DEVELOPING STUDENTS' CRITICAL THINKING THROUGH PROGRAMMING: INFORMATION TECHNOLOGY AS A TOOL FOR DISTANCE LEARNING

DESENVOLVENDO O PENSAMENTO CRÍTICO DOS ESTUDANTES POR MEIO DA PROGRAMAÇÃO: A TECNOLOGIA DA INFORMAÇÃO COMO FERRAMENTA PARA O ENSINO A DISTÂNCIA

Nataliia Pavliuk
ORCID 0000-0002-4456-5682

Rivne State University of the Humanities
Rivne, Ukraine
natakoval.kovalchyk@gmail.com

Tetiana Hryshyna
ORCID 0009-0005-1443-8278

Institute of Journalism and Mass Communication
Classic Private University
Zaporizhzhia, Ukraine
hryshyna65@gmail.com

Tetyana Chumak
ORCID 0000-0003-4723-1757

National University of Bioresources and Nature Management
Kyiv, Ukraine
tchumak27@ukr.net

Yurii Pivnenko
ORCID 0000-0002-6675-2649

O.M.Beketov National University of Urban Economy in Kharkiv
Kharkiv, Ukraine
yurii.pivnenko@kname.edu.ua

Nadia Kuravska
ORCID 0000-0001-7755-2899

Kharkiv National Air Force University
Kharkiv, Ukraine
kn649533@gmail.com

Abstract. The article attempts to summarize the visions concerning development of critical thinking in students, especially in software development disciplines, through programmed learning. The concept of programmed learning is considered within the two existing paradigms of software development – Waterfall and Agile. In this vein, the approach of Agile teaching is investigated, with highlighting its benefits for development students' critical thinking, including in distance learning.

Keywords: critical thinking, programmed learning, Agile teaching, software development.

Resumo. O artigo busca sintetizar visões sobre o desenvolvimento do pensamento crítico em estudantes, especialmente nas disciplinas de desenvolvimento de software, por meio do aprendizado programado. O conceito de aprendizado programado é analisado dentro das duas principais paradigmas existentes no desenvolvimento de software – Waterfall e Agile. Nesse contexto, é investigada a abordagem do ensino baseado no método Agile, destacando seus benefícios para o desenvolvimento do pensamento crítico dos estudantes, inclusive no ensino a distância.

Palavras-chave: pensamento crítico, aprendizado programado, ensino Agile, desenvolvimento de software.

1. INTRODUCTION

Education knowledge and creative goods are inextricably linked to the development of new learning capacities. The teacher's personality impacts the creative process, which is dependent on a number of elements, including eager pupils who want to learn, which is an important aspect in education, as well as mastering new creative processes that result in faster and more effective absorbing of new material. The efficacy of the educational system in the era of modern technologies, the extent of interdisciplinary knowledge, and the simultaneous increase in demand for specialists in changing domains of life are all hot topics (Kotyk et al., 2020; Kryshtanovych et al., 2021; Kryshtanovych et al., 2024; Zyazyun et al., 2022). The field of education uses the same assessment criteria as other branches: achievement as quickly as possible, cost-benefit ratio, reduction of time and money-consuming tasks, and so on (Chorna-

Klymovets et al., 2022; Gavrysh et al., 2020, 2023; Gorban et al., 2022). The programmed learning-teaching process deals with control and is quickly becoming an essential component of topic curricula (Beckman Soares da Cruz et al., 2024 Furtado da Silva et al., 2024). Despite its extensive history, its history is described only in decades: it incorporates technique, instructional aids, and a wide range of branches, thus professionals from diverse fields work to further its growth.

The theory of programmed learning as a professional subject has been recognized for over sixty years. The notion originated in the United States, and the first teaching machines began to grow and thrive. Pressey, Skinner, and Crowder, founding fathers and representatives of subsequent generations, have been known as programmed learning popularizers (Hošková-Mayerová & Rosická, 2012). Later, it was refined, improved, and enriched based on considerable knowledge, notably in pedagogy, psychology, cybernetics, information theory, and mathematical logic. The essential notion in this paradigm is algorithm. In programmed learning, an algorithm may not be described as the end concept. It is defined as the optimal logical framework for studying a specific issue, area, or course. The primary premise is "from data to knowledge" (Davies, 2007). The core need consists of reducing study material into manageable pieces and providing rapid feedback and confirmation of each phase consisting of content presentation, question, and learner answer. The approach is founded on the idea that the learner immediately understands whether or not he has made an error and proceeds with the program. Individual stages are organized into logical sequences of content called programs.

