A Tool-Centric Framework for Teaching Undergraduate Computer Science Students on Operating System Design

***Sulastri Putit*[1*]*, Lenny Yusrina Bujang Khedif*[2]**
[1]*College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Cawangan Sarawak, Kampus Samarahan, 94300 Kota Samarahan, Sarawak, Malaysia*
[1]*sulastri@uitm.edu.my,*
[2]*College of Computing, Informatics and Mathematics, Universiti Teknologi MARA Cawangan Sarawak, Kampus Samarahan, 94300 Kota Samarahan, Sarawak, Malaysia*
[2]*lennykhedif@uitm.edu.my*
**Corresponding Author*

Received: 04 November 2024
Accepted: 31 December 2024
Date Published Online: 31 January 2025
Published: 31 January 2025

Abstract: *Teaching and learning operating system (OS) design is a core part of computer science education, which is highly demanding for both teaching and learning. This paper presents a tool-centric framework that aims at enhancing teaching operating system design for undergraduate students in computer science. The framework integrates multiple educational tools and methodologies with the aim of making the course more appealing for students while improving their comprehension and hands-on experience in OS design. The proposed framework extends existing educational theories and practices by focusing on aspects such as computational thinking, collaborative learning, and gamification in the learning process. The theoretical basics of this framework, the practical use of such a framework, and the expected influence on students' learning outcome will be further explained in this paper. Empirical evidence, detailed examples and comparisons with existing frameworks underscore its potential impact on student learning outcomes.*

Keywords: *collaborative learning, computational thinking, gamification, operating system design, tool-centric framework*

1.0 INTRODUCTION

Operating systems represent the core of computer functionality; controlling hardware resources and providing essential services for application software. Therein lies such a significant complexity in the design of an OS, ranging from process management to memory management, file systems, security protocols, among others. Teaching such concepts requires multiple approaches in view of the diverse learning styles and backgrounds typical of students. Most traditional pedagogical approaches hardly engage students and deepen their conceptual understanding of such complex concepts. Research indicates that students often struggle with abstract concepts in OS design, leading to a lack of confidence and motivation (Gesing et al., 2022; Zahir, 2024). This paper presents a tool-centric framework that leverages modern educational technologies and methodologies in enriching the teaching and learning of OS design. The framework's novelty lies in its systematic incorporation of computational thinking, collaborative learning, and gamification, offering an engaging and effective learning experience.

2.0 PROBLEM STATEMENT

While traditional approaches often leave students disengaged and struggling with abstract ideas, the increasing complexity of operating systems concepts necessitates innovative teaching strategies that can simplify and clarify core concepts. Research indicates that integrating tools that support collaborative learning can significantly enhance educational outcomes in computer science (Gesing et al., 2022; Zahir, 2024). For instance, Anohah (2016) emphasizes that pedagogical principles must form the foundation for the inclusion of features in learning management systems, which can be adapted to create a more engaging learning environment for OS design. Furthermore, the use of simulation tools allows students to visualize and manipulate OS concepts, making abstract ideas more tangible (Weitl-Harms, 2023). By adopting a tool-centric approach, educators can provide students with access to a variety of resources, including simulation tools, collaborative coding environments, and gamified learning platforms.

3.0 OBJECTIVE

Objective 1: To propose a tool-centric framework for teaching operating system design to undergraduate computer science student.

Objective 2: To explore the theoretical foundations of the framework, its practical usage, and the expected impact on students' learning outcomes

4.0 INTEGRATING COMPUTATIONAL THINKING

Computational thinking (CT) is a critical skill in computer science education, and involves problem-solving, algorithmic thinking, and system design. For effective integration of CT in OS design education, unplugged activities may prepare better conceptual development before the effective use of technology. (Peel et al., 2022). This approach aligns with findings by Kharb (2023) , who identifies that logical and operational thinking must be developed through practical activities. This type of activity involves students in the conceptual issues of OS design and encourages them to also indulge in deeper abstract work. Additionally, frameworks that support the integration of CT into science education can be adapted to OS design, providing a structured approach to teaching complex topics (Cabrera et al., 2023). Real examples can be used to help students visualize some of the ways in which an operating system utilizes the resources. For example, the theoretical material regarding resource management by an operating system may be illustrated better using real-life examples of such operating system resource management.

