Research Statement

Ouh Eng Lieh
School of Computing and Information Systems, Singapore Management University
Tel: (65) 6828-9596; Email: elouh@smu.edu.sg
18 (Day) 12 (Month) 2025 (Year)

Background

My interest in the application of computing principles to secure industry software has shaped my research trajectory. Through mentoring computing students, I have recognized the foundational role education plays in equipping future professionals with essential software design and security skills. These experiences have motivated my ongoing research efforts in **computing education research (CER)** and **secure software engineering**. Figure 1 presents my research output over time.

My current focus lies in developing innovative **educational technologies for computing design curricula and creating tools that address security vulnerabilities using Large Language Models (LLMs)**. My current research focuses on two main questions: ***How can LLM-based educational technologies improve student's learning?*** and "***How can software development be made more secure using LLMs?*** To address these, I am developing the **TITAN security vulnerability detection tool and PromptTutor education technology.**

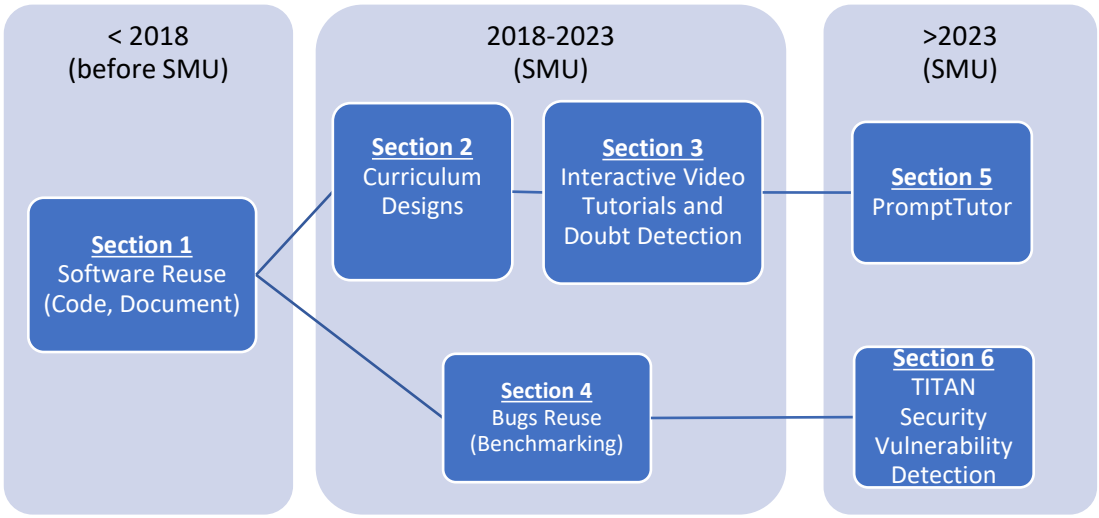

Figure 1. Research Output Over Time

Research Areas

1. Software Reuse (Before SMU)

I briefly describe my background research before describing my computing education research. Before joining SMU, my **practice research** on **software reuse** to help developers improve productivity with **code reuse** and **technical document reuse**.

The code reuse process involves clone detection and clone reuse. Code clones are semantically similar code fragment pairs that are syntactically similar or different. By automatically detecting and reusing code clones, developers can reduce the cost of software maintenance. I applied the reuse techniques in a study involving mobile apps for mood assessment. In this study, doctors need to use mobile apps to track mood assessment, and the required features of the mobile app depend on the patient's condition. The number of features and their inter-dependencies result in an exponential number of combinations, making the manual development of each possible mobile app challenging. **Our techniques customise these required features and automatically generate these mobile apps with required features by the doctors within minutes.** We used the Adaptive Reuse Technique (ART) [1] for the flexible customisation of mood scale apps. ART allowed us to partition mood scales and their features into distinct modules parameterised for ease of adaptation. ART engine performs synthesis of a required mood scale app from specifications of mood scales required and their features. App modules are composed after customisations to form an executable app that can be uploaded to the users' smartphones.

