AI-AFACT: Designing AI-Assisted Formative Assessment of Coding Tasks in Web Development Education

Franz Knipp[1,2][a] and Werner Winiwarter[3][b]
[1]University of Applied Sciences Burgenland, Eisenstadt, Austria
[2]University of Vienna, UniVie Doctoral School Computer Science DoCS, Vienna, Austria
[3]University of Vienna, Faculty of Computer Science, Vienna, Austria

Keywords: Computing Education, Artificial Intelligence, Formative Assessment, Web Development Education, Large Language Model, Design Science Research.

Abstract: Large Language Models (LLMs) are finding their way into computer science education. In particular, their natural language capabilities allow them to be used for formative assessment of student work, with the goal of reducing teacher time. However, initial research shows that there are still weaknesses in their use. To overcome this, this paper presents a design for an assessment tool that combines an LLM with a human-in-the-loop approach to ensure high-quality feedback. The proposed system focuses on the assessment of student submissions in the field of web technologies, which can be evaluated in different ways, including the content of the submitted files and the graphical output. Therefore, the use of a multimodal LLM is being considered. The innovative approach of a continuous learning system could significantly improve the efficiency of the assessment process, benefiting both teachers and students.

1 INTRODUCTION

Feedback is an essential part of the educational process, occurring in the evaluation of work as a summative assessment and through a formative assessment "defined as information communicated to the learner that is intended to modify his or her thinking or behavior to improve learning" (Shute, 2008). Formative assessments play a crucial role in deepening understanding. It significantly influences motivation and can improve learning outcomes (Leenknecht et al., 2021). In addition, open-ended assignments can improve the students' motivation by allowing them to be creative in finding solutions (Sharmin, 2022).

This paper focuses on the assessment of programming work in the field of web technologies that was specifically designed with HTML, CSS, and JavaScript. Submissions to such assignments are typically open-ended and can be evaluated in different ways, including the content of the files, their syntactic validity, the graphical implementation of the task, and by running automated tests in a browser (Siochi and Hardy, 2015).

Manual review of such web projects is time-consuming (Fu et al., 2008). In lessons with numerous feedback cycles, teachers accumulate substantial amounts of work. Learners often make similar mistakes, resulting in correspondingly similar feedback. Therefore, automation in this area can provide valuable support.

Automated assessment tools have been used in the field of computer science education for many decades. They check the submissions against a set of rules defined by the teacher. This restricts assignments to simple and constrained tasks. Therefore, they don't work well for open-ended assessments and their use for open-ended web projects has only been investigated to a limited extent (Messer et al., 2023).

Generative AI, based on Large Language Models (LLMs), is already in use to generate feedback to students (Neo et al., 2024; Azaiz et al., 2024). To overcome the limitations of traditional rule-based automated assessment tools, the integration of an LLM into the assessment process is a promising alternative. By leveraging the capabilities of LLMs, it is possible to automate the evaluation of web projects with a higher degree of flexibility and adaptability compared to rule-based systems.

[a] https://orcid.org/0009-0002-1811-9782
[b] https://orcid.org/0000-0002-8343-1257

Knipp, F. and Winiwarter, W.
AI-AFACT: Designing AI-Assisted Formative Assessment of Coding Tasks in Web Development Education.
DOI: 10.5220/0013430500003932
In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU 2025) - Volume 2*, pages 379-386
ISBN: 978-989-758-746-7; ISSN: 2184-5026

However, it is important to acknowledge the current limitations of LLMs in this domain. General-purpose LLMs and even programming-specialized variants often exhibit limited support for the breadth and depth of web technologies (Li et al., 2023).

To pave the way to filling this gap, the objectives of this paper are to specify the requirements for an automated assessment tool and – based on the literature – to derive an architectural design that is intended to serve as a blueprint for its implementation by the authors as well as other interested scientists and developers.

The research questions pursued in this paper are:

RQ1. Which requirements should an automated assessment tool for web development education fulfill to serve as an effective and efficient tool?

RQ2. Based on the requirements identified in RQ1, which architectural design can be derived following a design science research approach?

This paper proposes a design for an assessment tool for web development education that integrates a multimodal LLM with a human-in-the-loop approach to establish a continuous learning loop to ensure high-quality feedback, even for cases not yet covered by existing LLMs.

