Arguments for and Approaches to Computing Education in Undergraduate Computer Science Programmes

Quintin Cutts[*]
University of Glasgow
Glasgow, Scotland
quintin.cutts@glasgow.ac.uk

Maria Kallia[*]
University of Glasgow
Glasgow, Scotland
maria.kallia@glasgow.ac.uk

Ruth Anderson
University of Washington
Seattle, USA
rea@cs.washington.edu

Tom Crick
Swansea University
Swansea, Wales
thomas.crick@swansea.ac.uk

Marie Devlin
Newcastle University
Newcastle, England
marie.devlin@newcastle.ac.uk

Mohammed Farghally
Virginia Tech
Blacksburg, USA
mfseddik@vt.edu

Claudio Mirolo
University of Udine
Udine, Italy
claudio.mirolo@uniud.it

Ragnhild Kobro Runde
University of Oslo
Oslo, Norway
ragnhilk@ifi.uio.no

Otto Seppälä
Aalto University
Espoo, Finland
otto.seppala@aalto.fi

Jaime Urquiza-Fuentes
Universidad Rey Juan Carlos
Móstoles, Spain
jaime.urquiza@urjc.es

Jan Vahrenhold
University of Münster
Münster, Germany
jan.vahrenhold@uni-muenster.de

ABSTRACT

Computing education (CE), the scientific foundation of the teaching and learning of subject matter specific to computing, has matured into a field with its own research journals and conferences as well as graduate programmes. Yet, and unlike other mature subfields of computer science (CS), it is rarely taught as part of undergraduate CS programmes. In this report, we present a gap analysis resulting from semi-structured interviews with various types of stakeholders and derive a set of arguments for teaching CE courses in undergraduate CS programmes. This analysis and the arguments highlight a number of opportunities for the discipline of CS at large, in academia, in industry, and in school education, that would be opened up with undergraduate CE courses, as well as potential barriers to implementation that will need to be overcome. We also report on the results of a Delphi process performed to elicit topics for such a course with various audiences in mind. The Delphi process yielded 19 high-level categories that encompass the subject matter CE courses should incorporate, tailored to the specific needs of their intended student audiences. This outcome underscores the extensive range of content that can be integrated into a comprehensive CE programme. Based on these two stakeholder interactions as well as a systematic literature review aiming to explore the current practices in teaching CE to undergraduate students, we develop two prototypical outlines of such a course, keeping in mind that departments may have different preferences and affordances resulting in different kinds of CE offerings. Overall, input from external stakeholders underscores the clear significance of undergraduate CE courses. We anticipate leveraging this valuable feedback to actively promote these courses on a broader scale.

[*]Working group co-leaders

CCS CONCEPTS

• **Social and professional topics → Model curricula**.

KEYWORDS

undergraduate; computing education; argument; curriculum outline

ACM Reference Format:
Quintin Cutts, Maria Kallia, Ruth Anderson, Tom Crick, Marie Devlin, Mohammed Farghally, Claudio Mirolo, Ragnhild Kobro Runde, Otto Seppälä, Jaime Urquiza-Fuentes, and Jan Vahrenhold. 2023. Arguments for and Approaches to Computing Education in Undergraduate Computer Science Programmes. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR 2023), July 7–12, 2023, Turku, Finland*. ACM, New York, NY, USA, 36 pages. https://doi.org/10.1145/3623762.3633494

1 INTRODUCTION

Back in 1989, *Computing as a Discipline* [35] defined the curricular embedding of computer science (CS) by addressing three charges: (1) Providing a definitional description of CS as a research field on its own, (2) Devising a teaching paradigm for CS aligned with standards and quality criteria in other established fields, and (3) Outlining how CS could be taught in undergraduate education. Over the past decades, the structured discussion about what should be taught to undergraduate students has evolved from the exemplars set forth in Denning et al.'s report, as witnessed by research publications and curricular recommendations such as the ACM Computing Curricula [101]. At the same time, the teaching of CS has begun to change from a focus on self-selected students to much broader and more diverse student populations, culminating in initiatives working towards CS being taught to everyone in school [17, 117].

This working group set out to study Computing Education (CE), the scientific foundations of the teaching and learning of subject matter specific to computing, under a very similar overarching question: Has CE become a field that can be adopted in undergraduate CS programmes? [28] Considering the analysis of Denning et al., the first charge has been responded to affirmatively: In his doctoral dissertation, Simon [108] applied Fensham's set of criteria [42] to analyze the field of CE; the outcome of this analysis was that all structural and content-related criteria but one (the existence of a professional association with an exclusive focus on CE) are fulfilled and that, thus, CE can be considered a discipline of its own. This working group report intends to take the first steps towards addressing the remaining two charges along with assembling supporting motivation for the implementation and further development of CE courses in undergraduate CS programmes.

