Automated Feature Extraction from UML Images to Measure SOA Size

Samson Wanjala Munialo, Geoffrey Muchiri Muketha, Kelvin Kabeti Omieno

Abstract*—Enormous development has been experiences in the field of text and image extraction and classification. This is due to large amount of image data that is generated as a result of document sharing for collaborative software development and electronic storage of design documents. One of the recent technique for analyzing large dataset and discover underlying patterns is Deep learning technique. Deep learning is a branch of Machine learning inspired by human brain functionality for the purpose of analyzing unstructured data including images, sound and text. Unified Model Language (UML) is an architectural design which provides developers with a view of software components and scope. UML contain texts and notations which are mostly analyzed and interpreted manually for the purpose of system implementation and scope or size measurement. Consequently, manual processing of electronic design artifacts is prone to bias, errors and time consuming. Various researchers have attempted to automate the process of reading and interpreting design artifacts but still there is a challenge due to varying style of designing these artifacts. This study propose an automatic tool based on existing deep learning algorithms including ResNet50 CNN to read UML interface and sequence diagrams images to detect UML arrows, EAST test detector to detect text, Tesseract OCR with Long Short-Term Memory (LSTM) to recognize text and Multi-class Support Vector Machine to classify text for the purpose of measuring Service Oriented Architecture size. We subjected the tool to accuracy tests which returned encouraging results.

Keywords — Unified Modeling Language, Machine Learning, Deep Learning, image classification, text extraction.

I. INTRODUCTION

Digital image processing has become an important field of research due to vast amount of digital documents and images available currently in databases. Due to large volumes of digital information, human capacity to interpret digital documents and images is challenged requiring automation to capture details more efficiently and effectively. Consequently, researchers have made great contribution in this area by introducing techniques that have improved the accuracy and speed of extracting information from digital content [1].

Revised Manuscript Received on July 30, 2020.

***** Correspondence Author

Samson Wanjala Munialo*, Department of Information Technology, Meru University of Science and Technology, Meru, Kenya. Email: sammunialo@gmail.com

Geoffrey Muchiri Muketha, Department of Computer Science, Murang'a University of Technology, Murang'a, Kenya. Email: gimuchiri@gmail.com

Kelvin Kabeti Omieno, Department of Information Technology and Informatics, Kaimosi Friends University College, Kaimosi, Kenya. Email: komieno@kafuco.ac.ke

Design artifacts such as Unified Model Language (UML), Computer Aided Design (CAD) diagrams, maps, civil architectural diagrams and flowcharts are available in digital format which offers opportunity for interpreting these documents automatically based on algorithms such as machine learning techniques. Automated reading and interpretation brings more benefits including enhanced sharing of interpretation results with other applications and other users for further processing [2]. One of the most popular design artifacts used in Software engineering is Unified Modeling Language (UML). UML is a standard general purpose language developed by Object Management Group (OMG) to provide a visual representation of a software system. UML is not only used to model and document software systems but also to enable visualization of software scope or size [3] Various researchers have made use of UML to design and determine the system size[4][3][5] but the process of interpreting UML diagrams is done manually which compromise on interpretation accuracy. Software size measurement is an important activity as it provides the basis for planning and management of software development[4] [6]. With the growing demand for interoperability and agility, organizations are shifting to Application Programming Interface (API) applications such as Service Oriented Architecture (SOA) to adapt to changes in the dynamic business environment [7] [8] [9]. SOA architectural differences as compared to traditional software applications compelled researchers to introduce COSMIC-SOA [10] and SOA – Size Metrics (SOASM) [3] specifically to measure size of SOA applications. SOASM is one of the metrics that utilizes Unified Modeling Language (UML) diagram to identify attributes that contributes to SOA size [3]. Furthermore, with systems designers and developers working in collaboration at different sites they require sharing of design artifacts such as UML and CAD as images to enable collaborative design. It is from these backdrops that necessitate the need to automate the process of interpreting design artifacts to reduce human interpretation which is biased, less accurate and time consuming. Researchers have proposed automated tools to interpret design artifacts to improve on artifacts interpretation accuracy and time. Most Object Oriented Programming languages IDE have incorporated automatic conversion from UML to XML for further processing [11]. However, they are not able to read, interpret and convert UML from image files. Secondly, conversion to XML only focuses on UML implementation to a specific programming language and thus it cannot interpret UML for other purpose such as software size measurement. Karasneh & Chadron

