A Perspective-based Complexity Analysis Framework for UML Behavioral Diagrams

Ann Wambui King'ori
Department of Information Technology
Murang'a University of Technology
Kenya

Geoffrey Muchiri Muketha
Department of Computer Science
Murang'a University of Technology
Kenya

John Gichuki Ndia
Department of Information Technology
Murang'a University of Technology
Kenya

ABSTRACT
Software designers are rapidly adopting UML behavioral diagrams to communicate the dynamic behavior of software. As is the case with many other software artefacts, these diagrams tend to get more complex whenever they are modified for either corrective or enhancement purposes thus compromising on their quality. Several researchers have proposed different measurement frameworks to assess the quality of the various software artefacts. However, these existing frameworks cannot be directly applied to assess UML behavioral diagrams which come with unique complexity perspectives not seen in traditional software. This paper, therefore, proposes a perspective-based framework for assessing the complexity of UML behavioral diagrams. The proposed framework identifies three complexity perspectives, namely, element, control-flow, and interaction perspective. Each perspective in turn defines a set of measurable attributes. The framework was validated using an expert opinion survey. Because of the difficulty in getting UML experts, purposive sampling was adopted to select eleven industry participants. Descriptive statistics was used for data analysis. Findings indicate that the proposed framework is effective and adequate in form, which implies that it can be a good tool for defining new complexity metrics for UML behavioral diagrams. Such metrics can in turn be used to predict the behavioral quality of software.

Keywords
Measurement Frameworks, Software Metrics, UML Behavioral Diagrams, Software Complexity, Software Quality Control

1. INTRODUCTION
Modeling is critical in many disciplines because it makes easier the communication and construction of complex system from minor parts [1]. The focus of software quality assurance is shifting from system implementation towards system modelling (model verification and validation). Models are important in communicating the components of a system for productive analysis [1]. The Unified Modeling Language, (UML) is widely used by designers to develop analysis and design models [2, 3]. It provides models to show the static structure and the dynamic behavior of a system. The dynamic behavior illustrates how the system changes at run time while the static UML diagram focus on the structural components of a system [2, 4].

As software systems become more complex and a necessity in everyday activities, a lot of emphasis has been placed on software quality. To assess software quality, measurement process has been applied. Measurement can be defined as the process of discovering, planning, executing and assessing measurement of a project [5, 6]. Software measurement is critical in software engineering since it allows system developers to obtain reliable estimates concerning deadlines, cost, and quality for the development of their systems. The software measurement process is executed in all phases of software development life cycle [7, 8]. This measurement process is guided by use of metric measurement frameworks.

Metrics measurement frameworks provide a guide on how metrics should be defined and validated. A number of measurement frameworks have been proposed in the literature to aid in assessing the different software quality attributes. However, they do not consider all the measurable perspectives of UML behavioral diagrams. Considering the problem, a perspective-based complexity framework has been proposed by establishing all measurable perspective of these behavioral diagrams. In addition, the proposed framework is validated via an expert opinion survey.

The rest sections are organized as follows. Section 2 covers an overview of related work on software measurement and measurement frameworks. Section 3 presents the proposed perspective-based complexity framework. Section 4 resents the methodology, section 5 presents the validation of the framework, section 6 presents the discussion and finally the conclusion and future works are presented in section 7.

2. RELATED WORK
This section presents related work on software measurement and metrics measurement framework.

2.1 Measurement
Please Software measurement is the procedure of using numbers or symbols to evaluate an aspect of an object [9]. Software measurement is established on models such as Goal Attribute Measure (GAM), Goal Question Metric (GQM), Balanced Scorecard (BSC) and the Entity Attribute Metrics Model (EAM).

Goal-Attribute Measure (GAM) focuses on product, processes and resources [10]. To derive measures in GAM, first identify the measurement customers and their goals, and then identify their attributes, their driving attributes, and measurement objects. Lastly, attributes are further divided into measurable sub attributes from which metrics are defined [10]. In GAM, the scope of goals is on measurement objects while focus is on the structuring and definition of attributes.

The Balanced Scorecard (BSC) [11] is used to assess how an organization is making progress in achieving their goals. In BSC, an organization recognizes its mission and vision. Also, the drivers to aid in realizing their goals are identified. In addition, indicators for each driver are obtained. BSC provides four perspectives namely, financial perspective (shareholders' view), customer perspective (value-adding view), internal perspective (process-based view), and learning and growth perspective (future view). The first step in deriving BSC metrics starts with the analysis of the mission and vision of the organization. The second step is the definition of goals for financial and other perspectives. The next step defines drivers that aid in accomplishing the goals. Finally, indicators for each driver are defined [11].

