Applications*, 5, 483-491.

[8] Fitsilis, P., Gerogiannis, V. C., & Anthopoulos, L. (2013). Role of Unified Modelling Language in Software Development in Greece-results from an exploratory study. *IET software*, 8(4), pp. 143-153.

[9] Fenton, N. (1994). Software measurement: a necessary scientific basis. *IEEE Transactions on Software Engineering*, 199-206.

[10] Fenton N. and Pfleeger S. (1997). *Software Metrics: A Rigorous Approach.* 2nd. edition. London, Chapman & Hall.

[11] Genero, M., Miranda, D., & Piattini, M. (2002). Defining and validating metrics for UML statechart diagrams. *Proceedings of QAOOSE, 2002*.

[12] Jain, L., & Singh, S. (2019). Designing the Code Snippets for Experiments on Code Comprehension of Different Software Constructs. and outputs. International Journal of Computer Sciences and Engineering, Vol.7, Issue.3, pp.310-318. https://doi.org/10.26438/ijcse/v7i3.310318

[13] Jakhar, A.K & Rajnish, K. (2014). A New Cognitive Approach to Measure the Complexity of Software's. International Journal of Software Engineering & its Applications, vol. 8, no. 7, pp. 185-198.

[14] Jama, O.M., (2009). *A Case Study on Evaluating UML Modelling in Software Testing* (Master's thesis, University of OSLO).

[15] Kitchenham B., Pflegger S. and Fenton N. (1995). Towards a Framework for Software Measurement Validation. *IEEE Transactions of Software Engineering*, 21(12), 929-943.

[16] King'ori, A. W., Muketha, G. M., & Micheni, E. M. (2019). A literature survey of cognitive complexity metrics for statechart diagrams. International Journal of Software Engineering & Applications (IJSEA), 10(4).

[17] Kushwaha, D. S., & Misra, A. K. (2006). Cognitive complexity metrics and its impact on software reliability based on cognitive software development model. ACM SIGSOFT Software Engineering Notes, 31(2), 1-6.

[18] Maurya L.S et al. (2010). Comparison of Software Architecture Evaluation Methods for Software Quality Attributes, *Journal of Global Research in Computer Science*, 1 (4).

[19] Misra, S., Adewumi, A., Fernandez-Sanz, L., & Damasevicius, R. (2018). A Suite of Object Oriented Cognitive Complexity Metrics. IEEE Access, 6, 8782–8796.

[20] Misra, S., & Akman, I. (2008, May). A new complexity metric based on cognitive informatics. In International Conference on Rough Sets and Knowledge Technology (pp. 620-627). Springer, Berlin, Heidelberg.

[21] Misra, S. (2006). Modified Cognitive Complexity Measure. Lecture Notes in Computer Science 4263: 1050-1059. https:// doi:10.1007/11902140_109.

[22] Muketha, G. M., Ghani, A. A. A., Selamat, M. H., & Atan, R, (2010b) A Survey of Business Process Complexity Metrics. *Information Technology Journal*, Vol 9, No. 7, pp1336-1344.

[23] Ndia, J. G. (2019). *Structural Complexity Framework and Metrics for Analyzing the Maintainability OF Sassy Cascading Style Sheets* (Doctoral dissertation, MMUST).

[24] Padmanabhan, B. (2012). Unified modeling language (UML) overview. *EECS810–Principles of Software Engineering*.

[25] Rabani, S. T., & Maheswaran, K. (2017). Software Cognitive Complexity Metrics for OO Design: A Survey. International Journal of Scientific Research in Science, Engineering and Technology, 3, 691-698.90–101.

[26] Rambaugh, J., Jacobsen, I. & Booch, G. (2005). *The Unified Modelling Language Reference Manual*, second Edition. Pearson Higher Education

[27] Shao, J., & Wang, Y. (2003). A new measure of software complexity based on cognitive weights. Canadian Journal of Electrical and Computer Engineering, 28(2), 69-74.

[28] Siau, K., & Loo, P.P. (2006). Identifying difficulties in learning UML. *Information Systems Management*, 23(3), 43-51.

[29] Schneidewind N. (1992). Methodology for Validating Software Metrics. *IEEE Transactions of Software Engineering*, 18 (5), 410-422.

[30] UML, O. (2012a). Information technology-Object Management Group Unified Modelling Language (OMG UML), Infrastructure.

[31] Wang, Y. & Shao J., (2003). A new Measure of Software Complexity Based on Cognitive weights. Journal of Electrical and Computer Engineering,28(2),69-74.

[32] Wohlin, C., P. Runeson, M. Host, M. C. Ohlsson, and B. Regnell. "Experimentation in Software Engineering-An Introduction. Kluwer Academic Publishers." *Doedrecht the Netherlands* (2000).

[33] Weyuker, E.J. (1988). Evaluating software complexity measures. *IEEE Transactions on Software on Software Engineering* 14: 1357-1365.

AUTHORS

Ann Wambui King'ori is an ICT Lecturer at the Department of Information Communication Technology at Nkabune Technical Training Institute, Kenya. She earned her Bachelor of Technology Education (Computer Studies) from the University of Eldoret, Kenya in 2014, and her MSc. In Information Technology from Murang'a University of Technology, Kenya, 2021. She is currently pursuing her PhD. in Information Technology at Murang'a University of Technology, Kenya. Her research interests include software metrics, software quality, and business intelligence.

Geoffrey Muchiri Muketha is Professor of Computer Science and Director of Postgraduate Studies at Murang' a University of Technology, Kenya. He received his BSc. in Information Science from Moi University in 1995, his MSc. in Computer Science from Periyar University, India in 2004, and his PhD in Software Engineering from Universiti Putra Malaysia in 2011. He has wide experience in teaching and supervision of postgraduate students. His research interests include software and business process metrics, software quality, verification and validation, empirical methods in software engineering, and component-based software engineering. He is a member of the International Association of Engineers (IAENG).

Elyjoy Muthoni Micheni is a Senior Lecturer in Information Systems in the Department of Management Science and Technology at The Technical University of Kenya. She holds a Ph.D. (Information Technology) from Masinde Muliro University of Science and Technology, Master of Science (Computer Based Information Systems) from Sunderland University, (UK); Bachelor of Education from Kenyatta University; Post Graduate Diploma in Project Management from Kenya Institute of Management. She has taught Management Information System courses for many years at the University level. She has presented papers in scientific conferences and has many publications in refereed journals. She has also co-authored a book for Middle-level colleges entitled: "Computerized Document Processing". Her career objective is to tap computer-based knowledge as a tool to advance business activities, promote research in ICT and enhance quality service.