In turn, one of the most contributing disciplines in this vein is software development.

Software development and innovation are moving faster than ever before, but there is one talent that developers, architects, and coders must learn to remain competitive. This talent is called critical thinking. Critical thinking is not limited to troubleshooting and optimizing algorithms. It is about fully comprehending the problem. It motivates us to go beyond syntax to see patterns, rework elegantly, and optimize for clarity. As a result, these abilities aid in the study of other disciplines, such as the humanities and social science cycle. The challenges of developing critical thinking, are, however, more acute in distance learning.

2. THEORETICAL FRAMEWORK

According to Bastias et al. (2021), critical thinking is the process of examining and assessing the coherence of arguments. This skill is essential when discussing software quality (SQ). Students must analyze, assess, and form conclusions since SQ is strongly tied to the engineer's capacity to accurately appraise and distinguish between solutions. As a result, teaching software developers to think critically becomes essential. Finding precise suggestions is challenging, particularly in online situations, due to the variety of offers and the lack of rigor in previous experiences (English & Lehmann, 2024). However, it should be mentioned that without the right critical thinking abilities, it is also rare that one may successfully master other fields. However, critical thinking development is frequently dispersed across several fields without a cohesive framework or platform (Berry et al., 2022).

Simultaneously, software and its development processes are constantly evolving to become more and more integrated into our everyday lives. The software business is keen to embrace new and varied ideas and techniques that will give it a competitive edge. Because of the variety of these new methods and approaches, as well as the variety of clients and situations in the software business, software engineers must be able to make accurate judgments and effectively distinguish between them. Software developers must learn these skills during their official undergraduate studies. However, in terms of providing students with these unique and varied abilities, standard techniques in software engineering education (SEEd) fall short (Chouseinoglou & Bilgen, 2014). The above allows suggesting that programming can become

a foundation tool for developing critical thinking in students, regardless of IT, STEM, or other field of study.

3. METHODOLOGY

The methodological and theoretical basis of the study is the regulatory-activity approach in training, which is implemented through the principles of systemicity, reflection, and dialogicity.

The provisions of the theory of development of critical thinking, the theory of activity and application of the activity approach in education, informatization of education, the theory and methods of teaching computer science, information and communication technologies, research in the field of methods of teaching different paradigms and programming technologies were also used.

The methodological tools included content analysis and comparative analysis.

4. RESULTS AND DISCUSSION

As curricular standards have evolved, the number of requirements requiring a higher degree of thinking has grown. These standards promote critical thinking abilities and challenge students to interact with complicated materials and topics at a higher level. Employers need workers who can think critically, evaluate information, solve issues creatively, and communicate successfully, demonstrating the emphasis on higher level thinking.

In 2023, IBM conducted a study to assess the most important skills for the workforce. The question was formulated as follows: "How many of the skills most in demand require critical thinking?" The results are presented in Fig. 1.

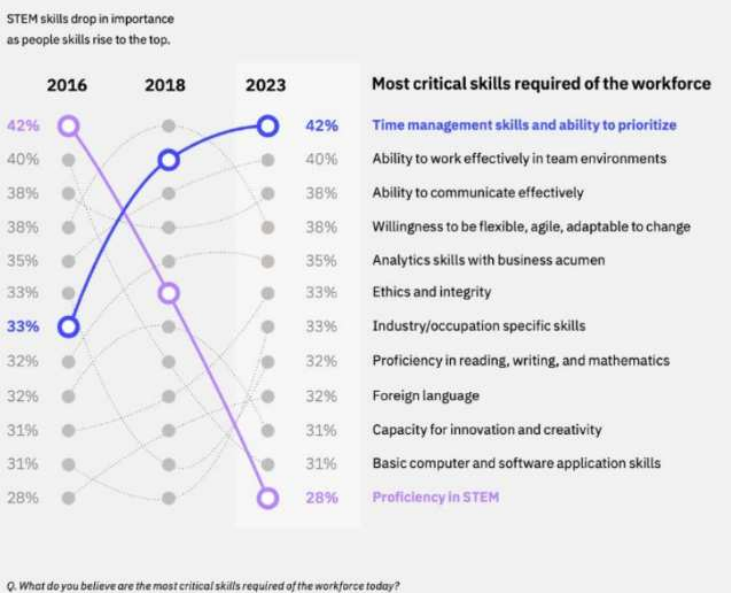


Figure 1. *Results of IBM 2023 survey on critical thinking skills* (Marlett, 2024)

There is a common misperception that critical thinking is an intrinsic aptitude or skill that develops over time. According to research, critical thinking abilities may be taught explicitly and improved via purposeful practice.

