Full paper: Exploring Instructors Insight' to a MATLAB Code Critiquer

Mary Benjamin, Michigan Technological University

PhD Student in the Dept. of Civil, Environmental, & Geospatial Engineering at Michigan Technological university.

Laura Albrant, Michigan Technological University

After completing a bachelor's degree in computer science, Laura Albrant decided to challenge how she viewed software development, by switching departments. Currently working towards a master's degree in human factors at Michigan Technological University, Laura pursues interests on both sides of the fence through education research.

Dr. Michelle E Jarvie-Eggart P.E., Michigan Technological University

Dr. Jarvie-Eggart is a registered professional engineer with over a decade of experience as an environmental engineer. She is an Assistant Professor of Engineering Fundamentals at Michigan Technological University. Her research interests include technology adoption, problem based and service learning, and sustainability.

Dr. Jon Sticklen, Michigan Technological University

Jon Sticklen is an Associate Professor with the Engineering Fundamentals Department (EF) and Affiliated Faculty with the Department of Cognitive and Learning Sciences (CLS). He served as Chair of EF from 2014-2020, leading a successful effort to design a

Dr. Laura E Brown, Michigan Technological University
Dr. Leo C. Ureel II, Michigan Technological University

Leo C. Ureel II is an Assistant Professor in Computer Science and in Cognitive and Learning Sciences at Michigan Technological University. He has worked extensively in the field of educational software development. His research interests include intelligent learning environments, computer science education, and Artificial Intelligence

Full Paper: Exploring Instructors Insight' to a MATLAB Code Critiquer

Introduction

Recent advancements in educational tools for computer programming have highlighted the need for specialized tools to address challenges faced by novice programmers. Among these tools, code critiquers have shown promise in other programming languages such as Java [1]. We developed a MATLAB version of a Java-based code critiquer called WebTA. This paper reports on initial beta tests of MATLAB WebTA in the spring semester of 2023 within an introductory engineering course, providing insights into its efficiency and areas for improvement.

Background

A code critiquer is an advanced software tool that analyzes programming codes and provides feedback [2]. It detects errors, identifies poor coding practices, and suggests improvements. Features such as autograding, debugging, and intelligent tutoring can also be present. The ability to provide immediate, context-specific feedback makes it a beneficial resource for learning and teaching programming [3].

What is an Antipattern?

An antipattern is a commonly found mistake within a novice programmer's code [3]. These mistakes range from syntax, interpreter, style, logic, and more. Teaching students to identify and correct antipatterns is essential for their growth into competent programmers who can write clean, efficient, and sustainable code.

WebTA's Features:

WebTA utilizes spotting antipatterns and good patterns, with a traffic light system for categorizing patterns. Spotting antipatterns means identifying common coding mistakes that beginners often make, such as errors in syntax or logic. Good patterns, on the other hand, are effective coding practices that should be reinforced. The traffic light system provides a simple way for students to understand their coding practices: a green light indicates good coding practices, a yellow light signals a warning for potential issues that may not cause immediate problems but should be avoided, and a red light indicates critical mistakes that need to be fixed to ensure the code runs correctly [1].

The Role of Code Critiquers in Programming Education:

Acknowledgment

This work was funded by the National Science Foundation award # XXXXXXX. Recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.
This work would not have been possible without the exemplary cooperation from the First Year Engineering Program at AU.

Typically, students learn syntax first, then move on to higher levels of problem-solving [4]. WebTA addresses the gap between these levels. It acts as an extension of the instructor,

providing timely, objective feedback and automating the identification of antipatterns, thus enhancing students' understanding of core concepts [5]. Integrating tools like MATLAB WebTA in programming courses is crucial for comprehensive and engaging learning [6]. The instantaneous formative feedback provided by code critiquers can be leveraged in active learning remote learning settings alike. However, MATLAB WebTA is still in development, and currently only used for a few assignments in the introductory engineering course.

As we continue to develop MATLAB WebTA, it is important to assess its efficacy, understand user experience, and refine the code critiquer. With this in mind, this study poses the following research questions:

RQ1: How effective is the MATLAB WebTA in identifying and providing feedback on antipatterns in novice programmers' code, as measured by instruments for student engagement and learning?

RQ2: What are the experiences and perceptions of instructors and students regarding the use of MATLAB WebTA in an introductory engineering course?

RQ3: What key challenges and areas for improvement are identified by instructors and students when using the MATLAB WebTA?

Methods

A multi-methods qualitative research approach examined the experiences of instructors and students using the MATLAB WebTA over two semesters, beginning with class observations [7] followed by semi-structured interviews [8].