[11] introduced Img2UML tool to extract UML class diagram from an image into an XML file for further processing by StarUML case tool. The approach allows users to upload UML images, detect rectangles that host class names and attributes based on geometrical detection. The tool also detects existence of relationships among classes and used Microsoft Office Document Imaging (MODI) OCR to extract class names and attribute names. They validated the tool's accuracy and capability to handle large classes of UML images. However, varied styles of representing UML diagrams by various UML designers, offers a challenge to tools that are not supported by machine learning techniques. Furthermore, the tool was not able to identify types of relationship among classes. Intwala et al. [2] proposed a multi-level thresholding geometrical tool to read and interpret CAD images. The tool allowed input of images which were converted to grayscale and eventually to binary images based on OTSU thresholding. They applied Black Top Hat morphology and White Top Hat morphological operations, to capture solid arrows and line based arrows respectively. The tool made use of contour detection feature to capture enclosed areas and they used area checks to detect arrows. They tested the tool with different images which returned promising results. However, geometrical morphological, contour and area check detection are challenged when arrow shapes, style and area vary from the training arrow. For example line based arrows are drawn differently in terms of thickness, arrow head shape and how they link to other objects in the diagram which may result to image arrow failing the test of morphological detection, contour and area checks. Problems of geometrical counter and area based detection can be solved by introducing machine learning techniques to take care of different types of shapes, shades, area and colors. Machine learning techniques use algorithms to learn and predict based on training sets [1] [12]. Machine learning is applied in various areas including image classification, text extraction, natural language and any other process that requires prediction and detection. Image classification is the process of extracting features from an image while text extraction is the process of capturing and recognizing text from an image [1][13]. In the past few years, image processing has experienced tremendous progress as a result of vast amount of digital image and development in machine learning techniques [1]. Most common machine learning techniques for image classification include Artificial Neural Network (ANN), Decision Tree, Naïve Bayes [14] [15], Support Vector Machine (SVM) [16] and Convolutional Neural Network (CNN) [17]. So far, CNN is more efficient and provides a better image classification platform as compared to other machine learning techniques. One advantage of CNN is reducing under-fitting and overfitting which give better result. CNN is made up of learnable neurons which are weighted accordingly, trained with various datasets to extract and classify features from an image[1][12][18]. The task of implementing machine learning is far much simpler with the introduction of machine learning frameworks such as TensorFlow released by Google in 2015 and Keras [19] [20]. TensorFlow and Keras are open source machine learning libraries designed for faster and easier implementation with various platforms including C++ and Python[19]. Text extraction from images has also experienced tremendous development in relation to techniques used to extract text and the amount and variety of available images that require text extraction. Text extraction technology has developed from Optical Character Recognition (OCR) that could only extract defined standard characters to currently where machine learning driven OCR that enables extraction of various shapes of characters including hand written [1][21][13]. Today, the most common text extraction implementation is Tesseract-OCR which has an inbuilt deep neural network technique called Long Short-Term Memory (LSTM) to enhance text extraction process. Lastly, Natural Language Processing (NLP) is another area that rely immensely on machine learning to classify text [22]. Text classification is the process of analyzing and categorizing text into defined groups or classes. Classification of text is made more efficient with inbuilt machine learning algorithms in implementation languages such as Python. Machine Learning techniques for NLP include Random Forrest and Support Vector Machine (SVM). This study proposes an automated tool that relies on existing deep learning techniques including ResNet50 CNN to detect and classify images, EAST text detector, Tesseract OCR and Multi-Class SVM to detect, recognize and classify operations names respectively from UML interface diagrams. The remaining part of this paper discusses the proposed solution in detail, results and discusses the findings and conclusion.