The remainder of this paper is organized as follows. In Section 2, we first present some relevant background and related work. In Section 3, we then introduce our research methodology based on the design science research paradigm, before describing the individual DSRM activities (problem identification, definition of objectives, architecture of the solution, demonstration and evaluation) in Section 4 in more detail. We conclude the paper in Section 5 with some thoughts on future work.

2 BACKGROUND

2.1 Automated Grading and Feedback

Tools for automatic assessment of programming courses date back to 1960 (Hollingsworth, 1960) and have evolved since then.

The systematic literature review conducted by Messer et al. (2023) highlights an increasing interest in the development and implementation of automated grading tools in computer science education. They review the period from 2017 to 2022, when unit testing emerged as the most prevalent method, limiting the applicability of the grading tools to short, well-defined assignments as opposed to more open-ended tasks. Machine learning techniques have already been integrated into some grading tools, but Messer et al. (2023) did not consider LLMs due to their recent emergence. Notably, two studies examined the grading time required by educators and reported a positive impact of semi-automated processes compared to manual grading (Insa et al., 2021; Muuli et al., 2017). Some of the tools use linters and software metrics designed for professional software developers, which can be overwhelming and sometimes confusing for students. Messer et al. (2023) recommend to further investigate automatic assessment of web applications and open-ended assignments.

2.2 Automated Assessment Tools for Web Development Education

Although Messer et al. (2023) categorize three of the tools under the category of web languages, only Nguyen et al. (2020) integrate CSS and HTML. This assessment system evaluates the coding style of group projects to determine the individual contributions of group members. It incorporates a code quality checker, security checker, and coding style checker into a Continuous Integration (CI) pipeline. This system is employed in a second-year course focused on web front-end development (Nguyen et al., 2020).

An older tool is ProtoAPOGEE by Fu et al. (2008). This tool is used for advanced courses, integrating concepts such as security, database integration, and reliability. To test the correctness of the submitted project, test cases created by the teachers are available, which test the submitted website step by step in a controlled browser.

A similar concept is followed by WebWolf by Siochi and Hardy (2015). The assessment is based on test cases, comparable to unit tests, that are created by the teachers and programmatically executed through a controlled browser.

ProgEdu4Web by Duong and Chen (2024) takes it a step further by integrating tools that check the syntax and quality of the submitted code. Here too, the test cases for functional tests must be provided by the teacher. Additionally, it allows for group work.

2.3 LLMs for Software Development

With the advent of LLMs, their application in software engineering tasks has been actively explored since 2018, as demonstrated by Hou et al. (2024). This field of research is highly dynamic, with dozens of publications emerging monthly at the time of writing. Several models are available, both commercial and open source, including specialized models such as Codex (Chen et al., 2021) and general models such as OpenAI's GPT-4. These models are capable of performing tasks such as generating code from textual input, summarizing or explaining code segments, creating tests, localizing and fixing bugs, or performing code reviews. Thus, the use of LLMs will bring fundamental changes in the field of software development, improving the productivity during coding and reducing errors (Hou et al., 2024).

Notably, an LLM such as GPT-4 is capable of solving coding problems used for software engineering interviews that are comparable to human performance, as shown in Bubeck et al. (2023).

2.4 Multimodal Models

Building on the success of LLMs in processing natural language, research is expanding to include the integration of non-textual information such as audio, video and images. According to Naveed et al. (2024), incorporating these modalities not only broadens the application fields of LLMs but also enriches the context, enhancing the situational understanding and decision-making capabilities.

2.5 Use of LLMs in Computer Science Education

Due to the extensive possibilities of LLMs in the field of computer science, they are finding their way into computer science lessons. Anishka et al. (2024) describe the use of an LLM based on GPT-3.5-Turbo in an advanced computing class on distributed systems as support tool for teaching assistants by generating exam questions and evaluating the answers. It can also check code snippets. The results show that LLMs are very effective in generating exam questions, while the feedback on the answers at times shows the effect of hallucination, which limits accuracy. Apart from this, the feedback was constructive and balanced, providing assistance to the teaching assistants. The feedback on the code snippets was comprehensive, but improvements are needed.