Here are some strategies for improving critical thinking (Egan, 2019):

1. Moving beyond rote memorization: Activities that demand students to memorize information do not encourage critical thinking. Instead, provide exercises that motivate kids to evaluate data, spot trends, and reach conclusions.

2. Asking open-ended questions encourages students to think critically and formulate their responses. These questions can help students examine a situation, evaluate facts, and create views.
3. Encouraging discussion and debate: Discussions enable students to express their viewpoints, question one another's ideas, and get a better grasp of the subject at hand.
4. Using problem-based learning: Problem-based learning challenges students to tackle real-world situations using critical thinking abilities. This technique encourages students to think creatively, examine information, and work together.
5. Modeling critical thinking: Showing students how to approach issues, examine information, and draw conclusions. This can assist students in improving their critical thinking abilities.

As it was mentioned above, learning programming represents a broad field for development of critical thinking. However, the right approach should be chosen within application of programmed learning.

Programmed learning ideas, methodologies, and operating arrangements differ; nonetheless, two primary programming approaches are defined: Skinner linear programming and Crowder branch programming.

B. F. Skinner pioneered the linear programming paradigm. The Linear Skinner program comprises of simple stages that any student may complete without difficulty. According to Skinner, 95% right responses indicate a successful program. First, the study material is divided into little information stages, then the information is supplied, the learner responds, the correct response is provided, and the process is repeated. Principles of linear programming in this concept are as follows (Davies, 2007):

1. Learners follow instructions in the same hierarchical sequence.
2. Skinner's linear program requires the learner's initial answer.
    a. Every question needs an active answer; every blank must be filled up.
    b. Active responses are preferred because:
        i. Recall is more effective than recognition.
        ii. Because response leads to learning, learners should not be presented with erroneous options.
3. A little step is typical of any linear program.
    a. Response leads to learning, thus every step must be modest enough to anticipate proper replies and avoid wrong answers.
    b. Too many erroneous responses do not encourage students, whereas right replies are positive stimulus.
4. S. Pressey' justification of retrieval choice linear programming is as follows:
    a. Numerosity law: The learner's response is occasionally incorrect; nonetheless, he eventually chooses a proper choice: we may conclude that correct responses prevail.
    b. The novelty law states that erroneous responses are insignificant, and the correct answer is always the last one remembered by the learner.

N. A. Crowder developed the branching or intrinsic approach of programming. His technique is based on three principles: the principle of exposition, the concept of diagnosis, and the theory of remediation, which entails presenting material, asking questions, and providing multiple choice responses (4 - 5); one is accurate, the others are false; the erroneous answers are not prima facie. The student moves through the main stream from number one to number two, then number three, and so on. If he picks the erroneous option, he is sent to a remedial item, where he is offered further assistance in comprehending the topic and solving

the problem using superior logic. He will then be sent back to the original frame so that he may read it again and correctly answer it using the remedial information he has received: the learner will go through the same frame again. Alternatives that "always go somewhere" differ from linear programming (Tripathi & Sasikala, 2016).

In reality, these two ways to program learning correlate to two methodologies in software development: waterfall and agile. Regardless of the technique selected, the primary goal is to educate pupils how to negotiate complexity, which involves the following principles:

- Software is not a single project; it is a symphony of interrelated components. Critical thinkers divide difficulties into small bits.
- They connect the dots by evaluating how each module fits into the overall design. Their solutions are compatible with the system.

Table 1 shows practical techniques for developing critical thinking skills in teaching programming.

Table 1. Practical steps for cultivating critical thinking in teaching programming (Oriogun, 2010)

Principle	**Core elements**
Curiosity and Inquiry	Maintain a lifelong learning mindset. Curiosity promotes critical thinking. Explore beyond the immediate domain by reading widely, attending conferences, and engaging with various perspectives.
	When confronted with an issue, ask questions incessantly. Why? How? What if?
	Pair programming is more than simply coding; it is about exchanging mental models. Interact with coworkers. Their perspectives inspire creativity.
	Discuss trade-offs, design options, and alternative approaches. Accept the richness of varied viewpoints.
Analytical Reasoning	Break down difficulties systematically. Analyze each component. Identify reoccurring motifs.
	Structured analysis yields effective methods. Critical thinkers approach difficult situations with clarity.