II. PROPOSED SOLUTION

This study proposes an automated tool to allow input of UML interface diagram images and sequence diagram images. The tool then uses CNN deep learning image classifier to detect and classify UML arrows which are relevant to compute SOA size. In addition, the automated tool provides a platform based on Tesseract OCR to recognize text contained in UML interface diagram. The tool then classified operations names based on their complexity as stipulated in SOASM [3]. SOASM [3] proposed SOA size metrics grounded on UML interface diagram and sequence diagram which formed the basis of this study which is to automate the process of computing SOA size by capturing UML features. SOASM [3] proposed Service Dependency Count (SDC) and Weighted Operation Count (WOC) extracted from UML service interface diagram and Weighted Message Count (WMC) extracted from UML sequence diagram. The summation of the three metrics was used to compute SOA size. A sample of UML service interface diagram for a Taxi management system is as shown in Fig. 1.

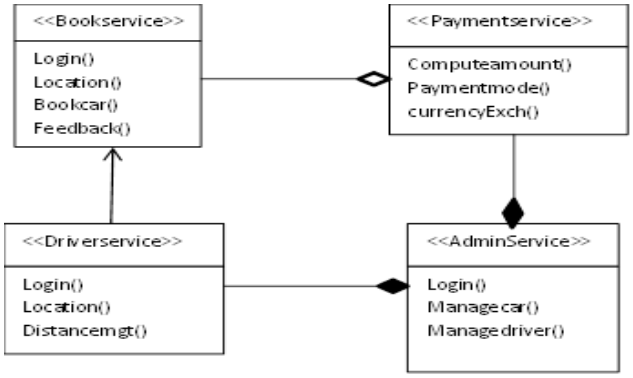

Figure 1: Taxi Service UML interface diagram

The task at hand for our proposed automated tool was to extract and classify text representing operations from UML interface diagram to compute WOC. Secondly, the tool was required to classify types of arrows from UML interface diagram and sequence diagram to compute SDC and WMC respectively. Text extraction and image classification were achieved with the support of existing deep learning algorithms and image processing libraries in the implementation language.

A. *Text Extraction*

Based on WOC SOA size estimation metrics, operation names contained in the lower rectangle in UML interface diagram shown in Fig. 1 formed the basis for computing WOC. The tool was required to extract operation names in the bottom rectangle, classify the names then assign weights to operation names based on their complexity. We employed deep leaning text detection technique to detect text, OCR to extract detected text and natural language processing algorithm to classify operation names.

1) Text detection

Text detection identifies and locates a group of characters without spaces contained in an image. We employed a deep learning technique known as Efficient and Accurate Scene Text detection (EAST) [23] pipeline to detect text contained in UML interface diagrams. EAST is faster, more accurate and capable of localizing text of different shapes including text affected by light and reflection as compared to other text detection methods. EAST text detector makes use of Fully Convolution Network (FCN) model to detect the presence of text in an image. We implemented EAST pipeline in Python supported by OpenCV library. We loaded the UML interface image and EAST algorithm into the system. We formed two layers for feature maps including a layer to give the probability of a region containing text and the second layer to represent the geometry of the image defining coordinates for the text bounding box. The final result was highlighted bounding boxes of text Region of Interest (ROI) contained in UML interface diagram as shown in Fig. 2.