Another area is technical document reuse. We apply software clone detection and reuse to **automate searching repeated fragments in technical software documentation to be reused**. Our approach [2-3] supports adaptive reuse, extracting "near duplicate" text fragments (repetitions with variations) and producing customisable, reusable elements. The proposed approach simplifies the document maintenance process and can be used in the context of variability management in software product line development and in areas such as Enterprise Architecture Modelling, where models are stored in XML format and irregular repetitions are possible in a large volume of information.

2. Computing Education Research in Curriculum Design

I started my CER research when I joined SMU in 2018 as education track faculty. After practicing software design and development in my previous jobs, I decided to contribute to computing education research, which aligns with my education track at SMU. One of my early contributions was to analyse the ***"effectiveness of case-based learning (CBL) in computing courses on software architecture designs"*** based on my longitudinal study of teaching software design courses. A competent software architect needs to make design decisions. For a student who is used to writing code and compiles to get a deterministic result, the mindset of justifying their key decisions with potential trade-offs without a concrete output is a frustrating divergence from what they are doing. CBL provides students with an opportunity to see theory in practice,

and students are required to analyse data and background information to make design decisions. We propose bullet cases, mini-cases, and descriptive cases and how to use them. Our assessments [4-6] of a decade-long student evaluation data show that our proposed case-based course design is effective for our students to learn better. Our analysis results also show that the student's educational background does not have significant co-relationships with the efficacy of the case-based design.

An observation of undergraduate challenges leading to reduced motivation and interest to study and appreciating software designs is their limited experience in the software industry. I propose and analyse the **"*effectiveness of a model adapting the four stages in the Kolb model of experiential learning with risk management process concepts to highlight the impact of inadequate quality designs in real-life scenarios".*** This course design [7] promotes learning activities to observe how different parts of design components work in the software industry, experience the impact of real software quality issues or risks, reflect on the root causes of these risks, conceptualise, and subsequently implement the countermeasure to mitigate the risk. Our experiment findings show that the students preferred this model consistently in their learning ratings when comparing each stage of the model against the traditional lecture-based lesson.

Another practical challenge for undergraduates joining the workforce is how they can demonstrate they have the right computing skills to work on projects immediately. Industry recognised certifications offer a way for them to do that, leading to another CER area I work on to analyse the **"effectiveness of integrating IT certifications into a course to prepare students to be career-ready".** I designed an approach to identify a list of highly sought-after IT certifications based on recent industry job postings and map them in our course design [8]. I wrote programs to automate the search job postings using the Google Job Search engine. The search uses terms of job roles recognised by Singapore SkillsFuture (e.g., software engineer, software architect, solution architect) and text analytics techniques to identify in-demand industry certifications within these job postings. These programs are open-sourced in GitHub. We identified AWS Cloud Practitioner and Solutions Architect - Associate certifications as the top two in-demand certifications and integrated the certifications as part of our computing courses. Students appreciate the effort and are motivated to do the certification to be recognised for their skills. They achieved a high passing rate of over 90% for the recent 3 semesters this initiative was implemented. However, we also observed considerable efforts to integrate certification requirements into course contents and manage examination logistics for the certifications.

A well-thought course design requires effective course conduct to deliver the required learning outcomes. A natural progression of my CER in course design is to look into improving the conduct. An area I extend in my CER is to analyse the "**effectiveness of methods to identify and clarify students' doubts".** We evaluate two in-class methods (lecture, quiz reflection) and four out-of-class methods (self-reflection, assignment work, online communication, and face-to-face communication) [9-11]. Our study involves implementing these methods to students in two undergraduate computing design courses over two semesters, one on user interaction design and another on architecture design. Our results show that in-class methods with immediate

timing (in-class lecture and in-class quiz reflection) are more effective for the students in identifying and clarifying doubts. Students feel that it can be challenging to communicate online, and there is a delay in the replies if the instructor is unavailable. There is no clear indication that individual or group participation effectively identifies and clarifies doubts. Our statistical analysis shows significant differences in efficacy between most methods to identify and clarify. The instructor has to select a mix of methods to achieve high effectiveness in identifying and clarifying doubts.

