NIET Journal of Engineering & Technology

Volume 14, Issue 2, 2023 ISSN: 2229-5828 (Print)

Why the High Attrition Rate for Computer Science Students:Some Thoughts and Observations

Shakti Sharma[1], Rishabh Sharma,[2] Alka Singh[3]

[1](Department of MCA, Noida Institute of Engineering and Technology India)
[2](Department of MCA, Noida Institute of Engineering and Technology India)
[3](Department of MCA, Noida Institute of Engineering and Technology India)

Abstract: *This paper is based on investigation of the possible causes for high attrition rates for Computer Science students. It is a serious problem in colleges/universities that must be addressed if the need for technologically competent professionals is to be met.*

Keywords: Attrition, Retention, Computer Science education

1. INTRODUCTION

Over seven hundred individuals have chosen computer science majors at our college. Still, the students efficiently complete their studies in this area every semester. The higher-level classes usually consist of a maximum of two parts, sometimes only one, but the initial classes for freshman include numerous parts to fit a large number of students. Advisers process equally many change-of-major applications during the prior registration and advise process equally as do authorization schedule recommendations.
What occurs when an applicant eventually opts to drop from the course and chooses to stay with the computer science major? There may be many causes why individuals leave school early, give up, or transfer to yet another institution. The field of computing turnover varies by factors, though attrition occur in all fields for factors like as friendships, illness, money active duty, or an additional job. A couple of of these features will be discussed in the parts that follow. Dropout rates for CS programmes have been observed as high as between thirty and forty percent at some institutions. All of this, a significant amount of the dialogue next will centre on concerns linked to the first of the two CS courses.

2. INADEQUATE PRE- AND DURING-COLLEGE PSYCHOTHERAPY

If asked, pupils give an array of often not ideal reasons for choosing to major in IT. Because carers, school counsellors, and students itself often have imperfect notions about an individual's aptitudes, often individuals are directed towards careers in computer science. They contend that since "anything is going to machines," computer engineering is an option that students cannot get right with. Those same pupils often feel ill-prepared for entry

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

into college with this major cause they weren't asked to take any more science, math, or linguistics courses beyond the bare demand.
Here are a few myths among learners regarding computing. Many of them enrol in computer-related courses, do well in them, and believe that technology is just word the process, Excel sheets, and surfing the web. They are being encouraged to enrol in more computer courses since they do well. Many other pupils have aspirations to be video game programmers given that they appreciate playing video games. a lot of the time, they are clueless of the computer and math skills required for such jobs.

A pupil's ability for achievement in education is additionally significantly affected by their guidance. One of the pupils in one of the writer's computer design classes hadn't previously taken a computer science course! He took part in the programme "since he needed an undergraduate level design course," stated to his adviser. Since acquiring a foreign language is difficult, pupils at the college where the authors worked before were often urged to take computer programming languages over foreign languages because CS1 as the and CS2 could be substituted for foreign languages. Others were disappointed with the fact that CS1 ended out to be more difficult still.

3. INADEQUATE PROBLEM SOLVING AND ARITHMETIC SKILLS

Because arithmetic is an essential part of computing, pupils who are interested in succeeding in the discipline must have a strong grasp of the topic at hand. The connection among algebra and a typical computer technology and business curriculum has been shown in [1]. Math-proficient students tend to do better with the field of computing. Students have improved grasp of the construction of algorithms, scientific processing, and data collaborations. As consequently, they're able to fix issues better and deliver designs of outstanding quality from standards.
Computer science pupils often have very weak problem-solving skills. Beaubouef, and colleagues [3] looked into addresses to help beginning learners improve their problem-solving abilities. The majority of the time to pick this up is when somebody can't solve basic word-related issues. While word problems are an essential part of most CS1 being presented courses, phrase solving is more than just a mathematics skill. It was observed by the researchers that pupils who score well on multiple-choice tests might not do nearly as well in exams which ask them to create a basic C++ (or Java, for example) application according to a hypothetical situation.
A comprehensive retesting showed that the issue had nothing to do about syntactical comprehension. Students who did well on multiple-choice tests were able to properly "head trace" correct syntactic code, identify syntactic mistakes in incorrect code, and perform perform a complex process that was laid out in plain English. The difficulty arose from the inability to first define the algorithm.
Naturally, grammar has no bearing on one's ability to create a procedural method (oriented towards objects techniques are addressed below); in reality, syntax has zero bearing upon the language selected until the CS1 grade. It exists to teach that ability. It turns out that many entering STEM students lack this ability, at the most basic level, is worrisome. Our CS1 programmes starts with two to three weeks of education committed teaching in basic solving issues.
This "Take-22" scenario is excruciating. In the words of Donald Knuth, "CS= problem handling." Learning for students how to design machines to solve issues is a big part of what they do. Yet, teaching fundamental reasoning is outside the scope of the programme because it is a prerequisite to CS1. But if these skills are lacking, more pupils certainly will drop the course, this will further aggravate our problem of high drop prices. These researchers speculate that upgraded CS1 achieves and greater retention in the first two years of the information technology curricula would arise by requiring students pass a course on basic computational thinking in language-based algorithm layout.