The GQM by Basili [12] is the most used framework. GQM defines a measurement model on three levels: Conceptual level (Goal) where a goal is defined for an object for a number of reasons, Operational level (Question) where a collection of questions is used to define models of the object under study to realize a specific goal and, Quantitative level (Metric) where a set of measures are based on the models, and are linked with every question in order to answer it in a measurable way [12]. The GQM is built on the assumption that in order to evaluate in an objective way, an organization must specify goals, identify them by means of questions pointing their important attributes and give measurements to answer these questions [12].

The Entity Attribute Model by Fenton & Pfleeger [9, 13] focuses on three steps which include identification of a measurable entity, identification of entity measurable attributes and definition of new measures to assess each of the identified attributes [9, 13]. An entity is an item such as a piece of software module while an attribute is a measurable feature of the entity. Entities are categorized into three, namely process, products and resource. A process is an activity undertaken to develop a software, a product is the object produced during software development and a resource is the hardware or software required for the process [9, 13].

2.2 Metrics Measurement Frameworks

Software metrics frameworks are used by researchers to aid in selecting metrics at a particular situation. Several frameworks have been proposed in the literature. For instance, Yue and Li [14] proposed an MOF-Based Framework to formally specify a number of quality measurements (completeness, correctness, redundancy) for MOF-based modelling languages. The framework enables definition of metrics at different complexity levels varying from coarse-grained metrics to fine-grained ones thus covering the whole metamodel. The framework defines metrics using a language's metamodel, enabling access to the details of the language's concepts and their relations [14]. The MOF framework enables the selection of metamodel subset that should be examined for each metric. In addition, the framework allows the allocation of weights on the metamodel details for the purpose of emphasizing the significance of these details. The framework has been validated by defining a set of metrics to evaluate UML state machines, class and sequence diagrams. However, it is not adequate since it does not provide a use interface support for defining metrics that are independent of the syntax or semantics of any MOF-based language. In addition, the framework has been used to only generate metrics for UML metamodel and other MOF-based metamodels has not been considered [14].

Macharial et al [15] proposed a metrics measurement framework by the name Metrics-Based Maintainability Estimation Framework for Object-Oriented software (MEFOOS) that assesses systems maintainability. Although MEFOOS has been validated on real life application and shown to be promising, it overlooks important attributes such as code complexity, polymorphism, abstraction and class complexity.

Amara et.al [16] proposed a reliability measurement framework to aid in producing reliable software. The framework includes models, tools, techniques and metrics which are incorporated in all phases of software development life cycle (SDLC). The framework enhances the measurement of reliability and minimize the cost and effort needed for corrections and improvements. The reliability measurement framework is not adequate since it has not been evaluated and validated in real applications [16].

Tempero & Ralph [17] proposed a framework for defining coupling metrics. The framework is language independent and aids in finding out the extent to which two or more coupling metrics measure the same thing. The framework enables unambiguous definitions of coupling metrics and comparable metric definitions. In addition, the framework resolves issues due to incomplete metric definitions, such as different language features [17]. The framework has been tested on its applicability of defining existing coupling metrics such as Chidamber and Kemerer's CBO. Even so, the framework is specific for coupling. In addition, not all non-coupling metrics can be defined naturally e.g. size, metrics do not fit.

Deraman et al. [18] proposed a software ageing measurement framework. The framework adopts the basic GQM structure to determine the various issues affecting the software ageing process. Also, it represents various objectives of measurement and list all the possible measurable metrics that could be captured from the real environment [8]. The framework is limited since it has not been evaluated and validated in real applications.

Basili et al. [12] GQM+ Strategies framework. The GQM+ Strategies is an extension of the GQM approach. It provides mechanisms for explicitly linking software measurement goals to higher-level goals for the software organization, and further to goals and strategies at the level of the entire business [12]. The entire model provides an organization with a mechanism not only to define measurement consistent with larger, upper-level organizational concerns, but also to interpret and roll up the resulting measurement data at each level. The framework is limited since it does not provide a guide that a measurement program responsible could take to communicate and elicit information from relevant stakeholders beyond the presentation of the definition and concepts, as well as the final result [19]. In addition, the framework lacks tool support that are important to make the approach more practicable and utilizable [20].

3. PROPOSED FRAMEWORK

The perspective-based complexity framework is proposed as follows, (Figure 1). The framework identifies different measurable perspectives that behavioral diagrams display during execution of behavior. Three types of perspectives were identified including, element, control flow and interaction. In addition, the framework classifies the identified perspectives into different categories. For instance, the element perspective is subdivided into measurable attributes such as action state, state and messages. The control flow is subdivided into sequential, decision, repetitive and parallel while interaction perspective is subdivided into incoming interaction and outgoing interaction.

