Adopting Model-Eliciting Activities in an Undergraduate Software Engineering Course through Real-World Projects

Young Lee
Dept. Computational, Engineering, and Mathematical Sciences
Texas A&M University-San Antonio
San Antonio, TX, USA
young.lee@tamusa.edu

Jeong Yang
Dept. Computational, Engineering, and Mathematical Sciences
Texas A&M University-San Antonio
San Antonio, TX, USA
jeong.yang@tamusa.edu

Young Rae Kim
Dept. Curriculum and Instruction
Texas A&M University-San Antonio
San Antonio, TX, USA
youngrae.kim@tamusa.edu

***Abstract*—This innovative practice work in progress paper discusses the integration of Model-Eliciting Activities (MEA) into in software engineering (SE) classes, and the challenges faced in preparing and delivering these activities. The preparation and implementation of MEA in SE classes can be challenging, as it requires creating self-assessable MEA questions that simulate real-world problems, ensuring the integration of MEA in SE course topics, providing feedback, and analyzing learning outcomes. In this research, the authors address these challenges and gain practical experience in designing and implementing MEA in SE classes.**

The authors conducted experiments over two consecutive semesters in SE courses that cover topics such as requirement solicitation, design and implementation, software testing techniques, secure software, and software quality assurance. They incorporated MEA questions that simulate real-world problems into both face-to-face and online classes, ensuring the integration of MEA in SE course topics, providing feedback, and analyzing the learning gains. This paper contributes to the distribution of MEA application for SE courses. It presents the authors' experiences, challenges, reports evaluations, and findings in implementing MEA in SE courses. Overall, this paper provides insights into the effective integration of MEA into SE courses, and the benefits it can bring to students' learning outcomes.

Keywords— software engineering, computer science education, model-electing activity, MEA

I. Introduction

This Innovative Practice work in progress paper discusses the integration of Model-Eliciting Activities (MEAs) in software engineering classes, and the challenges faced in preparing and delivering these activities. MEAs are a problem-solving approach that aims to enhance students' critical thinking, creativity, problem-solving, and communication skills by presenting real-world problems that require them to work in teams to create, test, and refine their models in response to the needs of a hypothetical client [1, 2, 3].

A few research on MEAs have been applied to computer science [4, 5]. Software engineering can benefit from MEAs as they are an effective way to deepen students' conceptual knowledge and evaluate their problem-solving processes. However, MEAs have not yet been applied extensively to the field of software engineering. Thus, further research is needed in this area.

The concept of correctness is essential for secure software development. That is why it is so important to have an effective learning model to help students gain a better comprehension of the concept. However, this can be a challenging task, as software engineering is a complex subject. Thus, it is important to find a way to deliver the material in a way that students can understand and apply the concept taught and assess students' capabilities.

The primary goal of this study is to create effective learning experiences within software engineering courses through integration of MEA into software engineering. In this study, we create an MEA entitled "Software Correctness Measurement" for computer science students within the context of software engineering (SE) courses and explore how students, and teams of students, navigate the problem-solving process of the MEA, and how the MEA impact student learning.

II. Methodology and Research Setting

This study uses a teaching experiment design methodology to investigate the nature of the intervention and its effectiveness in student learning. This methodology enables us to examine the impact of a particular intervention on student learning and teaching practices in a complex educational setting, and to effectively improve the design of the intervention based on instructor-researcher partnerships [6].

The authors incorporated MEA questions that simulate real-world problems into both face-to-face and online classes, ensuring the integration of MEA in SE course topics, providing feedback, and analyzing the learning gains. 18 undergraduate students participated in Fall 2022, software engineering I (SEI) class and 19 undergraduate students participated in Spring 2023 software engineering II (SEII) class.

