**On design of online learning environments for programming education:**

The introduction discusses challenges in programming education at the university level, emphasizing difficulties in grasping theoretical concepts and practical coding aspects. It highlights low motivation and high dropout rates, prompting a shift towards online self-learning alternatives. The current generation entering universities, labeled as Digital Natives, is proficient in basic computer skills and comfortable with online environments. Codecademy and MyProgrammingLab are recognized as promising online platforms for programming activities.

the study supports the idea of using online environments for self-learning in programming education. Design factors crucial for effective learning include usability, unambiguous exercises, clear feedback, GUI design, multi-modality, and gamification. Curriculum alignment is identified as key for achieving learning outcomes. The Codecademy environment is preferred by a majority due to its implementation of important design factors, while the MyProgrammingLab system is noted to work better for analytical learners. The conclusion suggests potential extensions like multi-lingual descriptions and adaptation for different learning styles.

# 

# Intelligent learning environments for programming: The case for integration and adaptation

The introduction discusses the historical context of intelligent tutoring systems (ITS) and their application in teaching programming. It acknowledges the success of AI in education, particularly in the development of ITS and Integrated Learning Environments (ILE) for programming. The text highlights the extensive coverage of programming languages and various aspects of the programming process in existing systems. It points out that despite the vast literature and numerous tools developed, there is a limited adoption of ITS or ILEs in real classrooms for teaching programming. The author raises the question of whether this limited adoption reflects the success or failure of intelligent technology in the field of programming education.

The paper justifies the need for future research on an integration-oriented architecture for Integrated Learning Environments (ILE) and reviews existing work in this area. The envisioned architecture for a future ILE is described as a collection of independent components integrated around a central student model. Knowledge-based components, like the curriculum planner or problem-solving support, would be instantiated from generic components. The ILE would include components developed by the authors, reused components from other projects, and standard commercially available systems, all capable of sending information about the student to the central model and using it for adaptation. The interaction between components would be implemented through various communication methods, including inter-module, inter-application, or network communication. The vision is that a common architecture could be adopted, allowing teachers to assemble ILEs for individualized teaching support, while developers focus on creating new components and reusing existing ones, fostering collaboration and efficiency in ILE development.

**A study of the relationships among learning styles, participation types, and performance in programming language learning supported by online forums:**

The introduction outlines the importance of learning styles in e-learning and their impact on academic competence. Previous research suggests that matching learning styles with instructional methods improves online course outcomes. This study focuses on the influence of learning styles on programming language learning, utilizing Internet discussion tools. Acknowledging the significance of feedback in computer learning environments, the study emphasizes the role of social behaviors in programming language learning. Social learning theories, particularly Bandura's, are employed to examine the relationships between participation behaviors in online forums and learning performance. The use of an online forum is seen as a simulation of a real-world collaborative community for programmers. The study aims to distinguish different online forum participation types and their associations with learning performance. Additionally, the study suggests implications for designing effective learning strategies and emphasizes measures to enhance programming language learning effectiveness.

the study discusses the creation of an artificial environment using an online forum to support programming language learning. The key findings are as follows.

1. Different learning styles, particularly the Accommodator type, influence learning performance in programming language learning with online forum support.

2. Different participation types, especially the Replier type associated with active participation, influence learning performance positively.

3. There is no significant association between learning styles and participation types, suggesting they cannot be used interchangeably.

4. Learning styles and participation types do not significantly impact learning satisfaction, although overall satisfaction is high.

5. The study highlights the success of the hybrid learning environment, combining instructor-led courses with an online forum for exercises and discussions.

that online forums, while not replacing traditional learning, act as supplementary tools to enhance programming language learning performance. Recommendations for future studies include exploring more precise definitions of participation types, comparing the impact of different web 2.0 tools on performance and satisfaction, and investigating the supplementary role of online forums in learning across various age groups and educational levels.