Critical thinking in software development is a fundamental talent that goes beyond typical issue resolution. It entails a rigorous intellectual discipline for studying and synthesizing information required to create robust, efficient, and user-friendly products.

Critical thinking promotes inventive problem solutions, improves quality assurance, and serves as the foundation for strategic decision-making. It encourages the creative and analytical thinking required to navigate the complexity of the software business. By stressing critical thinking, teams enhance efficiency and productivity, promote company development, innovate effectively, and create high-quality software solutions that meet user demands and corporate goals. This emphasis on critical thinking is vital for navigating the changing digital world and developing software solutions that are resilient, inventive, and strategically sound.

The Foundation for Critical Thinking defines critical thinking as an intellectual discipline that actively conceptualizes, applies, analyzes, synthesizes, and evaluates knowledge received from many sources. In software development, this entails a thorough study of every component of the product (Gunay et al., 2020). The goal is to create software that serves its intended function while standing out in terms of performance, dependability, and user happiness. This

method necessitates a thorough grasp of technology and the end user, ensuring that each feature, function, and user interface piece is carefully planned and tested.

Critical competencies within critical thinking in software development and testing can be summarized as follows:

1. Problem-solving: This talent is essential for recognizing and addressing complex issues. Finding hidden flaws that are not immediately visible is critical in software development for providing strong and trustworthy software solutions.
2. Quality assurance: A critical approach to testing is essential for designing comprehensive solutions that considerably enhance software quality. QA engineers should thoroughly test software, inspecting each component to detect and correct possible flaws.
3. Strategic decision-making: In the fast-paced software development environment, making timely and informed judgments is critical. Critical thinking helps experts to swiftly examine complicated events and make technical judgments that are consistent with overall corporate strategy and user demands.
4. Critical thinking helps to create innovation and creativity in software development. Challenging conventional knowledge and promoting unconventional thinking results in revolutionary software solutions that push the limits of technology and user experience.
5. Risk assessment and management: Critical thinking helps with practical risk assessment and management, which is an important component of software development. It helps teams to anticipate future issues and design risk-mitigation methods, resulting in easier project execution and increased chances of success.

Cultivating critical thinking within software development teams is an important activity that goes beyond traditional problem-solving techniques. This development entails many crucial strategies: seeking inspiration, adopting a beginner's attitude, questioning assumptions, framing issues as questions, being uncomfortable, and practicing immersive empathy (Groeneveld et al., 2021).

Seeking inspiration entails engaging with different and unique ideas, which can drive critical examination of current conceptions and promote the development of new viewpoints. Adopting a beginner's perspective can aid critical thinking. It fosters challenging the current quo and reexamining what is thought to be known, which is essential for complete study and fair appraisal of new information. Challenging assumptions is an essential component of critical thinking. It entails examining the fundamental assumptions of arguments and ideas, which is critical for deconstructing complicated situations and avoiding logical fallacies. Turning difficulties into questions is a critical thinking technique that enables more in-depth investigation. It promotes an adventurous and inquisitive attitude, which is required for full investigation and comprehension of difficult subjects. Getting uncomfortable is venturing beyond of one's comfort zone, and critical thinking entails confronting difficult and new concepts. This process is essential for developing adaptability and resilience in thought, as well as the capacity to properly examine and assess multiple points of view. Immersive empathy, while frequently associated with emotional intelligence, also contributes to critical thinking. Understanding multiple viewpoints and settings can help you conduct a more nuanced and complete examination of circumstances or problems (Bastias et al., 2021).

In addition to these strategies, it is critical to cultivate a culture of intellectual humility and persistent inquiry. Encouraging students to realize the limits of their knowledge and to be open to continual learning may considerably improve critical thinking. This entails not just searching out different perspectives and questioning established paradigms, but also developing a habit of critical thinking in which students routinely review their own thought processes and

judgments. This reflective approach, along with a dedication to continual learning, ensures that critical thinking becomes an inherent and ongoing part of the software development process.