3. Interactive Video Tutorials and Doubt Detection

With the need for online learning during COVID-19, I realise the importance of using education technologies to improve student's learning outcomes in their asynchronous learning of computing design concepts. Recent studies on the pedagogical analysis of online video tutorials show that increased interactions can increase learners' effectiveness in learning to code. However, there is a limited application within the educational process due to a lack or limited access to open-source tools that implement interactivity. In my research of education technologies, I implement and study the ***"effectiveness of interactivity features of video tutorials on students' learning outcomes"***. Video-based tutorials are the most common avenues for users to learn. However, the streaming nature of programming videos limits the ways to explore the captured workflows and interact with the output in the videos, e.g., copying the codes or text in the video. We develop an online web-based Interactive Tutorial Software System ITSS [12] (open-sourced in GitHub) with interactivity features and evaluate it in a course that teaches coding of **software design patterns**. Instead of recording video streams, we record the tutorial author's interactivity actions such as clicking, scrolling, highlighting, and typing in text-based format (e.g., what action at what time) using accessibility APIs. The system replays these actions to the student, giving the impression that the author is actively doing these on the web screen. The results of our two studies show that participants agree with the ease of recording when using our environment and feel that the increased interactivity can help us learn to program better. Our environment has a good usability level based on System Usability Scale (SUS). We also conduct a performance test to show that our environment can support up to 500 concurrent users with a total system response time of fewer than 5 seconds. This work was published at the ICSE conference and supported by **Singapore Management University Educational Research Fellowship Grant** and **Research Lab for Intelligent Software Engineering.**

Another challenge teachers encounter is to "***provide timely and relevant guidance to individual students according to their levels of understanding***". One option we studied is collecting students' reflections after each lesson to extract relevant and high-value feedback so that doubts or questions can be addressed promptly. We implemented an approach with an automated system to identify doubts from the informal reflections through features analysis and machine learning [13-15]. Our results show that selecting suitable features is essential and reflections with positive sentiment play a role in constructing a better model. The analysis of the reflection data suggests that the self-assessed level of understanding Likert scale rating can be adopted with the proposed automated approach to enable learner-centred learning and improve students' learning experience. I am a co-author in this piece of work.

4. Bugs Reuse (Benchmarking)

Besides my research in CER, I also branched out my practice research of software reuse to artefacts reuse, such as bugs for testing and debugging. We create a benchmark [16] database and tool that contain 493 real bugs from 17 real-world Python programs, making it the largest Python bug dataset to date. We present a framework to enable controlled studies requiring experiments on actual bugs in Python projects, such as work on testing and debugging. We hope this benchmark can help catalyse future work on testing and debugging tools on Python programs. I am a co-author in this piece of work.

5. PromptTutor – Supporting students during asynchronous learning using LLMs

We enhance ITSS (section 3) by integrating student analytics and AI video analytics to improve programming learning outcomes, such as concept understanding and problem-solving, during asynchronous learning. This PromptTutor initiative supports SMU's Vision 2025 digital transformation goals and aligns with Singapore's Ministry of Education Tertiary Education Research Fund Theme 4, aiming to design AI-enabled interventions that prompt students to reflect on completed tasks and address their doubts with personalized resources.

This research focuses on the design, deployment, and empirical evaluation of **AI-augmented learning systems**, with particular emphasis on how **Large Language Models (LLMs)** can be pedagogically grounded to improve both **learning outcomes and student motivation** in undergraduate computer science education. A central theme of this work is addressing scalability and feedback challenges in contemporary instructional designs, especially **flipped classrooms**, where students are expected to engage with complex material independently prior to class.

Flipped classrooms are widely adopted in computer science education due to their potential to free in-class time for higher-order discussion and applied problem solving. However, a persistent limitation of this pedagogy is the **lack of timely, personalized feedback during the pre-class learning phase**. In practice, students often encounter conceptual misunderstandings while engaging with readings or videos outside class, yet instructor feedback is delayed until scheduled contact time. Prior research has shown that such delays can lead to prolonged confusion, reduced motivation, and disengagement, particularly in large cohorts. Although recent advances in LLMs present opportunities for scalable, personalized educational support, existing work in this area has been largely qualitative or exploratory, with limited controlled evidence demonstrating impact on learning and motivational outcomes in undergraduate CS contexts.