4. BADLY CONSTRUCTED CS1 EXPERIMENTAL PROGRAMMES

Few could argue that that lab sessions are an essential aspect of CS1 and CS2 training. In the words of Denning

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

et al. [5], computation "rests at a junction among the key processes of technological innovation and applied arithmetic." Each of these methods are equally significant and vital to the subject, which is a unique blend of theory, detachment, and design interactivity.

Given how crucial laboratory courses are for inexperienced programmers, numerous institutions fall short in this area. As stated by Walker [8], CS1 laboratory classes "are utilised for pupil homework assignment execution" at several universities. On first look, this could seem to make logical sense, particularly as fresh developers need all the knowledge they can obtain.

However, such an approach is frequently disastrous. Wright adds that "teacher-assisted troubleshooting sessions that are unpleasant to learners and teachers tend to devolve into homework-centric laboratory." These fixing sessions do not foster private teacher-student relationship nor do they encourage a growth of programming skills. “One learner can consume the instructor's focus for the greater part of an experiment period," referring to the possibilities. The remaining pupils have to hold off until it is time to meet the the lab assistant or teacher. "Developing skills weren't acquired in these debugging discussions," is the reasonable assumption. Project logic and planning have been reduced to the phases of code, organise, and debug. Walker’s makes many really good suggestions that will improve the CS1 labs because they exist nowadays.

5. INSUFFICIENT PREPARATION AND ENCOURAGEMENT

There is commonly accepted among scholars that repeated practice creates perfect, nevertheless in a seventeen-week term, a beginner's course does not typically include more than dozen or fifteen programming projects. It has many explanations for that, the primary being the enormous section lengths of most CS1 classes. Both a computer-generated rating technique or a lot of time must be invested to examine a lot of scripts.

Certain educators have ideological problems to automated grading structures, yet many of them favour them. Learners ought to be allowed to have their code reviewed by a seasoned developer who can offer helpful feedback. Input of this kind cannot be generated by machines. Also, it is incapable to provide instructors the helpful input that would allow him to see the areas where pupils might profit from additional guidance or exercise.

By the same a direction, but if giving checks and "pop quiz" weren't such an enormous amount of time, many educators would hand out more of it. While multiple-choice questions is a possibility on numerous examinations and examinations, "hand marking" is a laborious procedure of various some sort. The period of time necessary for grading everybody grows as class sizes rise in an attempt to save expense for college decline out of desperation. That doesn't seem as big as it seems problem at higher education institutions if professors gain access to transferring students, because these pupils are often well-prepared of evaluating CS1 and CS2 projects and evaluations, even though their aren't sufficiently prepared to help teach these.

6. MASTER'S DEGREE INSTRUCTORS

Several universities do not have reservations about employing graduate learners to provide instruction in beginning technological programmes. Although many graduate pupils are thrust into a significant number of basic programming classes with little or no setup, many postgraduate students are in fact capable of delivering CS1 and CS2 programmes. Although all graduate students in the field of computing should be technically competent in instructing the class, if they suffer with speaking English or are timid they may not be at all prepared for speaking in sunlight to a group or pupils. Anyone should be competent in written and verbal communication, and individuals who work in computer science occupations especially have to be ready to do it [2]. Whenever an instructor suffers with spoken language, the calibre of the class is impaired and essential opportunities for learning have been lost upon learners. Additionally, a lot of freshmen don't treat alumni with the same reverence that they would give treat an educator or instructor. Many would disagree with the idea that being an effective educator includes more than simply having understanding the subject matter.