4.1 GAMIFICATION AS A LEARNING TOOL

Gamification is one of the successful learning strategies widely used in modern learning/teaching methods to encourage students and raise their motivation to learn. In an example related to computer science students, elements of gamification have been combined with the ZORQ framework to make learning more playful for them. The ZORQ framework, for instance, utilizes gamification elements to create an interactive learning environment for computer science students (Weitl-Harms, 2023). Game-based learning can make the OS design courses even more dynamic and engaging by showing students there is a way to experiment with OS concepts in a risk-free

environment. Gamification, as various studies prove, tends to make better learning outcomes and increase student satisfaction (Awada et al., 2020). Furthermore, the Scalable Game Design curriculum provides a chance to explore ways in which game design might contribute to computer science education by enhancing computational thinking skills (Webb et al., 2015). By introducing different tools-merit boards, badges, and challenges-into the learning process, a teacher encourages students to be responsible for their learning and builds up both competitive spirit within them and a collaborative learning environment.

4.2 COLLABORATIVE LEARNING ENVIRONMENTS

Collaborative learning environments, particularly those supported by computer-mediated communication, have been found to foster student learning in subjects of STEM. Evidence provided by Zahir (2024), and "The Efficacy of Students' Knowledge Construction Process in Computer-Supported Collaborative Learning (CSCL) Environment: A Malaysian View" (2023) affirms this notion. The CSCL approach maintains that collaboration, technology, and pedagogy are the triplet bases for facilitating effective learning experiences. Generally, designing the OS course with a collaborative project and peer-to-peer learning increases that sense of community in which students feel supported by others and encouraged to share knowledge in their learning journey. This also aligns with the findings from Goode et al. (2020), who indicated that equity-focused teacher professional development is an important factor for driving inclusive environments in learning. Collaborative learning not only increases the level of OS concept understanding, but also develops essential soft skills such as teamwork and communication, which are critical in the tech industry.

4.3 PRACTICAL APPLICATIONS OF THE FRAMEWORK

The proposed tool-centric framework can be implemented through a series of practical applications which relate to the core components of OS design. With such process scheduling algorithms, students are able to derive insight using the simulation tools, playing with different strategies by observing in real-time the results of their application. Group projects may make use of interactive coding platforms where students will be designing and implementing simplified operating systems, thereby reinforcing their

learning through putting key concepts into practice. The Lab4CE is an example of a remote laboratory that may be used in practical computer science to enhance engagement (Broisin et al., 2015). By providing students with access to real-world tools and environments, educators can bridge the gap between theory and practice, preparing students for future careers in technology.

4.4 ASSESSMENT AND FEEDBACK MECHANISMS

Effective assessment strategies are crucial for evaluating student learning and providing constructive feedback. The framework incorporates various assessment methods, including peer assessments, self-reflections, and project-based evaluations, to ensure a comprehensive evaluation of student performance (Pasterk et al., 2019). By utilizing a variety of assessment tools, educators can gain insights into student understanding and identify areas for improvement, ultimately enhancing the learning experience. This approach is supported by the findings of Nagai et al. (2019), which emphasize the importance of formative assessments in developing competencies in computer science education. Additionally, incorporating feedback mechanisms that allow for continuous improvement can help students take ownership of their learning and foster a growth mindset.

4.5 PROPOSED TOOL-CENTRIC FRAMEWORK FOR TEACHING OS DESIGN

Figure 1 shows the tool-centric framework for teaching operating system design. The framework consists of 8 main steps: identifying learning objectives, integrating educational tools and methodologies, integrating computational thinking, collaborative learning, gamification elements, student-centered learning process, evaluation and assessment, and improving the understanding of operating system design.