The research by Smolić et al. (2024) compares the effects of zero-shot prompting and few-shot prompting to obtain comments and grades of programming tasks in C. It uses GPT-3.5 and Gemini as LLMs. Few-shot prompting shows an improvement in grading and generating more accurate comments. However, context is not well maintained. Therefore, a human-in-the-loop approach is required.

Gabbay and Cohen (2024) investigate the use of LLMs to close the gap of automated test-based feedback tools in a programming course for Python. The presented program gives students feedback on submitted solutions. It uses customized prompts that prevent the display of the correct solution or irrelevant information. When comparing GPT-4 with GPT-3.5, the former shows better performance. Nevertheless, the rate of false positives is high. They are not recognized by beginners and thus have a negative impact on the learning process.

The approach described by Liu and M'Hiri (2024) goes beyond the evaluation and assessment of coding assignments. It uses GPT-3.5 as an LLM-based virtual teaching assistant that provides help in different categories such as homework questions, code feedback, and explanation of complex concepts in an undergraduate course in Python. To do this, it uses a multi-step process in which students' questions are categorized. Depending on the category, the answer is generated differently. The answer is then automatically quality checked. Unsatisfactory answers are regenerated with a different set of LLM parameters until they pass the quality check. On the one hand, the comparative study shows comparable results to human teaching assistants on non-assignment specific questions, combined with higher ratings for clarity and engagement. On the other hand, the virtual teaching assistant sometimes gives answers that are overwhelming for novices. This underscores the importance of human supervision.

3 METHODOLOGY

Design science research is characterized by providing solutions to real-world problems using information technology. It must produce a viable artifact in the form of a model, method, or implementation that is rigorously evaluated (Hevner and Chatterjee, 2010).

Design science research has evolved over the last decades with hundreds of scientific publications (Akoka et al., 2023). Recently, it found its way into higher education in a number of projects (Apiola and Sutinen, 2021).

Therefore, design science research fits well into addressing the problem mentioned in Section 1. The selected research method is Design Science Research Methodology (DSRM), as described by Peffers et al. (2007).

The process consists of six activities, several of which are run through iteratively or might be omitted as well as shown in Figure 1. These activities are (Peffers et al., 2007):

1. Problem identification and motivation
2. Define the objectives
3. Design and development

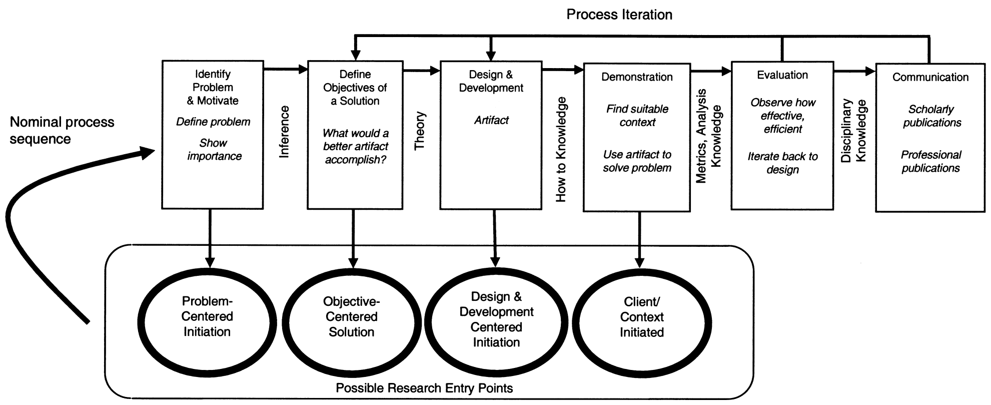
Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 1: DSRM process model (Peffers et al., 2007).

4. Demonstration
5. Evaluation
6. Communication

The model allows four entry points into the process (Peffers et al., 2007):

1. Problem-centered initiation
2. Objective-centered solution
3. Design & development centered initiation
4. Client/context initiated

The step-by-step application of DSRM ensures that the solution found solves the problem at hand, as numerous publications have shown, e.g. Clauss (2024) and Neo et al. (2024). In the case of this paper, the starting point is problem-centered.

4 APPLICATION OF DSRM

4.1 Problem Identification

The teaching of basic knowledge in HTML and CSS, as well as integration with JavaScript, is part of several curricula. Due to the positive impact on motivation, the use of open-ended assignments is recommended, where students develop their submissions using their creativity.

