TABLE I
USERS' PERCEPTION OF THE PROPOSED TECHNIQUE

Questions	Mean
Q1. Has it been difficult for you to understand the source code?	4.15
Q2. Has it become easier for you to understand open-source code using this tool?	4.3
Q3. Do you find it appropriate to use media for learning?	4.9
Q4. Do you find it appropriate to use media to learn source code and programming?	4.6
Q5. Has the use of media in this tool helped you better understand the source code?	4.4
Q6. Do you find it appropriate to use comments outside the source code as additional comments?	4
Q7. Has the additional comment in the tool helped to better understand the source code?	4.3
Q8. Do you see the technique used in this tool as an appropriate way to better understand the source code?	4.3
The average score of questions	4.35

TABLE II
USERS' SATISFACTION WITH THE TOOL

	Questions	Mean
Q9.	Has working with tools been easy for you?	4.35
Q10.	Do you believe it can be used without special knowledge about the tool?	4.55
Q11.	Do you want to work on the tool in newer versions?	3.95
Q12.	Do you recommend this tool to your friends?	4.15
Q13.	Was the order of the tool options in the proper order?	4.3
The average score of questions		4.26

TABLE III
THE EFFECTIVENESS OF THE TECHNIQUE

	Questions	Mean
Q14.	Do you want to check the source code of the MINIX operating system?	3.05
Q15.	Do you find it difficult to understand the source code of the MINIX operating system?	4.3
Q16.	Did using the tool on the source code of the MINIX operating system lead to a better understanding of it?	4.2
Q17.	Has the technique of adding multimedia helped better to understand the source code of the MINIX operating system?	4.3
Q18.	Has the extra comment technique helped better to understand the source code of the MINIX operating system?	4.15
The average score of questions		4.0

IV. CONCLUSION

Program understanding is one of the most critical tasks in using source code. The recent open-source programs are complex and complicated to understand because they were developed by many programmers using different languages and styles. Techniques that have been developed to understand programs have different strengths and weaknesses. The weaknesses of existing techniques motivate us to introduce a new technique to improve the understanding of open-source software.

Our proposed technique simplifies the understanding of open-source programs by supporting two unique features, i.e., the ability to add multimedia and additional comment to the complex open-source code. These additional features help with a better understanding of the source code. Moreover, the tool we developed also supports multiple programming languages to help users examine source code written in different languages.

The evaluation of the proposed techniques shows that the users have a positive perception because they agree that the technique is better at assisting them to understand the program. They also agree that it is easy to use. The tool's multimedia support and extra comment significantly improve user understanding of the source code. This proposed technique can be used via GitHub and design proper plugins for IDEs, such as Eclipse or IntelliJ IDEA. Users who access the source code from GitHub receive the multimedia and supplementary comment assigned to it.

The proposed software understanding tool currently supports five media types: video, audio, image, PDF, and PowerPoint. This tool considers a wide range of available media and their unique use. Each media can be used to improve source code understanding. In its current form, the software tool suffers several limitations. One of them is the lack of intelligence to some understanding process to be carried out automatically. For future research, we would like to explore the possibility of using some machine learning algorithms that can help enhance the program understanding process.

ACKNOWLEDGMENT

We thank Universiti Kebangsaan Malaysia for supporting this work under the GGPM-2020-026 research grant fund.

REFERENCES

[1] S. Butler *et al.*, "On Company Contributions to Community Open Source Software Projects," *IEEE Trans. Softw. Eng.*, vol. 47, no. 7, 2021.

[2] A. Khandelwal, "Impact of Open Source Software in Research," 2020.

[3] A. Azlen, M. Nordin, R. Latih, and N. M. Ali, "Using SaaS to Enhance Productivity for Software Developers: A Systematic Literature Review," *J. Theor. Appl. Inf. Technol.*, vol. 31, p. 24, 2020.

[4] Sumandeep Kaur, "Issues in Open-Source Software ," *Int. J. Comput. Sci. Commun.*, vol. 11, no. 2, pp. 47–51, 2020.

[5] G. M. Kapitsaki, N. D. Tselikas, K.-I. D. Kyriakou, and M. Papoutsoglou, "Help me with this: A categorization of open source software problems," *Inf. Softw. Technol.*, vol. 152, p. 107034, Dec. 2022.

[6] A. Mohd Zin, S. Ahmad Aljunid, Z. Shukur, and M. Jan Nordin, "A Knowledge-based Automated Debugger in Learning System," 2000.

[7] O. Levy and D. G. Feitelson, "Understanding large-scale software systems – structure and flows," *Empir. Softw. Eng.*, vol. 26, no. 3, p. 48, May 2021.

[8] S. A. Aljunid, Abdullah Mohd Zin, and Zarina Shukur, "A Study on the Program Comprehension and Debugging Processes of Novice Programmers," *J. Softw. Eng.*, vol. 6, no. 1, pp. 1–9, 2012.

[9] M. Hassan, "How do we Help Students 'See the Forest from the Trees?,'" in *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2*, 2022.