The Agile approach to programming makes it easier to create courses and curricula that incorporate all of the features stated above. Sprint-style teaching and learning allows for constant feedback. Overall, agile learning is a revolutionary educational method that brings the processes and ideas of agile software development to the environment of learning. Agile learning is distinguished by short project cycles known as sprints, during which a useable product is thoroughly planned, created, built, tested, evaluated, and deployed. Lang (2017) explains the effective practical introduction of this strategy. He discusses the findings of a pedagogical experiment in which an undergraduate elective Computer Information Systems course on web development was revamped to incorporate a semester-long agile learning experience. A student poll completed at the conclusion of the semester found that agile learning integrates learning with application while allowing students to fail more and quicker. At the same time, as Lang points out, Agile learning is slower than conventional project-based learning, making it easier for students to fall behind. Nonetheless, students strongly preferred agile learning over traditional project-based learning. Importantly, students' preferences and performance in agile learning were unaffected by their learning style. However, Agile learning needs extensive planning, combining the need to offer instructions with the need to provide explanations, as well as a large quantity of one-on-one student assistance.

Traditional project-based learning is frequently conducted in a linear approach that starts with theoretical lectures and then asks students to plan, develop, create, test, evaluate, and eventually launch a usable deliverable (Lee et al., 2015). Interestingly, conventional project-based learning became popular in the early 2000s, when the old "waterfall" systems development paradigm was dominant (Condliffe et al., 2015). Traditional systems development, like project-based learning, entails carrying out the aforementioned actions in a sequential order. Figure 2 displays a typical project-based learning method.



Figure 2. *Traditional project-based learning process* (Parsons & MacCallum, 2019)

However, Agile learning refers to the application of agile software development techniques and concepts to the learning context. During numerous sprints, developers iteratively grow and improve the software program. In the context of learning, project cycles and usable deliverables replace development cycles and operational software applications. In other words, an agile learning experience is made up of several short project cycles known as sprints, during which a workable product is completely planned, developed, produced, tested, evaluated, and launched. One of the distinguishing elements of agile software development, and by extension, agile learning, is that each sprint concludes with a usable product that is constantly being developed and improved upon. In this case, the learning process acquires the form depicted in Fig. 3.



Figure 3. *Agile learning process* (Parsons & MacCallum, 2019)

Br. J. Ed., Tech. Soc., v.18, n.se1, p.270-281, 2025
DOI http://dx.doi.org/10.14571/brajets.v18.nse1.270-281
ISSN 2316-9907

In addition to the methods listed above, agile learning brings the four principles of agile software development to the context of learning. The four principles of agile software development were initially expressed in the "Manifesto for Agile Software Development" by 17 top software development specialists who identified the need for an alternative to documentation-driven, heavyweight software development techniques (Justice, 2023).

The first agile concept is that "individuals and interactions take precedence over processes and tools". When applied to the context of learning, it indicates that the teacher focus on working one-on-one with students while being flexible in altering the methods and instruments employed in the classroom. The second agile concept is "working software over comprehensive documentation", which says that students should focus on generating something that can be utilized in a professional setting rather to just writing reports. "Customer collaboration over contract negotiation" is the third agile principle. In terms of learning, it implies that instructors work with students rather than rigorously enforcing tasks and associated norms. Finally, the fourth agile principle is "responding to change over following a plan", which highlights the instructor's willingness to deviate from the standard semester-long course schedule and instead adapt the timetable in response to students' needs as they emerge. The agile learning principles aim to increase the instructor's capacity to promote learning in an agile learning environment.

Chun (2004) investigated Agile Teaching/Learning Methodology (ATLM) in 2004 as a teaching/learning methodology created for higher education that is based on best practices and ideas from the area of software engineering and draws on principles from Agile development techniques. He also discusses the e-learning platform he created to complement the ATLM approach to teaching and learning, as well as the technology that underpin it. The platform takes use of a variety of current collaboration and knowledge sharing tools, including blogging, commenting, instant messaging, wikis, and XML RSS. Commenting, instant messaging, search, multilingual translations, and updated alerts via XML RSS are among the platform's shared features, which are accessible from all sites. Today, such platforms are far more extensive and advanced, using AI and immersive technology.

If to compare Waterfall and Agile learning with programmed learning approaches of Skinner and Crowder, the conceptual similarity becomes evident (see Fig. 4 and 5).

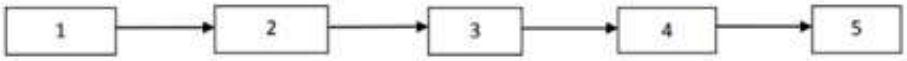


Figure 4. *The conceptual scheme of linear programmed learning* (Tripathi & Sasikala, 2016)

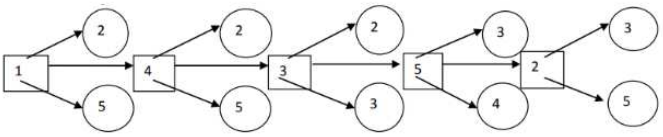


Figure 5. *Arrangement of branching programmed learning* (Tripathi & Sasikala, 2016)

Thus, it could be assumed that Waterfall and Agile learning can be successfully fitted into existing theoretical paradigms of programmed learning, and the best discipline to launch this approach is software development.