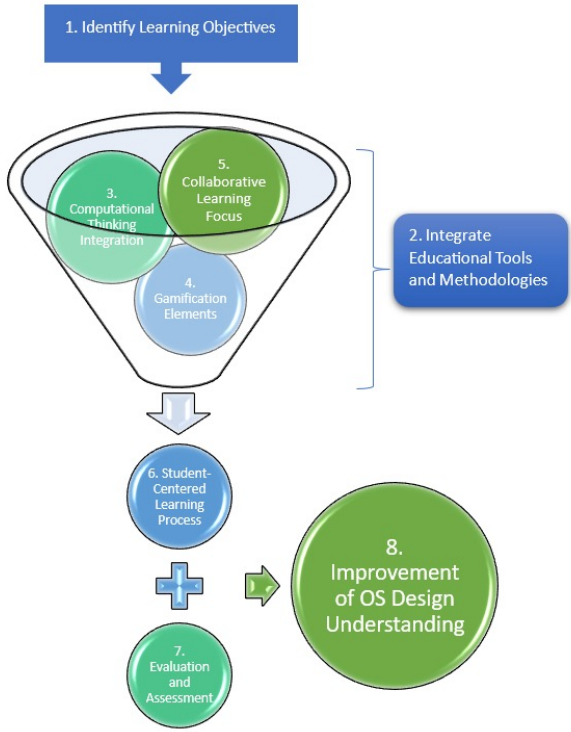

Figure 1: Tool-Centric Framework for Teaching OS Design

This shows a structured framework with the key components and their relationships.

1. Identify Learning Objectives:
Define specific learning goals, such as understanding OS concepts (e.g., memory management, process scheduling) and developing computational thinking skills. This foundational step sets clear expectations for students and aligns with the overall framework.

2. Integrate Educational Tools and Methodologies:
Introduce various tools, such as simulation platforms, collaborative coding environments, and game-based learning elements, tailored to the OS curriculum. The goal here is to equip students with interactive resources that support active learning and hands-on practice.

3. Computational Thinking Integration:
Use exercises that promote problem-solving, algorithmic thinking, and systems thinking in the context of OS design. For example, activities might include unplugged activities or real-world scenarios illustrating resource management.

4. Implement Gamification Elements:
Introduce game-based elements like leaderboards, challenges, and badges to boost engagement and motivation. By creating an interactive and competitive learning environment, gamification supports deeper exploration of OS concepts in a risk-free setting.

5. Collaborative Learning Focus:
Foster teamwork and knowledge sharing through peer-to-peer learning, group projects, and coding collaboration platforms. This collaborative element not only improves comprehension of OS concepts but also helps students develop essential soft skills.

6. Student-Centered Learning Process:
Adapt the framework to different learning styles and encourage self-paced learning. This approach makes the learning process more flexible, allowing students to engage with OS concepts in ways that suit their preferences.

7. Evaluation and Assessment:
Utilize formative assessments such as Quizzes and Practical Assignments, peer reviews, and self-reflections to gauge student understanding and provide feedback. This stage ensures students receive constructive guidance to improve their learning outcomes and understand areas for growth.

8. Improvement of OS Design Understanding:
Encourage an iterative process where students can revisit concepts and refine their skills based on feedback. This final stage supports a growth mindset, allowing students to deepen their understanding of OS design over time.

4.6 EXISTING FRAMEWORKS COMPARISON

Considering the current frameworks that exist for teaching undergraduate computer science students about operating systems (OS) design, approaches, tools, and pedagogical strategies implemented in the academic context have been evaluated. Below is a summary of the frameworks and methods traditionally in practice, plus their pros and cons.

