Flipping the grade: Empowering students through self-assessment in informatics

Maria GUTU

Technical University of Moldova, Chisinau, Moldova

maria.gutu.md@gmail.com

Abstract: *This paper illustrates how self-assessment and self-grading in Informatics education transform traditional grading into an active learning process. By shifting the focus from teacher-driven evaluation to student-led reflection, self-grading fosters autonomy, metacognition, and problem-solving skills. The paper shows how structured approaches – such as rubric-based self-grading, peer-reviewed assessment, and digital portfolios – enhance students' ability to critically evaluate their work and develop computational thinking. However, challenges such as grading reliability, student bias, and the need for structured guidance must be addressed. The paper highlights strategies to mitigate these issues, including standardized rubrics, justification mechanisms, and moderation through peer and teacher reviews. It demonstrates how balancing student autonomy with oversight ensures fairness and deepens engagement. By embedding self-regulation into assessment, the "Flipping the Grade" model shifts grading from a static measure to an iterative learning process. This paper presents self-grading as a powerful tool in competency-based Informatics education, fostering independence, accountability, and lifelong learning skills.*

Keywords: Self-assessment, Self-grading, Assessment criteria, Informatics, Active learning.

1. Introduction

Assessment has traditionally been a teacher-controlled process, where students receive grades based on external evaluations rather than their understanding of their learning progress. However, there is a growing shift towards student-driven self-assessment in modern education (McMillan & Hearn, 2008; Yan, Chiu & Ko, 2020; Gutu, 2022a; Wong & Taras, 2022; Gutu, 2023b), where learners actively evaluate and grade their work (Andrade, 2008; Weiss, 2018; Carroll, 2020; Panadero et al., 2023). This shift is particularly relevant in Informatics education, where problem-solving, debugging, and optimising code require continuous self-reflection and iterative improvement.

Implementing structured self-assessment models helps students develop critical thinking, self-regulation, and a deeper understanding of Informatics

https://doi.org/10.58503/icvl-v20y202530

concepts. This paper explores various models for integrating self-assessment and self-grading into Informatics education, including rubric-based evaluation, peer-reviewed self-grading, and digital portfolios. The focus is on how teachers can implement these models effectively, ensuring that students assess their work accurately and engage in meaningful self-reflection to enhance their programming skills.

2. Theoretical framework

"Flipping the Grade" is an innovative approach that transforms assessment from a teacher-centred practice into a student-led process. Instead of relying solely on teacher-assigned grades, students are provided explicitly stated assessment criteria and scoring indicator rubrics to evaluate their work, reflect on their learning, and assign themselves a grade based on their performance. This approach aligns with competency-based education, which emphasises the attainment of skill mastery rather than reliance on rote memorisation. Additionally, it aligns with the flipped learning model (Gutu, 2023a), which fosters student autonomy by encouraging learners to take an active role in their educational development.

The foundation of "Flipping the Grade" lies in constructivist learning theories, which emphasise the role of learners as active participants in their education. Constructivism posits that knowledge is actively constructed rather than passively received, meaning that students learn best when reflecting, self-regulation, and problem-solving. In Informatics education (Caspersen et al., 2022; Gutu, 2022b, 2023a), where students work with complex problem-solving tasks, algorithms, and debugging, critically evaluating one's work is crucial for developing computational thinking skills.

Metacognition, closely linked to constructivism, refers to the ability to reflect on one's thinking and learning processes. When students are engaged in self-assessment and self-grading, they actively practice metacognitive regulation (Giraldo & Herold, 2023; Nechyporuk & Romaniuk, 2024), a crucial component of independent learning. In Informatics education, particularly in programming, learning extends beyond the correct execution of syntax; it requires strategic problem-solving, logical reasoning, and continuous self-monitoring. According to Gutu (2022a, 2023a), students cultivate self-regulated learning skills through structured self-assessment, enabling them to plan their approach to programming tasks, monitor their progress, and assess their outcomes. These competencies are not only fundamental for academic achievement but also essential for long-term adaptability (Westover, 2024) and professional growth in technological fields (Verano-Tacoronte, Bolívar-Cruz & González-Betancor, 2015).