Besides the above perspective concerning CE's maturity as a discipline, additional motivations for planning a course in this field at undergraduate level arise from the potential benefits that could be envisaged on the part of the students and of society more broadly. On the one hand, reflection on the teaching and learning processes brings *metacognition* into play. Based on the compelling evidence reviewed in Bransford et al.'s monograph [10], the practice of metacognitive thinking can lead to improved understanding in several contexts as well as enable knowledge and skill transfer to new settings, hence enhancing students' achievements and ability to learn autonomously. Metacognition is in fact a peculiar trait exhibited by experts when monitoring their approach to problem solving. "In short, students need to develop the ability to teach themselves" [10, p. 50]. On the other hand, by being exposed to CE content and practices earlier, students in a computing track would have a chance to consider teaching- and education-related professions as real career opportunities, a prospect that is currently still rather uncommon in most CS departments. Robertson notes the invisibility of such career options as a particular issue in her work on CS teacher shortages [99]. This should be expected to have positive repercussions in the medium term on the quality of computing instruction at all levels. Moreover, the clarity and efficacy of communication could be improved in all social and professional circumstances where scientific and technical insights pertaining to the computing area are to be conveyed.

Let us now come back to the major aims of this report by stating the research questions it is meant to address:

RQ1: What are arguments in support of introducing CE in undergraduate CS programmes?

RQ2: What are topics that different stakeholders, including, but not limited to, instructors, researchers, and employers consider both important and fitting for an undergraduate audience?

RQ3: What are current practices[1] in teaching CE to undergraduate audiences?

On a surface level, the answers to the above questions may appear to be readily available. For example, one might think that comprehensive overviews such as the recent *Cambridge Handbook of Computing Education Research* [43] and *Past, Present and Future of Computing Education Research: A Global Perspective* [6] would lend themselves directly to the design of a course on this topic, thus answering RQ2. The handbook [43], however, represents the current state-of-the-art of a research field and, thus, might be suited for graduate students and researchers who wish to deep-dive into particular aspects of this field. For an undergraduate audience, on the other hand, such a handbook is much less suited; in fact, this is precisely what we can see in other fields, e.g., in Computational Geometry, where textbooks [33] and handbooks [47] co-exist and fulfill different purposes. Similarly, several arguments based on first principles come to mind; while we will outline them below, we strive to obtain stronger and more meaningful arguments by empirically deriving them through interactions with stakeholders.

Another important analogy, preparing K-12 teachers, comes to mind as well. Many countries have a well-established tradition of formally training K-12 CS teachers as can be witnessed by the existence of dedicated researchers' and practitioners' conferences, reference handbooks [48, 103], and textbooks [52, 55] in this area. In countries, such as Israel and Germany, that have mandatory teacher training programmes [46, 62], such textbooks, if used, represent consensus among a broad set of stakeholders. By design, however, such textbooks and courses are tailored towards a very clearly defined audience: prospective K-12 teachers, possibly even differentiated further by the age group these teachers are supposed to teach, see, e.g., [38]. The intended audience for CE courses as envisioned by this working group, however, is not only more diverse in terms of where they might apply the concepts, skills, and competences taught, but — depending on the educational context — might have a different set of skills and competences they bring to this course: on the one hand, students in a teacher training programme usually take classes on general educational topics, including learning theories, so standard textbooks such as, e.g., [52], do not need to discuss such matters. On the other hand, the time allocated to these general classes cannot be used for more in-depth subject-matter courses in CS and this has implications for the assumptions about the depth of subject-matter knowledge. As a consequence, this working group agreed that research output and informal resources regarding teacher training programmes should not be the main focus of our studies. Instead, exemplars of course descriptions as well as the above-mentioned textbook should be consulted for comparison, i.e., to establish a baseline and to investigate whether synergies between long-standing programmes in teacher training and the envisioned CE courses could be identified.

In the attempt to provide a convincing answer to our above research questions, it may also be worth referring to Clements's Research-based Curriculum Framework [19]. Starting from a critique of the non-univocal uses of the term "research-based" in connection with the curriculum development process, Clements

[1]For the purposes of this study, *practices* specifically pertain to the structural aspects and content of courses and not to their pedagogical subtleties and implementation.

identifies ten desirable phases organised into three categories. The first category is about the *a priori* foundations relative to the subject matter (phase 1), more general aims (phase 2) and pedagogical knowledge (phase 3). The second category concerns the curricular structure according to some learning model (phase 4). Finally, the last category is about evaluation: “market” research (phase 5) as well as formative and summative research at different scale levels (phases 6-10).

By thinking in terms of this framework, the *a priori* foundations (phases 1–3) are expected to result from the three main tracks of analysis introduced below in sect. 1.3 and detailed in sects. 2.2, 2.3 and 2.4. The discussion within the working group will then provide insight into possible course structures in connection with learning models (phase 4). Finally, the first evaluation step (phase 5) should be at least partly covered by the outcome of the interviews as planned in sect. 2.2 as well as by the first round of the Delphi process introduced in sect. 2.4. As to the research-based evaluation phases 6–10, this endeavour cannot of course be undertaken by this working group and then is left as a future work perspective.

More generally, as observed by Coşkun Yaşar and Aslan in their review of curriculum theories [20], we have to be aware that any curriculum definition incorporates some socio-political perspective about the role and goals of education. In particular, while trying to coalesce some differences in terminology, Coşkun Yaşar and Aslan identify four main content-driven approaches to curriculum development (plus one focusing on the functioning of the development process itself): learner-centered, society-centered, knowledge-centered, managerial/social efficiency-oriented — the latter being focused on the assessment of observable behaviours. In this respect, since the main motivations underlying our present proposal rest on the “gap analysis” outlined in Section 1.2, we would aspire to reach a balance between the first three orientations, perhaps by prioritising some “society-centered” perspectives.