In-class Observations

In the 2023 Spring semester at Michigan Technological University, WebTA was introduced in ENG1101, the first course in a two-course sequence for first-year engineering students. Approximately 54 students in a "trailer section" used WebTA for three MATLAB assignments, submitting screenshots as proof. Graduate students observed and logged student interactions with WebTA [1].

During the in-class observation, graduate students assessed student engagement with WebTA, usage patterns, challenges faced, behavioral responses to feedback, and instructor interaction with the tool.

Detailed digital notes and memos captured immediate insights, informing the development of interview questions to address significant aspects observed.

Interviews

In Fall 2023, pre-interviews with two ENG1101 instructors assessed their baseline knowledge of antipatterns. Each instructor identified coding errors and discussed their teaching methods. A post-interview with one instructor captured reflections on WebTA's impact on teaching and learning. These interviews were conducted online and lasted about 45 minutes each.

Thematic analysis [9] was used to transcribe and code the interviews, categorizing initial codes into larger themes relevant to the research questions as seen in table 1. For instance, codes related to "student engagement" were categorized under "impact on learning," addressing research question 3.

Table 1. Themes, Codes and Examples.

Theme/code	Code Description	Example Quote
Recognition of Antipatterns: Identification issues	Identifying common mistakes in students' code	"The sign looks a little weird, and um, the step seems a little off."
Recognition of Antipatterns: Context Issue	Struggling to identify specific antipatterns without clear context or prior explanation.	"Without the context, I was definitely struggling to understand what errors we were looking for."
Recognition of Antipatterns: Importance of Understanding Errors	Importance of understanding common errors students make and the benefits of addressing these errors.	"Making sure that (educators & graders) know what the most common errors are, will make a much faster process for us to assist students."
Recognition of Antipatterns: Specific Example	Examples of common antipatterns students struggle with such as missing semicolons or reestablishing variables	"Participants felt the tools can be way too nitpicky for first-year engineers. Example given is identifying missing semicolons on every line."
Teaching Approaches: Pedagogical Differences	Differences in pedagogical approaches to coding errors, focusing on syntax or broader problem-solving techniques	"You never know how much you depend on the command line to tell you what the error is before you're just looking at it."
Teaching Approaches: Use of Analogies	Use of analogies and familiar examples to explain complex coding concepts.	"Coding is like an essay. MATLAB is a big calculator… really seems to resonate with students."
Feedback on WebTA: Utilization and Improvement	Instructors' and students' feedback on WebTA tool utilization and areas of improvement.	"Making sure that (educators & graders) know what the most common errors are, will make a

		much faster process for us to assist students.”
Feedback on WebTA: Mixed Reactions	Mixed feelings on the tool’s feedback mechanism, with suggestions for improvements.	“Participants noted that the emphasis on syntax was over highlighted by WebTA.”
Feedback on WebTA: Specific Features	Specific feedback on tools features, focusing on syntax and integration with other platforms.	“Mixed feelings about WebTA integration with other platforms (e.g canvas) and its focus on syntax.”
Feedback on WebTA: Challenges and Improvements	Challenges faced by students due to WebTA feedback being focused on syntax (good patterns)	“Participants noted that renaming and reestablishing variables proved to be a common mistake, and integrating the previous lesson knowledge and trying it to be a compound code seemed to be a struggle for the students.”
Teaching Insights: Good Habits	Insight into how WebTA helped instill good programming habits in students.	“It's not memorizing words, it's a tool, more than a big calculator.”
Teaching Insights: Reducing Fear	Observations on how the tool helped students not to be afraid of coding.	“Not be afraid to code. Voiced how overwhelming it was to learn programming ‘as a whole new language.’ ”
Student Engagement and Learning: Impact on Engagement	Observations on how WebTA impacted student engagement and learning.	“Some students found coding overwhelming, likening it to learning a new language, while others were more receptive.”
Student Engagement and Learning: Self-Efficacy	Impact of the tool on students’ willingness to engage and their sense of self efficacy.	“Students in the 2nd semester were more willing to fail things.”

Results and Discussion

The three classroom interventions demonstrated WebTA’s usefulness and highlighted areas where students’ expectations and WebTA’s functionality did not align [1]. These observations informed the interview questions in Study 2, providing a foundation for future steps. Interviews revealed WebTA’s strengths and areas for improvement.

Themes From Pre-Interviews

Recognition of Antipatterns

Instructors varied significantly in identifying antipatterns. One instructor struggled to pinpoint specific issues, highlighting a lack of detailed understanding or confidence. Another focused on superficial aspects, missing core antipatterns. This underscores the need for better instructor training on common coding mistakes.