Software engineering is primarily a problem-solving process. Every piece of software, whether it is an operating system, a mobile app, or an enterprise system, is designed to solve a

specific problem or group of problems. These issues might include automating a corporate process, offering a platform for social engagement, or making sense of enormous datasets.

When engineers begin a software development project, they first identify the problem they are entrusted with solving. This includes understanding the subtleties of the problem, anticipating the demands of the users, and describing the limitations and criteria that govern the situation. Once the problem is understood, the following stage is to consider potential remedies. During this phase, several problem-solving strategies are used, including decomposition (breaking the problem down into smaller, more manageable parts), pattern recognition (identifying similarities between the current and previous problems), and abstraction (removing unnecessary details to focus on the core problem). Using these methodologies, the engineer creates a comprehensive solution that can be implemented as a software system.

Along with problem resolution, critical thinking is the core of software engineering. Critical thinking is the objective investigation and evaluation of a topic in order to develop a judgment. It is used throughout the software development process.

During the design process, critical thinking is used to choose amongst various alternative solutions or design patterns. The engineer must assess the benefits and downsides of each choice, taking into account scalability, maintainability, and performance. This necessitates a thorough grasp of computer science fundamentals, as well as the ability to predict how the system will change in the future. In the implementation phase, critical thinking is required to write effective, efficient code. It include choosing the appropriate data structures and algorithms, assuring code readability, and preserving program security and integrity. Furthermore, engineers must anticipate and handle any mistakes and exceptions, which necessitates critical thinking to detect potential dangers and edge cases. During testing, engineers use critical thinking to identify faults that may not be obvious from first inspection. This involves not only checking for obvious faults, but also detecting possible design flaws, usability concerns, and performance bottlenecks.

Bastias et al. (2021) present principles for evaluating critical thinking in software development in the online environment (see Fig. 6). Based on the findings of the systematic mapping study, the authors propose a preliminary framework for evaluating critical thinking in software engineer training in the context of online higher education, claiming that this proposal will serve as a foundation for discipline instructors when evaluating critical thinking in an online teaching context.

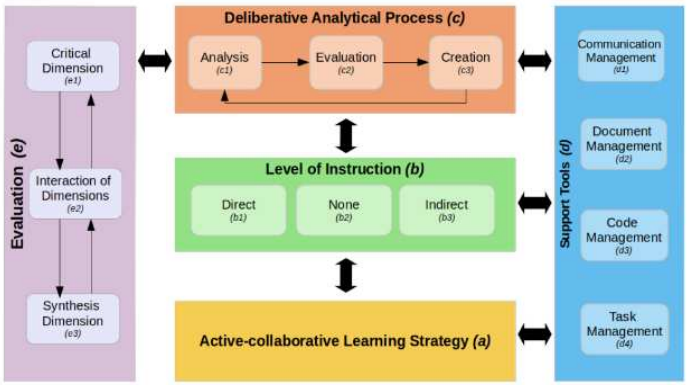


Figure 6. *Possible guidelines for the evaluation of critical thinking in the context of software development* (Bastias et al., 2021)

Accordingly, these moments should be present in curricula and teaching policies both within software development courses and in other disciplines.

The extensive usage of digital technology contributes to further improvement in learning results. In this regard, it is worth mentioning that fusion and immersion are the most often employed methods nowadays. The fusion approach combines in-depth topic training with specialized critical thinking education, as well as the integration of everyday scenarios, all with the purpose of teaching students how to use critical thinking talents in specific settings (Silva, 2009). The immersion technique contends that students learn critical thinking skills as a result of studying course content, rather than as distinct components of the course (Bastias et al., 2021). Furthermore, digital technologies promote collaboration and participation in education. Virtual classrooms, discussion forums, and video conferencing platforms enable real-time communication and collaboration between students and teachers, fostering a sense of community and peer-to-peer learning. These technologies also allow educators to provide quick feedback and assistance, enhancing the learning experience, which is crucial in an Agile teaching environment.