As far as the curriculum development process is concerned, a few scholars suggest interpreting it as an outcome of given power patterns, so emphasising a political/economic dimension — see e.g. Lau’s review [65]. In light of the models considered in Lau’s essay, an interesting and comprehensive analysis lens is represented by the *actor-network theory*, developed in the 80s by Callon and Latour in an attempt to understand social and technical change ([15], [64]) in terms of relationships and interactions between *heterogeneous* agents. In this context, “non-human agents, such as machines, texts and money [...] are equally as significant as human agents” [65, p. 40]. According to Carroll [16], this perspective can also help “to understand why specific curriculum changes are successful or unsuccessful, and how we can go about initiating the process of change in rational and sensitive ways” [16, p. 247]. Although it was not a deliberate decision, retrospectively we could also make sense of the process implied by our approach to curriculum development and implementation within an actor-network framework. Not only can this very report be seen as one of the non-human actants playing a role in the network (we hope, of course, it will be influential for future interventions), but the “stakeholders” involved in our research may somehow prefigure a complex relational network of mutually influencing actors shaping and supporting the introduction of CE courses in undergraduates programmes.

1.1 Early Steps Along the Road to CE Instruction

Since our literature review is restricted to the years between 2010 and 2023 (see sect. 2.4), we start with a short, non-exhaustive summary of past evolution(s) of CE instruction. We structure this early work using the four emerging themes identified by Tedre et al. in their overview of the changing nature of computing education [116], according to which CE is variously seen as (1) training about technology, (2) training for software development, (3) a field of academic recognition, or (4) education for computational problem-solving. According to the authors, “[e]ach theme has played a role throughout the history of modern computing, but their relative emphases have changed over the years.”

Among the earliest initiatives to introduce CE as a learning subject, we can mention the efforts described by Moulton and Moursund [79] and Poirot [86], the latter author having been particularly influential in the first attempts to devise teacher training programmes. In accordance with Tedre et al.’s “training about technology” perspective, Moulton and Moursund saw the “field of computers in instruction” as “divided into *teaching about computers* and *teaching using computers*.”

Poirot, on the other hand, while anticipating a growing individual and societal impact of computers, outlined a *Computers in Education* course, intended for secondary teacher education, covering the subject matter to be taught, the related motivations, the pursued objectives and potential teaching strategies. Then, in the mid 80s Taylor and Poirot [114] re-designed the course structure by submitting a survey to expert CS educators in order to identify major computing topics to cover in a curriculum for prospective secondary teachers. More generally, several of the earliest attempts to devise the role of computing in pre-college instruction mostly focused on training in-service or pre-service teachers about how to profitably use computers in K–12 education — see for instance [53, 56, 80] — or were centered on how to select the basic disciplinary content to convey, e.g. [44].

Regarding the “training for software development” perspective, we refer to Tomayko’s review [118] since in that case, the topic is how to train software engineers, not how to prepare educators. However, more recent work has started to think more broadly about software engineering education and skills (encompassing “software carpentry” and “codemanship”) [25, 26], for example major national skills initiatives such as the UK’s Institute of Coding [31], as well as in the wider framing of sustainability [124, 125].

When taking a “field of academic recognition” perspective, *research* in CE gains particular relevance. In this respect, Berglund et al. [9] report on issues concerning the design and organisation of a course on CE research with a methodological focus; their focus, however, is on a doctoral context, which is not a target of our present work. On the other hand, experiences of teaching this field of study to undergraduates are still scarce. In a column for *ACM Inroads*, Kaczmarczyk [58] listed five potential benefits:

- “[I]ntroducing students [. . .] to rigorous qualitative research” helps them develop the worthwhile “ability to listen, observe, and analyze other perspectives thoughtfully yet methodically.”

- For those students who are not going to attend graduate programs, this may be the only "opportunity to showcase the value of CS Ed Research" — a potentially inspiring value in a range of possible contexts of their future careers.
- The experience for "those who attend graduate programs" will be enriched "because they will have conducted research [...] in their field".
- "CS graduate school attendees might become interested in conducting further CE research," that would be valuable for our community.
- "[S]tudents who eventually embark upon an academic career in CS, regardless of research specialty, may well bring with them a respect for CS Ed Research and for teaching they otherwise, as experience has often shown, might not have developed."

As to the "education for computational problem-solving" perspective, the Israeli research-led, comprehensive approach to teacher education [45] is the earliest carefully designed such programme, and is well-documented in literature and books such as the "Methods of Teaching CS in the High School" course, presented more extensively in [63, 93]. This focus on computational problem-solving has emerged more recently as the primary purpose of K–12 education, often referred to as teaching Computational Thinking.

As an aside, based on the systematic literature review of Mirza et al. [78], the first proposal of a structured training programme for undergraduate teaching assistants was proposed back in the 1980s [94]. While this technically is an education-related course for undergraduate students, the authors mostly restricted themselves to organisational issues and do not address the specific content conveyed by the programme. We shall return to such courses as part of our literature review (see sect. 3.3),

We can finally mention Bell and Lambert's experience [8] as a possible model for the kinds of courses addressed to undergraduates we have in mind here. Their course, indeed, was not meant to qualify students for teaching, but to give "a taste of teaching and [...] to explore issues in CE, including the very education [students] are receiving." The general aims of Bell and Lambert's proposal include being able to create educational resources, contributing to mentoring activities while working in a variety of organisations, and developing knowledge about and possibly interest in careers as teachers, professors or researchers in CE.