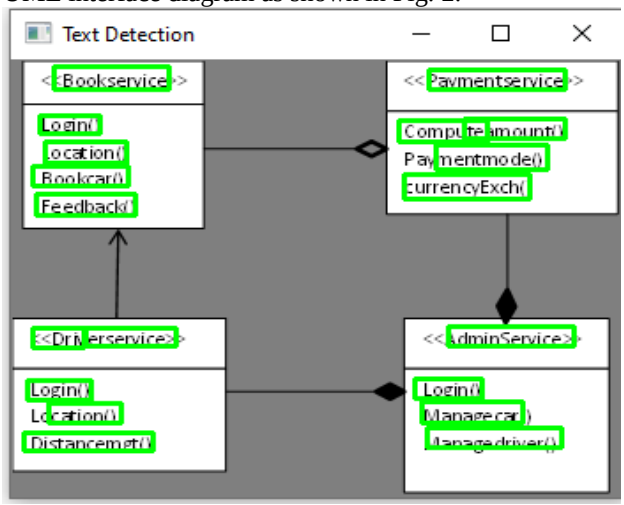

Figure 2: Detected text in UML interface

2) Text recognition

The next step was to recognize detected text ROI by extracting text from images and store in an array. We used Tesseract OCR to read images and text contained in the UML interface diagram. The current version of Tesseract OCR is fitted with Long Short term Memory (LSTM) deep learning algorithm to improve on text recognition accuracy. The process of text recognition was implemented in Python which also provided image processing libraries to enable extracted text to be loaded into as array for text classification. Tesseract OCR captures both service names at the top rectangle and operation names at the bottom rectangle from UML interface diagram. However, to compute WOC, we only needed operation names in our classification and computation. Therefore, the tool separates operations names from service names by use of python 'IF' condition and wildcard characters to consider only text with characters '()' at the end to represent operation names. Fig. 3 shows a snapshot of a section of text extracted by Tesseract-OCR from the sampled UML interface diagram image sorted in ascending order. Both EAST and Tesseract OCR are existing models and therefore they did not require training.

```
RECOGNIZED TEXT
==============
Bookcar()

Distancemeti)

Feedback),

Location()

Login()
```

Figure 3: A section of text recognized from UML

3) Text classification

Weighted Operation Count (WOC) defined in [3] classified operations based operation complexity as simple, average and complex. The next task of our proposed tool was to classify operations automatically. The tool uses Multi-class Support Vector Machine (SVM) implemented in Python to analyze and classify text into simple, average and complex operations. The process involved training text classification model, evaluate and execute the model. We used "The Bag of words model" to convert text to numbers for classification. We collected a wide variety of possible operation names to train the classification model. The text classification model was designed based on training dataset of 1200 operation texts. In this case the model was working with three finite classes. We used 100 service operation names to test both the text recognition model and text classification model.

4) WOC Computation

Through an implementation program counter, the tool is able to compute the number of classified operations into simple, average and complex categories. The number of operations in each category is then multiplied by the assigned weights and summed to give the total weighted operation count (WOC). Operations that are not captured from UML interface diagram by EAST detector and Tesseract OCR are not included in the count. Therefore, further verification is required to ascertain inclusion of all operations in the final count. Text extraction from UML interfaces for the purpose of computing WOC includes input image, text detection, and text recognition and text classification as shown in Fig. 4.

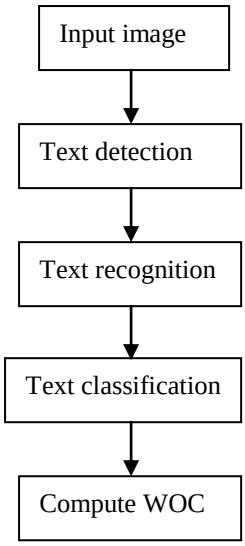

Figure 4: Text extraction process

B. Image classifier

This study used existing ResNet50 CNN[12] to classify UML interface and sequence diagrams images arrows. In UML interface diagram, the type of arrow in Fig. 5 is determined by the type of dependency among services. On the other hand, the type of arrow in a sequence diagram is classified as synchronous, asynchronous and reply message centered on the type of data movement. The objective of ResNet50 CNN in this study was to classify arrows types based on their arrow head shape and dashed line.

