Humanities & Social Sciences Communications

ARTICLE

https://doi.org/10.1057/s41599-024-02794-z OPEN

Applying Q-methodology to investigate computer science teachers' preferences about students' skills and knowledge for obtaining a degree

Rita Takács [1✉], Szabolcs Takács [2,3], Judit T. Kárász [4,5], Attila Oláh[6,7] & Zoltán Horváth[1]

Students' dropout of Computer Science (CS) education is a crucial issue. This study aims to investigate one of the aspects which can help to retain our students. It is vital to examine CS education on the challenge of competence transition within the BSc curriculum from faculty perspectives. Teachers' expectations about students' skills and knowledge are important to understand because they influence learning outcomes and teaching methodologies. Acquiring hard skills and professional skills has outstanding significance in preparing students for their future careers. This study uses Q-methodology to identify the different viewpoints on the skills necessary to obtain a CS degree. Teachers from CS bachelor's programmes at a large European university participated in the study and shared their opinions. The participants were asked to rank the statements along a spectrum of "the most important skill" to "unimportant skill" (containing hard skills as well as soft skills). Factor analysis revealed five factors that are key components to obtaining a degree in CS: 1. analytical and technical skills, 2. teamwork and self-study experience, 3. group programming experience, 4. communication and problem-solving skills, 5. mathematical foundations and process modelling ability. This exploratory study applied a new research instrument and approach to incorporate teachers' perspectives into research and practice. These findings could help administrators develop new curricula in order to increase students' retention. We confirmed the need for acquiring professional skills and highlighted the need for designing new programmes which can improve students' soft skills to prepare them for work in the IT field.

[1] Faculty of Informatics, Eötvös Loránd University Budapest, Budapest, Hungary. [2] Institute of Psychology, Department of General Psychology and Methodology, Jena, Germany. [3] Károli Gáspár University of the Reformed Church in Hungary, Budapest, Hungary. [4] Doctoral School of Education, Eötvös Loránd University Budapest, Budapest, Hungary. [5] Institute of Education, Eötvös Loránd University Budapest, Budapest, Hungary. [6] Doctoral School of Psychology, Eötvös Loránd University Budapest, Budapest, Hungary. [7] Institute of Psychology, Eötvös Loránd University Budapest, Budapest, Hungary.
✉email: takacsrita@inf.elte.hu

Introduction

The increasing need for computer scientists, along with the importance of advancing innovations in education, led to conducting this research. Success in college is a highly complex phenomenon: it does not only depend on students' academic ability, but it also requires mastering academic skills. In addition, students must understand teachers' expectations and apply their professional skills effectively to meet the requirements. A high dropout rate requires a transformation of the roles of universities and colleges; what is more, it has the potential to disrupt traditional approaches to university programmes (Cabrera et al., 2006; Whittle and Rampton, 2020).

Our study took place in a European university, which made the findings unique. The relationships among teachers' perceptions and teaching practices, student outcomes, and dropout rates need to be investigated. Teaching professional skills in CS programmes may help increase student retention in CS. The most important question seemed to be how CS teachers teach or prioritise those hard and soft skills instead of understanding their preferences and perceptions of those professional skills. This article used data from teachers to examine university faculty members' expectations towards students. Similar surveys using the Q-methodology among members of the faculty have not yet been found in the field of CS. We found striking novelties among the results. Our aim was to investigate the expectations toward students and, thus, to improve their ability to respond to faculty expectations. In conclusion, we discuss the theoretical and practical implications of developing these skills.

Background

The dropout rate among CS students has emerged as a significant concern in higher education over the past two decades. Attrition from CS programmes has notable repercussions at the individual, economic, and educational levels, as highlighted in various studies (e.g., Nagrecha et al., 2017; Di Pietro, 2006; Belloc et al., 2011; Cabrera et al., 2006). As a worldwide phenomenon, this issue has garnered attention from policymakers, stakeholders, and scholars alike, emphasising the need for effective solutions. Addressing this dropout crisis necessitates multifaceted interventions aimed at fostering student retention. It calls for collaborative interdisciplinary efforts, fostering partnerships among stakeholders and academics to implement actionable strategies.

According to Vision 2030 European Union studies, education plays a crucial role in economic advancement by directly impacting entrepreneurship, productivity growth, and the expansion of employment opportunities, as well as promoting women's empowerment. Education serves to reduce unemployment and enhance the abilities and skills of students. The high attrition rate among computer science students also adversely affects the economy, as individuals with a computer science degree generally make a greater contribution to the GDP compared to those without one (Whittle and Rampton, 2020).

Given the extensive work on how universities can support students to meaningfully address those sustainability challenges, the question is 'How can we further help students' adaptation process to university challenges?' Our aim was to examine teachers' data and highlight the points in the education system about what kind of further improvements could be made at an institutional level.

Skills and attributes required for CS students to meet the challenges of their profession

Hard or soft skills?. Reynders et al. (2020) claim that soft (professional, process) skills are fundamental for students to acquire during their bachelor studies (ABET Engineering Accreditation Commission, 2012; Singer and Smith, 2013; The Royal Society, 2014).

These skills include problem-solving, initiative, critical thinking, creativity, digital literacy, communication, and teamwork, etc. (Clausen et al., 2020; Karatsolis et al., 2011). Soft skills such as problem-solving, critical thinking, information processing, and communication are broadly applicable to many academic curricula and receive increased attention in undergraduate programmes (ABET Engineering Accreditation Commission, 2012). These skills are not only important from an educational point of view but also from the perspective of the industry (Grey and Koncz, 2018; Pearl et al., 2019). Annual surveys of around 200 hiring managers from different companies indicated that problem-solving skills and the ability to work in a team are the most desired attributes of an employee.