Moreover, self-grading within the flipped learning paradigm further amplifies student engagement and autonomy. In a flipped classroom model (Gutu, 2023a; Xia, 2023; Deng, Feng & Shen, 2024), theoretical instruction is delivered outside the classroom through digital resources, allowing in-class time devoted to

active problem-solving and applied learning. Extending this model to the grading process reinforces students' responsibility for their learning by requiring them to critically assess their work, identify areas for improvement, and assign themselves a grade based on provided assessment criteria (Stevens & Levi, 2005; Gutu, 2022a). This approach aligns with competency-based education, prioritising mastery of skills over accumulating grades. By self-grading, students transition from a fixed mindset, where assessment is seen as an endpoint, to a growth-oriented perspective, wherein evaluation is viewed as an iterative improvement process. Rather than perceiving grades as static indicators of ability, students develop a reflective mindset emphasising continual skill refinement.

According to Gutu (Gutu, 2022a), the effectiveness of self-assessment and self-grading in Informatics education is contingent upon providing clear and structured assessment criteria. The absence of explicit grading guidelines may result in inaccurate self-evaluation (Dixon et al., 2020), either due to overestimating one's abilities or an undue lack of confidence in one's work. Teachers must employ structured rubrics that define specific evaluation parameters (Andrade, 2008; Muhammad, Lebar & Mokshein, 2018), including code correctness, efficiency, readability, debugging methodologies, and problem-solving approaches to mitigate such inconsistencies. These rubrics serve as objective benchmarks, ensuring student evaluations align with established learning objectives and maintaining consistency across self-assessments.

In addition to rubrics, guided reflection prompts enhance the depth of self-assessment by prompting students to justify their evaluations. Reflection questions (e.g., "What were the primary challenges I encountered while coding?" "How does my solution compare to alternative approaches in terms of efficiency?" and "What modifications could improve my implementation?") encourage higher-order cognitive engagement.

By shifting the responsibility of assessment from teachers to students, the "Flipping the Grade" model fosters a learning culture centred on accountability, self-regulation, and lifelong learning. In Informatics education, continuous learning and adaptability are essential for success (Caspersen et al., 2022); this approach enhances students' technical competencies and cultivates critical thinking and self-directed learning behaviours. When implemented effectively, self-grading transforms traditional assessment into a dynamic and reflective learning process, equipping students with the skills necessary for both academic and professional excellence in computational fields.

3. Models of implementation for "Flipping the Grade" in informatics

3.1 Self-grading rubrics with assessment criteria and scoring indicators

Implementing self-assessment and self-grading in Informatics requires structured approaches that ensure both objectivity and student engagement. One of the most effective methods for guiding students in this process is the use of detailed

grading rubrics. As outlined by Stevens and Levi (2005), Brookhart (2013), Dawson (2017), and Muhammad, Lebar, and Mokshein (2018), rubrics are designed with distinct elements tailored to specific aspects of assessment, such as task-specific criteria, task descriptions, and analytic cumulative scoring. These elements aim to clarify the task, identify the level of mastery, and guide students through new concepts.

In this context, the self-grading rubric developed in this study provides students with explicit assessment criteria and scoring indicators, enabling them to evaluate their work effectively. This structured approach ensures that students comprehend key aspects of quality programming, including code correctness, readability, and debugging strategies. Furthermore, the assessment criteria facilitate critical analysis of students' coding practices and other Informatics-related activities, thereby fostering self-regulation (Andrade & Brookhart, 2016; Gutu, 2022a) and enhancing independent problem-solving skills.

The process of self-grading begins with the teacher designing a comprehensive description rubric that clearly defines the parameters for assessment. This rubric typically includes multiple dimensions of evaluation, such as functionality (whether the code produces the expected output), efficiency (how well the algorithm optimises computational resources), readability (the clarity and organisation of the code), and debugging (the student's ability to identify and correct errors). Before engaging in self-assessment, students receive explicit instruction on using the rubric, often through guided examples where they practice grading sample code snippets. This preparatory step ensures consistency and accuracy in the self-assessment process.

Once students have completed their programming tasks, they apply the rubric with descriptions to their work, carefully assessing each criterion. This process encourages them to critically reflect on their problem-solving approach, identifying strengths and areas for improvement (Andrade & Brookhart, 2016; Dawson, 2017; Muhammad, Lebar & Mokshein, 2018; Gutu, 2023b). To enhance accountability, students are required to justify their self-assigned grades through written explanations or brief verbal reflections. This justification process reinforces metacognitive skills and minimises the risk of inflated or inaccurate self-grading. By articulating the reasoning behind their evaluations, students become more conscious of the quality of their work and develop a deeper understanding of Informatics concepts.