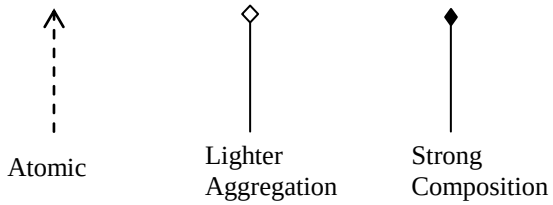

Figure 5: Types of UML composition arrows

First of all, we prepared datasets consisting of 1000 UML arrows for UML interface diagram and 1000 arrows for UML sequence diagram. For both UML interface and UML sequence diagrams we used 900 arrows to train each ResNet50 CNN algorithm and 100 arrows to test each ResNet50 CNN model. Selected arrows for training set and validation were varied accordingly to capture different arrow head shapes, dotted lines types, varied arrow head area and shades. After preparing training datasets and test sets, we supplied ResNet50 CNN with the training dataset, and then we tested the model with test dataset. We used python and openCV to implement ResNet50 which was supported by Tensorflow and Keras. We constructed image pyramid and sliding window in Python to identify and extract arrow images. ROI captured was passed through CNN for classification which returned positive results. UML interface images loaded into the tool were exposed to CNN model which extracted identified arrow types accordingly as shown in Fig. 6.

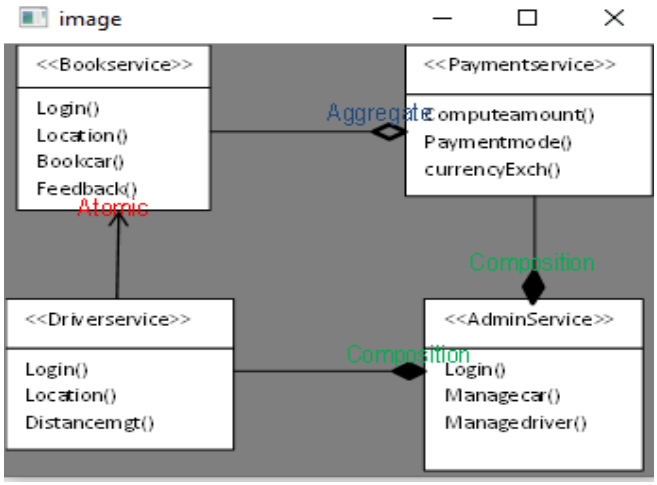

Figure 6: Arrow classification based on dependency

To compute Service Dependency Count (SDC) metric as defined in SOASM[3], we used a loop to count the number of each type of arrow then multiplied with their respective weights and summed to compute SDC. Lastly, we applied the same principle of arrow detection to identify arrow types in UML sequence diagram. Based on SOASM, data movement among services represented in sequence diagram are indicators of SOA size. We exposed UML sequence diagrams to the trained ResNet50 CNN which detected data movement arrows as shown in UML sequence diagram in Fig. 7 indicated by letters a, s and r representing asynchronous, synchronous and reply messages respectively.

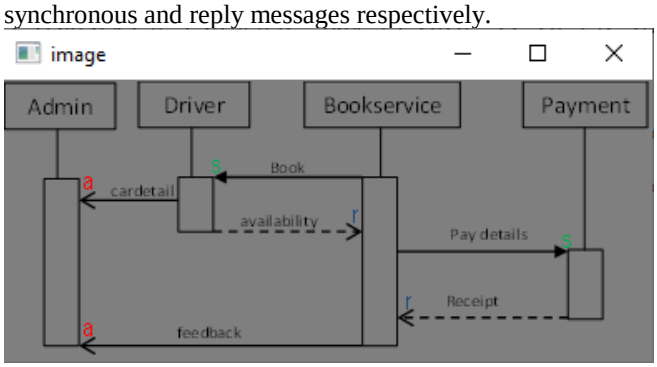

Figure 7: UML sequence diagram arrow classification

Having identified data movement arrows as represented in sequence diagram, the tool is able to compute WMC by counting the number of each arrow type, multiply with the assigned weight for each type then sum the result to calculate total WMC. Lastly, the summation of SDC, WOC and WMC is computed to produce the final result which is SOA size.