Previous studies (e.g., Stehle and Peters-Burton, 2019) suggest that STEM education can prepare students for their future careers and improve their professional skills. CS degree programmes are typically strong in developing essential technical (hard) skills but weaker in their capacity to strengthen soft skills (Anderson et al., 2015; Bean, 2011; Begel and Simon, 2008). Numerous research papers offer special methods for developing professional competencies (for example communication and collaboration skills) within a CS curriculum, and a few have proven the usefulness of these skills within subjects (Burge et al., 2012; Burge et al., 2014; Carter, 2007; Carter et al., 2011; Dugan and Polanski, 2006; Falkner and Falkner, 2012; Herbert et al., 2021; Oliver et al., 2019) The term competence means work-readiness in the education systems. Competency is a combined collection of skills that makes a student able to perform a task in the academic context or fulfill a duty in the professional environment (Herbert et al., 2021; Hoffman et al., 2014). A CS graduate has acquired a set of skills enabling them to combine the acquired knowledge to meet professional requirements. These competencies are transferable from the educational context to the working environment (Danczak et al., 2017). Work-readiness is a combination of technical (hard skills) and non-technical (professional, generic, or soft) skills (Holmes and Oakleaf, 2013). The acquisition of professional skills requires extra time and attention. In this article, we use the terminology professional skills when referring to soft skills.

Previously researchers provided good examples and practices of how soft skills can be improved by certain subjects, e.g., Pollock (2001) and Hoffman et al. (2014) showed evidence of how written communication skills can be developed without undermining the technical content of the course. Other researchers (Anderson et al., 2015; Carter, 2007; McDonald and McDonald, 1993) suggested methods of how oral communication skills and teamwork can be strengthened within basic CS courses. Some researchers studied whole curricula regarding skills and competencies and did not narrow down their studies to one or two subjects. Falkner and Falkner (2012) developed another approach to examine communication skill development throughout the entire curriculum. Coleman and Lang (2012) applied a similar approach to measure teamwork by qualitative student feedback. Burge et al. (2014) also implemented a curriculum-wide framework for examining communication skills and teamwork in CS education. Anewalt and Polack (2017) and Anderson et al. (2015) implemented a curriculum-wide approach to communication skill development in hard-skill subjects. Surveys of students indicated that oral or written communication skills were effectively improved by teachers' intentional methods.

The computer science (CS) programme stands out as one of the most challenging programmes due to its dual demands: a fully comprehensive understanding of the theoretical background of computer science and an improved ability to put it into practice in order to meet all the academic requirements (Wu et al., 2014). Many students encounter difficulties with mathematical subjects, and the strong correlation between success in programming courses and performance in mathematical courses remains prevalent (Ali et al., 2014; Balmes, 2017; White and Sivitanides, 2015).

The Association for Computing Machinery has published a general report as a guidance to curricula in CS education (Sabin et al., 2015). Every CS curriculum in the EU contains mathematics subjects for at least 12 compulsory credits. In most education systems students must collect 30 credits per semester by successfully completing 8–10 subjects. Recent reports about several countries' higher education (Singer and Smith, 2013; The Royal Society, 2014; Woodin et al., 2010) have highlighted that logical skills are preferred in several undergraduate programmes. Students who retain in higher education, i.e., successfully accomplish their studies, have successfully acquired these professional skills. This also serves as an indicator of students' readiness to enter the world of work. Researchers can help and enable teachers to effectively support the acquisition of hard and soft skills during academic years.

One of the key factors in retaining students can be understanding teachers' expectations. It is important to develop within a curriculum both technical and non-technical skills. However, there is a tension between balancing discipline-specific content and soft skill development within a standard 3-year bachelor's degree programme curriculum (Al-Mahmood and Gruba, 2007; Herbert et al., 2021).

A comparative examination of previous research was undertaken to explore the myriad factors contributing to student attrition. Of particular interest is the performance of students following their initial academic year, where a lack of engagement in academic life and unpreparedness are identified as primary drivers of dropout during this critical period. Various scholars have advocated for significant reforms within academic institutions to better support CS students. Overall, there is a consensus on the importance of fostering students' professional skills and self-regulation to mitigate dropout rates (Trevors et al., 2016; Alarcon and Edwards, 2013; Moulin et al., 2013; Ruiz-Gallardo et al., 2016). Developing self-study methods helps students to stay on track and be able to achieve successful test scores. Additionally, the literature contains numerous concepts and programmes designed specifically for first-year students (Takács, Horváth, 2017). These programmes aimed at promoting students' professional skills aim to develop and improve their communication skills, interpersonal relationships, and critical thinking abilities. However, more actions are needed to retain CS students. According to university success requires the mastery of the "college student" role.

Study purpose. The rapidly changing environment of computer scientists requires new adaptations and provides new challenges. This is not different in the world of education. As previous research has shown, CS graduates require a far broader range of skills and attributes than technical capability but also need to have competence in non-technical skills and attributes such as communication, problem-solving, and management skills. It is time to investigate teachers' expectations because their perceptions and attitudes toward learning greatly influence the learning outcomes of whether students are successful or not during their studies (Brown, 2004; Fessakis and Prantsoudi, 2019).

High dropout rates have various effects on the CS education sector. One of the new perspectives that we wish to introduce in this article is that professional skills are vital for obtaining a degree and being successful in the labour market. We focused on understanding what teachers expect from students in order to obtain a degree. Developing a clearer approach could help students to retain their studies and better focus on teachers' expectations. Besides the technical skills, we also examined the non-technical ones in the curricula in the light of teachers' expectations.

As a step towards understanding the divergent perspectives of faculty members, this article investigates teachers' perceptions of the factors involved in student success at the university levels. We propose that, in addition to academic skills, university success requires soft skills (including the understanding of teachers' expectations) on the part of the student. What are computer science instructors' viewpoints on the technical and professional skills required to successfully complete a computer science bachelor's degree programme?

Hypotheses: 1. Teachers highlighted the stronger importance of hard skills (such as understanding mathematical principles) than soft skills (such as communication or teamwork skills).

2. Teachers regarded basic mathematical and programming foundations as the most vital factor.

Methods

Developing a clearer approach could help students to retain their studies and better focus on teachers' expectations. Besides the technical skills, we also examined the non-technical ones in the curricula in light of teachers' expectations. Q-methodology has been widely applied in humanities such as sociology, education, and political sciences but it has not yet gained much attention from experimental fields (Gao and Soranzo, 2020). We propose that teachers should possess a comprehensive understanding of students' skills and how to facilitate their development.