Furthermore, students are encouraged not to refer to the assessment criteria before completing the task. They should only consult these criteria if they do not fully understand the task requirements. In this regard, within the same classroom, the same criteria can serve different purposes: either as a guide for task completion or as a tool for assessing the level of mastery. This approach fosters student engagement and promotes independent task execution.

An example of this implementation in an Informatics classroom involves a programming task focused on loop structures in C++. Students are assigned a task

that requires them to write a program utilising loops to perform a computational task (Table 1).

Table 1. Example of a problem statement

The Bunker Access Code Puzzle A secret bunker conceals essential information in a deserted city of ruins and mysteries. To uncover the events in this city, a group of explorers has decided to enter the bunker and investigate its contents. Access to the bunker is restricted by a unique numerical code, which can only be deciphered by solving a mathematical puzzle left behind by the engineers of the previous civilisation. Engraved on the massive surface of the bunker door are *n* numbered panels, each containing a distinct number. These numbers serve as the key to decrypting the bunker's access code. **Puzzle Rules:** • Examine each panel and read the inscribed number. • Determine whether the given number is prime. • Sum all the prime numbers identified on the *n* panels to obtain the final sum. • The sum of the identified prime numbers represents the bunker's access code. **Input Data:** • A natural number *n*, indicates the total number of panels. • *n* distinct natural values, each representing a number engraved on a panel. **Output Data:** • A natural number represents the sum of the prime numbers on the panels, constituting the bunker's access code. **Constraints and Specifications:** • $1 \leq n \leq 100$ • $2 \leq$ engraved value on a panel ≤ 1000000

After writing and debugging their code, students assess their performance using the assessment rubrics (Table 2). They evaluate whether their loops execute correctly, whether their algorithm is optimised for efficiency, their understanding of the problem statement, the correctness of summing prime numbers, the implementation of an optimised prime-checking function, input handling and adherence to constraints, and whether their code follows best practices for readability and clarity. They are encouraged to revise their work before finalising their self-assessment if they encounter errors. In addition to assigning a numerical grade, students provide a short reflection explaining how they arrived at their score, detailing any challenges they faced and the strategies they used to overcome them.

Table 2. Provided assessment criteria for the bunker access code solution

1.	**Criteria: Input Handling and Adherence to Constraints**	
	Excellent (20 pts)	Correctly processes all constraints, including boundary values (e.g., n=1, n=100, $2 \leq$ engraved value on a panel ≤ 1000000). Fully adheres to input rules and uniqueness conditions.
	Good (15 pts)	Handles most edge cases but overlooks one or two. Minor constraint issues (e.g., fails to enforce distinct values or mishandles n=99).
	Satisfactory (10 pts)	Manages some edge cases but fails in extreme cases. Partial constraint adherence (e.g., incorrect range handling).
	Needs improvement (5 pts)	Fails to manage boundary conditions, leading to errors. Do not enforce constraints, resulting in incorrect outputs.