III. RESULTS AND DISCUSSION

The success of implementing our proposed tool was determined by the level of accuracy in detecting, extracting and classifying text and arrow types captured from UML interface diagram images and sequence diagram images. First of all, we tested the accuracy of extracting text which involves the process of text recognition. Accuracy was tested against the number of operations contained in a UML diagram as shown in Fig. 8.

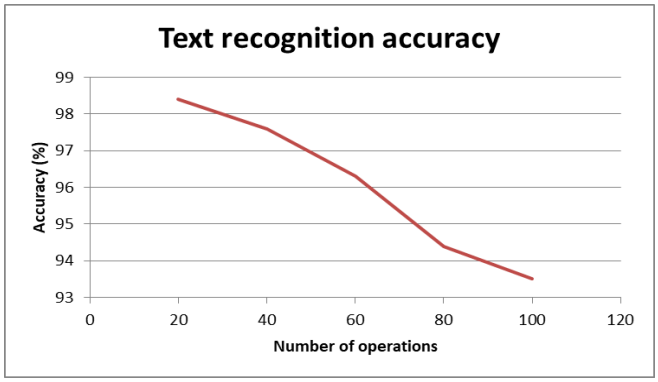

Figure 8: Text recognition accuracy

Text extraction accuracy was affected in instances where spaces appeared within a word causing the text to be recognized as two different words or operations. Another aspect that affected text recognition is images with unclear text appearance. Secondly, operation names classification was tested for accuracy based on correct classification of operation names as simple, average and complex. According to SOASM [3] WOC metric, simple operations include operations with simple algorithm such as add, compute, delete, admit, book and so on. On the other hand average operations include algorithms to sort, search and so on while complex operations are intelligence based operations such as forecasting and predictions algorithms. One main challenge of classifying this type of text is lack of standard for naming operations which sometimes the operation name does not necessarily reflect the underlying algorithm. Lack of standard and consistency in naming operations affected the model's accuracy as shown in Fig. 9.

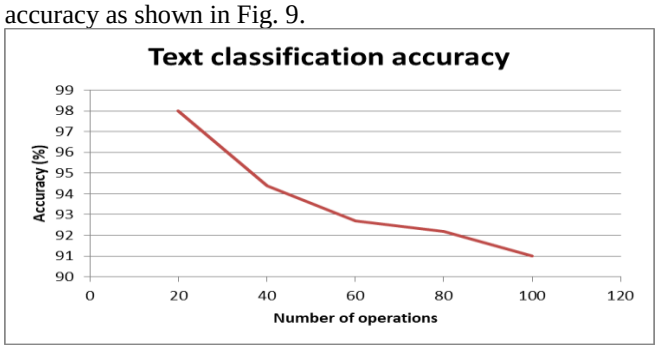

Figure 9: Text classification accuracy

Thirdly, accuracy of classifying arrows in UML interface and sequence diagram was validated. We used different types of arrows as training data to enable the automated tool capture a wide range of arrow types. This included varying arrow head shape and area, use of straight line, curved lines and cornered lines, different types of dashed lines and arrow line shades. The result of validation is as shown in Fig. 10

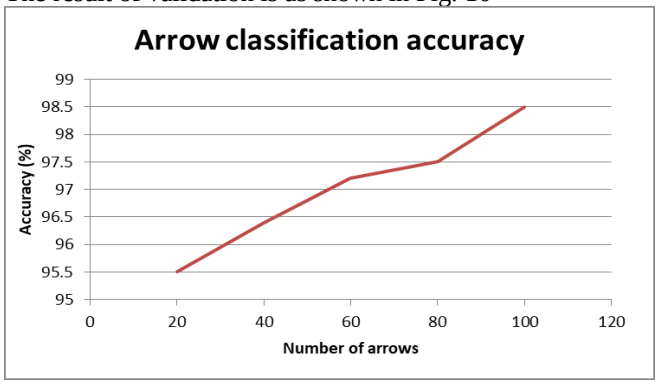

Figure 10: Arrow classification accuracy

Results from analysis show that the tool is accurate and applicable. A summary of our validation results are shown in Table 1.