Framework Type	Strengths	Weaknesses	Best Suited For
Lecture-Based Approach	Conveys the theory well with proper structure	Doesn't really cover practical work	Good understanding on theory
Project-Based Learning	Involves practical experience of the subject	Very time exhausting and detailed	Exposure in practical learning
Hybrid Approach	Mix of theoretical and practical knowledge	Simulation might be unrealistic	Overall knowledge
Case Study Based Learning	In-depth knowledge of the Industry workings & large scale systems	Is harder to grasp most times	In-depth knowledge
Gamification and Interactive Tools	Interesting and rapid feedback techniques	Sometimes, the concept can be too vague	New learners
Blended Online Learning	Works with a broader audience since it is very flexible	There is limited time in to interact other students	Self-learners, students who are not on campus
Modular or Layered Approach	Good would be the modular design focus	There is a chance of losing the bigger picture	Concentrated study of OS design|

Table 1: Summary of Framework Type for Teaching OS Design

Existing frameworks, such as ZORQ (Weitl-Harms, 2023) and Scalable Game Design (Repenning et al., 2015), focus on specific aspects like gamification or computational thinking. Unlike these, the proposed framework integrates multiple pedagogical strategies, providing a holistic approach to OS education.

For instance:

ZORQ Framework: Emphasizes gamification but lacks collaborative learning components.

Scalable Game Design: Focuses on game-based learning for computational thinking but does not address hands-on OS design tasks.

The proposed framework bridges these gaps by combining gamification, collaborative learning, and simulation tools to address diverse learning objectives.

4.7 EMPIRICAL EVIDENCE

The framework's effectiveness was evaluated through a pilot program involving 30 undergraduate students. Students were introduced to simulation tools, collaborative platforms, and gamification elements. Pre- and post-assessments revealed a 35% improvement in comprehension scores, while qualitative feedback indicated increased engagement and motivation.

1. Case Study: A group project required students to design a simplified OS kernel using a collaborative platform. The project fostered teamwork and critical thinking, with 85% of students reporting enhanced understanding of core OS concepts.

2. Result and Discussion: The pilot study demonstrated that students engaged with the framework achieved deeper understanding and higher retention of OS concepts. Key findings include:

    a. Improved Comprehension: Students demonstrated significant improvement in process management and resource allocation tasks.
    b. Enhanced Engagement: Gamification elements, such as leaderboards and badges, motivated students to participate actively.
    c. Skill Development: Collaborative projects improved teamwork and communication skills, essential for industry readiness.

3. Limitations: Limitations of the study includes limited sample size, duration of the study, and challenges in integrating the framework into existing curricula.

4.8 CHALLENGES AND CONSIDERATIONS

While the framework presented in this proposal has many advantages, it also needs to present and address the existing or potential challenges that might arise with the application of this framework. Educators may face difficulties in integrating new technologies into existing curricula, particularly in institutions with limited resources. The ideal scenario would be that the need for equitable access of all students to the required apparatus and tools is guaranteed in the success of the framework. Surmounting these would require efficient planning and team coordination from educators, administrators, and providers of technology. The insights from Kulikova and Yakovleva (2022) regarding pedagogical management in digital environments can inform strategies for overcoming these obstacles. Furthermore, ongoing professional development for educators is crucial to ensure they are equipped to effectively implement the framework and adapt to evolving educational technologies.

5.0 CONCLUSION

The proposed tool-centric framework in teaching operating system design to undergraduate computer science students is an unprecedented step forward in educational methodology. This should integrate active and collaborative learning with gamification and computational thinking to further increase student motivation and elicit deep understanding of the more difficult operating system concepts. While computer science continues to evolve, innovative methodologies by educators also must do so, which in turn will provide students with challenges they have to surmount in the future. When this framework is successfully implemented, it could be the framework that would revolutionize a method of education in OS, with knowledgeable and skilled students to maneuver in a world of ever-changing technologies.

6.0 ACKNOWLEDGEMENTS

The authors would like to extend their gratitude to lecturers of Universiti Teknologi MARA (UiTM), Cawangan Sarawak, Kampus Samarahan, for their valuable input and contribution in this paper.

7.0 FUNDING

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

8.0 AUTHORS' CONTRIBUTION

The article's premise was devised, written, and revised by Putit, S. Sulastri. Putit conducted the review and revisions, and gave the article submission approval Khedif, LYB. conceived the article and managed its development.