2.	**Criteria: Prime number identification**	
	Excellent (20 pts)	Implements an optimised prime-checking function with O(√n) time complexity.
	Good (15 pts)	Correctly identifies primes but with minor inefficiencies (e.g., unnecessary divisibility checks, no separate function).
	Satisfactory (10 pts)	Partially correct implementation; some incorrect results for certain numbers (e.g., incorrectly classifies *one* or other *numbers* as prime).
	Needs improvement (5 pts)	Incorrect prime-checking logic, leading to significant errors (e.g., missing actual primes or misidentifying composites).
3.	**Criteria: Summing prime numbers**	
	Excellent (20 pts)	Correctly sums only prime numbers as required and ensures accuracy.
	Good (15 pts)	Works correctly but lacks efficiency (e.g., unnecessary checks or redundant calculations).
	Satisfactory (10 pts)	Some errors in summation (e.g., including non-primes or missing primes in the sum).
	Needs improvement (5 pts)	Incorrect or missing logic summation leads to completely wrong results.
4.	**Criteria: Time complexity & Memory efficiency**	
	Excellent (20 pts)	Implements prime checking with O(√n) complexity. Uses minimal and necessary memory without redundancy.
	Good (15 pts)	Efficient but with minor optimizations missing (e.g., redundant calculations, extra loop iterations). Slightly higher memory use but no major inefficiencies.
	Satisfactory (10 pts)	Uses a less optimal approach (e.g., checks divisibility up to n instead of √n). Unnecessary memory usage (e.g., storing extra data instead of direct processing).
	Needs improvement (5 pts)	Highly inefficient (e.g., nested loops, brute force checking). Excessive memory usage due to redundant storage or poor management.
5.	**Criteria: Code clarity and readability**	
	Excellent (10 pts)	Well-structured, properly indented, and includes meaningful variable names and inline comments explaining key sections.
	Good (8 pts)	Mostly clear, with minor readability issues (e.g., inconsistent spacing or some unclear variable names). Includes some comments.
	Satisfactory (5 pts)	Somewhat readable but lacks consistent formatting. Does not include comments.
	Needs improvement (2 pts)	Poorly structured, making it difficult to follow. No comments, making it hard to understand the logic.
6.	**Criteria: Understanding of the problem statement**	
	Excellent (10 pts)	Clearly understands the problem and implements all requirements correctly, including constraints and conditions.
	Good (8 pts)	Demonstrates good understanding but misses minor details (e.g., slight deviations in expected input/output format).
	Satisfactory (5 pts)	Basic understanding, but some requirements are incomplete or misinterpreted (e.g., incorrect assumption about the number range).
	Needs improvement (2 pts)	Major misunderstanding of the problem, an incorrect or incomplete implementation that does not align with the given statement.
Total Score: /100 Points **90 - 100:** Outstanding implementation with correct logic, efficiency, and clarity. **75 - 89:** Good implementation with minor efficiency or edge case handling issues. **50 - 74:** Satisfactory but with noticeable logic, efficiency, or constraints errors. **Below 50:** Needs significant improvements in correctness, efficiency, and understanding.		

The assessment criteria in rubrics can be presented more explicitly by incorporating concrete aspects of code development, using specific code sequences to support students' understanding. This approach is particularly useful when introducing students to a new topic or concept. Alternatively, the criteria can be presented in a less detailed manner, as shown in Table 2, to encourage a more critical self-evaluation process. This type of approach requires a final collective analysis to ensure that all students have correctly applied and assessed their work using these criteria.

The selection of approach depends on the desired level of competency mastery in student development. When the criteria are used frequently, aspects related to *Code Clarity and Readability, Time Complexity & Memory Efficiency, Input Handling, and Adherence to Constraints* tend to be implicitly considered by students, thereby streamlining the self-evaluation process and reducing the time required for assessment.

To enhance the efficiency of the assessment process, these evaluation rubrics can be implemented online using platforms such as Google Forms, Microsoft Forms, Moodle, or any other system that enables automated score calculation. Additionally, these platforms allow students to review their selected responses for further reflection and reevaluation. Consequently, the option to complete the rubrics multiple times could be enabled to foster continuous learning and improvement.

Therefore, this model fosters a greater sense of responsibility and engagement among students. Rather than viewing assessment as an external judgment imposed by the teacher, students take ownership of their learning process, recognising assessment as a personal and academic growth tool. Furthermore, this approach aligns with competency-based education by emphasising skill mastery over task completion. Students develop an iterative mindset by repeatedly engaging in self-assessment, refining their problem-solving techniques and enhancing their programming abilities.

3.2 Peer-reviewed self-grading model

The peer-reviewed self-grading model is a structured approach that integrates collaborative assessment with self-reflection, fostering a deeper engagement with learning while promoting accuracy in self-assessment. Unlike traditional grading, where the teacher solely determines evaluation, this model introduces a two-step verification process: peer feedback (Topping, 2018; Gutu, 2022a) and self-grading based on structured reflection. This process strengthens students' ability to analyse their work critically and that of their peers, encouraging constructive dialogue, iterative improvement, and metacognitive awareness in Informatics education (Gutu, 2023a, 2023b).

The implementation of this model begins with students exchanging their programs before assigning themselves a grade. The exchange ensures that each

student evaluates a peer's work using teacher-provided assessment criteria. This stage encourages students to apply their analytical skills to a peer's program, providing structured feedback highlighting strengths and improvement areas. Peer feedback is a formative checkpoint (Kumar, Kenney & Buraphadeja, 2013; Simonsmeier et al., 2020), offering an external perspective that may uncover unnoticed errors or suggest optimisations that enhance code efficiency. Furthermore, peer assessment develops communication and critique skills (Double, McGrane & Hopfenbeck, 2020), which are essential for collaborative problem-solving in Informatics.