To support students in the learning process, submissions are not only assessed summatively but also provided with extensive feedback. The goal is for students to avoid repeating mistakes in future assignments and to apply what they have learned correctly.

When reviewing submissions, different perspectives must be taken: Are the files syntactically correct? Have the HTML elements been used in a semantically correct way? Are the requirements specified in the assignment met? What does the generated page look like?

If JavaScript is also part of the course content, the questions expand: Does the JavaScript code work correctly? Are the specified functionalities implemented?

The feedback must be adapted to the students' level of knowledge. It must be structured in such a way that it helps students to independently correct the noted points. Understandably, the feedback must not contain any incorrect or misleading information.

This feedback is manually created in many courses by the teacher reviewing the submitted code and the resulting website. Sometimes, syntax checkers and similar tools are used. This process is time-consuming and therefore limits the number of assignments or the size of the group that a single teacher can evaluate. At the same time, the errors in the submissions are very similar, so the points in the feedback are repeated.

To remedy this, an automation of the assessment process is suggested, which detects possible errors and generates feedback. Unfortunately, existing solutions do not seem suitable as they are based on rules or tests that must be created in advance by the teacher. These rigid rules are only partially compatible with open-ended assignments. The rules can only cover the cases considered by the teacher, so some properties of the solutions remain unconsidered in the feedback process, and other errors do not trigger feedback.

Conversely, LLM-based approaches to the assessment of programming tasks might provide natural language feedback without having to define all possible cases in advance. However, research also highlights the shortcomings of such systems: The context of the task is inadequately considered. There is misinformation in the answers. The answers can be overwhelming, especially for beginners.

Current LLM-based assessment systems focus on the implementation of individual functions in languages like Python. Web sites implemented using syntactically different technologies such as HTML, CSS, and JavaScript in multiple files are much more complex and require different perspectives as described above, so existing solutions cannot simply be ported to this application area.

Given the importance of web applications in daily life, web development education is part of many curricula. Improving the teaching in this area is therefore a worthwhile goal.

4.2 Definition of Objectives

The requirements for the solution to the presented problem are defined as objectives. For clarity, the objectives are grouped into three categories and numbered.

Base Objectives

B1. The solution must accept students' work.

B2. The solution must provide feedback to students on their submissions.

Domain-specific Objectives

D1. The submitted work will be checked for syntactic correctness.

D2. The submitted work will be checked for the semantically correct use of HTML tags.

D3. The submitted work will be checked for adherence to code style guidelines.

D4. The submitted work will be checked for correct implementation of the assignment requirements.

D5. The appearance of the submitted website will be considered during the review.

D6. The functionality of the submitted website will be considered during the review.

Quality Criteria

Q1. The quality of the feedback must be ensured.

Q2. The feedback should be consistent.

Q3. The feedback must consider the context of the assignment. This includes the content of the assignment itself as well as the current course progress and the already taught content.

Q4. The use of the system must lead to a time saving for the teacher compared to the manual review of each submission.

4.3 Architecture of the Solution

Based on these objectives a prototype will be developed. It will implement the assessment process shown in Figure 2, consisting of the following steps:

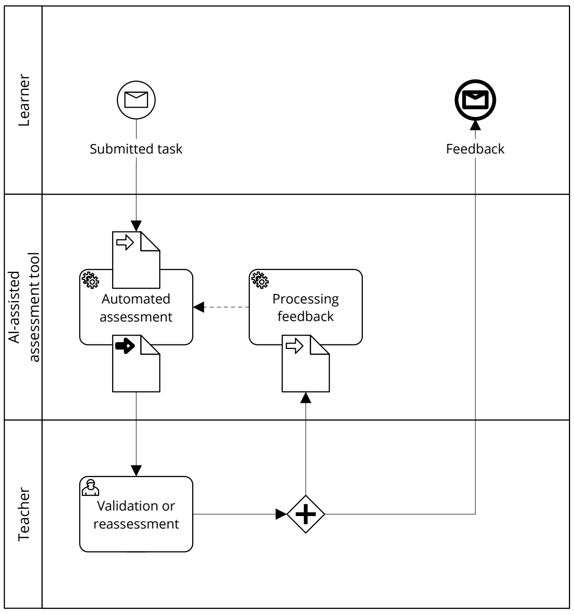

Figure 2: Assessment process.