We applied the Q-method procedure because it was less time-consuming and more engaging for the participants (Klooster et al., 2008); moreover, it required more abstract thinking than just completing questionnaires with Likert scales (Gao and Soranzo, 2020). Participants were given the opportunity to systematically evaluate and compare all the questions presented at the same time. The Q-sorting technique helped the decision-making process and allowed participants to correctly differentiate their judgments. Hence, a greater coherent and correct evaluation of the decision process could be achieved (Gao and Soranzo, 2020). Therefore, the Q-sorting final results were primarily based totally on holistic thinking rather than on isolated ratings. In addition to this, the Q-sorting method managed to minimise the order effect, which psychologists frequently complain about in experiments (Atmanspacher and Römer, 2012; Brown, 1980; Brown, 1993; Brown, 2008; Desing and Kajfez, 2020).

The data collection and analysis of this study followed a standard Q-methodological procedure.

Concourse development. Our research applied the Q methodology. The first step in Q-methodological research demands extensive field knowledge and sources to collect statements about the topic (Brown, 2008). The items that were selected consisted of a combination of questions previously used in examining CS education and were mentioned as key factors (ACM: Computing Curricula, 2016). 11 teachers examined different descriptions and the Computing Curricula (2016) about skills and competencies which are essential for a computer science student. All in all, they collected 238 skills and competencies. After taking a vote, they agreed on the 15 most important skills and competencies, which we then included in our research.

Q-set construction. The Q-set construction process used the theoretical basis of key factors in CS education. 15 statements were selected because this was the result of the experts' opinion. The statements come from the university skills requirements list for computer science students. The final statements were the following: (1) Basics of mathematics (definition of graphs and basic algorithms, statistical terms, basics of logics and probability theory) (2) Foundation of computer science (basic algorithms and implementations, graph algorithms and basic code optimisation techniques) (3) Technical competences (e.g., Linux/Unix system knowledge, knowledge of development environments, automated testing and debugging) (4) Basics of programming (e.g., knowledge of an OOP language, a functional language, a scripting language, and simple programme design) (5) Group software development with agile tools (6) Software and project modelling (7) Test-driven development (8) Business process management (9) Machine learning (10) Numerical methods application (11) Career design and self-knowledge (12) Teamwork (13) Presenting the work (14) Individual learning, self-development, staying up-to-date on the latest trends of their profession (15) Abstract and analytical skills.

Participant Q-sorting. A specifically designed questionnaire in the form of a pyramid was used for the collection of the data, where we recorded teachers' responses. The questionnaire, which took 10–15 min to answer, was developed in a ppt file. The items that were selected consisted of a combination of questions previously used in examining CS education and were mentioned as key factors (ACM: Computing Curricula, 2016). The questionnaire was delivered through e-mail to 17 CS teachers at a large, public university in Hungary, who were asked to sort their opinions about the most important skills necessary to complete a CS degree. Participants were instructed to provide rank order of the 15 statements according to the importance of the skills they represented. Each statement was assigned a hierarchical position from "less important" (−3) to "rather important" (+3) in a forced-choice, quasi-normal, and symmetrical distribution grid (see Fig. 1).

Questionnaire. The teachers received the following list of skills and attributes:

(1) Basics of mathematics (definition of graphs and basic algorithms, statistical terms, basics of logic and probability theory)

(2) Foundation of computer science (basic algorithms and implementations, graph algorithms, and basic code optimisation techniques)

(3) Technical competencies (e.g., Linux/Unix system knowledge, knowledge of development environments, automated testing and debugging)

(4) Basics of programming (e.g., knowledge of an OOP language, a functional language, a scripting language, and simple programme design)

(5) Group software development with agile tools

(6) Software and project modelling

(7) Test-driven development

(8) Business process management

(9) Machine learning

(10) Numerical methods application

(11) Career design and self-knowledge

(12) Teamwork

(13) Presenting the work

(14) Individual learning, self-development, staying up-to-date on the latest trends of their profession

(15) Abstract and analytical skills

Ethical consideration. Before starting this research, teachers were asked for their voluntary consent, and they were allowed to stop anytime during the research. To guarantee their rights, privacy, and confidentiality of personal information, all the data collected through this study were processed, encoded, and Q-sorted as described in the process of data analysis for confidentiality. We did not prepare feedback for the respondents because of the exploratory nature of the study and participants conducted the questionnaire anonymously. In the university community anonymity is important: we received ethical permission to process it in an ethical way to preserve anonymity.

Q-factor analyses. For Q-sort correlations and factor analysis, we used IBM SPSS 27.0 Software (Du et al., 2022). Principal component analysis and varimax rotation were used to condense the data. The final factor structure was determined by comparing different factor solutions. Here, generally applied statistical criteria, such as an eigenvalue greater than 1.00, communalities were above 0.5 for all items, and the minimum was 0.6 (Brown, 2008; Brown, 1993; Brown, 1980; Desing and Kajfez, 2020). The researchers discussed the participants' responses until they reached a final agreement that led to the most informative factor solution. Based on expert discussions, it seemed that the number of factors is greater than 2. Being a good programmer is a complex phenomenon and the excellent programmer attitude can be described as a combination of several factors. The selected five-factor solution explains 82% of the study variance.

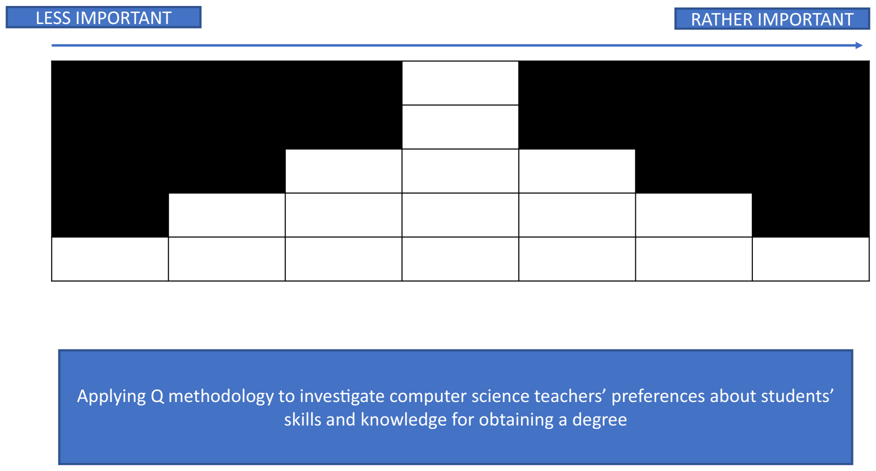
LESS IMPORTANT
RATHER IMPORTANT
Applying Q methodology to investigate computer science teachers' preferences about students' skills and knowledge for obtaining a degree

Fig. 1 The triangle of the Q methodology.