Big educational institutions are particularly notorious for this terrible treatment of recent graduates and undergrad.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

Before seeing a professor within the lecture hall, every one of the contributors was a second quarter lower ranks, or a full semester into his or her technical major. Under the four-year level, graduates taught all of the computational science modules. When put alongside the added pressure of a demanding major, this might occasionally give students an appearance that they are inconsequential. For certain learners, this may be excessive pressure for them to handle.
.

7. WEAKNESSES IN THE ADMINISTRATION OF PROJECTS

In the initial or subsequent decade, technological trainees typically abandon out of their curricula. Frequently, these students who are younger possess the appropriate routines for learning. Even those who have excellent routines for learning, however, frequently fail to develop certain essential organisational abilities that must be developed to successfully complete programming projects before deadline.
In respective initial or subsequent calendar year, technological students frequently transfer out of their degrees. Frequently, these younger students lacked the appropriate routines for learning. But even kids with strict study habits frequently miss some vital arguably the most significant problems that teachers and learners alike have is calculating the amount of time required to develop a piece software effectively. The processes of analysing, designing, coding, testing, and documentation are certainly nearly finished during this time. Professional software creation projects are legendary for being beyond budget in everyday life [9]. It shouldn't be surprising that student-written programmes, or initiatives, and tend to fall considerably farther beyond budget.
Learners that fail usually want to move immediately to writing in programming as opposed to conducting evaluation and architecture.
Maintaining records is a necessity only one brief believed, and not much else essential management of projects talents, which must be developed to finish assignments in programming on schedule Research concluded. The learner will usually wait unless the very last moment to start completing the assignment and keep going until it concludes or space runs out since he planned on approaching the project this manner from the beginning. These young people will succumb to disappointment, dissatisfaction and costly duplication.
Upgraded learners start activities early and generally conclude them on a period of time yet they remain not exceptionally good at estimating the quantity of labour and time involved. Nevertheless, when projects take longer to finish than originally anticipated owing to insufficient guidance or inadequate design, those pupils may continue to feel frustration and worry.
Individuals considering technological subjects need to possess proficient in organising resources in addition to time management. Whenever required, they need to have the required CDs, documents, folders for them, etc. on available. They must to encourage vital efforts that helps avoid effort loss. In addition, learners who require lab machines must be allowed arrange their work during periods when the machines are accessible and while generally aren't plenty of interruptions in the lab. Learners who operate from home have responsibility for extra gadget maintenance. Systems crash, networks collapse, electricity and communication delays happen, either in a laboratory or at residence. In order to successfully handle their time and finances in inadequate job situations, students need to develop particular skills.
Once learners acquire the requisite discernment to be competent to manage their time and money prudently, it usually requires individuals an extended period of computing and confronting real-life obstacles. Technology research could seem extremely challenging, laborious, and irritating when compared to research in other fields of study. Individuals who aren't totally invested are likely to give up and transfer to less rigorous degrees.

8. CONSIDERING EARLIER VS. LATE LANGUAGE AND ARTEFACTS

It might first appear like an insignificant issue since it is clearly just a matter of individual choice as to if we teach