**NoobLab: An Intelligent Learning Environment for Teaching Programming**

The introduction discusses the common perception among students that programming is "boring" and "difficult." It introduces the challenges faced by tutors in teaching programming, emphasizing the limitations of traditional lecture-style teaching. The constructivist approach, particularly Wulf's, is highlighted, which encourages practical experimentation and collaboration among students. However, practical considerations often lead to a compromise, combining lectures to establish concepts with hands-on sessions. The importance of tutor guidance during workshops is stressed, creating a dynamic learning loop between tutor and student. The challenges of managing large cohorts, finding enough tutors, and assessing student work efficiently are acknowledged. Multiple-choice questions are criticized for not effectively assessing practical programming skills. The introduction sets the stage for discussing the limitations of current teaching methods in programming and the need for more effective assessment strategies.

The results section describes the positive outcomes of using an online learning environment (NoobLab) in a programming module. The environment reduced the need for human resources, provided timely feedback, and simplified the marking process for Small Tests. Approximately 78% of students who completed more than half the summative tasks achieved a passing grade. The satisfaction index for various positive statements about the NoobLab environment and the module was high, with 83% agreeing that they did better than expected. There was a significant correlation between time spent in the NoobLab environment and a student's final mark. General feedback was positive, highlighting the environment's effectiveness and its integration into the module structure. Continuous assessment through Small Tests received a high satisfaction index of 94.12%, positively impacting attendance and formative usage of the learning environment.

emphasizing the potential of online delivery and automated assessment tools in enhancing the learning experience in programming courses. The holistic approach, integrating technology, pedagogy, and course content, received overwhelmingly positive feedback. The author suggests that the true potential lies in using student statistics to drive feedback and create an adaptive learning environment, similar to the learning loop between a human tutor and a student.

**A case study investigating programming students’ peer review of codes and their perceptions of the online learning environment**

The text discusses the historical context of Logo, the first programming language introduced at the K-12 level in the 1960s, and its limited adoption in mainstream schools. Despite a shift towards teacher-centric approaches, there is a renewed interest in teaching programming to K-12 students, facilitated by visual programming languages like Scratch and Alice. The importance of programming in developing computational thinking and crucial skills such as critical thinking and problem-solving is emphasized.

The challenges students face in learning programming, such as difficulty in reading and writing code, understanding concepts, and tracing through codes, are highlighted. The need for teachers to facilitate students' exploration of alternative programming solutions, individually or collaboratively, is emphasized. The study's focus is on investigating how students learn in an online programming environment through peer review of codes, examining their perceptions of pedagogical, social, and technical design. The goal is to engage students in constructing meaningful representations and problem-solving, providing insights for programming educators to develop effective teaching methods.

The study finds that students perceive Google apps for education as beneficial for learning programming, suggesting potential exploration of similar cloud-based technologies in one-to-one computing classrooms. Emphasizing the significant role of teachers as facilitators in collaborative learning activities, the recommendation is for programming educators to engage in continuous professional development through workshops and conferences.

The research indicates that involving students in learner-centered activities supports the development of their programming skills. It proposes a cross-sectional study spanning multiple school terms to observe and record students' experiences in a learner-centric programming learning environment, focusing on grades 9–11.

While the study acknowledges the positive impact of peer review on students' programming learning, it identifies a challenge with non-participating students. Future research is suggested to enhance the online programming learning environment by incorporating a reward system to encourage participation. Strategies are proposed, including facilitating discussions on alternative solutions, emphasizing coding approaches' advantages and disadvantages, and highlighting individual contributions in the programming community.

Recognizing the technological landscape shaping the future, the study stresses the importance of enhancing students' programming skills, given the influence of technologies like big data, augmented reality, virtual reality, and artificial intelligence. Learning programming is seen as crucial for developing twenty-first century skills such as innovation, problem-solving, collaboration, creativity, and critical thinking, aligning with STEM activities' role in preparing students for the future workforce.