TABLE- I: MODELS VALIDATION RESULTS

Models	Training dataset	Testing dataset	Average accuracy
EAST detector	-	100	96.4%
Tesseract OCR	-	100	95.8%
Multi-class SVM	1200	100	93.1%
ResNet50 CNN (UML Interface)	900	100	97 %
ResNet50 CNN (UML sequence)	900	100	97.4%

Most instances where the tool was not able to capture text or arrow correctly were due to issues with input image or text. When standard UML diagram and notations were used, the tool recognized and classified text and arrows more accuracy.

IV. CONCLUSION

In this study we proposed an automated tool supported by deep learning techniques to detect and classify service operations and arrows extracted from UML interface diagram and UML sequence diagram. We used EAST deep learning algorithm to detect text, Tesseract OCR with Long Short-Term Memory (LSTM) to recognize text and Multi-Class SVM to classify service interface operations into simple, average and complex. In addition, we classified UML interface and sequence diagram arrows using ResNet50 CNN. We tested the automated tool accuracy with regard to text extraction and image classification and the results were encouraging. The result from this study implies that automatic extraction of text and arrow images from UML diagram images offers a more accurate method of reading and interpreting UML images. Future research is required to automate more design artifacts with are available electronic format.

REFERENCES

1. R. Deepa and K. N. Lalwani, "Image Classification and Text Extraction using Machine Learning," *Proc. 3rd Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2019*, pp. 680–684, 2019, doi: 10.1109/ICECA.2019.8821936.
2. A. M. Intwala, K. Kharade, R. Chaugule, and A. Magikar, "Dimensional arrow detection from CAD drawings," *Indian J. Sci. Technol.*, vol. 9, no. 21, pp. 1–7, 2016, doi: 10.17485/ijst/2016/v9i21/89259.
3. W. S. Munialo, M. G. Muketha, and K. . Omieno, "Size Metrics for Service-Oriented Architecture," *Int. J. Softw. Eng. Appl.*, vol. 10, no. 2, pp. 67–83, 2019.
4. M. Harizi, "The Role of Class Diagram in Estimating Software Size," Int. J. of Comp. Appl.,vol. 44, no. 5, pp. 31–33, 2012.
5. L. Marcos, "Modelling of Service-Oriented Architectures with UML," vol. 194, pp. 23–37, 2008, doi: 10.1016/j.entcs.2008.03.097.
6. G. Albrecht, A., Gaffney, "No Title," *A Softw. Sci. Validation, IEEE Trans Softw. Eng.*, 1983.
7. H. Chindove, L. F. Seymour, and F. I. Van Der Merwe, "Service-oriented Architecture : Describing Benefits from an Organisational and Enterprise Architecture Perspective," vol. 3, no. Iceis, pp. 483–492, 2017, doi: 10.5220/0006383604830492.