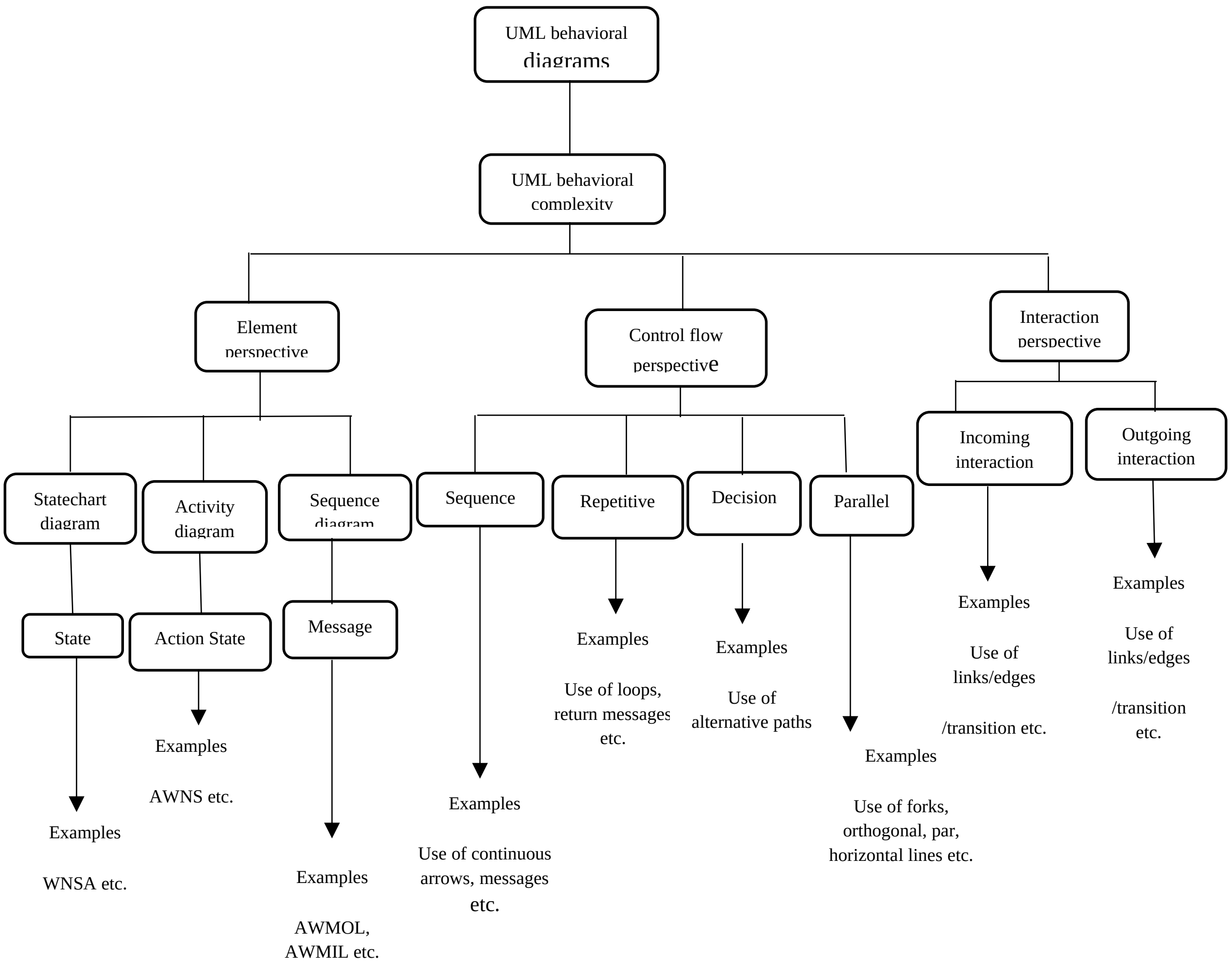
UML behavioral diagrams
UML behavioral complexity
Element perspective
Control flow perspective
Interaction perspective
Statechart diagram
Activity diagram
Sequence diagram
Sequence
Repetitive
Decision
Parallel
Incoming interaction
Outgoing interaction
State
Action State
Message
Examples WNSA etc.
Examples AWNS etc.
Examples AWMOL, AWMIL etc.
Examples Use of continuous arrows, messages etc.
Examples Use of loops, return messages etc.
Examples Use of alternative paths
Examples Use of forks, orthogonal, par, horizontal lines etc.
Examples Use of links/edges /transition etc.
Examples Use of links/edges /transition etc.

Fig 1: A perspective-based complexity framework for UML behavioral diagrams

3.1 Element Perspective

The structural element perspective is based on the building elements of the behavioral diagram. Each diagram has unique elements that compose it. When the size of this elements increase, the complexity of these diagrams increase. For example, the building elements of a statechart diagram are a state, event and a transition. A state depicts a situation where the object satisfies some condition, performs some activity, or waits for some event. A state is represented using a rounded rectangle. A transition connects two states and is represented by an arrow. Events cause transitions of states in state machines. Events can be illustrated externally by transitions and are written as text strings. Figure 2 shows elements a statechart diagram.