Weatherly and Malott (2014) did an intriguing research on employing computer-based programmed instruction to train goal-directed system design. The study's objective was to compare two versions of a programmed instruction training program intended to teach undergraduate college students a goal-directed systems approach to understanding organizational systems. The initial version was a paper-based programmed teaching module that had previously been found to be successful at educating fundamental understanding of the ideas but poor at training their application. To improve the application of these principles, a computer-based programmed instruction (CBPI) version was developed and assessed using a series of three open-ended posttests, each with progressively clear prompts. The study's findings indicated that the CBPI versions performed better across all three dependent variables. CBPI outperformed paper-based programmed instruction in all three posttests, with performance improving when explicit prompts were introduced for each subsequent posttest. The initial iteration of the computer-based programmed instruction (N = 19) had a posttest #1 mean score of 51%, a posttest #2 mean score of 63%, and a posttest #3 mean score of 87 percent. During the next semester, students (N = 32) received paper-based programmed training, with posttest #1 mean score of 38%, posttest #2 mean score of 55%, and posttest #3 mean score of 78%. The final version of computer-based training was given to students (N = 45) the following semester, with posttest #1 mean score of 51%, posttest #2 mean score of 68%, and posttest #3 mean score of 88%.

5. CONCLUSION

The software development landscape has shifted considerably in recent years, owing to the emergence of digital transformation. As a result, what was formerly an isolated and linear process has become considerably more dynamic and interrelated. Developers must now deal with a steady influx of new technology and tools. However, digital transformation has greatly aided the software development process. For example, it has enabled developers to construct more usable and efficient programs. It has also enabled the faster and more effective rollout of software upgrades. As the software development landscape evolves, digital transformation will become increasingly important. Thus, considering these tendencies while building curriculum for software development courses is vital for developing in students adaptable critical thinking abilities that may be effectively implemented within Agile teaching.

REFERENCES

Bastias, O., Diaz-Arancibia, J., & Rodriguez, C. (2021). Evaluation of critical thinking in online software engineering teaching: A systematic mapping study. *IEEE Access, 9,* 167015.

Beckman Soares da Cruz, A. K., de Salles Soares Neto, C., Meireles Teixeira , M. A., & Torres Maia Beckman da Cruz, P. (2024). Individualized Recommender Systems for Teaching: A Systematic Literature Mapping. *Cadernos De Educação Tecnologia E Sociedade*, *17*(3), 878–908. https://doi.org/10.14571/brajets.v17.n3.878-908

Berry, A., Buntting, C., Corrigan, D., Gunstone, R., Jones, A. (2022). *Education in the 21st Century: STEM, Creativity and Critical Thinking*. Springer.

Chorna-Klymovets, I., Semeriak, I., Mordous, I., Kryshtanovych, S., Zainchkivska, I. (2022). Modern technologies for the development of distance education. IJCSNS. *International Journal of Computer Science and Network Security, 22*(9), 103-108. https://doi.org/10.22937/IJCSNS.2022.22.9.16

Chouseinoglou, O., Bilgen, S. (2014). Introducing Critical Thinking to Software Engineering Education. In: Lee, R. (eds). *Software Engineering Research, Management and Applications. Studies in Computational Intelligence,* vol 496. Springer, Heidelberg. https://doi.org/10.1007/978-3-319-00948-3_12

Chun, A. (2004). The Agile Teaching/Learning Methodology and Its e-Learning Platform. Advances in Web-Based Learning - ICWL 2004, Third International Conference, Beijing, China, August 8-11, 2004.

Condliffe, B., Visher, M. G., Bangser, M. R., Drohojowska, S. & Saco, L. (2015). Project-based learning: A literature review. *MDRC*. https://s3-uswest1.amazonaws.com/ler/MDRC+PBL+Literature+Review.pdf

Davies, I. (2007). *Programmed learning in perspective: A guide to program writing*. Routledge.

Egan, A. (2019). *Confidence in critical thinking: Developing learners in higher education*. Routledge.

English, L., & Lehmann, T. (2024). *Ways of Thinking in STEM-based Problem Solving: Teaching and Learning in a New Era*. Routledge.