Table 1 Varimax rotated factors on the preference of teachers' perspectives of students' skills.

Type	F1	F2	F3	F4	F5
1. Basics of mathematics	0.75	0	−1.19	0.26	1.8
10. Numerical methods application	0.41	−0.42	−2.1	−0.26	−0.15
11. Career design and self-knowledge	−0.96	1.43	−0.7	−1.41	−0.07
12. Teamwork	−0.33	0.93	0.63	−1.08	0.36
13. Presenting the work	0.15	0.2	0	−0.73	−0.5
14. Individual learning, self-development, staying up-to-date on the latest trends of their profession	−0.67	1.96	0.39	1.63	−0.14
15. Abstract and analytical skills	1.08	1.09	1.08	1.09	−0.32
2. Foundation of computer science	1.23	0.39	−0.86	−0.01	0.69
3. Technical competences	1.61	−0.75	0.61	−1.1	−1.37
4. Basics of programming	0.96	v0.78	1.76	0,4	0.68
5. Group software development with agile tools	−0.97	−0.57	1.15	−1.6	0.98
6. Software and project modelling	−0.85	−0.33	−0.23	0.93	0.17
7. Test-driven development	0.32	−0.79	−0.63	0.72	−1.63
8. Business process management	−1.08	−1.77	0.09	0.99	1.11
9. Machine learning	−1.63	−0.58	0	0.18	−1.6

We have a 5-factor model, total variance explained: 82.28. In this case, the fit of the factor analysis was not highly important, because each factor represents an arrangement. In the following section, the three most important (2 or 3 weights) and the three least important (−3, −2 weights) factors are highlighted (Table 1). Appendix 1 shows the full outcome of the factor analysis which shows how each item loads on each factor.

Results

Structure of Q-type analysis. Next, we present the key findings as well as the results of the research, organised in sections corresponding to the research questions (Table 2).

There were five factors according to the results of the Q-factor analysis on the preference of teachers' perspectives of students' skills:

First factor:

The least important skills:

9. Machine learning
8. Process modelling
5. Group software development with agile tools

The most important skills:

15. Abstraction and analytical skills
2. Basics of computer science
3. Technical skills

Those who attached a higher value to the skills in the first factor regarded the IT/programming background as not so important, as opposed to the level of abstraction and the acquisition of technical skills. It means that the most important skill for them is the acquisition of stable CS foundations and thorough technical knowledge—and not hot topics of informatics (like modelling or AI). It means these teachers emphasised the importance of a stable foundation.

Second factor:

The least important:

8. Process modelling
7. Test-driven development
4. Basics of programming

The most important:

15. Abstraction and analytical skills
11. Career planning and self-knowledge
14. Independent learning, self-development, and staying up-to-date on trends

Those who attached a higher value to the skills in the second factor regarded programming as less important, and emphasised the necessity of development of individual learning/professional skills and self-management skills.

Third factor:

The least important:

10. Numerical methods and their application
1. Mathematical basics
2. Basics of computer science

The most important:

15. Abstraction and analytical skills
5. Group software development with agile tools
4. Basics of programming

Those who attached a higher value to the skills in the third factor were convinced that the development of the basics of programming is important, and they prioritise a deep mathematical foundation less. Their goal is to educate a kind of basic programmer who can code but not designers or computer analysts.

Fourth factor:

The least important:

5. Group software development with agile tools
11. Career planning and self-knowledge
3. Technical skills

The most important:

8. Process modelling
15. Abstraction and analytical skills
14. Independent learning, self-development, and staying up-to-date on the latest trends

Unlike the previous group, these teachers would like to enable their students of a high level of abstractions. This means that they would like to train professional computer experts and scientists rather than coding experts.

Fifth factor:

The least important:

7. Test-driven development
9. Machine learning
3. Technical skills

The most important:

5. Group software development with agile tools

Table 2 The placement of the statements in each factor.

Statement	F1	F2	F3	F4	F5
1. Basics of mathematics	9. Machine learning (−1.64)	8. Business process management (−1.77)	10. Numerical methods application (−2.09)	5. Group software development with agile tools (−1.6)	7. Test-driven development (−1.63)
10. Numerical methods application	8. Business process management (−1.07)	7. Test-driven development (−0.78)	1. Basics of mathematics (−1.19)	11. Career design and self-knowledge (−1.41)	9. Machine learning (−1.59)
11. Career design and self-knowledge	5. Group software development with agile tools (−0.97)	4. Basics of programming (−0.77)	2. Foundation of computer science (−0.85)	3. Technical competences (−1.09)	3. Technical competences (−1.37)
12. Teamwork	11. Career design and self-knowledge (−0.95)	3. Technical competences (−0.75)	11. Career design and self-knowledge (−0.703)	12. Teamwork (−1.08)	13. Presenting the work (−0.49)
13. Presenting the work6. Software and project modelling (−0,85)	6. Software and project modelling (−0.85)	9. Machine learning (−0.57)	7. Test-driven development (−0.63)	13. Presenting the work (−0.73)	15. Abstract and analytical skills (−0.31)
14. Individual learning, self-development, staying up-to-date on the latest trends of their profession	14. Individual learning, self-development, staying up-to-date on the latest trends of their profession (−0.67)	5. Group software development with agile tools (−0.56)	6. Software and project modelling (−0.22)	10. Numerical methods application (−0.26)	10. Numerical methods application (−0.14)
15. Abstract and analytical skills	12. Teamwork (−0.33)	10. Numerical methods application (−0.42)	13. Presenting the work (−0.002)	2. Foundation of computer science (−0.007)	14. Individual learning, self-development, staying up-to-date on the latest trends of their profession (−0.14)
2. Foundation of computer science	13. Presenting the work (0.15)	6. Software and project modelling (−0.32)	9. Machine learning (0.004)	9. Machine learning (0.18)	11. Career design and self-knowledge (−0.06)
3. Technical competences	7. Test-driven development (0.32)	1. Basics of mathematics (0.003)	8. Business process management (0.08)	1. Basics of mathematics (0.25)	6. Software and project modelling (0.17)
4. Basics of programming	10. Numerical methods application (0.4)	13. Presenting the work (0.19)	14. Individual learning, self-development, staying up-to-date on the latest trends of their profession (0.39)	4. Basics of programming (0.39)	12. Teamwork (0.35)
5. Group software development with agile tools	1. Basics of mathematics (0.74)	2. Foundation of computer science (0.39)	3. Technical competences (−0.75)	7. Test-driven development (0.72)	4. Basics of programming (0.67)
6. Software and project modelling	4. Basics of programming (0.95)	12. Teamwork (0.92)	12. Teamwork (0.62)	6. Software and project modelling (0.93)	2. Foundation of computer science (0.69)
7. Test-driven development	15. Abstract and analytical skills (1.07)	15. Abstract and analytical skills (1.08)	15. Abstract and analytical skills (1.08)	8. Business process management (0.98)	5. Group software development with agile tools (0.97)
8. Business process management	2. Foundation of computer science (1.22)	11. Career design and self-knowledge (1.43)	5. Group software development with agile tools (1.14)	15. Abstract and analytical skills (1.08)	8. Business process management (1.105)
9. Machine learning	3. Technical competences (1.6)	14. Individual learning, self-development, staying up-to-date on the latest trends of their profession (1.95)	4. Basics of programming (1.75)	14. Individual learning, self-development, staying up-to-date on the latest trends of their profession (1.62)	1. Basics of mathematics (1.79)