Following peer feedback, students return to their programs, reviewing the suggestions received before conducting their self-assessment. This reflection phase allows students to revise their code, address logical errors, improve structure, or optimise functionality. Once revisions are complete, students proceed with self-grading, assigning themselves a score based on the same rubric used for peer evaluation. This step requires them to justify their grade, explaining how their program aligns with the provided criteria and what changes were made based on peer feedback. The requirement for justification reinforces accountability and self-regulation, ensuring that students engage deeply with the assessment process rather than arbitrarily assigning grades.

An example of this model in practice can be illustrated through a sorting algorithm task. A student initially submits an implementation of bubble sort and exchanges it with a peer. The peer identifies that while the algorithm produces correct output, its efficiency could be improved by reducing unnecessary iterations. They provide this feedback, suggesting the optimised version with an early termination condition. Upon receiving the input, the students revisit their code, incorporate the proposed improvement, and re-evaluate their work. They then assign themselves a grade, justifying their decision by explaining how the peer feedback led to a more efficient implementation.

This iterative process of peer review, revision, and self-grading enhances students' ability to critically engage with their work while developing a more accurate understanding of evaluation criteria. It also reduces grading bias, as self-assessment is informed by both external feedback and assessment criteria reflection. By shifting the grading process from a teacher-centric activity to collaborative and self-directed practice, this model empowers students to take ownership of their learning and fosters a culture of continuous improvement in Informatics education.

3.3 Digital portfolios with self-grading reflections

In the context of Flipping the Grade, digital portfolios serve as an effective tool for fostering self-assessment, metacognition, and long-term learning reflection (Yancey, Cambridge & Cambridge, 2023) in Informatics education. Unlike traditional grading methods, which offer students only a final score with limited

feedback, digital portfolios provide a structured way to document their coding progress, evaluate their work, and justify their grading decisions over time. This process enhances student autonomy and encourages continuous improvement by allowing learners to revisit their past work, identify patterns in their learning, and develop a growth mindset.

To implement self-grading through digital portfolios, students maintain an organised collection (Berbegal Vázquez et al., 2021) of their coding exercises, challenges, and projects, each accompanied by a self-assessment reflection. The structure of these portfolios typically includes three key components: the initial problem statement or task, the student's solution (code and/or algorithm), and a self-reflection entry where they analyse their performance based on predefined grading criteria. The self-reflection component is crucial, as it requires students to critically evaluate their strengths, weaknesses, debugging strategies, and overall approach to problem-solving. Students engage in deeper cognitive processing by assigning and justifying their grades, reinforcing their understanding of Informatics concepts and programming logic.

In our educational practice, we utilize Moodle and Google Classroom for managing digital portfolios, providing a structured framework for student reflection and self-assessment. Within this process, teachers assume a facilitative role, offering periodic feedback on students' reflections rather than assigning direct grades to their work. Comments provided on the platform focus on the accuracy of self-assessment, the depth of reflection, and areas requiring improvement, rather than merely validating or correcting student-assigned grades. The primary objective is to support students in developing evaluative judgment, enabling them to accurately assess their performance over time. In cases where there is a significant discrepancy between a student's self-assigned grade and expected performance standards, teachers intervene through guided questioning or structured discussions rather than direct correction. This approach reinforces student responsibility and autonomy in the learning process.

A practical example of this implementation can be seen in project-based assessments, where students develop a mini-programming project aligned with specific learning objectives. As part of their portfolio, students document their development process, challenges encountered, and problem-solving approaches. Upon completing the project, they use a rubric or checklist provided by the teacher to assess their work, assigning themselves a grade based on parameters such as functionality, efficiency, code readability, and debugging quality. In their reflection, they justify their grade by addressing key questions: Did the program meet the intended goals? What were the main challenges? How did I improve my code? What would I do differently in a future iteration? This iterative process promotes self-directed learning and continuous improvement, enabling students to develop a more sophisticated understanding of programming principles.