1.2 Gap Analysis

As mentioned in the above subsection, CE has reached the stage of a mature discipline [108]. It is thus reasonable to ask whether CE should be taught on a regular basis and, if so, to which audience. While these questions could be asked with respect to any mature discipline, we will argue below that there is a particular urgent need to close the gap that arises from currently not teaching CE to an undergraduate audience. We focus on these issues first from a position of schools and academia, then turn to industry needs, and conclude by outlining the need for a full ecosystem around the teaching and learning of CE.

1.2.1 School needs. The most obvious direct issue for CE is the availability of suitably qualified teachers for the burgeoning school CS provision. While this is not a problem in all countries, it certainly affects many. Both the UK and the US, for example, have significant shortages and recruitment is not easy, see, e.g., [22, 113]. Most CS graduates go into industry careers; CS teacher training in both countries is being delivered often by teachers who have no significant prior CE experience [34]. Instead, they receive a crash course in programming and other CS material that may last just a few weeks; or else they may have studied a little computing in the past. While it is commendable that such teachers are attempting to fill the breach and while a recent study has shown that trade-offs are possible [12], it is hard to imagine other established subjects, like mathematics or physics, accepting student teachers with such limited disciplinary backgrounds. Disciplinary expertise comes with years of practice and the lack of subject matter knowledge has been identified by teachers as one of the main obstacles to effective teaching [104].

A national school system will need leadership in discipline-specific education at many levels. All of the following roles will be more effective if staffed by those with expertise in CE: government-level staff leading curriculum planning, design and oversight processes; national teacher professional development agencies; the officers of sub-national administrative areas, such as local authorities or local school boards, to coordinate, lead and strengthen local teacher communities; and finally, the staff of companies producing CE resources for the education sector. Staff in all these positions, for long-standing school subjects, will have had school experience and hence have at least some understanding of the subject. As research has shown the strong indirect effects of leadership on students' learning [51], CS in schools will benefit from leadership personnel with experience in CS and CE.

1.2.2 Academia needs. Academia, the established seat of CE practice, is faced with different challenges. First of all, in contrast to Shulman's emphasis on the equal importance of content knowledge, pedagogical knowledge, and pedagogical content knowledge [105], anecdotal evidence suggests that hiring decisions are often made based on an overemphasis on *either* content knowledge *or* pedagogical (content) knowledge. As Archibald and Feldman point out, however, there is an immense benefit if those who teach also engage in research, in particular, if this results in undergraduate or graduate research projects [3, Ch. 8]. Given the high workload imposed on those teaching first-year courses, it seems a natural choice to conduct CE research in these settings. With the presence of undergraduate CE courses, a two-fold advantage would be observable. First, instructors, especially at undergraduate-only institutions, would be able to recruit trained research assistants and, second, as pointed out by Kaczmarczyk [58] and noted above, undergraduate students would be able to conscientiously choose graduate studies in CE and eventually enter a teaching or CE research career well-prepared and with a clearer picture of such a careers' specifics.

1.2.3 Industry needs. While the obvious need for those knowledgeable and interested in both CS and education is in the school sector, students destined for the software industry will also benefit from some understanding of education in the discipline.

To begin with, IT professionals are always learning [37]. Whether picking up new languages and systems or working in new problem domains, quickly learning complex topics is required. Highly successful industry professionals move jobs on a regular basis. This requires highly-developed learning skills, to accommodate new problems or computing domains – but this is not always made explicit. Undergraduate programmes vary in how well they prepare students for this: for example, the more languages they are exposed to, the better; the more new topic areas they must develop software in, the faster they will pick up and be productive in a new problem domain. A successful programme teaches students how to learn, however much of this is part of the so-called *hidden curriculum* (where “hidden” stands for “not explicitly intended,” see e.g. [123]). This essential skill will be surfaced in the course frameworks outlined in section 4.

Once students progress in industry to become team leaders, they will need to coach and mentor their team members in order to get the most out of them. Team members themselves will often be asked to support and work with new staff [111]. Coaching and mentoring skills can be part of a CE course, but also CE research methods and methods in empirical software engineering overlap [84].

Both students and engineers learn a lot using informal learning resources [37]. In addition to taking courses, they browse documentation, blogs, tutorials, videos, etc. [111]. Some of the materials are pedagogically great, others bad. The lack of a great tutorial can drive someone to select a different technology just because they cannot get started. Adoption of new technologies is about making them understandable and easy to adopt: this justifies why creators of new technologies should know the basics of CE and pedagogy.