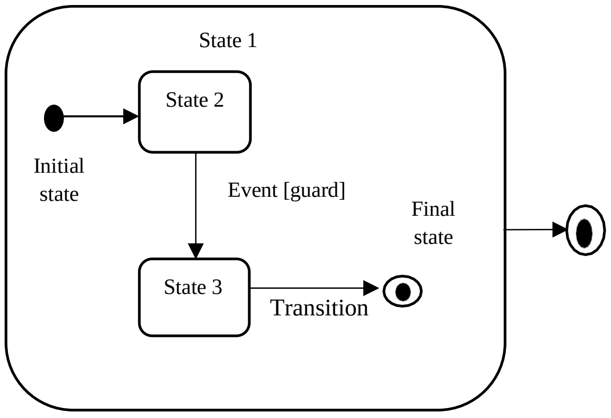

Fig 2: Elements of a statechart diagram

A sequence diagram is made up of a group of objects and messages. Objects are represented by lifelines while messages are represented by arrows among the objects. Messages show an association among the objects. Figure 3 shows a sequence diagram. A vertical rectangle represents a lifeline and arrows represent messages.

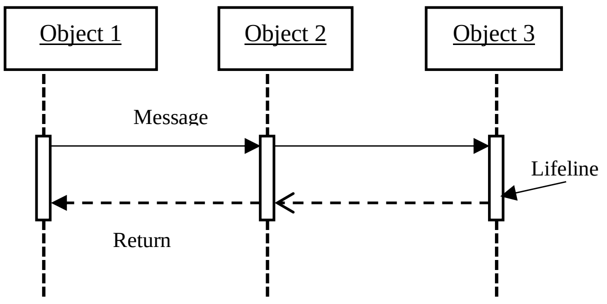

Fig 3: Elements of a sequence diagram

An activity diagram is a flowchart that illustrates the flow from one activity to another. An activity diagram is composed of an action state, edge, initial and final state. Action state represent the behavior of an object while an activity edge is a connection between two action states. Figure 4 illustrates the building elements of an activity diagram.

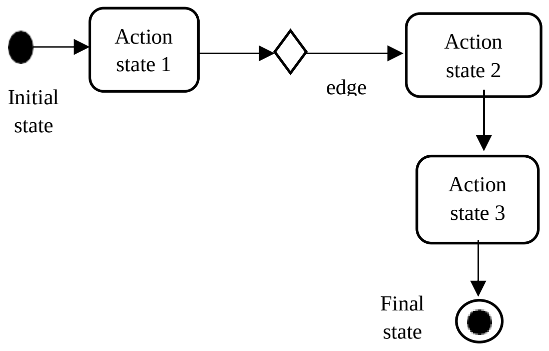

Fig 4: Elements of an activity diagram

3.2 Control Flow Perspective

Control flow in a software is the order in which instructions are executed. The control flow perspective is the behavior flow from one object to another. The perspective borrows from the traditional aspect of control flow structure of a software. They include: Sequential control flow: This control flow represents execution of behavior one after the another e.g. execution of signals one after the other; Decision control flow: It analyses the types of alternative paths that a system follows when executing behavior; Repetitive control flow: It is based on the analysis of how a system repeats a certain behavior a number of times; Parallel control flow: It is based on the analysis of the activities that happen simultaneously during execution of behavior. Figure 5 shows the different types of control structures in behavioral diagrams.