1. The task to be assessed is submitted by the learner.
2. AI-powered software makes a preliminary assessment of the task. The software knows the task definition.
3. The teacher reviews the automatic assessment. This step ensures quality assurance, allowing the teacher to control the feedback and add additional information or address any shortcomings not identified by the AI.
4. The manual feedback is used to fine-tune the AI for future automatic assessments.
5. The feedback is returned to the learner.

A more detailed view in Figure 3 shows the phases of the data processing in the assessment tool:

1. **Input.** The submission of the student serves as the input data for the workflow.
2. **Preprocessing.** The data is prepared for the next phase through:
   - **Transformation.** The file contents are normalized. A parser might generate an abstract syntax tree of the JavaScript files.
   - **Derivation.** Additional data is retrieved by calling programs or calling external APIs to augment the submission. This may include integrating linters, validators, execution outputs, or screenshots.
3. **Processing.** The raw submission, along with the results of transformation and derivation, is assessed by an LLM. The LLM is aware of

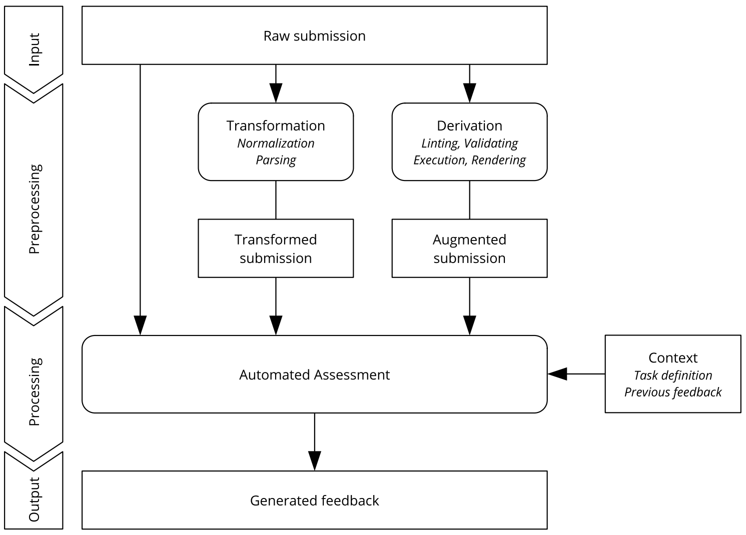

Figure 3: Data flow.

the context, including the task definition, previous submission-feedback pairs, and other relevant data.

4. **Output.** The generated feedback is delivered to the teacher for further processing.

Processing both text and images is feasible if a multimodal LLM is utilized at the processing phase.

4.4 Demonstration and Evaluation

Part of the DSRM process is the evaluation of the created artifact, in this case, the architecture for a solution. Since there is no implementation yet, the demonstration activity is omitted. It is examined how and to what extent the architecture can meet the required objectives.

The base objectives B1 and B2 are addressed through integration into a Learning Management System (LMS) or by providing a web interface that offers the required functionalities. This interface also provides the teacher with interaction possibilities with the assessment tool.

The domain-specific objectives are met through data preparation and the chosen LLM. Objectives D1 and D3 are addressed by integrating various sources in the preprocessing stage. For D2, D4, and D6, the understanding of the LLM is utilized, which considers the assignment and previous feedback from the teacher. The use of a multimodal LLM enables the fulfillment of objective D5.

To meet quality criterion Q1, a human-in-the-loop approach is followed, ensuring that all feedback is reviewed by the teacher before being sent to the student. The use of an automated system ensures Q2. Since the LLM uses previous feedback as a basis, Q3 is fulfilled. To test the achievement of objective Q4 in implementation, analytics functions are built in to evaluate the time expenditure.

As the system continuously learns with each submission, the hypothesis is that the effort for the teacher will decrease over time. Testing this hypothesis is part of further research.

5 DISCUSSION AND FUTURE WORK

Following the DSRM process model, the research questions posed in Section 1 are addressed.

The requirements for an automated assessment tool for web development education are established as objectives in Subsection 4.2, thereby providing an answer to RQ1. These requirements are derived from the problem definition, which encapsulates the findings from the literature reviewed in Section 2.