The use of digital portfolios for self-grading has significant pedagogical advantages (Berbegal Vázquez et al., 2021; Domene-Martos et al., 2021; Yancey, Cambridge & Cambridge, 2023). First, it shifts the focus from external validation to intrinsic motivation, as students become active participants in their assessment rather than passive recipients of grades. Second, researchers highlight that the digital portfolio provides a comprehensive record of student learning, allowing both students and teachers to track progress, identify learning gaps, and make informed instructional adjustments (Chang et al., 2018; Berbegal Vázquez et al., 2021). Finally, this approach aligns with modern competency-based education models, where assessment is not solely about correctness but also about the process of learning, problem-solving, and self-regulation.

By integrating self-grading through digital portfolios, teachers can create a more reflective and autonomous learning environment in Informatics. Students are assessed not only on their final output but also on their ability to evaluate and improve their work. This model ensures that grading is not an endpoint but an ongoing learning process, preparing students to be self-reliant and critical thinkers in their future academic and professional pursuits.

4. Challenges and considerations

The integration of self-grading in Informatics education offers significant benefits in fostering autonomy, metacognitive skills, and critical thinking. However, its implementation is accompanied by challenges that must be carefully addressed to ensure fairness, accuracy, and pedagogical effectiveness. Key concerns include grading reliability, balancing student autonomy with teacher oversight, and providing structured guidance to facilitate accurate self-assessment.

A primary challenge in self-grading is the risk of inaccurate assessment due to over- or underestimation of one's abilities. Without clear evaluation criteria, students may inflate or deflate their grades, leading to inconsistencies that compromise the reliability of assessments. Establishing detailed, standardized rubrics with explicit performance indicators mitigates this risk, enabling students to align their evaluations with objective benchmarks rather than personal biases. Additionally, incorporating justification mechanisms, such as reflective explanations or comparative analysis with exemplars, enhances grading accuracy by encouraging deeper metacognitive engagement.

Balancing student autonomy with teacher oversight is another critical consideration. While self-grading promotes student ownership of learning, the absence of teacher intervention may result in discrepancies or manipulation of grades. A moderation system, where teachers review a subset of student-assigned grades, ensures consistency while maintaining student agency. Furthermore, peer assessment serves as an intermediary validation step, allowing students to refine their self-assessment based on external feedback. A multi-layered approach – integrating self, peer, and moderated teacher assessments – preserves fairness while reinforcing evaluative skills.

Effective self-grading implementation necessitates explicit scaffolding strategies. Self-assessment is a learned skill, requiring structured guidance and modeling. Teachers should introduce self-evaluation techniques through demonstrated grading exercises, allowing students to practice using rubrics with sample work before applying them to their own assignments. Collaborative discussions and peer comparisons further refine evaluative abilities by exposing students to diverse perspectives on quality and performance standards. Iterative self-assessment, wherein students engage in repeated grading cycles with feedback-driven refinement, cultivates more reliable self-evaluation practices over time.

Ultimately, the success of self-grading in Informatics education depends on a well-structured framework that addresses grading biases, ensures an appropriate balance of autonomy and oversight, and fosters the development of self-assessment competencies. By embedding reflective evaluation into learning processes, students not only enhance their ability to critically assess their programming work but also cultivate essential problem-solving and self-regulation skills necessary for academic and professional growth. This structured approach positions self-grading as a transformative tool in competency-based education, shifting assessment from a static grading mechanism to an iterative learning process.

5. Conclusions

Implementing self-assessment and self-grading in Informatics education carries significant pedagogical implications, particularly in transforming students' perceptions of grading. Traditionally, students view grades as external judgments imposed by teachers, often prioritizing high scores over the learning process itself. By engaging in self-assessment, grading becomes an active, reflective learning experience rather than a summative evaluation. This shift fosters deeper cognitive engagement, requiring students to critically analyze their work, justify their assessments, and identify areas for improvement. Rather than passively receiving grades, they develop a growth-oriented mindset where feedback and self-reflection become integral to their learning. In Informatics education, where problem-solving and debugging demand iterative thinking, this approach strengthens students' ability to independently evaluate their coding practices and enhance their computational skills over time.

Beyond fostering a mindset shift, self-grading cultivates responsibility and accountability by requiring students to apply structured assessment criteria to their work. This process enhances self-regulation, an essential skill in programming and software development. When students justify their self-assigned grades using predefined rubrics, they engage in metacognitive evaluation, improving their ability to detect errors, optimize solutions, and critically assess their problem-solving approaches. Moreover, by assuming responsibility for their grading, students become more invested in the quality of their work, leading to increased motivation and engagement. This approach also reduces dependency on teacher

validation, fostering independence and self-confidence in their programming abilities.

