**Measuring Undergraduate Understanding in Computer Science**

Emma Frampton

University of Delaware

**Introduction**

It is almost impossible to go a day without using technology. We utilize software in education, healthcare, recreation, and government. Technology is an integral part of life and is part of almost every industry; however, software does not appear out of nowhere. So much of daily life and humanity relies on these platforms, thus, the creation and maintenance of software has created a rapidly increasing need for programming developers. The U.S. Bureau of Labor Statistics projects employment in computer and information technology occupations “to grow 15 percent from 2021 to 2031, much faster than the average for all occupations.” This increase is expected to result in “about 682,800 new jobs over the decade.” (U.S. Bureau of Labor Statistics, 2022) As a result, the computer science major inside American universities has grown exponentially, with the number of bachelor’s degrees rising “by 74 percent between 2009 and 2015, compared to 16 percent growth across all fields.” (Hambrusch, 2017) With lecture halls speedily filling with more and more computer science students and prospective developers, the effectiveness of computer science education at the university level has become increasingly important. Consequently, there has been a great debate between college educators about the proper way to assess students' understanding of computing principles and practices in the classroom setting. Although computer science exams have a historical foundation in measuring learning, projects are much more effective at evaluating undergraduate computer science students’ learning experiences, encouraging good programming skills, and increasing diversity retention in computing.

**The Origin and History of Assessment**

Assessments were developed to reflect the measured merit of someone’s skill or knowledge, therefore, the validity of assessments is determined by their ability to measure understanding. According to Gwei-Djen Lu, a Cambridge scientist and historian, the first assessments were originally designed to test “medical and surgical proficiency for the protection of the public from unskilled practitioners” (Gwei-Djen Lu, 1963) during the ancient Chinese Sui dynasty in 605. Without exams and practitioner assessments, doctors could operate without any qualifications or repercussions. Examinations created the distinction between the “qualified'' and the “unqualified”. Thus, raising the status of those who became qualified, “medical men were in dignified, if not the most exalted, company.” (Gwei-Djen Lu, 1963) Having a doctor with a passed examination would guarantee a higher likelihood of living for patients. The effectiveness of exams compared to the lack of learning metrics led to a boom in assessments in a variety of skills and fields. Although the origins of examination assessments may feel like a disconnected relic from the past, they help us understand that assessments are supposed to measure the ability of someone's particular skill. In this scenario, efficient computer science assessments should aim to test computer science students on their programming skills, problem-solving abilities, and understanding of software development principles.

**Failure of Computing Exams**

One of the earliest metrics of assessment, written examinations, are a tried and true method of measuring students’ knowledge, rooted in learning principles with a history of success. Computing exams tend to be quite standard with test writers borrowing from other computing educators. Multiple researchers at the RMIT University Melbourne City Campus found that “few exams are written entirely from scratch, with most being modified versions of previous papers,” (Sheard et al. 2013) showing the standardization between computer science curriculums. Examinations are a simple way to provide an easily comparable measurement of all students on a specific subject set. Furthermore, many instructors expressed “satisfaction with their assessments effectiveness,” stating they were comfortable with how well the examinations did their job “as an assessment tool.” (Sheard et al. 2013) Examinations are an easy way for instructors to systematically evaluate all of their students; overall instructors are satisfied with the gauging of their students. Finally, exams frequently align with and utilize educational learning objectives. The RMIT researchers found that “Bloom’s Taxonomy [was] mentioned most frequently” (Sheard et al. 2013) between educators. Bloom’s Taxonomy uses a classification system to distinguish different levels of human cognition, this principle allows teachers to classify the depth of assessment methods and improves learning evaluations. Examinations have been used for decades and rely on educational principles and allow instructors to easily standardize material for large pools of students.

On the contrary, written exams are perceived as outdated and do not consider the nuances of computing making them an insufficient assessment tool. Ibrahim Albluwi, an Associate Professor at the Department of Computer Science Princess Sumaya University for Technology, explains the grading discrepancies between CS1 (introductory computer science courses) exams. His study revealed that “graders scoring simple pieces of code could disagree significantly,” with a “variance in how different logic errors are perceived,” and “variance in how important syntax is,” (Albluwi, 2018) between instructors in the same department. Exams aim to standardize grading, but often fail when different instructors are grading the same piece of broken code as coding principles are unequally weighted between educators. More importantly, it was discovered that although the occasional computer science instructors will use an educational learning objective, the majority of exam writers will not utilize this component, with the practice of exam writing being “largely based on intuition, experience and pragmatics rather than any expert educational knowledge or training.” (Sheard et al. 2013) Unchecked exam creations can create errors thus impacting the validity of the exam’s evaluation. The combination of unrestrained exam modeling and grading discrepancies can send mixed messages to new collegiate programmers. When students “receive 2/8 for a solution with an off-by-one error or for a solution that has only syntax error,” they are “implicitly learning something different than those who get an 8/10 for the same piece of code,” (Sheard et al. 2013) making the standardization of exams insufficient on a broad scale. Although exams appear to properly measure the learning between students in computer science courses, they ultimately fail at the grading level as variances between graders and exam writers can greatly impact the exams.