8. Process modelling
1. Mathematical basics

Those who chose these skills as most important believe that the primary task of CS education is the translation and interpretation of programming tasks into mathematical models and then encoding them in software. Therefore, their aim is to educate classical programmers, who have a strong mathematical foundation.

Limitations of the study

Even though this study presented interesting results, the authors believe that the conclusions should be interpreted carefully. It is important to note that this study has several limitations, especially regarding the selection of the research participants, thus we suggest that further research should be extended with additional variables. This study lacks qualitative, post-sorting data to support the factor analysis results. Concerning the limitations of the research, since the participation in the research was voluntary, the sample can be considered occasional and there was no prior mechanism to ensure its representativeness.

While this is adequate for generating hypotheses, the generalisation of findings to a larger population should be applied carefully, (with keeping these limitations in mind). It is crucial to examine the universality of these arguments by replicating this research across various types of universities. The question here is whether the degree to which teachers respond to these questions is influenced by institutional characteristics, such as public, or private universities, etc. In the future, students' levels of knowledge could be measured prior to entering university, allowing for better adjustments in the curriculum, which can be more tailored to their levels of knowledge. Moreover, this type of analysis can generate ideas for teachers (e.g., to introduce more study-supporting mechanisms) and course managers (e.g., to improve retention, and scholarship system).

Discussion

Universities around the world have already made significant progress towards purposefully developing curricula which include both hard and soft skills. According to researchers (e.g., Reynders et al., 2020; Singer and Smith, 2013), soft skills help less gifted or talented students to take exams and complete their academic studies. This exploratory study focused on examining teachers' opinions on important professional skills in the CS curriculum. Our studies aimed to investigate teachers' viewpoints about the balance of hard skills and soft skills in CS degree programmes.

The teachers' expectations about skills. According to our results, faculty members were quite articulate and explicit about the skills that they expect students to acquire. Both of our hypotheses were confirmed. Some teachers voted that acquiring basic hard and analytical skills is the most important to obtain a degree. First and foremost, students could have a deep understanding of their field of science and have good technical skills.

Regarding students' skills faculty members' expectations fell into five main categories:

Factor 1: Analytical and technical skills: it is important to develop a basic understanding of what computer science is really about.

Factor 2: It is essential to make students experience teamwork and programming/coding and master studying independently.

Factor 3: According to this factor the experience of working together on a project, i.e., programming in a team is the key to success.

Factor 4: Communication, problem-solving skills, and the ability to make complex abstractions are crucial elements for the success of a CS graduate.

Factor 5: According to the teachers' responses, in this factor the mathematical foundations and process modelling are important, i.e., the architecture of system development, the ability to see the whole picture, and the ability to manage it.

The first factor referred to the first step of programming: acquiring the necessary technical skills to solve basic programming tasks. Process modelling and teamwork turned out to be less important at this level, but precision in coding matters. Work focused on details and functional outcomes was essential; however, these tend to overshadow the importance of understanding the whole process of programming.

According to the respondents of the second factor, independent study, self-development, career planning, and self-knowledge were the most important for success in the field of CS. In this factor, the understanding of whole processes and test-driven development were less important aspects. All in all, self-development and self-management were found to be the primary skills.

In the third factor, the mathematical background was less important; on the other hand, working in a team, developing in a group, and having programming experience were vital skills in obtaining a degree in CS. According to the teachers in the fourth factor, the ability to imagine complex abstractions and independent study were essential; while, teamwork, career planning, and self-knowledge were the least important skills. According to the teachers' answers in the fifth factor, in-depth knowledge of artificial intelligence (test-driven development and machine learning) was the least important, but mathematical foundations and process modelling, and collaboration in a development team were highly valued.

Expectations related to professional skills. In the beginning faculty members had very clear expectations about what skills and knowledge students should have. They had strong beliefs that students should dedicate time and effort to learning programming. Moreover, students need to recognise the amount of time their schoolwork requires, and they need to prioritise a deep understanding of the logic of programming so that they can meet teachers' expectations. In addition, faculty members are convinced that university-level coursework is more demanding and requires self-management skills and a lot of independent study. Students who fail to understand this might not spend sufficient time to master key skills of precise coding and dropout.

Explicitness of expectations and teamwork. Faculty members had clear standards and expected students to be as explicit as possible to be able to understand the whole process of programming and address problems in an appropriate way. Teachers expect students to become more precise and structured to be able to put pieces of information together to develop their teamwork skills as well. Students should pay attention to their own development, and be responsive and precise about their work.