To address this gap, I led a research team that investigated **PromptTutor**, an LLM-based chatbot designed specifically to support flipped classrooms. PromptTutor moves beyond generic LLM usage by embedding **reflective learning and scaffolding strategies** directly into prompt engineering. The system provides immediate, targeted feedback, adaptive pacing, and structured reflection prompts aligned with explicit learning outcomes, transforming pre-class preparation from passive content consumption into guided, self-directed learning.

The study employed a **randomized crossover experimental design** with undergraduate computer science students, enabling robust within-subject and between-subject comparisons. Learning outcomes were measured using quizzes aligned with Bloom's taxonomy, while motivation was assessed using validated subscales of the Motivated Strategies for Learning Questionnaire. The results demonstrated that students using PromptTutor achieved **significantly larger individual learning gains**, with particularly notable improvements in **higher-order thinking skills**, compared to traditional flipped classroom conditions. In addition, students reported **substantially higher intrinsic value and self-efficacy**, indicating that the intervention positively influenced key motivational beliefs known to support engagement and self-regulated learning.

This work makes several substantive contributions. First, it demonstrates how LLMs can be **pedagogically operationalized**, rather than opportunistically adopted, by grounding system behaviour in established learning theories such as reflective practice and scaffolding. Second, it provides **rigorous quantitative evidence** of the cognitive and motivational benefits of LLM-based tutoring in flipped classrooms, addressing a critical gap in the CS education literature. Third, it offers **actionable insights for educators and institutional leaders** on how AI-enabled tools can scale personalized feedback without increasing instructor workload, supporting sustainable teaching practices in large and diverse learning environments.

Collectively, this research strengthens my broader agenda on **evidence-based integration of generative AI in education**, contributing not only methodological rigor but also practical relevance. It positions AI not as a replacement for instruction, but as a thoughtfully designed augmentation that enhances learning effectiveness, motivation, and equity in computer science education.

6. TITAN - Software Vulnerabilities Detection using LLMs

This research advances the state of **AI-assisted software security** by developing principled methods that enable **reliable, interpretable, and scalable vulnerability detection using large language models (LLMs)**. A central concern motivating this work is that, while LLMs have shown promise in code understanding tasks, their application to vulnerability detection remains limited by **unreliable reasoning, high false positive rates, and poor trustworthiness**, which constrain real-world adoption in security-critical settings.

Vulnerability detection differs fundamentally from code generation and other reasoning tasks. It requires **binary judgements under uncertainty**, strong inductive bias toward security semantics, and explanations that practitioners can trust and act upon. Prior approaches relying on chain-of-thought prompting or supervised fine-tuning often produce explanations that are superficially plausible yet incorrect, while traditional sequence classification methods sacrifice interpretability altogether. These limitations motivate the need for models that can both **detect vulnerabilities accurately and justify their decisions with well-founded, security-aware reasoning**.

To address this challenge, I work with a research team on **R2VUL**, which introduces a novel framework for learning vulnerability detection through **reinforcement learning from AI feedback (RLAIF)** combined with **structured reasoning distillation**. Rather than training models solely on correct reasoning paths, R2VUL explicitly teaches models to **prefer valid over flawed reasoning** by contrasting high-quality and misleading explanations generated by a stronger teacher LLM. This preference-based learning signals a more robust inductive bias toward sound security reasoning, leading to improved detection accuracy and explanation quality, even in relatively small student models.

A key enabling contribution of this work is the construction of the **first high-quality multilingual preference dataset for function-level vulnerability detection**, comprising approximately 18,000 samples across five widely used programming languages (C#, JavaScript, Java, Python, and C). The dataset pairs high-confidence vulnerability labels, grounded in real-world CVE and CWE metadata, with **contrastive structured reasoning traces**. Rigorous re-annotation and validation procedures ensure substantially higher label correctness than existing vulnerability datasets, addressing a long-standing bottleneck in training reliable vulnerability detection models.