1.2.4 *Developing a full ecosystem.*

CE as a crucial topic within the wider computing world is *coming of age.* When we educated a relatively tiny proportion of engineers and scientists who self-selected into university-level programmes, the lack of status for the topic was perhaps understandable; now that computing is becoming part of mandatory school curricula worldwide, the study of how we teach the subject is of national importance. With international conferences and respected journals, and an increasing capacity for training PhD students in the field, and some research funding, we are piecing together the kind of ecosystem that other CS topic areas already have. Crucially, however, we do not have a presence in the undergraduate CS curriculum, and so CS students are not aware of CE as a topic of interest. Furthermore, the relatively few who do enter Masters and PhD programmes have to start their learning about CE, and education, from scratch. Finally, those who graduate and take up roles in academia may be seen as teaching faculty or generalists, rather than specialists with a strong research area, both to teach and to continue investigating.

Periodically, new topics appear and there are challenging times as they are recognised as valid subjects of study within the wider discipline. HCI, for instance, has travelled this rocky road and is now largely accepted, although there are purists who would argue against it. We anticipate similar challenges for CE, and so will need to develop strong arguments for its inclusion.

Indeed, advocacy for CE generally is sorely needed across society. CE researchers have often been asked why specific pedagogical research is needed for computing. Such questions are not asked of mathematics, by comparison. Of course, our discipline is so little understood generally and its education has not been experienced by the vast majority of the population, so it is a valid question for them. This further underlines the case for bringing forward more potential advocates for CE.

1.3 Objectives Implied by the Research Questions

At the beginning of this introduction, we stated three research questions which, collectively, aim at a better understanding of three key issues: (1) Arguments for undergraduate CE education, (2) Curriculum content, and (3) Underpinning on known practices. In light of the contextualization in previous endeavours (sect. 1.1) as well as reflecting the motives outlined in sect. 1.2, we are now in a position to elaborate on the working groups’ approach to address these research questions.

Arguments for undergraduate CE courses (RQ 1). We know that the undergraduate curriculum is crowded and that departmental battles rage about what to include. We intend to work both within and beyond the working group to develop and test arguments to persuade departmental committees that CE should have at least some footing within the undergraduate curriculum. A multi-national team like ours should indeed be able to take into account the highly situated nature of education: an argument fit for German academia may not work in Italian institutions, for example.

Curriculum content (RQ 2). There are many issues and alternatives to work through in considering curriculum outlines: does the curriculum principally concern best CE practices, or research frontiers, or both; what are the key topics and what material is (also) relevant in industry? In order to develop meaningful course outlines, a range of potential topic areas should be identified and coalesced into the ones that are most popular among stakeholders.

Underpinning on known practices (RQ 3). While we are not aware of CE courses in typical undergraduate curricula in many institutions, we do know of tutor/TA training courses with significant CE research underpinnings, as well as Masters and PhD level preparation, and also of teacher training programmes with strong CE input. It is then worth trying to expand as much as possible the range of sources from which to draw on existing expertise.

1.4 Outline of the Report

The remainder of this working group report is structured along our three research questions and the corresponding objectives. Section 2 presents the three different methodologies employed for addressing our research questions: Semi-structured interviews with different types of stakeholders for eliciting arguments for CE courses (RQ 1), a Delphi process with different types of stakeholders for converging on a set of relevant topics for such courses (RQ 2), and a systematic literature review for identifying practices in designing CE courses (RQ 3). Section 3 then presents the results for each of the three research questions.

To showcase our findings and illustrate how a CE course might be designed as either a stand-alone course or as a course embedded into an already existing course, we present two proof-of-concept exemplars for CE courses in sect. 4: Section 4.1 summarizes our findings regarding how a training course for teaching assistants (TAs) might look, and sect. 4.2 presents how a course on professional soft skills might be augmented to also address CE content relevant to a career in industry.

We summarize our work in sect. 5, connecting our findings back to the research questions and outlining next steps to turn our research into actionable advice for motivating, designing, and ultimately implementing CE courses in undergraduate education.

2 METHODOLOGY

The aim of the working group is to advocate the adoption of CE in undergraduate CS (CS) programmes. Such adoption requires both arguments sufficient to persuade our departmental colleagues and our education committees, and also curricular outlines to assist our colleagues in delivery. The goal of the group is to develop examples of both arguments and curricular outlines.

2.1 Operationalizing Our Research Questions

To achieve the aims of our study, we operationalized our research questions (see Section 1) as follows:

- Perform and analyze interviews to elicit how different CS stakeholders advocate for the importance of CE courses in undergraduate CS programmes (RQ 1).
- Initiate a Delphi process to understand possibly different perspectives on what the topics in CE are that stakeholders consider important and fitting for a CS undergraduate audience (RQ 2).
- Run a literature review and explore other less formal sources to gauge what practices are reported formally and informally related to undergraduate CE courses (RQ 3).

In consequence, our study considers three tracks that run simultaneously. The aims of each of these lines of investigation are described below along with the corresponding methods employed.

2.2 Arguments for CS Education Courses – Semi-structured Interviews

The aim of this track was to develop arguments that would be effective in persuading university CS departments to adopt CS education courses or material into their undergraduate curricula, and four of the working group members were involved. The broad approach was to draft some initial arguments and use these as the basis for a discussion with a number of interviewees drawn from academia, external education-related organisations and industry. Academics were necessary for their knowledge of academic processes and policy; the other groups provided a validation of the arguments we had drafted that concerned their (not our) spheres of expertise. The stages of our process were as follows.