One area to which faculty members devote considerable attention to is the explicitness of programming; these expectations appeared in more than one factor. We can observe that teamwork skills are also highly prioritised in the list. We can conclude that students should have technical skills, such as writing precise code sequences and structure, and have a good ability to flexibly connect with others.

Expectations related to self-development and teamwork. Each group of teachers agreed that it was essential to improve students' soft skills of working together to increase the likelihood of student success. Establishing a rapport between students can be a crucial part of developing teamwork skills. Professors could even dedicate some of their lecture time to sharing tips about how to improve them.

We conclude our results by examining teachers' expectations regarding the different skills.

Our results demonstrate the importance of understanding professors' expectations for students' academic success at college. The analyses of the Q-method show that students' success depends both on developing expertise in programming and mastering logical skills, which can be as important as the acquisition of soft skills.

Conclusions

Students' lives have been supported in many ways. A different approach can be highlighted from an educational organisation perspective, e.g., it would be useful to support teachers by developing pedagogical strategies. We could confirm the result of

Reynders et al. (2020) that soft (professional, process) skills are fundamental for students to acquire during their bachelor studies (ABET Engineering Accreditation Commission, 2012; Singer and Smith, 2013; The Royal Society, 2014). According to proposals from the industry, every position requires professional skills, so prevention and promotion programmes are useful for both universities and companies. Prevention and promotion programmes aim to engage and empower students to choose adaptive coping behaviours, and make changes that reduce the risk of dropping out university degree programmes. It would be also vital to give a list of completed subjects to those who wish to interrupt their studies, so that they can continue their studies when they wish to come back to the university.

Familiarising students with their teachers' expectations can be crucial to help resolve the dropout phenomenon. Encouraging them to contact the professors during their office hours and ask them questions like what kind of strategies could be advantageous to develop professional skills. Faculty members' communication often fails because their efforts are wasted because they cannot achieve the goal; however, communicating their expectations plays an important role in students' success in obtaining a degree.

As both hard skills and professional skills possess immense value from a student's perspective, it would be advantageous to effectively balance work and study. Our research group investigated how much programming competence students should have to complete their first courses in CS education. In order to explore options for assessing students, teachers were asked about the skills that they found important.

Previous studies (e.g., Clausen et al., 2020; Du et al., 2022; Grey and Koncz, 2018; Karatsolis et al., 2011; Pearl et al., 2019; Stehle and Peters-Burton, 2019) suggest that STEM education can prepare students for their future career and improve their professional skills. The underlying goal of this work was to initiate a dialogue in the CS community on how to develop such skills. Based on this research we developed a structure of expectations for students and suggestions for further work to develop more comprehensive courses.

One of the most difficult issues faculty members faced was what kind of skills were required to succeed within the three-year bachelor's degree programme. According to the teachers, the degree programme should be developed in a way that it will provide students with a balanced education in the technical and non-technical disciplines of computing. Developing both hard and professional skills by strengthening the non-technical study areas will broaden students' skills and knowledge and increase their job prospects by better meeting industry requirements.

Receiving a degree in higher education and being successful in many workplaces in the field of CS are crucial. Therefore, understanding all the factors that teachers may expect from students can be important in order to prevent student dropout. In this article, we examined a previously under-explored area: what kind of skills teachers expect students to have in order to complete an undergraduate programme in CS. This problem disproportionately affects students, whose lack of background information about higher education may limit their awareness of how to engage themselves in their studies. These elements can have an important part in explaining student retention as well.

The conceptual model organises these elements into five factors that—according to teachers' opinion—contribute to student academic success.

Looking beyond the limitations of this study, this research has implications for the development of theory, future research, and practice at both micro and macro levels. At the micro level, this research demonstrates the value of the importance of teachers' expectations towards students. In particular, we demonstrated how teachers think that the combination of hard and soft skills affects of obtaining a degree.

At the macro level, the theoretical implications of this research point to the especially valuable component of bachelor's study programmes in the importance of acquiring soft skills to retain our students. Developing professional skills are an essential part of prevention programmes at universities, which aim to prevent students from dropping out. From this perspective, these bachelor's programmes provide many individuals with a higher level of competencies, thus enabling them to set advantages in workplaces. Effectively, we agree with Colliers and Morgan (2008) that it is highly appreciated that responding appropriately to teachers' expectations leads to higher rates of graduation and better jobs. Turning to practical implications, this research can inform stakeholders about the importance and content of soft skill-promoting programmes. Our findings emphasise the potential importance of creating such programmes which are designed around the needs of students. Another goal of these programmes would be to provide students with the skills to be able to recognise and respond to teachers' expectations. For example, self-study materials might be made available through online platforms to help students who do not have a grasp on the extent of their problems in meeting faculty expectations—and thus stop them from considering dropping out.

In order to prepare CS students to meet these new challenges, curricula should be revised and modernised. Changes to the existing curricula should be directed towards overcoming current challenges. However, further improvements are necessary to maintain the career development of junior scholars. The main findings of this study open new perspectives of research. Hopefully, other researchers will consider examining the potential impact of dropout on educational management and student grade tracking systems. Our results could give a deeper understanding of the dropout phenomenon. Analysing students' results could help administrators develop new programmes in order to increase retention.

Data availability

According to the ethical standards of the Faculty of Informatics, Eötvös Loránd University Budapest, the datasets analysed during the current study are available upon reasonable request by contacting the correspondence author.