Through extensive evaluation, R2VUL demonstrates **consistent and statistically significant improvements** over a wide range of baselines, including static analysis tools, sequence classification models, supervised fine-tuning approaches, recent reasoning-based vulnerability detectors, and leading commercial LLMs. Notably, a **1.5B-parameter R2VUL model outperforms its 32B teacher model and proprietary systems such as Claude-4-Opus**, highlighting the effectiveness of preference-based reasoning distillation for specializing compact models. The approach also scales smoothly with increasing data and exhibits greater robustness under severe class imbalance, a critical requirement for practical deployment.

Beyond detection accuracy, this work directly addresses practitioner concerns by introducing a **lightweight calibration mechanism** that substantially reduces false positive rates with controlled trade-offs in recall. Qualitative evaluations further show that both human security experts and independent LLM judges consistently prefer R2VUL's explanations over competing reasoning-based methods, reinforcing its interpretability and actionability.

Collectively, this research makes foundational contributions to **LLM alignment for software engineering**, demonstrating how preference learning and structured reasoning can be harnessed to improve both performance and trust in AI-based security tools. It advances my broader research agenda at the intersection of **software engineering, security, and trustworthy AI**, with clear implications for building deployable AI systems that support, rather than hinder, real-world decision making in high-stakes domains.

7. Summary

In the coming years, I plan to continue pursuing research in computing education and secure software engineering, focusing on deriving better, secure and innovative tools using LLMs. The interactive tutorial software system, doubt identification system, PromptTutor and TITAN can be broadened to benefit more students and developers.

Selected Publications and Outputs

[1] Khue, Le Minh, Eng Lieh Ouh, and Stanislaw Jarzabek. 2015. Mood self-assessment on smartphones. In Proceedings of the conference on Wireless Health (WH '15). Association for Computing Machinery, New York, NY, USA, Article 19, 1–8. https://doi.org/10.1145/2811780.2811921

[2] Koznov, Dmitrij, Dmitry Luciv, Hamid Abdul Basit, Ouh Eng Lieh, and Mikhail Smirnov. (2016). Clone Detection in Reuse of Software Technical Documentation. In: Mazzara, M., Voronkov, A. (eds) Perspectives of System Informatics. PSI 2015. Lecture Notes in Computer Science, vol 9609. Springer, Cham. https://doi.org/10.1007/978-3-319-41579-6_14

[3] Lutsiv, Dmitry V., Dmitry Koznov, Hamid A. Basit, Eng Lieh Ouh, Mikhail N. Smirnov, and Konstantin Y. ROMANOVSKY. "An approach for clone detection in documentation reuse." Scientific and Technical Journal of Information Technologies, Mechanics and Optics 14, no. 4 (2014): 106.

[4] Ouh, Eng Lieh, and Yunghans Irawan. 2019. "Applying Case-Based Learning for a Postgraduate Software Architecture Course." In Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19). Association for Computing Machinery, New York, NY, USA, 457–463. https://doi.org/10.1145/3304221.3319737

[5] Ouh, Eng Lieh, Benjamin Kok Siew Gan and Yunghans Irawan, "Did our Course Design on Software Architecture meet our Student's Learning Expectations?," 2020 IEEE Frontiers in Education Conference (FIE), 2020, pp. 1-9, doi: 10.1109/FIE44824.2020.9274014.

[6] Ouh, Eng Lieh, Benjamin Kok Siew Gan and Yunghans Irawan, "Teaching Adult Learners on Software Architecture Design Skills," *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1-9, doi: 10.1109/FIE.2018.8658714.

[7] Ouh, Eng Lieh, and Yunghans Irawan, "Exploring Experiential Learning Model and Risk Management Process for an Undergraduate Software Architecture Course," *2018 IEEE Frontiers in Education Conference (FIE)*, 2018, pp. 1-9, doi: 10.1109/FIE.2018.8659200.

[8] Ouh, Eng Lieh, and Kyong Jin Shim, "Integration of Information Technology Certifications into Undergraduate Computing Curriculum," in *2021 IEEE Frontiers in Education Conference (FIE)*, Lincoln, NE, USA, 2021 pp. 1-9. doi: 10.1109/FIE49875.2021.9637266

[9] Ouh, Eng Lieh, and Benjamin Kok Siew Gan, "Evaluating Methods for Students to Identify and Clarify Doubts in Computing Design Courses," *2020 IEEE Frontiers in Education Conference (FIE)*, 2020, pp. 1-9, doi: 10.1109/FIE44824.2020.9274170.