The concepts of Software Requirement analysis, Software Design and implementation, Software Process Models, Software Unit Testing, Software Acceptance Testing, Risk Analysis were taught at Fall 2022 SEI class; Software Testing, Static Analysis,

2023 IEEE Frontiers in Education Conference (FIE) | 979-8-3503-3642-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/FIE58773.2023.10343438

Table 1. Principles for guiding the Software Correctness Measurement MEA development.

Principle	Description	Software Correctness Measurement
Reality	Ensures that the activity is contextualized in a realistic situation.	Student teams develop a program measurement formula for software developers in the process of designing and implementing a social network platform like an Instagram.
Model Construction	Asks students to construct an explicit description, explanation, or procedure for a significant system.	Students are asked to develop a formula that reads a source code to measure the correctness of the social network platform.
Model Documentation	Mandates that students create a form of documentation to clearly communicate their solution process.	Student teams prepare a report, presentation, and demonstration which feature the essential problem-solving strategies utilized in their formula.
Self-Assessment	Has criteria in the activity to which the students can test and revise their ways of thinking.	Student teams test to see if their formula meets the requirements and detect any issues through their own test cases. If necessary, they then modify the formula based on the tests conducted.
Generalizability	Requires students to create solutions that can be shared with others, and adapted for other engineering scenarios that are closely related.	Students' formula should be reusable for measuring any social network software. It should allow others to reuse it as a program correctness measuring tool for future social network software projects.
Effective Prototype	Ensures the model created is simple yet involves grounded concepts or principles from engineering.	The purpose of this MEA is to enhance technological literacy related to static analysis as a method of software quality assurance, such as correctness achieved through source code analysis without executing a program.

Secure Software Development, Software Quality and Metrics, Software Planning and Estimation were taught in Spring 2023, SEII class with the incorporation of some security modules introduced and developed in [9, 10]. The book used for the course was Software Engineering: A Practitioner's Approach with other related papers, software testing tools, and static analysis tools.

A pilot MEA construction began with initial implementations in Fall 2022. The researchers identified problem topics that are cornerstones or capstones in software engineering courses, such as requirement solicitation, design and implementation, software testing techniques, secure software, and software quality assurance.

Read the following email individually before beginning teamwork.

To: Software Engineering II Students,
From: Software Development Team, Jaguar metasoft
Subject: Measuring Software Correctness

Software Engineering II Students,

Software Development team at Jaguar metasoft has invited our class to help them create a **program measurement formula** for their software developers. This company, Jaguar metasoft, is in the process of designing and implementing a social network platform like an Instagram. But Jaguar metasoft software engineers have been struggling with proving whether their social network platform application is correct.

Static Measuring Software is best described as a method of software quality such as correctness by source code analysis before a program is run.

As a software engineer or programmer, you will be required to develop a formula that reads a source code to generate the correctness rating between 1 to 10 (higher metric value means higher quality).

In line with this view, Jaguar metasoft requests our help with program analysis. This is where your team comes in. Your team needs to develop a formula to measure the correctness of the social network platform. The formula should be reusable for measuring of any social network software. It is important that you offer a detailed explanation for the reasons behind all the decisions you made in this formula, because it will be used as a program correctness measuring tool for Jaguar metasoft's future social network software projects. The formula should increase overall product quality in terms of correctness and reliability with shorter verification time and less development cost.

I look forward to hearing from you.

Figure 1. Letter from the Industry.

However, the preparation and implementation of MEA in SE classes posed challenges, as it requires addressing the six principles for a well-designed MEA: the Reality Principle, the Model Construction Principle, the Model Documentation Principle, the Self-Assessment Principle, the Generalizability Principle, and the Effective Prototype Principle. Adherence to these design principles for MEAs is necessary to enhance students' understanding of engineering concepts, in addition to improving their problem-solving, communication, and teamwork capabilities [1, 2].