2.2.1 Draft arguments. We constructed a set of arguments based on our motivations for this work as outlined in the Introduction (Section 1.2). As a group of practising academics, we had strong confidence in the arguments that relate to *academia*, such as TAs/tutors, new academics, PhD students considering academia, CS learners generally, and the larger CS education academic ecosystem. Our confidence derived from our own experience/contexts and so the appropriateness of these arguments in other kinds of academic contexts needed to be checked. We also speculated that our proposed course material would be valuable for *non-university teaching contexts*, such as schools, and industry. We recognised that these contexts were beyond our specific expertise and that the associated arguments would need a level of validation. The arguments were summarised on a single sheet of paper for participant interview preparation and are listed below. The draft arguments used in the interviews were organised into seven primary categories, distinguishing between targeted learners, and ordered such that those we thought to be most persuasive to a general academic audience or leadership team came first. Each category included additional detail, in the format shown below. This argument list provided a basis for the discussion, but interviewees were free to bring up topics outside the list.

(1) TAs/TUTORS: Improve our own TA/tutor body through in-depth education and training.
    - Influences our overall education provision: retention, reputation, employability, etc.
    - Includes informal interactions between students who enjoy helping one another.
    - Course participants can create educational resources for use locally or beyond.
(2) SOFTWARE ENGINEERS: Better prepare prospective software engineers, who will be involved in a range of formal / informal learning and teaching activities.
    - Software engineering is constant learning: new systems, APIs, problem domains.
    - Creating instructional videos and documentation for web consumption.
    - Stack Overflow and help desk responding.
    - Coaching / mentoring as team lead/member; currently picked up more/less on the fly.
    - Science communication skills generally - sharing technical issues to diverse audiences.
(3) ALL LEARNERS: Engage with topics relevant to the practice of learning CS.
    - Encourages independent and self-regulated learners
    - Enhances their enjoyment of CS study, and that of their teachers.
    - Prepares students to describe what is at the heart of our subject, to improve how we are viewed societally – good teachers must be aware of this, and much is lost by our specialised, fragmented courses.
    - Awareness of rigorous qualitative research develops the valuable ability to consider and analyse other perspectives with care.
(4) ACADEMICS: Provide subject-specific education skills for incoming or current academics.
    - A driver for improved student experience and retention – influencing enjoyment, depth of learning, assessment and feedback practices.
    - Add to PgCert courses in academic practice for new staff; these are often very general.
    - Bring latest CE findings from conferences or literature into the school/department.
    - Could be of interest to academics outside CS who are tasked to teach digital skills, Computational Thinking, programming.

(5) PhD STUDENTS: Prepare incoming PhD students for teaching activities in PhD studies.
   - Students get some research experience at the undergraduate (UG) level and then, going on to PhD, they're expected to be both researchers and often teachers as well – as tutors/TAs. Better then if they came in with both research and teaching background.
   - Such students may be interested in contributing to CE research activities.

(6) FUTURE EDUCATORS: Address chronic shortfall in folk knowledgeable in CS and education
   - Encourage folk early on to realise that this is a viable career path, as otherwise industry can easily be seen as the only option.
   - Teachers and education leaders for the school system.
   - Currently little deep CS understanding absorbed in schools. Hence prospective great teachers need significant university-level CE, giving a broad subject perspective – but must be alerted to teaching as an option.
   - Effective for post-first-degree teacher education systems. [Note however, not effective for 4/5 year combined bachelors and masters education programmes.]
   - CS-oriented educational technology specialists.
   - Researchers and academics.

(7) ECOSYSTEM: Place CE as a mainstream topic, like systems, AI, HCI, etc.
   - In an ecosystem, there is funding, PhD training, academic positions - and UG specialist courses so that students can find out about the topic.
   - Research students for UG, Masters and PhD projects in CE will have topic-specific prior knowledge, currently lacking for such students.
   - Researching CE academics will have specialist courses rather than being consigned to teaching only generalist courses, or courses quite outside their field.
   - CE outputs can be REF-able (a UK-based argument!) – but more broadly, it is crucial to emphasise that CE research is "real" research.

2.2.2 *Participants.* Our participants represent a convenience sample of those within our networks fitting the requirements of our argument evaluation and hence drawn from academia, industry and the pre-tertiary education world. They were recruited via personal email invitation. The roles and backgrounds of the participants who accepted are given below in the Table 5.

2.2.3 *Interview Protocol.* We devised an interview protocol as follows. Interviews involved one author and one participant, except for one case where there were two participants and one case where there were two authors. Prior to an interview, we sent the participant an information sheet, a consent form, and the summarised arguments. The information sheet outlined our principles and approach to data collection including informed consent, safe storage of data and anonymity. The interview followed a semi-structured format: the participant was first asked to give consent to the study, and then asked about their career background justifying their inclusion in the process; the interviewer then took the participant through each argument on the sheet, drawing out their views on it. The order in which the arguments were considered varied according to the participant's background. For example, the arguments around industry and software engineers were considered first by the industry participants. All interviews were carried out using Zoom with recording and transcript generation enabled. The whole approach was submitted to the University of Glasgow College of Science and Engineering Ethics Panel for approval, which was duly obtained.