Received: 6 December 2022; Accepted: 12 February 2024;
Published online: 29 February 2024

References

ABET Engineering Accreditation Commission (2012) Criteria for accrediting engineering programs. Retrieved from http://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2016-2017/. Accessed 1 February 2021

ACM: Computing Curricula (2016) Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, Association for Computing Machinery and IEEE Computer Society; Retrieved from https://www.acm.org/education/curricula-recommendations. Accessed 6 April 2023

Alarcon GM, Edwards JM (2013) Ability and motivation: assessing individual factors that contribute to university retention. J Educ Psychol 105(129). https://doi.org/10.1037/a0028496

Ali P, Ali S, Farag W (2014) An instrument to measure math attitudes of computer science students. Int J Inf Educ Technol 4(5):459–462

Al-Mahmood R, Gruba P (2007) Approaches to the implementation of generic graduate attributes in Australian ICT undergraduate education. Comput Sci Educ 17(3):171–185. https://doi.org/10.1080/08993400701538054

Anderson P, Heckman S, Vouk M, Wright D, Carter M, Burge J, Gannod G (2015) CS/SE instructors can improve student writing without reducing class time devoted to technical content: experimental results. in Proceedings of the 37th International Conference on Software Engineering, 2(2), 455–464., IEEE Press, Piscataway

Anewalt K, Polack J (2017) A curriculum model featuring oral communication instruction and practice. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, 33–37. https://doi.org/10.1145/3017680.3017775

Atmanspacher H, Römer H (2012) Order effects in sequential measurements of non-commuting psychological observables. J Math Psychol 56(4):274–280. https://doi.org/10.1016/j.jmp.2012.06.003

Balmes IL (2017) Correlation of mathematical ability and programming ability of the computer science students. Asia Pac J Educ Arts Sci 4(3):85–88

Bean JC (2011) Engaging ideas: The professor's guide to integrating writing, critical thinking, and active learning in the classroom (2nd edn.). Jossey-Bass, San Francisco, CA

Begel A, Simon B (2008) Novice software developers, all over again. Proceeding of the Fourth International Workshop on Computing Education Research—ICER '08, 3–14. https://doi.org/10.1145/1404520.1404522

Belloc F, Maruotti A, Petrella L (2011) How individual characteristics affect university students drop-out: a semiparametric mixed-effects model for an Italian case study. J Appl Stat 38:2225–2239. https://doi.org/10.1080/02664763.2010.545373

Brown G (2004) Teachers' conceptions of assessment: implications for policy and professional development. Assess Educ Princ Policy Pract 11:301–318. https://doi.org/10.1080/0969594042000304609

Brown SR (1993) A primer on Q methodology. Operant Subjectivity, 16. https://doi.org/10.22488/okstate.93.100504

Brown, SR (1980) Political subjectivity: applications of Q methodology in political science. Yale University Press

Brown SR (2008) Q methodology. in The SAGE Encyclopedia of Qualitative Research Methods. Given LM (ed) SAGE Publications, Inc., Thousand Oaks, CA. pp. 699–702

Burge JE, Gannod GC, Anderson PV, Rosine K, Vouk MA, Carter M (2012) Characterizing communication instruction in computer science and engineering programs: Methods and applications. 2012 Frontiers in Education Conference Proceedings, 1–6. https://doi.org/10.1109/FIE.2012.6462496

Burge JE, Gannod G, Carter M, Howard A, Schultz B, Vouk M, Wright D, Anderson P (2014) Developing CS/SE students' communication abilities through a program-wide framework. Proceedings of the 45th ACM Technical Symposium on Computer Science Education. pp. 579–584. https://doi.org/10.1145/2538862.2538984

Cabrera L, Bethencourt JT, Pérez PAAfonso MG (2006) El Probl del Abandon de los estudios universitarios Rev Electrón Invest Eval Educ 12:171–203

Clausen J, Rutledge D, Borthwick A (2020) Understanding stakeholder perspectives on technology infusion in teacher preparation. American Education Research Association Conference, San Francisco, CA

Carter M (2007) Ways of knowing, doing, and writing in the disciplines. Coll Compos Commun 58(3):385–418

Carter M, Vouk M, Gannod GC, Burge JE, Anderson PV, Hoffman ME (2011) Communication genres: Integrating communication into the software engineering curriculum. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE T). pp. 21–30. https://doi.org/10.1109/CSEET.2011.5876091

Coleman B, Lang M (2012) Collaboration across the curriculum: a disciplined approach to developing team skills. Proceedings of the 43rd ACM technical symposium on Computer Science Education. pp. 277–282. https://doi.org/10.1145/2157136.2157220

Collier PJ, Morgan DL (2008) Is that paper really due today?": Differences in first-generation and traditional college students' understandings of faculty expectations. High Educ 55(4):425–446

Danczak S, Thompson C, Overton T (2017) 'What does the term Critical Thinking mean to you?'A qualitative analysis of chemistry undergraduate, teaching staff and employers' views of critical thinking. Chem Educ Res Pract 18:420–434

Desing R, Kajfez R (2020) How to Use Q Methodology in Engineering Education Research. Paper presented at American Society for Engineering Education Annual Conference and Exposition. Virtual. https://doi.org/10.18260/1-2-34737

Di Pietro G (2006) Regional labour market conditions and university dropout rates: evidence from Italy. Region Stud 40:617–630. https://doi.org/10.1080/00343400600868770

Du X, Lundberg A, Ayari MA, Naji KK, Hawari A (2022) Examining engineering students' perceptions of learner agency enactment in problem- and project-based learning using Q methodology. J Eng Educ 111(1):111–136. https://doi.org/10.1002/jee.20430

Dugan Jr RF, Polanski VG (2006) Writing for computer science: a taxonomy of writing tasks and general advice. J Comput Sci Coll 21(6):191–203

Falkner K, Falkner NJG (2012) Integrating communication skills into the computer science curriculum. Proceedings of the 43rd ACM Technical Symposium on Computer Science Education-SIGCSE '12, 379. https://doi.org/10.1145/2157136.2157248

Fessakis G, Prantsoudi S (2019) Computer science teachers' perceptions, beliefs and attitudes on computational thinking in Greece. Inform Educ 18(2):227–258. https://doi.org/10.15388/infedu.2019.11

Gao J, Soranzo A (2020) Applying Q-methodology to investigate people' preferences for multivariate stimuli. Front Psychol 11:3445. https://doi.org/10.3389/fpsyg.2020.556509

Gray K, Koncz A (2018) The key attributes employers seek on students' resumes. Retrieved from http://www.naceweb.org/about-us/press/2017/the-key-attributes-employers-seek-on-students-resumes/. Accessed 2 February 2022

Herbert N, Herbert D, Wapstra E, de Salas K, Acuña T (2021) Integrating the development of professional skills throughout an ICT curriculum improves a graduate's competency. In: Arabnia HR, Deligiannidis L, Tinetti FG, Tran Q-N (eds.). Advances in software engineering, education, and e-learning. pp. 55–67. Springer International Publishing. https://doi.org/10.1007/978-3-030-70873-3_5