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

things early (prior to typical traditional coding) or in the future (after conventional programming). But there's mounting evidence that the place (and manner) of object instructions matters tremendously. Over a three-decade duration, McConnell & Burhans [7] evaluated texts for basic information technology.
Researchers noticed that the median length of older basic programming textbooks was less than 500 pages. For example, an ordinary Cobol text encompassed 378 pages. A typical Java text is 876 page long, with more recent courses far longer.
The fact because simple types of data, mathematical gestures, structural and reasoning phrases, recurrence assertions, modules, and matrices are, on typical, less covered in newer courses is perhaps even more concerning. "I observe an unsettling pattern in the inclusion of both fundamental programming concepts and components," Mclaughlin and Burhans write in their assessment. Around forty percent fewer has been spotted in the covering of basic ideas in programming with a 62 percent decrease in the amount of coverage of topics associated with parts. Due solely to the emergence of GUI challenges there is a nearly threefold rise in input and output issues at the same period. What might be presented in a CS1 curriculum must definitely change in view of this an important rise.
A series of talks on the SIGCSE email list during March 2004 gave additional proof that a lot of educators have reservations about if objects should be introduced either early or late (or possibly at all) during a CS1 programme. Given the potency of the disagreement, Bruce [4] generated an essay summarising the majority of the theoretical claim.
Elliot Koffman initiated a lot of the debate with an article he posted, suggesting that a lot of instructors are reluctant to employ teaching strategies that place a focus on teaching materials early in essential programmes. Learning the Java language has been compared by Koffman to the 1960s "new arithmetic" project. According to him as a worker the primary explanation for this denial is that numerous teachers lack the necessary education to teach pupils on OD programming. They therefore depend on their current understanding while offering instructions. Still, he argues that "beneficial" learners have no difficulty implementing the objects in the first technique, while "less strong" CS1 learners find it challenging to fully understand the additional degree of conceptualization associated with bringing together items and graphical user interfaces.
Expanding on Koffman's "new math" comparison, Stuart Reges published an essay titled "The new CS." When that "the discussion moved from What to teach them early to the issue of When to introduce elements early in the morning," Reges inquiries. He requested opponents of the original encampment objection to provide evidence that:

1. Subsequently can be accomplished for an extensive number of instructors to teach merchandise effectively.
2. It is may be grasped by an extensive range of learners, including
3. The purpose's early methodology minimises as opposed to enhances the assortment of challenges as well.

This post understandably generated a lot of conversation amongst CS1 faculty members. Numerous teachers genuinely maintain that instructing everything at an early age generates greater problems than it addresses. Even while Graham himself favours teaching everything early, he concedes that "it is sometimes difficult to impart knowledge about objects immediately." This happens to be the one aspect concerning which the group reached almost total concurrence during the debate. It is evident that the matter is a contentious question that will certainly not go anywhere immediately.

9. CONCLUSION

Our handle the extremely actual problem of retention in mathematical curricula in this article. There can be plenty of reasons underlying low retention rates for pupils. We encompassed an extensive variety of topics that we consider are significant to digital science curricula at all institutions. Yet, it should be emphasised that additional variables could also be significant for some organisations, such as detrimental high schools, controversial instructors, management reorganisations, and transferring students to other adjacent colleges and universities.
Whereas the article examines several important issues associated with information technology preservation, we lack remedies for these pertains here. It has the goal that by discovering some of the main reasons of elevated

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.

dropping out rates, additional studies in these areas and measures to lower the high attrition rates could possibly be initiated.

References

[1] Beaubouef, T., "Why Computer Science Students Need Math," *SIGCSE Bulletin (inroads*), vol. 34 No. 4, December 2002.
[2] Beaubouef, T., "Why Computer Science Students Need Language," *SIGCSE Bulletin (inroads)*, Vol. 35, No. 4, December, 2003.
[3] Beaubouef, T., R. Lucas, and J. Howatt, "The Unlock System: Enhancing Problem Solving Skills in CS1 Students," *SIGCSE Bulletin (inroads)*, vol. 33, no. 2, pp. 43-46, June, 2001.
[4] Bruce, K. "Controversy on How to Teach CS1: A Discussion on the SIGCSE-members Mailing List". *SIGCSE Bulletin (inroads)*,Dec. 2004 (36:4). pp. 29-34.
[5] Denning, P. (ed) "Computing as a Discipline". *Communications of the ACM*, Jan. 1989 (32:1). pp. 202-210.
[6] Kendall, K. and Kendall, J. *Systems Analysis and Design*, Prentice Hall, New Jersey, 1999.
[7] McConnell, J. and Burhans, D. "The Evolution of CS1 Textbooks. Proceedings of Frontiers in Education. 2002. pp. T4G-1-T4G-6
[8] Walker, G. "Experimentation in the Computer Programming Lab". Inroads, Dec. 2004 (36:4). pp 69-72.
[9] Whitten, J. and Bentley, L. *Systems Analysis and Design Methods*, Irwin McGraw-Hill, Boston, 1998.

Publisher: Noida Institute of Engineering & Technology,
19, Knowledge Park-II, Institutional Area, Greater Noida (UP), India.