[10] Ouh, Eng Lieh, and Benjamin Kok Siew Gan. "Effective digital learning practices for IS design courses during COVID-19." In 27th Annual Americas Conference on Information Systems, AMCIS 2021.

[11] Benjamin Kok Siew Gan, and Ouh, Eng Lieh. "Designing learning activities for experiential learning in a design thinking course." In *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, pp. 1-8. IEEE, 2019.

[12] Ouh, Eng Lieh, and Benjamin Kok Siew Gan, and David Lo. "ITSS: Interactive Web-Based Authoring and Playback Integrated Environment for Programming Tutorials." ICSE 2022: Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering, Virtual Conference, May 8-13, (pp. 1-7) Pittsburgh, PA: IEEE/ACM.

[13] Lo, Siaw Ling, Tan, Kar Way, and Ouh, Eng Lieh. "Automated doubt identification from informal reflections through hybrid sentic patterns and machine learning approach." Research and Practice in Technology Enhanced Learning 16, no. 1 (2021): 1-24.

[14] Lo, Siaw Ling, Tan, Kar Way, and Ouh, Eng Lieh. "Do my students understand? Automated identification of doubts from informal reflections." Proceedings of the 27th International Conference on Computers in Education. Taiwan: Asia-Pacific Society for Computers in Education, 2019.

[15] Tan, Kar Way, Lo, Siaw Ling, Ouh, Eng Lieh, and Neo, Wei Leng, "AI-Enabled Adaptive Learning using Automated Topic Alignment and Doubt Detection" (2022). *PACIS 2022 Proceedings*. 110. https://aisel.aisnet.org/pacis2022/110

[16] Ratnadira Widyasari, Sheng Qin Sim, Camellia Lok, Haodi Qi, Jack Phan, Qijin Tay, Constance Tan, Fiona Wee, Jodie Ethelda Tan, Yuheng Yieh, Brian Goh, Ferdian Thung, Hong Jin Kang, Thong Hoang, David Lo, and Eng Lieh Ouh. 2020. "BugsInPy: a database of existing bugs in Python programs to enable controlled testing and debugging studies." Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, 1556–1560. https://doi.org/10.1145/3368089.3417943

[17] Ouh, Eng Lieh, Benjamin Kok Siew Gan, Kyong Jin Shim, and Swavek Wlodkowski. "ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an

undergraduate Java Programming Course." In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, pp. 54-60. 2023.

[18] Ouh, Eng Lieh, and Stanislaw Jarzabek. (2016). An Adaptability-Driven Model and Tool for Analysis of Service Profitability. In: Nurcan, S., Soffer, P., Bajec, M., Eder, J. (eds) Advanced Information Systems Engineering. CAiSE 2016. Lecture Notes in Computer Science, vol 9694. Springer, Cham. https://doi.org/10.1007/978-3-319-39696-5_24

[19] Ouh, Eng Lieh, and Stanislaw Jarzabek. "Understanding Service Variability for Profitable Software as a Service: Service Providers' Perspective." In *CAiSE (Forum/Doctoral Consortium)*, pp. 9-16. 2014.

[20] Ouh, Eng Lieh, Stanislaw Jarzabek, Geok Shan Lim, and Ogawa Masayoshi. "Cloud, edge, and fog computing: Trends and case studies." In Data Science and Innovations for Intelligent Systems, pp. 181-208. CRC Press, 2021.

[21] Zhang, Yuhao, Eng Lieh Ouh, Adam Ho, Siaw Ling Lo, Kar Way Tan, and Feng Lin. "PromptTutor: Effects of an LLM-Based Chatbot on Learning Outcomes and Motivation in Flipped Classrooms." In Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1, pp. 445-451. 2025.

[22] Ouh, Eng Lieh, Kar Way Tan, Siaw Ling Lo, and Benjamin Kok Siew Gan. "Evaluating ChatGPT to Answer Multi-Modal Exercises in Computer Science Education." In Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1, pp. 58-64. 2025.