Teaching Approaches

There were notable differences in teaching strategies. Some instructors emphasized syntax and technical precision, while others focused on broader concerns like code commenting and variable naming. One instructor likened coding to essay writing, stressing clarity and structure, while another was frustrated with students' over-reliance on syntax feedback, suggesting it overshadowed conceptual understanding.

Feedback on WebTA

Instructors provided mixed feedback. One appreciated its quick identification of common errors, streamlining the teaching process. Another felt the emphasis on syntax could be overly nitpicky for first-year students, suggesting a balanced approach integrating both syntactic and conceptual feedback.

Themes From Post-Interviews

Teaching Insights

WebTA helped instill good programming habits in first-year engineering students. Instructors noted students became more diligent in their coding practices, paying closer attention to code commenting and variable naming, crucial for long-term success.

Student Engagement and Learning

Student reactions varied widely. Some found coding overwhelming, likening it to learning a new language, while others were more receptive. The WebTA feedback mechanism helped reduce fear of coding, encouraging students to experiment and learn from mistakes.

Tool Feedback

Instructors had mixed feelings about WebTA's integration with other platforms, like Canvas. While the focus on syntax was appreciated for its precision, some felt it was overly stringent. They suggested future iterations should balance syntactic feedback with a broader understanding of coding principles to help students develop a deeper comprehension.

Conclusions and Future Directions

The study found WebTA effective in providing feedback on antipatterns, enhancing teaching and learning, reducing coding fears, and promoting good programming habits. Challenges included integration with other platforms and a focus on syntax.

Future development should enhance WebTA's integration with educational platforms and expand its use across different programming courses to evaluate its adaptability. Addressing identified challenges will improve usability and efficiency.

Further research should measure WebTA's impact, focusing on student and instructor perceptions. This involves gathering quantitative data on its effectiveness (RQ1) and examining its impact on teaching and learning (RQ2 and RQ3). Analyzing these perspectives will offer insights into WebTA's overall impact and guide future enhancements.

References

[1] L. Albrant, P. Pendse, L. E. Brown, J. Sticklen, M. Jarvie-Eggart, and L. C. Ureel, "Work-in-Progress: Preliminary Work Introducing Automated Code Critiques in First-Year Engineering MATLAB Programming," in 2023 IEEE Frontiers in Education Conference (FIE), College Station, TX, USA: IEEE, Oct. 2023, pp. 1–5. doi: 10.1109/FIE58773.2023.10343067.

[2] L. C. Ureel II, "Integrating a Colony of Code Critiquers into WebTA," in Seventh SPLICE Workshop at SIGCSE 2021 "CS Education Infrastructure for All III: From Ideas to Practice," 2021.

[3] L. C. Ureel II, L. E. Brown, J. Sticklen, M. Jarvie-Eggart, and M. Benjamin, "Work in Progress: The RICA Project: Rich, Immediate Critique of Antipatterns in Student Code," in Educational Data Mining in Computer Science Education (CSEDM) Workshop, Jul. 2022.

[4] P. Kinnunen and B. Simon, "Experiencing programming assignments in CS1: the emotional toll," in Proceedings of the Sixth international workshop on Computing education research, Aarhus Denmark: ACM, Aug. 2010, pp. 77–86. doi: 10.1145/1839594.1839609.

[5] Z. Gan, Z. An, and F. Liu, "Teacher Feedback Practices, Student Feedback Motivation, and Feedback Behavior: How Are They Associated With Learning Outcomes?," Front. Psychol., vol. 12, p. 697045, Jun. 2021, doi: 10.3389/fpsyg.2021.697045.

[6] C. M. Amerstorfer and C. Freiin Von Münster-Kistner, "Student Perceptions of Academic Engagement and Student-Teacher Relationships in Problem-Based Learning," Front. Psychol., vol. 12, p. 713057, Oct. 2021, doi: 10.3389/fpsyg.2021.713057

[7] O'Leary, M. (2020). Classroom Observation: A Guide to the Effective Observation of Teaching and Learning (2nd ed.). Routledge. https://doi-org.services.lib.mtu.edu/10.4324/9781315630243.

[8] O. A. Adeoye-Olatunde and N. L. Olenik. (2021). Research and scholarly methods: Semi-structured interviews. Journal Of The American College Of Clinical Pharmacy. https://doi-org.services.lib.mtu.edu/10.1002/jac5.1441

[9] G. Terry, N. Hayfield, V. Clarke, and V. Braun. (2017) Chapter 2: Thematic Analysis. Sage Handbook of Qualitative Research in Psychology, 2nd ed. C. Willig and W. Stainton-Rogers, Eds.. Sge Publications, Ltd, London. ISBN 978-1-4739-2521-2. 17-37.