For example, when developing the initial version of Static Analysis MEA, the Self-Assessment Principle posed a challenge for the research team, requiring the integration of MEA into a junior Software Engineering course to ensure students could gain experience in self-assessment. After conducting a pilot study using the initial version of the Static Analysis MEA, the research team modified it in order to create the Software Correctness Measurement MEA utilized in this study. Table 1 provides a brief description of each of the principles that map to the Software Correctness Measurement MEA presented in Figures 1 and 2. The MEA project served as a practical application of Software Quality Measurement Concept, presented in the lecture materials of SEII.

Table 2. SEII, Spring 2023.

Definition of the Correctness	Formular	Variables
A system or software must function correctly. Correctness can be defined as the degree to which software performs its specified function. Correctness is achieved when the program behaves exactly as intended for all the uses-cases. It can be measured in terms of defects per KLOC.	Nr = Number of Functional Requirements, No of tests passed/total number of requirements. 0: one indicates none of the tests were completed, while 1 means all the tests were passed	Requirement, Test case
software program meets its specified requirements and performs its intended functions accurately and without unexpected behavior or errors	Definition of correctness in our formula is based on performance, error handling, correctness, security, code readability, testability. Each factor has a score of 1-10 and its own individual weight. performance 10%m error handling 20%, correctness 35%, security 25%, code readability 5%, testability 5 %	performance, error handling, correctness, security, code readability, testability
Source code has a low detected issue rate and there are no errors that crash the application.	Correctness = |1- ((I * (1 - F) / L) | * 10 I = # of issues detected by static analyzer L = # of lines in source code F = Estimated percentage of false positives	Error, size, security
Correctness is measured on how well the code implements the functional requirements and specified quality standards.	Correctness (0-10pts) = Security + Readability + Unit Testing + Performance	Security, Readability, Unit Testing, Performance
Correctness can be defined as the degree to which the social network platform software meets its specified requirements and operates as intended without unexpected behavior or errors. The correctness rating generated by our formula should reflect the extent to which the software satisfies its requirements and meets the expectations of Jaguar metasoft stakeholders.	The formula considers various factors such as the complexity of the code, the presence of potential security vulnerabilities, and the frequency and severity of any detected errors or bugs. To arrive at a final correctness rating between 1 and 10, our formula assigns weights to each of these factors based on their relative importance.	Security vulnerabilities, code complexity, frequency bug, severity of bug
the ability to run code efficiently and effectively. Run time of individual methods does not go below the average time. When compiling code, there should be zero errors.	Formula starts off at the maximum (score of 10) and the longer an application takes to finish, the more points will be taken off.	Reusability, Speed (run time, static time), Space, Accuracy
number of bugs found by spotbugs and the number of lines of code.	10 - (No. of bugs detected by spotbugs/total LOC) * 100	Error, size

III. Preliminary Results and Discussion

For the study to explore the students' experiences with the Software Correctness Measurement MEA, the outcomes of the MEA were collected and analyzed, including student team reports containing solutions, processes with written and visual descriptions, team presentations, and individual students' self-reflections on the MEA. An open-coding technique was utilized to explore how the student teams navigated the problem-solving process of the MEA and how the students reflected on their experiences with the MEA in terms of learning, benefits, and challenges. Discrepancies in the coding were discussed and resolved by consensus amongst the researchers.

A. Problem-Solving Processes

The MEA created in the study requires the problem-solving processes, "definition building" and "operationalizing definitions" [3]. Definition building is the act of forming definitions, which are commonly qualitative constructs, to solve an MEA, while operationalizing definitions is the process of quantifying those definitions involving qualitative information and providing evidence to prove they meet the needs of the situation. Business and industry often use these problem-solving processes to tackle real-world issues. In the MEA, students are required to measure software quality factor of Correctness. Each student team must define their own definitions of the correctness and to develop a formular that reads a source code to generate the correctness rating between 1 to 10 (higher metric value means higher quality).

The problem-solving process on the Software Correctness Measurement MEA consists of four steps: 1) definition building (developing the definition of Correctness), 2) operationalizing definitions (developing the formula to measure Correctness), 3) providing an example of how the formula works, and 4) providing information on how other programmers apply this formula to their projects (See Figure 2).