2.2.4 *Transcript analysis.* Initially, the transcripts were tidied to remove extraneous speech parts, combining sequential statements by the same speaker and correcting errors in the automatic transcription with reference to the audio recording. Following the typical steps of an inductive thematic analysis [21], the four members of the interviewing team carried out an open coding process on the interview transcripts from the interviews they had conducted, identifying codes and selecting related quotes. Each transcript was read by at least two members of the team. In discussion, the team reviewed each other's codes and merged codes as appropriate. We then undertook axial coding in order to come up with themes and corresponding quotes which we collated in groups broadly corresponding to the argument categories. These are presented in the Results section (3.1). Following the guidelines presented by McDonald et al. [73], we did not carry out a statistical test of rater reliability, since we resolved differences and reached agreement in our coding through direct conversation, given all transcripts were read by more than one researcher.

2.3 Content for CS Education Courses – A Delphi Study

For this part of the study, we employed the Delphi method; the aim was to investigate informed CS stakeholders' perspectives on Computing Education topics that are important to be integrated into the curricula of CS departments.

2.3.1 *The Delphi Study and Consensus.* Delphi studies are useful when consensus is needed on a topic and particularly, on topics with limited existing evidence. Considering that research in the current area we are investigating is limited, whereas at the same time expertise is informally distributed, exploring the collective opinions of informed stakeholders while they negotiate their perspectives, was deemed suitable for the aims of this study.

In a Delphi method, the researchers form a panel of experts that fulfil some criteria, and whose identity is kept hidden. The aim is for the group to participate in a number of rounds answering a specific question or set of questions. After each round of the Delphi, the participants are provided with feedback regarding both the opinions articulated in the previous round and the motivations reported in support of these. The process is repeated until consensus is achieved.

In this study, three of the working group members were involved, and followed the so-called "Policy Delphi" rather than the more conventional one. In the conventional Delphi, the researchers design an original questionnaire and collect the participants' responses; after that, the researchers design a second questionnaire based on the results and give the opportunity to the participants to re-evaluate their original responses based on the group's responses in order to reach a consensus. In the Policy Delphi, the researchers are not interested in having a group generating a decision or consensus [18] but the aim is to provide a factual basis for or against a problem [119]. Consensus may occur but the main focus is for the participants to critique, comment, and re-evaluate their ratings in the light of their fellow participants' arguments. Another important aspect of the Policy Delphi is the participants. Informed people representing multiple different perspectives are suitable as the Delphi's panel so that multiple issues and opinions about an issue can be identified [70]. Once the Policy Delphi is completed, a small committee can use the results to formulate a "policy".

2.3.2 *Participants.*

Since the aim of our study is to suggest CE curriculum content for undergraduate students with a variety of future/current career roles (e.g., tutors, teachers/faculty, research students, industry), we wanted to include the perspectives of experienced individuals representing each of these categories. Again, we emphasize that this study intended to use teacher training, for which content discussions have been conducted in the past and still are being conducted, as a post-hoc baseline; thus we did not include experts in teacher training in our Delphi process.

In the first round of the Delphi, 47 participants took part. Table 1 depicts the percentage of the participants per country/continent in which they were employed at the time of the study and table 2 depicts the percentages for each of the roles included in the Delphi panel. In the following paragraphs, we highlight further information on the participants belonging to each of these groups.

Table 1: Distribution of participants in the first round of the Delphi process by country/continent

Country/Continent	Number of participants	Percentage
UK	23	49%
Europe	16	34%
US	8	17%
Total	47	100%

Table 2: Distribution of participants in the first round of the Delphi process by role

Group	Number of participants	Percentage
Tutors/TAs	8	17%
CE Academics	17	36%
CS Academics	6	13%
PhD students in CE	11	23%
Industry practitioners	5	11%
Total	47	100%

Tutors. In total, 17% (8 in total) of the participants were tutors or TAs in CS departments. Most of them have been employed in this role for more than 4 years. Further to this role, 4 of them had also experience working in the industry, 3 of them working as CS teachers, and 2 of them working as research assistants.

In relation to their training, almost all of them (7 out of 8) had some training prior to or during their role as tutors/TAs. The training included teaching methods, pedagogy, assessment, learning theories, and practices for engaging students.

CS Academics. A 13% of our participants were CS academics with no specialisation in CE. Most of them (5 out of 6) had also been trained specifically to teach in academic settings.

CE Academics. The largest group in our study (36%) were academics engaged with CE research. About half of them (52%) had been teaching in academia for more than 10 years and 41% of them had already been teaching a CE course of some kind. Most of them (82%) had also been trained specifically to teach in academic settings.

PhD students. In total, 23% of the participants were PhD students in the CE field. Most of the PhD students had previously been employed in other positions such as in industry, as teaching assistants, research associates, teachers of CS, and teacher training.

Industry. The smallest group in our study (5 participants, corresponding to 11%) were industry practitioners. 60% of them had more than 10 years of experience in the industry and some of them (40%) had previously been involved with academic settings (doing a PhD and postdoc). Apart from their technical skills, 80% of them had received special training such as Leadership and management, Project management, Organisation structure, influence, communication and collaboration, and Unconscious bias.

In the second round of the Delphi study, 37 participants from the 47 completed the survey: 2 were CS academics, 14 were academics in CE, 4 were practitioners in industry, 10 were PhD students in CE, and 7 were tutors/TAs.