[10] Z. Ahsan, U. Obaidellah, and M. Danaee, "Is Self-Rated Confidence a Predictor for Performance in Programming Comprehension Tasks?," *APSIPA Trans. Signal Inf. Process.*, vol. 11, no. 1, 2022.

[11] N. Al Madi and M. Zang, "Would a Rose by any Other Name Smell as Sweet? Examining the Cost of Similarity in Identifier Naming," in *The 33rd Psychology of Programming Interest Group (PPIG 2022)*, 2022.

[12] H. Eicken *et al.*, "Connecting Top-Down and Bottom-Up Approaches in Environmental Observing," *Bioscience*, vol. 71, no. 5, pp. 467–483, May 2021.

[13] S. Letovsky, “Cognitive processes in program comprehension,” *J. Syst. Softw.*, vol. 7, no. 4, pp. 325–339, Dec. 1987.

[14] A. Fekete and Z. Porkoláb, “A comprehensive review on software comprehension models,” *Ann. Math. Informaticae*, vol. 51, pp. 103–111, 2020.

[15] A. A. Shargabi, S. A. Aljunid, M. Annamalai, and A. M. Zin, “Performing Tasks Can Improve Program Comprehension Mental Model of Novice Developers,” in *Proceedings of the 28th International Conference on Program Comprehension*, 2020.

[16] P. Lima, J. Melegati, E. Gomes, N. S. Pereira, E. Guerra, and P. Meirelles, “CADV: A software visualization approach for code annotations distribution,” *Inf. Softw. Technol.*, vol. 154, p. 107089, Feb. 2023.

[17] E. Fregnan, J. Fröhlich, D. Spadini, and A. Bacchelli, “Graph-based visualization of merge requests for code review,” *J. Syst. Softw.*, vol. 195, p. 111506, Jan. 2023.

[18] Stephan Diehl, *Software Visualization - Visualizing the Structure, Behaviour, and Evolution of Software*. 2007.

[19] N. Chotisarn *et al.*, “A systematic literature review of modern software visualization,” *J. Vis.*, vol. 23, no. 4, pp. 539–558, Aug. 2020.

[20] Azila Adnan and Muhamad F B Noor Hassim, “Infographics in Teaching and Learning: An Attention Grabber,” in *International University Carnival on E-Learning (IUCEL) Proceedings 2022*, 2022.

[21] M. Dias, D. Orellana, S. Vidal, L. Merino, and A. Bergel, “Evaluating a Visual Approach for Understanding JavaScript Source Code,” in *Proceedings of the 28th International Conference on Program Comprehension*, 2020.

[22] M. Kargar, A. Isazadeh, and H. Izadkhah, “Improving the modularization quality of heterogeneous multi-programming software systems by unifying structural and semantic concepts,” *J. Supercomput.*, vol. 76, no. 1, pp. 87–121, Jan. 2020.

[23] D. Limberger, W. Scheibel, J. van Dieken, and J. Döllner, “Procedural texture patterns for encoding changes in color in 2.5D treemap visualizations,” *J. Vis.*, Oct. 2022.

[24] L. Bedu, O. Tinh, and F. Petrillo, “A Tertiary Systematic Literature Review on Software Visualization,” in *2019 Working Conference on Software Visualization (VISSOFT)*, pp. 33–44, 2019.

[25] R. Ishizue, K. Sakamoto, H. Washizaki, and Y. Fukazawa, “PVC.js: visualizing C programs on web browsers for novices,” *Heliyon*, vol. 6, no. 4, p. e03806, Apr. 2020.

[26] M. Mladenović, Ž. Žanko, and M. Aglić Čuvić, “The impact of using program visualization techniques on learning basic programming concepts at the K–12 level,” *Comput. Appl. Eng. Educ.*, vol. 29, no. 1, 2021.

[27] M. Altherwi, “An empirical study of programming language effect on open source software development,” in *Proceedings Companion of the 2019 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity*, 2019.

[28] Mohan Krishna Kagita and Li Xiujuan, “Machine Learning Techniques for Multimedia Communications in Business Marketing,” *J. Mult. Log. Soft Comput.* , vol. 36, no. 1, pp. 151–167, 2021.

[29] H. He, “Understanding source code comments at large-scale,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019.

[30] S. Panthaplackel, J. J. Li, M. Gligoric, and R. J. Mooney, “Deep Just-In-Time Inconsistency Detection Between Comments and Source Code,” *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 1, pp. 427–435, May 2021.

[31] X. Song, H. Sun, X. Wang, and J. Yan, “A Survey of Automatic Generation of Source Code Comments: Algorithms and Techniques,” *IEEE Access*, vol. 7, pp. 111411–111428, 2019.

[32] J. Nielsen, J. Lewis, and C. Turner, “Determining Usability Test Sample Size,” in *International Encyclopedia of Ergonomics and Human Factors, Second Edition - 3 Volume Set*, CRC Press, 2006.