The correctness formula report must include the following:

A. *You can assume that source code analysis tools are available to collect data from the source code.*
B. *Active participants names*
C. *You need to provide the definition of correctness.*
D. *A detailed explanation of how your team's formula measures a source code and decides the metric values (1 – 10) in terms of correctness. Metric value 10 means the perfect software in terms of the correctness. So that it can be used to implement the automated correctness measurement tool for social network software.*
E. *An application of your formula to a sample program which will be provided as an example of how your formula works.*
F. *Any Explanations and information to help Jaguar Metasoft engineers can implement and apply it to their social network projects.*

Figure 2. MEA Team Activity.

Table 3. Variables in Correctness Formular.

Variables	Test case	Run time	No of errors	No of lines (size)	Compl -exity	Bug severity
SEII	3	3	5	2	1	1
SEI	4	4	4	3	2	1
Variables of requirements, security, readability, space, accuracy are found only at formular in SEII MEA activity; Variables of efficiency, flexibility are found only at formular in SEI MEA activity.						

A total of 9 teams submitted their reports at SEI, while 7 teams submitted theirs at SEII. Firstly, each team defined their own definition of Software Correctness. Student teams exhibited relatively clear goals, such as defining software correctness as "the degree to which software performs its specified function." Table 2 illustrates that most definitions revolve around whether the software meets the functional requirements without any errors.

Secondly, to develop the formula, the students identified various variables, including the number of requirements, test cases, program size, and number of errors, among others. They also recognized abstract variables such as performance, security, readability, and complexity, as shown in Table 3.

Thirdly, the teams provided a formula to determine the metric values, ranging from 1 to 10, that represent the level of correctness achieved using these variables. A metric value of 10 indicates a perfect software in terms of correctness. This problem-solving process provided the student teams with an opportunity for modeling to establish a relationship between the abstract software quality factor, correctness, and the static and dynamic features of the software system (See Table 2).

Fourthly, the subsequent step required the students to apply the formula to a sample program, which was provided as an example to demonstrate how the formula functions and information on how other programmers apply this formula to their projects. This step served as a self-assessment, enabling them to review whether their formula generated the metrics as defined while illustrating it using examples and demonstrating how to apply it to other projects.

B. Impact of MEAs on Student Learning

According to the students' self-reflections on the MEA, the majority of them felt that they had a better understanding of a software quality modeling concept after completing the MEA. The students expressed their experiences on the MEA as highly rewarding, citing its advantages of team collaboration, multiple perspectives, communication, and better understanding. The most rewarding aspect of the activity was illustrated in their experiences, such as: "communicating with my partner and working together to figure out the formula"; "getting a better grasp on what it takes to have a correct program"; "what I found most rewarding was the fact that we all figured out the formula quickly and then just started to test it to see if the formula would work to complete the parameters. As we were going along, we came up with two different formulas that could work using various ideas"; and "I feel that the team collaboration was the most rewarding. Hearing and experiencing the process other people take to the same problem always serves as a good way to view other perspectives to the same problem"; and "we got to see the bigger picture of the activity on how this report could essentially be used for other programs not just a social networking application."

The majority of students also indicated that the most challenging aspect of the activity was operationalizing their definitions of correctness, which involved quantifying qualitative information and choosing appropriate variables and mathematical operations that were consistent with the definition constructed. This is illustrated by the following statements: "The most challenging aspect of the activity was trying to find the correct attributes that would define a correct formula"; "Coming up with the different factors for the formula and how they should be weighed against each other"; "Finding the correct attributes to use in the formula to help find the proper correctness of the program"; "Understanding how to come up with a formula and trying to follow along with my teammates who had previous experience."