In response to RQ2, the architecture of a solution is detailed in Subsection 4.3. This section illustrates the processing of submissions and outlines the essential components to meet the previously specified requirements.

In the subsequent iteration of the DSRM, the focus will be on selecting an appropriate LLM that demonstrates a foundational “understanding” of the web technologies–namely HTML, CSS, and JavaScript–and their interdependent functionality. Current benchmarks for evaluating LLMs are insufficient for these technologies, necessitating the development of a new benchmark. This benchmark will be tailored to assess the competencies essential for web development education and will facilitate the comprehensive evaluation of LLMs.

Upon identifying a suitable LLM, a prototype encapsulating the proposed process will be implemented. This prototype will undergo testing through hypothetical scenarios created by the authors, drawing upon substantial teaching experience in the field. Subsequently, as part of an advanced iteration, there are plans to integrate the developed system into university courses to evaluate the efficacy in real-life teaching.

6 CONCLUSIONS

Manually assessing student submissions is a time-consuming task. For this reason, tools that automate the assessment and thus simplify the teacher's work have long been used in computer science. However, these methods reach their limits with complex tasks, especially with open-ended tasks that can be solved creatively by the students. Homework in web development education often falls into this category.

In order to solve this problem, this paper presents an architecture for automated assessment. It is based on the use of Large Language Models (LLMs), which in recent years have penetrated many areas of software development as well as education, where they can cope with numerous tasks.

The approach using an established methodology, specifically the Design Science Research Methodology (DSRM), demonstrates how tools for the educational sector can be systematically created. DSRM was developed from the literature available at the time and can be considered comprehensive. It ensures that no steps are overlooked. The accompanying documentation of each step increases the traceability of the research project. The iterative approach allows the solution to be built and evaluated step by step.

In the first iteration documented here, the problem to be solved is described. Objectives for the solution are derived from the problem statement, thus defining the requirements for an automated assessment tool. The artifact created is an architectural design that is evaluated against the objectives.

The authors' vision is a tool that helps teachers to give more consistent feedback in less time. The architectural design lays the groundwork for the development of such a tool, ensuring it meets the defined requirements and establishes the foundation for later evaluations. A critical metric for these evaluations will be the quantification of time savings for instructors, which stands as a principal objective of automation.

In conclusion, this research highlights the need for automated assessment tools for web development education. Furthermore, it has showcased the efficacy of design science research as a robust approach for tackling complex use cases, affirming its value as methodological framework in the field of educational technology.

ACKNOWLEDGEMENTS

The authors would like to thank the University of Applied Sciences Burgenland, the University of Vienna and the UniVie Doctoral School Computer Science (DoCS) for their support and contribution to this research.

DeepL, DeepL Write, ChatGPT and Paperpal provided help in translating and refining the text.

REFERENCES

Akoka, J., Comyn-Wattiau, I., and Storey, V. C. (2023). Design Science Research: Progression, Schools of Thought and Research Themes. In Gerber, A. and Baskerville, R., editors, *Design Science Research for a New Society: Society 5.0*, volume 13873, pages 235–249. Springer Nature Switzerland, Cham.

Anishka, Sethi, D., Gupta, N., Sharma, S., Jain, S., Singhal, U., and Kumar, D. (2024). TAMIGO: Empowering Teaching Assistants using LLM-assisted viva and code assessment in an Advanced Computing Class. arXiv:2407.16805 [cs.HC].

Apiola, M. and Sutinen, E. (2021). Design science research for learning software engineering and computational thinking: Four cases. *Computer Applications in Engineering Education*, 29(1):83–101.

Azaiz, I., Kiesler, N., and Strickroth, S. (2024). Feedback-Generation for Programming Exercises With GPT-4. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, pages 31–37, Milan Italy. ACM.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., Nori, H., Palangi, H., Ribeiro, M. T., and Zhang, Y. (2023). Sparks of Artificial General Intelligence: Early experiments with GPT-4. arXiv:2303.12712 [cs.cL].

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG].

Clauss, A. (2024). Facilitating Competence-Oriented Qualification in New Work: Evaluation of a Platform Prototype:. In *Proceedings of the 16th International Conference on Computer Supported Education*, pages 659–668, Angers, France. SCITEPRESS - Science and Technology Publications.