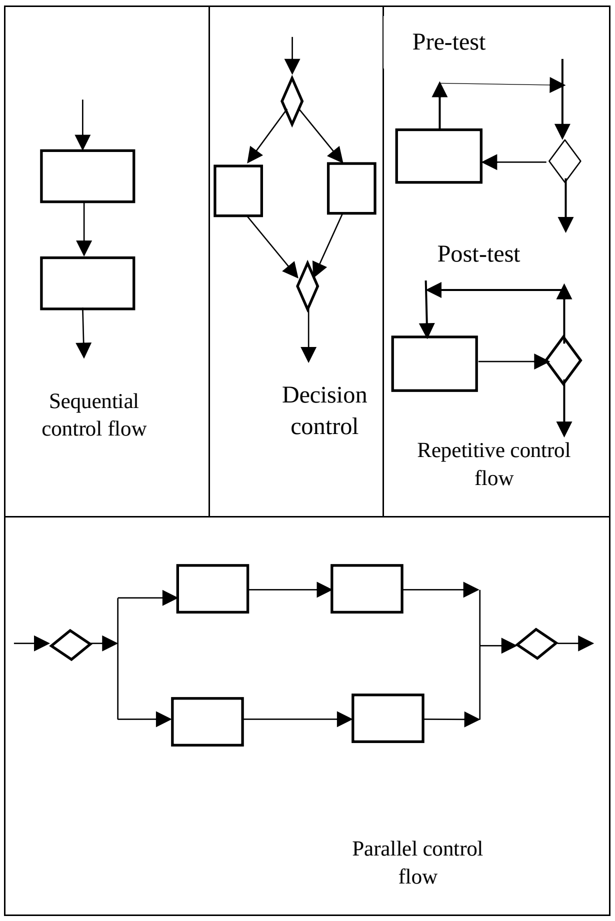

Fig 5: Control flows in behavioral diagrams

3.3 Interaction Perspective

Interaction occurs when an element/ object such as state, an action state or a lifeline of a UML behavioral diagram communicates to another element/ object. Interaction is illustrated by use of edges, links, transitions or messages which are elements of different behavioral diagrams. An object/ element with the highest number of links, messages, transitions or edges is said to interact more. High interaction of an object is associated with more complexity. This perspective can be further subdivided into; Incoming interaction which is based on the number of incoming edges, links, transitions or messages to an element/ object and outgoing interaction. It is based on the number of outgoing edges, links, transitions or messages from an element/object.

For instance, In Figure 6 below, the state named Checking has 3 outgoing transition, the waiting state has 2 outgoing transitions and the dispatching state has 1 transition. Therefore, the checking state interacts more than other states thus contributing to more complexity.

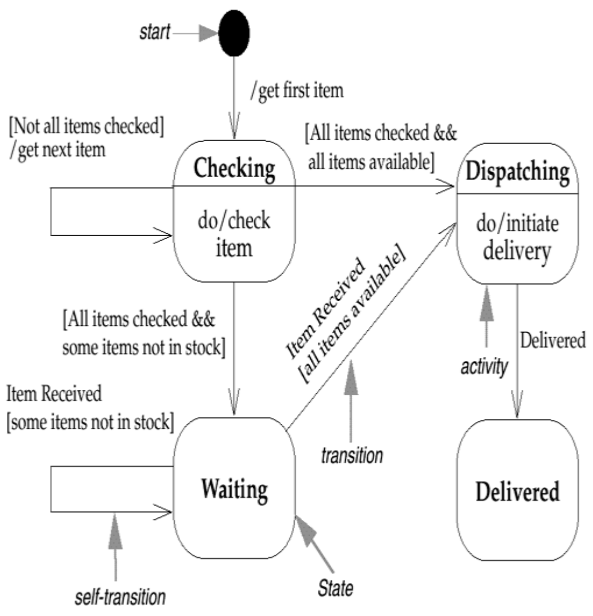

Fig 6: Order processing statechart diagram

4. METHODOLOGY

The study involved carrying out an expert opinion to establish whether the proposed framework is inclusive and can be adopted by software engineers for practice as a complexity taxonomy for UML behavioral diagrams.

Research design: The study employed a mixed method approach namely, qualitative and quantitative approaches for the evaluation of the framework.

Population: To validate the perspective-based complexity framework, the target population selected for this research were industry experts in UML modeling within Kenya.

Sampling strategy and sample size: The research employed purposive sampling method to get a sample size of 11 UML experts for validation of the framework. Purposive sampling was employed because the researcher was only interested with UML modeling experts.

Pilot study: The expert opinion survey questionnaire was pretested by involving 5 UML experts. The pretest was carried out to help the researcher improve the questionnaire's reliability. Feedback from the pilot study was used to restructure the questionnaire before the final study.

Data collection instrument: The data collection tool used for the study was a structured questionnaire for the expert opinion survey. The questionnaire used a five-point Likert scale with a view to uninformed opinions. The questionnaire focused on aspects namely, reliability, inclusivity, and adoptability.

Reliability of the research instrument: To ensure the validity of the research instrument, a pretesting was done using Cronbach's alpha reliability test. In addition, pilot study responses were analyzed and the required modifications were made on the questionnaire to improve its validity.

Data analysis: Descriptive statistics was applied to analyze the data collected included frequencies. Mean and standard deviation. Frequencies were used to analyze the distribution of expert responses while the mean calculated the overall opinions of the experts. The standard deviation measured the variations, ensuring consistency in expert opinions.

5. RESULTS

5.1 Reliability of the Research Instrument

Reliability of the questionnaire was carried out on the relevance and inclusiveness of the framework to ensure consistent results are achieved when different persons are using the same instrument. As shown in Table 1, the inclusiveness of the element perspective obtained a Cronbach's alpha of 0.794, inclusiveness of control flow complexity achieved a Cronbach's alpha of 0.714 while interaction perspective obtained a Cronbach's alpha of 1.0. Therefore, the instrument is reliable since it exceeded the considered threshold of 0.7 [19, 20].