Furtado da Silva, W., Andrade da Silva, F. C., Mariani Braz, R. M., & Rodrigues Leta, F. (2024). The teacher facing information and communication technologies. *Cadernos De Educação Tecnologia E Sociedade*, *17*(4), 1240–1259. https://doi.org/10.14571/brajets.v17.n4.1240-1259

Gavrysh, I., Olenych, I., Kryshtanovych, S., Saukh, I., & Khltobina, O. (2023). Formation of professional culture of future specialists in finance and credit in higher educational institutions. *Financial and Credit Activity Problems of Theory and Practice, 3*(50), 477-486. https://doi.org/10.55643/fcaptp.3.50.2023.4072

Gavrysh I., Kryshtanovych M., Kholtobina O., Melnychuk I., Salnikova N. (2020). Prospects, problems and ways to improve distance learning of students of higher educational institutions. *Revista Romaneasca pentru Educatie Multidimensionala, 12*(2), 348-364 https://doi.org/10.18662/rrem/12.2/282

Gorban I., Kornat L., Dykyi A., Kryshtanovych M., Marushko N. (2022). Investment support for the digitalization of socio-economic systems in the context of ensuring security. IJCSNS. *International Journal of Computer Science and Network Security, 22*(6), 733-738. https://doi.org/10.22937/IJCSNS.2022.22.6.92

Groeneveld, Luyten, L., Vennekens, J., Aerts, K. (2021). Exploring the role of creativity in software engineering. *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Society* ICSE-SEIS 21 (pp. 1-9).

Gunay, C., Doloc-Mihu, A., Barakat, R., Gluick, T. (2020). Improving critical thinking in software development via interdisciplinary projects at a most diverse college. Proceedings of The 21st Annual Conference on Information Technology Education.

Helesh, A., Eremenko, O., & Kryshtanovych, M. (2021). Monitoring the quality of the work of experts when they conduct accreditation examinations of educational programs. *Revista Tempos E Espaços Em Educação, 14*(33), e16535. https://doi.org/10.20952/revtee.v14i33.16535

Hošková-Mayerová, S., & Rosická, Z. (2012). Programmed learning. *Procedia - Social and Behavioral Sciences, 31*, 782-787.

Justice, J. (2023). *The Agile Educator: A different thinking approach to teaching and learning*. GRIN Verlag.

Kotyk, T., Kryshtanovych, M., Tiurina, T., Kovrei, D., & Dzhanda, H. (2020). Pedagogical and psychological aspects of the implementation of model of the value attitude to health. *BRAIN. Broad Research in Artificial Intelligence and Neuroscience, 11*(2Sup1), 127-138. https://doi.org/10.18662/brain/11.2Sup1/99

Kryshtanovych, S., Bezena, I., Hoi, N., Kaminska, O., & Partyko, N. (2021). Modelling the assessment of influence of institutional factors on the learning process of future business managers. *Management Theory and Studies for Rural Business and Infrastructure Development, 43*(3), 363-372. https://doi.org/10.15544/mts.2021.33

Kryshtanovych S., Fedir Z., Dulibskyy A., Ilkiv O., Odnovorchenko I., Chyzh V. (2024). Innovative technologies in the work of a teacher of physical culture and sports. *AD Alta – Journal of Interdisciplinary Studies 14(1), Special issue XL*, 220-225. https://doi.org/10.33543/j.140140.220225

Lang, G. (2017). Agile learning: Sprinting through the semester. *Information Systems Education Journal, 15*(3), 14-20.

Lee, D., Huh, Y. & Regeluth, C. (2015). Collaboration, intragroup conflict, and social skills in project-based learning. *Instructional Science, 43*(5), 561-590.

Marlett, D. (2024). Increasing critical thinking in education: How to prepare students for the future. *Learning-Focused*. https://learningfocused.com/increasing-critical-thinking-in-education-a-pathway-to-preparing-students-for-the-future/

Oriogun, P. (2010). *Software engineering education: Towards understanding and improving the process of small group collaborative learning*. LAP LAMBERT Academic Publishing.

Parsons, D., & MacCallum, K. (2019). *Agile and Lean concepts for teaching and learning: Bringing methodologies from industry to the classroom*. Springer.

Silva, E. (2009). Measuring skills for 21st-century learning. *Phi Delta Kappan, 90*(9), 630-634.

Tripathi, H., & Sasikala, M. (2016). Developing programmed learning material for teaching science of CBSE Class IX. *The International Journal of Indian Psychology, 3*(2), 1-7.

Weatherly, N., & Malott, R. (2014). Using computer-based programmed instruction to train goal-directed systems design. *Acta de Investigación Psicológica, 4*(3), 1747-1757.

Zyazyun, L., Vykhrushch, N., Huzii, I., Kryshtanovych, M., & Kalinska, O. (2022). Philosophical aspects of determining the main components of the formation of professional competence for students. *WISDOM, 22*(2), 130-137. https://doi.org/10.24234/wisdom.v22i2.606