Duong, H.-T. and Chen, H.-M. (2024). ProgEdu4Web: An automated assessment tool for motivating the learning of web programming course. *Computer Applications in Engineering Education*, 32(5):e22770.

Fu, X., Peltsverger, B., Qian, K., Tao, L., and Liu, J. (2008). APOGEE: Automated project grading and instant feedback system for web based computing. *ACM SIGCSE Bulletin*, 40(1):77–81.

Gabbay, H. and Cohen, A. (2024). Combining LLM-Generated and Test-Based Feedback in a MOOC for Programming. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, pages 177–187, Atlanta GA USA. ACM.

Hevner, A. and Chatterjee, S. (2010). *Design Research in Information Systems: Theory and Practice*, volume 22 of *Integrated Series in Information Systems*. Springer US, Boston, MA.

Hollingsworth, J. (1960). Automatic graders for programming classes. *Communications of the ACM*, 3(10):528–529.

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., and Wang, H. (2024). Large Language Models for Software Engineering: A Systematic Literature Review. arXiv:2308.10620 [cs.SE].

Insa, D., Pérez, S., Silva, J., and Tamarit, S. (2021). Semi-automatic generation and assessment of Java exercises in engineering education. *Computer Applications in Engineering Education*, 29(5):1034–1050.

Leenknecht, M., Wijnia, L., Köhlen, M., Fryer, L., Rikers, R., and Loyens, S. (2021). Formative assessment as practice: The role of students' motivation. *Assessment & Evaluation in Higher Education*, 46(2):236–255.

Li, A., Wu, J., and Bigham, J. P. (2023). Using LLMs to Customize the UI of Webpages. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–3, San Francisco CA USA. ACM.

Liu, M. and M'Hiri, F. (2024). Beyond Traditional Teaching: Large Language Models as Simulated Teaching Assistants in Computer Science. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 743–749, Portland OR USA. ACM.

Messer, M., Brown, N. C. C., Kölling, M., and Shi, M. (2023). Automated Grading and Feedback Tools for Programming Education: A Systematic Review. *ACM Transactions on Computing Education*, page 3636515.

Muuli, E., Papli, K., Tõnisson, E., Lepp, M., Palts, T., Suviste, R., Säde, M., and Luik, P. (2017). Automatic Assessment of Programming Assignments Using Image Recognition. In Lavoué, É., Drachsler, H., Verbert, K., Broisin, J., and Pérez-Sanagustín, M., editors, *Data Driven Approaches in Digital Education*, volume 10474, pages 153–163. Springer International Publishing, Cham.

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2024). A Comprehensive Overview of Large Language Models. arXiv:2307.06435 [cs.CL].

Neo, G., Moura, J., Almeida, H., Neo, A., and Freitas Júnior, O. (2024). User Story Tutor (UST) to Support Agile Software Developers:. In *Proceedings of the 16th International Conference on Computer Supported Education*, pages 51–62, Angers, France. SCITEPRESS - Science and Technology Publications.

Nguyen, B.-A., Ho, K.-Y., and Chen, H.-M. (2020). Measure Students' Contribution in Web Programming Projects by Exploring Source Code Repository. In *2020 International Computer Symposium (ICS)*, pages 473–478, Tainan, Taiwan. IEEE.

Peffers, K., Tuunanen, T., Rothenberger, M. A., and Chatterjee, S. (2007). A Design Science Research Methodology for Information Systems Research. *Journal of Management Information Systems*, 24(3):45–77.

Sharmin, S. (2022). Creativity in CS1: A Literature Review. *ACM Transactions on Computing Education*, 22(2):1–26.

Shute, V. J. (2008). Focus on Formative Feedback. *Review of Educational Research*, 78(1):153–189.

Siochi, A. C. and Hardy, W. R. (2015). WebWolf: Towards a Simple Framework for Automated Assessment of Webpage Assignments in an Introductory Web Programming Class. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, pages 84–89, Kansas City Missouri USA. ACM.

Smolić, E., Pavelić, M., Boras, B., Mekterović, I., and Jagušt, T. (2024). LLM Generative AI and Students' Exam Code Evaluation: Qualitative and Quantitative Analysis. In *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, pages 1261–1266, Opatija, Croatia. IEEE.