Table 1. Framework inclusiveness reliability statistics

Scale	Cronbach's Alpha
Inclusiveness of the element perspective	0.794
Inclusiveness of the control flow perspective	0.714
Inclusiveness of the interaction perspective	1.0

5.2 Results from the Questionnaire

Data from the respondents was received and checked for completeness. All the questionnaires were duly filled. For that reason, all were accepted for data analysis.

First, the researcher inquired the characteristics of the respondents such as the level of education, number of years served in the software industry and the level of knowledge in modeling with UML diagrams.

Respondents were asked to state their highest level of academic qualification. The results show that 8 (72.7 %) of the respondents had attained a bachelor's degree while the remaining 3 (27.3 %) had attained a master's degree. The results indicate that all the UML experts involved in this study had attained a bachelor's degree and therefore they could study the provided framework and respond appropriately as illustrated in Figure 7.

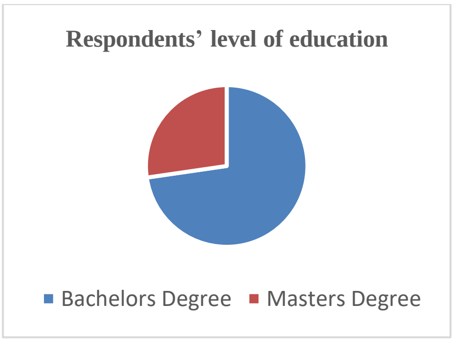

Fig 7: Level of Education

The researcher also established the number of years the respondents had served in the software industry. Results reveal that 2 (18.2 %) respondents had served for 4-5 years, 3 (27.3

%) had served for 6-7 years while 6 (54.5%) had served for 8 and above years. The results reveal that the respondents had experience in the software engineering field thus considered experts. Figure 8 shows this information.

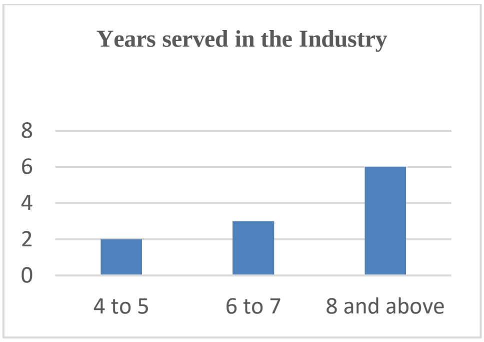

Fig 8: Years served in the industry

The level of knowledge in modeling with UML diagrams was also established as shown in Table 2. (9.1 %) respondent had moderate knowledge in UML modeling, 7 (63.6 %) had high knowledge in UML while 3 (27.3 %) had extremely high knowledge in modeling with UML. Analysis suggest that the responded can be trusted in the validation of the framework.

Table 2. Knowledge in UML modeling

Knowledge in UML Modeling	Frequency	Percent (%)
Moderate	1	9.1
High	7	63.6
Extremely High	3	27.3

The researcher investigated whether the developed perspective-based framework was relevant for the software industry experts in analyzing the complexity of UML behavioral diagrams. Analysis reveal that the framework is relevant with a mean of 4.55 which lies between agree and strongly agree. Also, the standard deviation 0.688 which is less than 1 indicates that the respondents did not differ from one another. These information is represented in Table 3.

Table 3. Relevance of the perspective based framework

	Relevance of the framework
Mean	4.55
Standard deviation	0.688

Also, Table 4 shows that the majority (63.6%) respondents strongly agree that the framework is relevant implying that the proposed framework is useful in analyzing the complexity of UML behavioral diagrams.

Table 4. Relevance of the framework in percentage

Likert Scale	Frequency	Percent (%)
Neither agree	1	9.1
Agree	3	27.3
Strongly agree	7	63.6

Findings indicate that the element perspective is inclusive. The state attribute obtained a mean of 4.45 and a standard deviation of 0.522, the action state attribute had a mean of 4.45 and a standard deviation of 0.522 while the message attribute had a mean of 4.27 and a standard deviation of 0.65. The mean values of the element perspective ranges between agree and strongly agree while the standard deviations of the element perspective indicate that the responds opinions did not differ. These information is represented in Table 5.

Table 5. Inclusiveness of the element perspective

Element Attributes	Mean	Standard deviation
State	4.45	0.522
Action state	4.45	0.522
Message	4.27	0.65

Table 6 shows the percentage of respondents based on their level of rating on the inclusiveness of the element perspective. Majority of the respondents (54.4 %) agree and (45.5 %) strongly agree that the state and action state attributes contribute to complexity of these diagrams while 54.5 % respondents and 36.4% agree and strongly agree respectively on the inclusiveness of the message attribute. Therefore, the element perspective can be relied on analyzing the complexity of UML behavioral diagrams

Table 6. Rating of the inclusiveness of element perspective

	State		Action state		Message	
Likert Scale	Frequency	Percent (%)	Frequency	Percent (%)	Frequency	Percent (%)
Neither agree	___	___	___	___	1	9.1
Agree	6	54.4	6	54.4	6	54.5
Strongly agree	5	45.5	5	45.5	4	36.4

Analysis of the control flow perspective indicated that the perspective is inclusive. The sequence obtained a mean of 4.55, decision a mean 0f 4.64 while the repetitive and parallel obtained a mean of 4.73 and 4.64 respectively. The standard deviations computed were 0.69 for the sequence attribute, 0.67 for decision, and 0.65 for repetitive attribute while the parallel attribute obtained a standard deviation of 0.50 as shown in Table 7.