9.0 CONFLICT OF INTEREST DECLARATION

We certify that the article is the Authors' and Co-Authors' original work. The article has not received prior publication and is not under consideration for publication elsewhere. This research/manuscript has not been submitted for publication, nor has it been published in whole or in part elsewhere. We testify to the fact that all Authors have contributed significantly to the work, validity and legitimacy of the data and its interpretation for submission to IJELHE.

10.0 REFERENCES

Anohah, E. (2016). Pedagogy and design of online learning environment in computer science education for high schools. International Journal of Online Pedagogy and Course Design, 6(3), 39-51. https://doi.org/10.4018/ijopcd.2016070104

Awada, A., Zeshan, F., Khan, M. S., Marriam, R., Ali, A., & Samreen, A. (2020). The impact of gamification on learning outcomes of computer science majors. ACM Transactions on Computing Education, 20(2), 1-25. https://doi.org/10.1145/3383456

Broisin, J., Venant, R., & Vidal, P. (2015). Lab4ce: a remote laboratory for computer education. International Journal of Artificial Intelligence in Education, 27(1), 154-180. https://doi.org/10.1007/s40593-015-0079-3

Cabrera, L., Ketelhut, D. J., Mills, K., Killen, H., Coenraad, M., Byrne, V. L., ... & Plane, J. (2023). Designing a framework for teachers' integration of computational thinking into elementary science. Journal of Research in Science Teaching, 61(6), 1326-1361. https://doi.org/10.1002/tea.21888

Gesing, S., Stirm, C., Klimeck, G., Zentner, L., Wang, S., Martinez, B. M. V., ... & Kalyanam, R. (2022). Open science via Hubzero: Exploring five science gateways supporting and growing their open science communities. Proceedings of the Annual Hawaii International Conference on System Sciences. https://doi.org/10.24251/hicss.2022.090

Goode, J., Skorodinsky, M., Hubbard, J., & Hook, J. (2020). Computer science for equity: teacher education, agency, and statewide reform. Frontiers in Education, 4. https://doi.org/10.3389/feduc.2019.00162

Hashim, S., Masek, A., Zahir, N. Z. M., & Khamis, N. (2023). The efficacy of student' knowledge construction process in computer-supported collaborative learning (CSCL) environment: A Malaysian view. International Journal of Information and Education Technology, 13(9), 1452-1461. https://doi.org/10.18178/ijiet.2023.13.9.1949

Kharb, L. and Chahal, D. (2023). Exploring the psychological advantages of early computer education in early phases of development. International Journal of Information and Technology, 41, 17-22. https://doi.org/10.55529/ijitc.41.17.22

Pasterk, S., Kesselbacher, M., & Bollin, A. (2019). A semi-automated approach to categorise learning outcomes into digital literacy or computer science. IFIP Advances in Information and Communication Technology, 77-87. https://doi.org/10.1007/978-3-030-23513-0_8

Peel, A., Sadler, T. D., & Friedrichsen, P. (2022). Algorithmic explanations: An unplugged instructional approach to integrate science and computational thinking. Journal of Science Education and Technology, 31(4), 428-441. https://doi.org/10.1007/s10956-022-09965-0

Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S., Brand, C., & Gutiérrez, K. D. (2015). Scalable game design. ACM Transactions on Computing Education, 15(2), 1-31. https://doi.org/10.1145/2700517

Weitl-Harms, S., Spanier, A., Rokusek, M., & Hastings, J. (2023). Assessing user experiences with Zorq: A gamification framework for computer science education. Proceedings of the Annual Hawaii International Conference on System Sciences. https://doi.org/10.24251/hicss.2023.141

Zahir, N. Z. M., Hashim, S., Abdul Rahman, K. A., Zulkifli, N. N., Riyadi, S. & Siswantoro, J. (2024). Unveiling effective CSCL constructs for STEM education in Malaysia and Indonesia. Journal of Advanced Research in Applied Sciences and Engineering Technology, 46(1), 97-106. https://doi.org/10.37934/araset.46.1.97106