The "Flipping the Grade" approach redefines assessment practices in Informatics education by encouraging a shift in student mindsets and reinforcing self-regulation skills. By integrating self-assessment and self-grading, teachers can cultivate a more autonomous, reflective, and engaged learning environment that better prepares students for the demands of computational problem-solving and lifelong learning.

REFERENCES

Andrade, H. (2008) Self-Assessment Through Rubrics. *Educational Leadership*. 65 (4), 60–63.

Andrade, H. & Brookhart, S.M. (2016) The Role of Classroom Assessment in Supporting Self-Regulated Learning. In: D. Laveault & L. Allal (eds.). Assessment for Learning: Meeting the Challenge of Implementation. The Enabling Power of Assessment. *Cham.* Springer International Publishing. pp. 293–309. doi: 10.1007/978-3-319-39211-0_17.

Berbegal Vázquez, A., Merino Orozco, A., Arraiz Pérez, A. & Sabirón Sierra, F. (2021) The e-portfolio in higher education: The case of a line of teaching innovation and complex change management. *Tuning Journal for Higher Education*. 9 (1), 29–64. doi: 10.18543/tjhe-9(1)-2021pp29-64.

Brookhart, S.M. (2013) *How to Create and Use Rubrics for Formative Assessment and Grading*. ASCD member book. Alexandria, ASCD.

Carroll, D. (2020) Observations of student accuracy in criteria-based self-assessment. *Assessment & Evaluation in Higher Education*. 45 (8), 1088–1105. doi:10.1080/02602938.2020.1727411.

Caspersen, M.E., Diethelm, I., Gal-Ezer, J., McGettrick, A., Nardelli, E., Passey, D., Rovan, B. & Webb, M. (2022) Informatics Reference Framework for School. *National Science Foundation.* doi:10.1145/3592625.

Chang, C.-C., Liang, C., Chou, P.-N. & Liao, Y.-M. (2018) Using e-portfolio for learning goal setting to facilitate self-regulated learning of high school students. *Behaviour & Information Technology*. 37 (12), 1237–1251. doi: 10.1080/0144929X.2018.1496275.

Dawson, P. (2017) Assessment rubrics: towards clearer and more replicable design, research and practice. *Assessment & Evaluation in Higher Education*. 42 (3), 347–360. doi: 10.1080/02602938.2015.1111294.

Deng, R., Feng, S. & Shen, S. (2024) Improving the effectiveness of video-based flipped classrooms with question-embedding. *Education and Information Technologies*. 29 (10), 12677–12702. doi: 10.1007/s10639-023-12303-5.

Dixon, C., Dixon, P.E., Sultan, S., Mustafa, R., Morgan, R.L., Murad, M.H., Falck-Ytter, Y. & Dahm, P. (2020) Guideline developers in the United States were inconsistent in applying criteria for appropriate Grading of Recommendations, Assessment, Development and Evaluation use. *Journal of Clinical Epidemiology*. 124, 193–199. doi: 10.1016/j.jclinepi.2020.01.026.

Domene-Martos, S., Rodríguez-Gallego, M., Caldevilla-Domínguez, D. & Barrientos-Báez, A. (2021) The Use of Digital Portfolio in Higher Education before and during the COVID-19 Pandemic. *International Journal of Environmental Research and Public Health*. 18 (20), 10904. doi: 10.3390/ijerph182010904.

Double, K.S., McGrane, J.A. & Hopfenbeck, T.N. (2020) The Impact of Peer Assessment on Academic Performance: A Meta-analysis of Control Group Studies. *Educational Psychology Review*. 32(2),481–509. doi:10.1007/s10648-019-09510-3.

Giraldo, M.A. & Herold, J.-F. (2023) *Development of a New Metacognitive Self-regulated Model of Competency*. In: 23 June 2023 pp. 181–185. doi: 10.36315/2023v1end039.

Gutu, M. (2023a) Flipped Classroom: An Effective Approach For Developing And Assessing Informatics Competencies. *Journal of Social Sciences*. 5(4), 45–51. doi: 10.52326/jss.utm.2022.5(4).05.

Gutu, M. (2022a) Improving the Informatics Competencies Through Assessment for Learning. In: *Proceedings of the 12th International Conference on "Electronics, Communications and Computing", December 2022*. Technical University of Moldova. pp. 300–305. doi: 10.52326/ic-ecco.2022/KBS.04.