Table 7. Inclusiveness of the control-flow perspective

Control-flow Attributes	Mean	Standard deviation
Sequence	4.55	0.69
Decision	4.64	0.67
Parallel	4.73	0.65
Repetitive	4.64	0.50

In addition, Table 8 shows that the majority (72.7 %) respondents strongly agree that the sequence and decision control flow contribute to complexity of behavioral diagrams. 81.8 % and 63.6 % of the respondents strongly agree on the inclusiveness of the repetitive and parallel control flow respectively implying that the control flow can be relied on in analyzing the complexity of UML behavioral diagrams.

Table 8. Rating of the inclusiveness of element perspective

	Sequence		Decision		Repetitive		Repetitive	
Likert Scale	**Frequency**	**Percent (%)**	**Frequency**	**Percent (%)**	**Frequency**	**Percent (%)**	**Frequency**	**Percent (%)**
Neither agree	1	9.1	1	9.1	1	9.1	___	___
Agree	3	18.2	2	18.2	1	9.1	4	36.4
Strongly agree	7	72.7	8	72.7	9	81.8	7	63.6

In addition, the analysis of interaction perspective reveals that the interaction perspective contributes to complexity of UML behavioral diagrams. The incoming interaction obtained a mean of 4.09 and a standard deviation of 0.7 while the outgoing interaction obtained a mean of 4.09 and a standard deviation of 0.7 as shown in Table 9.

Table 9. Inclusiveness of the interaction perspective

Interaction Attributes	Mean	Standard deviation
Incoming interaction	4.09	0.700
Outgoing interaction	4.09	0.700

Table 10 shows the percentage of respondents based on their level of rating on the inclusiveness of the interaction perspective. 54.5 % of the respondents agree while 27.3 % of the respondents strongly agree that the incoming and outgoing interaction contribute to complexity of UML behavioral diagrams.

Table 10. Rating of the inclusiveness of interaction perspective

	Incoming interaction		Outgoing interaction	
Likert Scale	**Frequency**	**Percent (%)**	**Frequency**	**Percent (%)**
Neither agree	2	18.2	2	18.2
Agree	6	54.5	6	54.5
Strongly agree	3	27.3	3	27.3

Finally, the respondents were asked to rank the extent to which they agree the perspective based framework can be adopted for practice as a complexity taxonomy for UML behavioral diagrams. Analysis indicate that the framework can be adopted for practice with a mean of 4.55 and a standard deviation of 0.69 as shown in Table 11.

Table 11. Adoption of the perspective based framework

	Relevance of the Framework
Mean	4.55
Standard deviation	0.69

Also, Table 12 shows that the majority (63.6%) respondents strongly agree that the proposed framework can be adopted by software engineers for practice as a complexity taxonomy for UML behavioral diagrams.

Table 12. Rating on the adoption of the framework

Likert Scale	Frequency	Percent (%)
Neither agree	1	9.1
Agree	3	27.3
Strongly agree	7	63.6

6. DISCUSSION

63.6% of the respondents strongly agree respondents that the framework is relevant, 27.3 % agreed with this statement. This means that the framework is appropriate in analyzing the complexity of UML behavioral diagrams. Also, majority (54.4%) of the respondents agreed that the state, action state and message contribute to complexity of statechart, activity and sequence diagram respectively. This means that the element perspective attributes cause complexity in UML behavioral diagrams and should not be overlooked.

It is noted that majority 72.7 % strongly agree that the sequence and decision control flows contribute to complexity of behavioral diagram while 81.1 % and 63.6 % strongly agree that the repetitive and parallel control flows cause complexity of behavioral diagrams. This implies that the control flow perspective can be relied on in analyzing the complexity of UML behavioral diagrams.

Analysis also indicate that 54.4 % of the respondents strongly agree that the interaction perspective is well represented while

27.3 % respondents agreed with the inclusiveness of outgoing and incoming interaction. Further, on the issue of adoptability of the framework, 63.6 % strongly agree that framework can be adopted by software modelers, 27.3% agreed with this statement. This implies that the proposed framework can be relied on as a taxonomy for classifying the complexity of UML behavioral diagrams.