8. S. Bilgaiyan, S. Sagnika, S. Mishra, and M. Das, "A systematic review on software cost estimation in Agile Software Development," *Journal of Engineering Science and Technology Review*, vol. 10, no. 4. pp. 51–64, 2017, doi: 10.25103/jestr.104.08.
9. Z. A. Siddiqui and K. Tyagi, "A critical review on effort estimation techniques for service-oriented-architecture-based applications," *Int. J. Comput. Appl.*, vol. 7074, no. October, pp. 1–10, 2016, doi: 10.1080/1206212X.2016.1237132.
10. COSMIC, *Guideline for Sizing SOA Software*. 2015.
11. B. Karasneh and M. R. V Chaudron, "Extracting UML Models from Images," no. March, 2013, doi: 10.1109/CSIT.2013.6588776.
12. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
13. T. Kumuda and L. Basavaraj, "Edge based segmentation approach to extract text from scene images," *Proc. - 7th IEEE Int. Adv. Comput. Conf. IACC 2017*, pp. 706–710, 2017, doi: 10.1109/IACC.2017.0147.
14. P. Dong-Chul, "Image Classification Using Naive Bayes Classfier," *Int. J. Comput. Sci. Electron. Eng.*, vol. 4, no. 3, pp. 2320–4028, 2016, http://www.isaet.org/images/extraimages/P1216004.pdf.
15. S. C. Hsu, I. C. Chen, and C. L. Huang, "Image classification using naive bayes classifier with pairwise local observations," *J. Inf. Sci. Eng.*, vol. 33, no. 5, pp. 1177–1193, 2017, doi: 10.6688/JISE.2017.33.5.5.
16. L. H. Thai, T. S. Hai, and N. T. Thuy, "Image Classification using Support Vector Machine and Artificial Neural Network," *Int. J. Inf. Technol. Comput. Sci.*, vol. 4, no. 5, pp. 32–38, 2012, doi: 10.5815/ijitcs.2012.05.05.
17. F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," *Proc. - 2018 4th IEEE Int. Conf. Res. Comput. Intell. Commun. Networks, ICRCICN 2018*, pp. 122–129, 2018, doi: 10.1109/ICRCICN.2018.8718718.
18. Z. Wu, C. Shen, and A. van den Hengel, "Wider or Deeper: Revisiting the ResNet Model for Visual Recognition," *Pattern Recognit.*, vol. 90, pp. 119–133, 2019, doi: 10.1016/j.patcog.2019.01.006.
19. [19] K. R. Tripathi and R. Kumar, "Image Classification using small convolutional Neural Network," *IEEE*, pp. 483–487, 2019, doi: 978-1-5386-5933-5/19.
20. K. Chauhan and S. Ram, "Image Classification with Deep Learning and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras," *Int. J. Adv. Eng. Res. Dev.*, 2018.
21. C. Patel, A. Patel, and D. Patel, "Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study," *Int. J. Comput. Appl.*, vol. 55, no. 10, pp. 50–56, 2012, doi: 10.5120/8794-2784.
22. K. Kowsari, D. E. Brown, M. Heidarysafa, K. J. Meimandi, M. S. Gerber, and L. E. Barnes, "HDLTex : Hierarchical Deep Learning for Text , Classification,", IEEE, 2017, doi: 10.1109/ICMLA.2017.0-134.
23. X. Zhou *et al.*, "EAST : An Efficient and Accurate Scene Text Detector," IEEE, pp. 5551–5560, 2015.

AUTHORS PROFILE

Samson Wanjala Munialo is an assistant lecturer in Meru University of Science and Technology, Kenya. He has BED. Degree from Catholic University of Eastern Africa, Kenya, MSc Information Technology Management from University of Sunderland, UK and currently he is pursuing his PHD Information Technology at Masinde Muliro University of Science and Technology. His area of research includes software metrics, Machine Learning, software effort estimation and IT project management.

Geoffrey Muchiri Muketha received the BSc degree in Information Science from Moi University in 1995, the MSc degree in Computer Science from Periyar University in 2004, and the PhD degree in Software Engineering from Universiti Putra Malaysia in 2011. He is Associate Professor and Dean of the School of Computing and Information Technology at Murang'a University of Technology, where he has taught and supervised both undergraduate and postgraduate students for many years. 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).

Kelvin Omieno is a Senior Lecturer in the Department of Information Technology and Informatics, School of Computing and Information Technology, Kaimosi Friends University College, Kenya. He holds a PhD in Information Systems of Jaramogi University of Science and Technology, MSc in Information Technology and BSc in Computer Science. His research interests are in ICT4D, eLearning, Internet of Things, Software metrics and Health Informatics. He is a professional member of Association of Computing Machinery (ACM).