**Computing Projects and Collaborative Exams**

Computer science project assessments are difficult to implement in CS1 courses and can encourage plagiarism when compared to exams. Lydia Fritz, a Computer Science Senior Lecturer at Grand Canyon University, describes that projects can deter new “students at the introductory level who have yet to develop the necessary skills to identify the source of errors and make necessary corrections,” as “a single mistake can cause an entire assignment to fail.” (Fritz, 2019) With more-open ended projects there are more problems for students to get stuck on, this can make many students frustrated and can lead them to drop the course altogether. Furthermore with projects, there is more of an opportunity for cheating between students, with even the “well-intentioned student finding themselves assembling resources to create a solution they don’t fully understand.” (Fritz, 2019) This leaves course objectives unmet and defeats the purpose of assessments. Projects are regarded as easier to cheat on resulting in the rewarding of undeserved merit to students who have not learned the material. Although projects attempt to prevent the inequity in grading from exams, they fail to prevent misconduct which can undermine the importance of assessments.

Ultimately, projects are the best method of assessing student course knowledge as they develop communication and career skills, increase performance for women programmers, increase student interest and scores. Hans Yuan and Paul Cao, researchers at the University of California, San Diego Jacobs School of Engineering, prove that “existing research has shown that students exposed to pair programming and collaborative exams performed more strongly,” than those who did not. (Cao and Yuan, 2019) Pair programming collaborative assessment projects are found to increase student scores and work quality, thus, increasing learning outcomes for more students. Projects have also been found to be good for the retention of women in computing. Associate professors of computer science at the University of South Florida Zachariah Beasley and Ayesha Johnson found that projects and pair programming increase “participation and [improve] outcomes, particularly for female computer science students.” (Beasley and Johnson, 2022) Scores for female computer science students significantly increased as “female students increased their confidence in asking questions in class” and subsequently scored “12% higher in the course when utilizing pair programming.” Women are historically underrepresented in the computer science industry and providing more opportunities for women to succeed in the classroom is important to diversify the field of computing. Finally, they discovered that projects encourage students to communicate technical information in an efficient way. Projects and presentations better prepare learners for industry as undergraduate “computer science students must be able to communicate technical information to a variety of audiences, and therefore must be engaged in written and oral presentation activities.” (Fritz, 2019) By building assessments around the intersection of course and real-world applications, students are prepared for both the understanding of their topic and its further purposes and applications. Projects effectively create better computer scientists as they better prepare students for industry, improve communication and collaboration, and increase performance.

**Conclusion**

Essentially, project-like assessments perform better than exams by preparing students for industry, supporting the learning styles of women in computing, and encouraging the development of programming logic understanding. To preserve and grow modern technology, computing education is more important than ever and creating proper assessments that prepare students for industry at the undergraduate level is vital. Without the proper emphasis of project delivery, communication, group management, and collaboration, students will have to develop their team management skills outside of the classroom and in the workplace. Providing project assessments that best prepare students for the future and gauge the understanding of the material is the most effective way to increase student productivity and understanding in an ever-evolving world of technology and software.

**References**

Albluwi, I. (2018). A closer look at the differences between graders in introductory computer science exams. Ieee Transactions on Education, 61(3).

Beasley , Z., & Johnson, A. (2022, July 1). *The impact of remote pair programming in an upper-level CS Course: Proceedings of the 27th ACM Conference on on Innovation and Technology in computer science education vol. 1*. ACM Conferences. Retrieved December 5, 2022, from https://dl.acm.org/doi/10.1145/3502718.3524772

Fritz, L. (2018, November 30). *Effective assessment for early courses in Computer Science: Instruments other than out-of-class programming assignments.* Journal of Instructional Research. https://eric.ed.gov/?id=EJ1242643

Hambrusch, S.(2018, January 18) *NAS report investigates the growth of computer science undergraduate enrollments*. CRN. https://cra.org/crn/2017/11/nas-report-investigates-growth-computer-science-undergraduate-enrollments/

Lu, G. (1963). China and the origin of examinations in medicine. Proceedings of the Royal Society of Medicine, 56, 63-70.

Sheard, J., Carbone, A., D'Souza, D., Simon, & Hamilton, M. (2013, July 1). *Assessment of Programming: Proceedings of the 18th ACM Conference on Innovation and Technology in computer science education*. ACM Conferences. https://dl.acm.org/doi/pdf/10.1145/2462476.2465586

U.S. Bureau of Labor Statistics. (2022, September 8). *Computer and Information Technology Occupations : Occupational Outlook Handbook*. U.S. Bureau of Labor Statistics. https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Yuan, H., & Cao, P. (2019, August). *Collaborative Assessments in Computer Science Education: A Survey*. https://ieeexplore.ieee.org/document/8660529