Gutu, M. (2022b) Instructional Design for Developing Informatics Competencies. In: *Proceedings of the International Conference on Virtual Learning - VIRTUAL LEARNING - VIRTUAL REALITY (17th edition), 23 December 2022*. pp. 49–61. doi: 10.58503/icvl-v17y202204.

Gutu, M. (2023b) Metacognition and self-assessment in informatics classes: exploring the impact of assessment criteria, motivation, and task complexity. In: *Proceedings of the International Conference on Virtual Learning - VIRTUAL LEARNING - VIRTUAL REALITY (18th edition), 19 October 2023*. pp. 243–261. doi: 10.58503/icvl-v18y202321.

Kumar, S., Kenney, J. & Buraphadeja, V. (2013) Peer Feedback for Enhancing Students' Project Development in Online Learning. In: *Cases on Online Learning Communities and Beyond: Investigations and Applications*. IGI Global. pp. 345–360. doi:10.4018/978-1-4666-1936-4.

McMillan, J.H. & Hearn, J. (2008) Student self-assessment: The key to stronger student motivation and higher achievement. *Educational Horizons*. 87 (1), 40–49.

Muhammad, A., Lebar, O. & Mokshein, S.E. (2018) Rubrics as Assessment, Evaluation and Scoring Tools. *International Journal of Academic Research in*

Business and Social Sciences. 8 (10), Pages 1417-1431. doi: 10.6007/IJARBSS/v8-i10/5309.

Nechyporuk, M. & Romaniuk, V. (2024) Methodological Aspects of researching National Security Students' Metacognitive control in the Context of Self-regulated Learning. *Scientific Notes of Ostroh Academy National University: Psychology Series*. 1 (17), 46–55. doi: 10.25264/2415-7384-2024-17-46-55.

Panadero, E., Pérez, D.G., Ruiz, J.F., Fraile, J., Sánchez-Iglesias, I. & Brown, G.T.L. (2023) University students' strategies and criteria during self-assessment: instructor's feedback, rubrics, and year level effects. *European Journal of Psychology of Education*. 38 (3), 1031–1051. doi: 10.1007/s10212-022-00639-4.

Simonsmeier, B.A., Peiffer, H., Flaig, M. & Schneider, M. (2020) Peer Feedback Improves Students' Academic Self-Concept in Higher Education. *Research in Higher Education*. 61 (6), 706–724. doi: 10.1007/s11162-020-09591-y.

Stevens, D. D. & Levi, A. (2005) *Introduction to rubrics: an assessment tool to save grading time, convey effective feedback, and promote student learning*. Sterling, Va, Stylus.

Verano-Tacoronte, D., Bolívar-Cruz, A. & González-Betancor, S.M. (2015) Self-assessment: A critical competence for Industrial Engineering. *DYNA*. 82 (194), 130–138. doi: 10.15446/dyna.v82n194.47097.

Weiss, K. (2018) Student Self-Assessment Re-assessed. *Journal of Academic Writing*. 8 (2), 161–175. doi: 10.18552/joaw.v8i2.448.

Westover, J. (2024) Better Decisions Through Self-Reflection and Assessment. *Human Capital Leadership Review*. 12 (3). doi: 10.70175/hclreview.2020.12.3.11.

Wong, H.M. & Taras, M. (2022) *Student Self-Assessment: An Essential Guide for Teaching, Learning and Reflection at School and University*. 1st edition. London, Routledge. doi: 10.4324/9781003140634.

Xia, X. (2023) Exploration and Practice of Flipped Classroom Model in Education Research Methods Course Based on Blended Learning. In *2023 International Conference on Computer Applications Technology (CCAT)*. 15 September 2023 Guiyang, China, IEEE. pp. 243–247. doi: 10.1109/CCAT59108.2023.00052.

Yan, Z., Chiu, M.M. & Ko, P.Y. (2020) Effects of self-assessment diaries on academic achievement, self-regulation, and motivation. *Assessment in Education: Principles, Policy & Practice*. 27 (5), 562–583. doi: 10.1080/0969594X.2020.1827221.

Yancey, K.B., Cambridge, B. & Cambridge, D. (2023) *Electronic Portfolios 2.0: Emergent Research on Implementation and Impact*. 1st edition. New York, Routledge. doi: 10.4324/9781003444428.