7. CONCLUSION AND FUTURE WORKS

In this study, a new perspective based complexity framework was proposed to analyze the complexity of UML behavioral diagrams. The identified perspectives were element, control flow and interaction. The perspectives were further subdivided into measurable attributes. The element perspective was categorized into state, action state and message. The control-flow perspective was subdivided into sequence, decision, repetitive and parallel while the interaction was categorized into incoming and outgoing interaction. The proposed framework was validated through an expert's opinion survey and the respondents agree that the framework is relevant and inclusive hence it can be adopted for practice as a taxonomy of complexity for UML behavioral diagrams.

This study proposes an extension of the perspective-based framework to include other behavioral diagrams not captured under the element perspective such as use case, interaction, and timing diagrams. Future work should incorporate these diagrams and identify attributes relevant to their structure.

Although the framework has been validated through an expert opinion survey, future work should focus on further validation with real-life software engineering scenarios to test its practical adaptability.

8. REFERENCES

[1] Alshayeb, M., Mumtaz, H., Mahmood, S., & Niazi, M. (2020). Improving the security of UML sequence diagram using genetic algorithm. IEEE Access, 8, 62738-62761.

[2] Estivill-Castro, V., & Hexel, R. (2019, February). The Understandability of Models for Behaviour. In *International Conference on Model-Driven Engineering and Software Development* (pp. 50-75). Springer, Cham.

[3] 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), 143-153.

[4] Shailesh, T., Nayak, A., & Prasad, D. (2022). Transformation of sequence diagram to timed Petri net using Atlas Transformation Language metamodel approach. Journal of Software: Evolution and Process, 34(1), e2412.

[5] Aloysius, A., & Arockiam, L. (2012). Coupling complexity metric: A cognitive approach. *International Journal of Information Technology and Computer Science (IJITCS)*, *4*(9), 29-35.

[6] Unterkalmsteiner, M., Gorschek, T., Islam, A. M., Cheng, C. K., Permadi, R. B., & Feldt, R. (2011). Evaluation and measurement of software process improvement—a systematic literature review. *IEEE Transactions on Software Engineering*, 38(2), 398-424.

[7] Gencel, C., Petersen, K., Mughal, A. A., & Iqbal, M. I. (2013). A decision support framework for metrics selection in goal-based measurement programs: GQM-DSFMS. *Journal of Systems and Software*, 86(12), 3091-3108.

[8] Hatzivasilis, G., Papaefstathiou, I., & Manifavas, C. (2016). Software security, privacy, and dependability: Metrics and measurement. *IEEE Software*, 33(4), 46-54.

[9] Fenton, N., & Bieman, J. (2014). Software metrics: a rigorous and practical approach. *CRC press.*

[10] Nilsson, A.T and Rise, J.L. (1996). "Performance Measurements, Procedure to Design Measures – Goal Attribute Measure (GAM)", Ericsson Quality Institute, LME/Q-93:332, Rev.E, (Internal Publication).

[11] Martinsons, M., Davison, R., & Tse, D. (1999). The balanced scorecard: a foundation for the strategic management of information systems. *Decision support systems*, 25(1), 71-88.

[12] Basili, V., Heidrich, J., Lindvall, M., Münch, J., Regardie, M., Rombach, D., ... & Trendowicz, A. (2014). GQM+ Strategies: A comprehensive methodology for aligning business strategies with software measurement. *arXiv preprint arXiv:1402.0292*.

[13] Fenton N., and Pfleeger, S. L (1997). "Software Metrics: A Rigorous and Practical Approach", 2nd Edition, *IT Publishing Company.*

[14] Yue, T., & Ali, S. (2014, July). A MOF-Based Framework for Defining Metrics to Measure the Quality of Models. *In European Conference on Modelling Foundations and Applications* (pp. 213-229). Springer, Cham.

[15]Macharia, E, "A Metrics-based Framework forEstimating the Maintainability of Object-Oriented Software," *J. Inf. Eng. Appl.*, vol. 9, no. 4, pp. 12–24, 2019, doi: 10.7176/jiea/9-4-02.

[16] Amara, D., & Rabai, L. B. A. (2017). Towards a new framework of software reliability measurement based on software metrics. *Procedia Computer Science*, *109*, 725-730.

[17] Tempero, E., & Ralph, P. (2018). A framework for defining coupling metrics. *Science of Computer Programming*, 166, 214-230.

[18] Deraman, A., Yahaya, J. H., Abidin, Z. N. Z., & Ali, N. M. (2014). Software Ageing Measurement Framework Based on GQM Structure. *Journal of Software and Systems Development*, 2014, 1.

[19] Nunnally, J. C. (2008)., *Psychometric theory* (2nd ed.). New York: McGrawHill.

[20] Doğaner, A. (2021). The approaches and expectations of the health sciences students towards artificial intelligence. *Karya Journal of Health Science*, *2*(1), 5-11.