Hoffman ME, Anderson PV, Gustafsson M (2014) Workplace scenarios to integrate communication skills and content: a case study, SIGCSE' 14, Atlanta, GA, March 5–8, pp. 349–354

Holmes C, Oakleaf M (2013) The official (and unofficial) rules for norming rubrics successfully. J Acad Librariansh 39(6):599–602. https://doi.org/10.1016/j.acalib.2013.09.001

Karatsolis A, Cervasato I, Harras K, Cooper Y, Oflazer K, Abu-Ghazaleh N, Sans T (2011) Getting CS undergraduates to communicate effectively. Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education-ITiCSE '11, 283. https://doi.org/10.1145/1999747.1999827

Klooster PM, Visser M, de Jong MDT (2008) Comparing two image research instruments: The Q-sort method versus the Likert attitude questionnaire. Food Qual Prefer 19(5):511–518. https://doi.org/10.1016/j.foodqual.2008.02.007

McDonald G, McDonald M (1993) Developing oral communication skills of computer science undergraduates, in Proceedings of the Twenty-fourth SIGCSE Technical Symposium on Computer Science Education, pp. 279–282. Indianapolis

Moulin S, Doray P, Laplante B, Street MC (2013) Work intensity and non-completion of university: longitudinal approach and causal inference. J Educ Work 26:333–356. https://doi.org/10.1080/13639080.2011.653554

Nagrecha S, Dillon JZ, Chawla NV (2017) MOOC dropout prediction: lessons learned from making pipelines interpretable. In: Proceedings of the 26th International Conference on World Wide Web Companion. International World Wide Web Conferences Steering Committee, Perth, WA. pp. 351–359

Oliver KH, Ehrman JD, Marasco CC (2019) Vigilante innovation (VIX): case study on the development of student skills through a team-based design process and environment. Int J STEM Educ 6(1):36. https://doi.org/10.1186/s40594-019-0190-3

Pearl AO, Rayner GM, Larson I, Orlando L (2019) Thinking about critical thinking: An industry perspective. Ind High Educ 33(2):116–126. https://doi.org/10.1177/0950422218796099

Pollock L (2001) Integrating an intensive experience with communication skills development into a computer science course. Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, 287–291. https://doi.org/10.1145/364447.364603

Reynders G, Lantz J, Ruder SM, Stanford CL, Cole RS (2020) Rubrics to assess critical thinking and information processing in undergraduate STEM courses. Int J STEM Educ 7(1):9. https://doi.org/10.1186/s40594-020-00208-5

Ruiz-Gallardo JR, González-Geraldo JL, Castaño S (2016) What are our students oing? Workload, time allocation and time management in PBL instruction. a case study in Science Education. Teach Teach Educ 53:51–62. https://doi.org/10.1016/j.tate.2015.10.005

Sabin M, Alrumaih H, Impagliazzo JLB, Tang CZhang M (2015) Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology. ACM/IEEE-CS Information Technol Curricula. 2017. pp. 75–76

Singer S, Smith KA (2013) Discipline-based education research: understanding and improving learning in undergraduate science and engineering. J Eng Educ 102(4):468–471. https://doi.org/10.1002/jee.20030

Stehle SM, Peters-Burton EE (2019) Developing student 21st Century skills in selected exemplary inclusive STEM high schools. Int J STEM Educ 6(1):39. https://doi.org/10.1186/s40594-019-0192-1

Takács R, Horváth Z (2017) Dropping-out prevention of computer science students: developing studying, thinking and soft skills among students, using training programs. In: 11th International Association of Technology, Education and Development (IATED). Chova LG, Martinez AL, Torres IC (eds). pp. 1–6

The Royal Society (2014) Vision for science and mathematics education: The Royal Society Science Policy Centre. London, England

Trevors G, Feyzi-Behnagh R, Azevedo R, Bouchet F (2016) Self-regulated learning processes vary as a function of epistemic beliefs and contexts: mixed method evidence from eye tracking and concurrent and retrospective reports. Learn Instr 42:31–46. https://doi.org/10.1016/j.learninstruc.2015.11.003

White G, Sivitanides M (2015) An empirical investigation of the relationship between success in mathematics and visual programming courses. J Inf Syst Educ 14(4):409–416

Whittle M, Rampton J (2020) Towards a 2030 vision on the future of universities in Europe. Publications Office of the European Union. Retrieved from http://op.europa.eu/en/publication-detail/-/publication/a3cde934-12a0-11eb-9a54-01aa75ed71a1/, Accessed: 1 February 2021

Woodin T, Carter VC, Fletcher L (2010) Vision and change in biology undergraduate education, a call for action—initial responses. CBE Life Sci Educ 9(2):71–73. https://doi.org/10.1187/cbe.10-03-0044

Wu HT, Hsu PC, Lee CY, Wang HJ, Sun CK (2014) The impact of supplementary hands-on practice on learning in introductory computer science course for freshmen. Comput Educ 70:1–8

Acknowledgements

This work is supported by the Digital Education Development Competence Centre at Eötvös Loránd University, Budapest, with project number 2022-1.1.1-KK-2022-00003.

Author contributions

TR contributed to the conception and design of the study; led data collection, analysis, and interpretation; and wrote and revised the manuscript. TKJ aided in study design and participated in data collection and interpretation. TSZ also contributed to the study's design, conducted data analysis and interpretation, and assisted in manuscript revisions. HZ was involved in study design and manuscript revisions. OA contributed to the study design and manuscript revisions. All authors reviewed and endorsed the final manuscript.

Funding

Open access funding provided by Eötvös Loránd University.

Competing interests

The authors declare no competing interests.

Ethical approval

The Ethics Committee of the Institute of Psychology of Eötvös Loránd University Budapest approved the ethical permission, which was registered under the following number: 75/2020/P/ET. The study protocol was designed and executed in compliance with the code of ethics set out by the university in which the research was conducted, as required by the Helsinki Declaration.

Informed consent

Written informed consent was obtained from the participants.

Consent for publication

The data were anonymized before analysis and publication.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1057/s41599-024-02794-z.

Correspondence and requests for materials should be addressed to Rita Takács.