Introductory Programming - Teaching

Sondre S. Bolland

March 6, 2024

Within the realm of Introductory Programming (CS1), an extensive body of literature exists, focusing on pedagogical approaches. Luxton-Reilly et al. [13] conducted a comprehensive literature review, which establishes five overarching categories: course orientation, teaching delivery methods, learning theories, teaching tools, and teaching infrastructure.

Course orientation pertains to diverse methodologies designed to enhance student learning outcomes by shaping the overall framework of the course. These methodologies encompass a spectrum of approaches, each offering unique advantages and disadvantages. One notably popular approach, currently being explored in a number of courses at the University of Bergen, is the implementation of *inverted classrooms*. In this model, students actively participate in exercises, collaborative problem-solving, and teamwork, fostering deeper engagement with the material. Despite initial development costs, inverted classrooms have been shown to yield significant benefits, including heightened student engagement, improved knowledge retention, and enhanced academic performance [4] [7] [10] [11].

Teaching delivery encompasses the methods through which the curriculum is imparted to students. The context in which CS1 is taught has demonstrated significant influence on student motivation, attitude, engagement, and perception of the discipline. Diverse contexts have been examined in literature, including CS1 integrated into real-world applications [8] [12], ecology [15], mobile phones [14], real-world data analysis [1], and gaming environments [2] [5]. While varying contexts possess potential to enhance student motivation, a majority of publications offer case studies focusing on contextual delivery rather than robustly evaluating the course's impact on students.

Collaborative learning strategies have been acknowledged for their numerous benefits in introductory programming environments, such as enhanced motivation, deeper comprehension of content knowledge, and the cultivation of soft skills like teamwork and communication. These strategies encompass peer review, pair programming, and cooperative group activities, with pair programming being particularly acclaimed for its effectiveness.

Another notable delivery method is testing. Several studies advocate for the early introduction of testing in CS1, highlighting anticipated benefits such as

fostering reflective programming over trial-and-error approaches [3] [6], improving comprehension of programming concepts [9] [?], familiarizing students with industry practices and tools [?], and bolstering student confidence [?]. However, empirical evidence supporting these potential benefits in introductory programming remains somewhat limited.

A previous literature review [?] suggests that papers within computing education often lack a solid theoretical foundation. **Theories of learning** are notably absent within the realm of CS1. Furthermore, despite widespread discrediting of learning styles as pseudoscience [?], 19 papers have addressed this concept [?, ?, ?]. It is evident that research under this topic is needed. That one should go further than simply doing case studies of tools and teaching methods, but rather consider theoretical implications and construct knowledge which is more general.

Teaching tools are frequently designed to augment CS1 instruction. While the majority of papers concentrate on the development of such tools, only a limited few are dedicated to their evaluation. Among the array of tools, programming editors [?], pedagogical environments [?], libraries, and APIs [?, ?, ?] are notable examples. Initially, tool developers prioritized creating resources to aid students in code composition practice. These tools are tailored to provide grading, feedback, and targeted assistance on specific programming concepts.

Teaching infrastructure encompasses various components, including physical resources such as labs and hardware, digital resources like software, and human resources such as teaching assistants and mentors. At the Department of Informatics, University of Bergen, a prominent practice involves the employment of student workers, particularly teaching assistants and mentors, which has long been integral to delivering high-quality education. The literature also highlights the benefits of utilizing teaching assistants [?, ?].

A growing research direction is the development and use of electronic textbooks which incorporate a number of active components such as video, code editing and execution, and code visualization [?]. Such a resource has been developed at the University of Bergen[1], which has (anecdotally) received praise from both instructors and students.

The existing CS1 literature offers a substantial reservoir of knowledge that can be leveraged to enhance the educational practices within our institution. However, there are notable gaps in understanding, particularly concerning the complex interplay of various factors and the generalizability of educational theories underlying interventions and tools. [Reference] serves as an invaluable resource that the didactics group can utilize for further research and aiding in informed decision-making regarding our instructional strategies.

References

[1] Ruth E. Anderson, Michael D. Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. A data programming cs1 course. In *Proceedings of the*

[1] https://inf101v23.stromme.me/

46th ACM Technical Symposium on Computer Science Education, SIGCSE '15, page 150–155, New York, NY, USA, 2015. Association for Computing Machinery.

[2] Jessica D. Bayliss and Sean Strout. Games as a "flavor" of cs1. *SIGCSE Bull.*, 38(1):500–504, mar 2006.

[3] Tom Briggs and C. Dudley Girard. Tools and techniques for test-driven learning in cs1. *J. Comput. Sci. Coll.*, 22(3):37–43, jan 2007.

[4] Jennifer Campbell, Diane Horton, Michelle Craig, and Paul Gries. Evaluating an inverted cs1. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 307–312, New York, NY, USA, 2014. Association for Computing Machinery.

[5] Daniel C. Cliburn. The effectiveness of games as assignments in an introductory programming course. In *Proceedings. Frontiers in Education. 36th Annual Conference*, pages 6–10, 2006.

[6] Stephen H. Edwards. Using software testing to move students from trial-and-error to reflection-in-action. *SIGCSE Bull.*, 36(1):26–30, mar 2004.

[7] Ashraf Elnagar and Mahir Ali. A modified team-based learning methodology for effective delivery of an introductory programming course. In *Proceedings of the 13th Annual Conference on Information Technology Education*, SIGITE '12, page 177–182, New York, NY, USA, 2012. Association for Computing Machinery.

[8] Mark Goadrich. Incorporating tangible computing devices into cs1. *J. Comput. Sci. Coll.*, 29(5):23–31, may 2014.

[9] Michael Hilton and David S. Janzen. On teaching arrays with test-driven learning in webide. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '12, page 93–98, New York, NY, USA, 2012. Association for Computing Machinery.

[10] Patricia Lasserre and Carolyn Szostak. Effects of team-based learning on a cs1 course. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, page 133–137, New York, NY, USA, 2011. Association for Computing Machinery.

[11] Celine Latulipe, N. Bruce Long, and Carlos E. Seminario. Structuring flipped classes with lightweight teams and gamification. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, page 392–397, New York, NY, USA, 2015. Association for Computing Machinery.

[12] Hai-Ning Liang, Charles Fleming, Ka Lok Man, and Tammam Tillo. A first introduction to programming for first-year students at a chinese university using lego mindstorms. In *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 233–238, 2013.

[13] Andrew Luxton-Reilly, Simon, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, et al. Introductory programming: a systematic literature review. In *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education*, pages 55–106, 2018.

[14] Dermot Shinners-Kennedy and David J. Barnes. The novice programmer's "device to think with". In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, SIGCSE '11, page 511–516, New York, NY, USA, 2011. Association for Computing Machinery.

[15] Michael Wirth. Ecological footprints as case studies in programming. In *2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH)*, pages 188–193, 2009.