These students' reflections are likely to support previous research studies illustrating how MEAs can help improve students' core professional skills, such as conceptual learning, problem-solving, communication, and teamwork, by engaging them in critical thinking and metacognitive learning environments [3, 7]. Research shows that the processes of operationalizing definitions is a key element of critical thinking, which is a metacognitive process involving the ability to interpret problem situations, make decisions, solve problems, and take actions. The process of critical thinking involves identifying and examining an issue, understanding its meaning, gathering evidence, assessing the evidence, drawing conclusions, considering other relevant information, and forming an overall judgment [7, 8]. Kim et al. [3] demonstrated that MEAs promote metacognition by encouraging problem-solvers to consider their own and others' thought processes, while also allowing them to monitor and evaluate potential alternative strategies.

IV. Conclusions

The authors found that the majority of the students valued the MEA project topic using real-world problems. Reasonable solutions to the MEA project also indicated a strong understanding of the Software Quality Measurement concept especially in the SEII. This paper contributes to the distribution of MEA application for SE courses in the classroom and distance learning. It presents the authors' experiences, challenges, and findings in implementing an MEA in SE courses. Overall, this paper provides insights into the effective integration of MEA into SE courses, and the benefits it can bring to students' learning outcomes.

Acknowledgement

This material is based upon work supported by the National Science Foundation (NSF)'s Grant No #1832433. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] R. Lesh, M. Hoover, B. Hole, A. Kelly, and T. Post, "Principles for developing thought- revealing activities for students and teachers," in *The handbook of research design in mathematics and science education*, A. Kelly and R. Lesh, Eds. Mahwah, NJ: Lawrence Erlbaum Associates, 2000, pp. 591–646.

[2] R. Lesh and H. Doerr, "Foundations of a models and modelling perspective on mathematics teaching, learning and problem solving," in *Beyond constructivism: A models and modeling perspectives on mathematics problem solving, learning and teaching*, R. Lesh and H. Doerr, Eds, Mahwah, NJ: Lawrence Earlbaum Associates, 2003, pp. 3-33.

[3] Y. R. Kim, M. S. Park, T. J. Moore, and S. Varma, "Multiple levels of metacognition and their elicitation through complex problem-solving tasks," *Journal of Mathematical Behavior*, vol. 32, no 3, pp. 377-396, 2013.

[4] J. Yang, B. Earwood, Y. R. Kim, and A. Lodgher, "Implementation of security modules with model-eliciting activities in computer science courses," in *Proceedings of the 2020 American Society for Engineering Education (ASEE) Virtual Annual Conference*, 2020.

[5] J. Yang, Y. R. Kim, and B. Earwood, Brandon, "A study of effectiveness and problem solving on security concepts with model-eliciting activities," in *Proceedings of the 2022 IEEE Frontiers in Education Conference (FIE)*, 2022.

[6] R. Lesh and A. Kelly, "Multitiered teaching experiments." in *Research design in mathematics and science education*, A. Kelly and R. Lesh, Eds. Mahwah, N.J.: Lawrence Erlbaum Associates, 2000, pp. 197-230.

[7] G. X. Chen, X. Q. Qu, L. P. Huang, L. Huang, C. Zhou, and Y. Qiao, "Modeling-eliciting activities in an online engineering course for improving conceptual learning, professional skill, interaction," *IEEE Access*, vol. 10, pp. 87767-87777, 2022.

[8] C. P. Dwyer, M. J. Hogan, and I. Stewart, "An integrated critical thinking framework for the 21st century," *Thinking Skills and Creativity*, vol. 12, pp. 43–52, 2014.

[9] J. Yang, A. Lodgher and Y. Lee, "Secure Modules for Undergraduate Software Engineering Courses," 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 2018, pp. 1-5, doi: 10.1109/FIE.2018.8658433.

[10] A. Lodgher, J. Yang and U. Bulut, "An Innovative Modular Approach of Teaching Cyber Security across Computing Curricula," 2018 IEEE Frontiers in Education Conference (FIE), San Jose, CA, USA, 2018, pp. 1-5, doi: 10.1109/FIE.2018.8659040.