2.3.3 *Data Collection and Analysis.*

The Delphi study was carried out over a period of six and a half weeks during the spring/summer 2023. The participants were invited via email and anonymity was kept during all rounds. The invitation explained the nature and aim of the study and included the link to the first survey. The participants were given about three weeks to answer the first survey, two weeks for the second, and about a week for the last.

In the first round, the participants were asked to fill out a survey with two sections. The first section included questions about the participants' academic and professional background while the second section included an open-ended question inviting the participants to reflect and suggest CE topics that can be introduced to undergraduate CS departments. In total, 47 participants answered the first round survey (the characteristics and background are described in the "participants" subsection above).

After collecting all of the responses, a qualitative analysis was performed by one researcher. The aim was to create a list of high-level topics along with more concrete sub-topics based on the participants' responses. To this end, the participants' responses were grouped into themes representing a higher category of (CE) topics. A second researcher then read all the responses and verified that all participants' suggestions were accurately represented in one or more of the high-level categories. Two more researchers verified that the sub-topics were accurately listed in the high-level categories. Having completed this analysis, the second survey was designed.

In the second round of the Delphi, the participants were asked to indicate their level of agreement (how important is the specific topic to be taught in CE courses) with each of the suggested high-level topics and for each of the undergraduate sub-groups (tutors and teaching assistants, potential PhD students in CE, faculty and potential CS teachers, undergraduates aiming for jobs in industry, and general undergraduates). The responses were given on a 5-level Likert scale. For each question, there was also an open-ended field in which the participants could leave comments or arguments in support of their ratings. In total 37 participants answered the questionnaire. In this round, the data were quantitatively analysed and the percentage of agreement and disagreement was measured for each question and for each sub-group we were interested in. Agreement was achieved when at least 60% of the participants (>=60%) agreed or strongly agreed with a specific statement.

Finally, in the third round, the participants were given the results of the previous round along with the participants' comments and arguments and were asked again to indicate a new level of agreement only if they wanted to change their previous rating and to comment further on their decisions. In total, 5 participants changed their ratings. The data were again quantitatively analysed as before.

2.4 Literature Review and Related Computing Education Courses

The aim of this track was to collate existing knowledge on CE courses by drawing formally on the CE literature and informally on online resources and on members of the CE community known to run various styles of CE courses.

For the first part, a systematic literature review was conducted following the steps from the PRISMA 2020 checklist [85]. The review was conducted between May and July 2023 and, in total, six members of the working group were involved in this process.

2.4.1 Search approach for the Systematic Literature Review (SLR). During the first meeting, the sub-group discussed and agreed on the aims and scope of the literature review, the inclusion and exclusion criteria of the retrieved papers, the search databases and the search string. For the purposes of this working group and given our goal of retrieving CE courses' contents, the scoping parameters we considered were the following: We considered peer-reviewed papers that are specific to CS education/training, and that describe and provide details of a course or training or workshop for the population we were interested in, that is CS undergraduates, tutors/TAs, and faculty. As mentioned in the introduction, we also retrieved papers on pre-service teacher training to be able to establish a base line. Being concerned that our study could expand in scope, we did not consider the large body of papers referring to in-service training or continued professional development of teachers. Similarly, while we did consider papers that convey knowledge about the learning and teaching of CS, we did not consider papers that describe an intervention to improve students' learning outcomes even if these papers explicitly described applying theories or strategies that might be part of a CE course.

With respect to the time frame from which papers should be retrieved and analyzed, we note that Sahami et al. published a paper in SIGCSE 2010 [100] in which they set out to initiate revisions to CS curricula motivated by the goals of

> - providing students greater awareness of the breadth of options in CS and opportunities to pursue these areas in depth,
> - incorporating relatively new, but already mature, sub-fields of CS on par with more traditional topics within the curriculum,
> - highlighting and promoting multi-disciplinary connections,
> - establishing a structure with sufficient flexibility to allow for lightweight revision in response to the evolution of the field. [100, p. 47]

This motivation and gap analysis being exactly aligned with our working groups' intentions, we concluded that papers published before 2010 were rather unlikely to provide meaningful answers to our research question; we acknowledge that this time frame does not apply to teacher training, but the de-facto standard textbook [52] in this area, summarizing many theoretical and practical results relevant to teacher training, was published in that time frame and thus contributed to the sought baseline. In consequence, we defined the search parameters of our queries to exclude papers published prior to January 1, 2010.

2.4.2 Information Sources. As mentioned in the introduction, Simon's doctoral dissertation concluded that CE Research had manifested itself as an independent academic field. We thus decided to start our literature review by examining the venues that Simon had used in his line of reasoning:

- *ACM International Conference on Computing Education Research* (ICER)
- *ACM International Conference on Innovation and Technology in CS Education* (ITiCSE)
- *ACM Technical Symposium on Computer Science Education* (SIGCSE TS)
- *Australasian Computing Education Conference* (ACE)
- *Koli Calling*

We added one more conference that had started after Simon had concluded his research:

- *United Kingdom and Ireland Computing Education Research Conference* (UKICER)

Given that our research questions also touched upon practical aspects of teaching about Computing Education, we also included five venues that are open to publishing papers on such aspects:

- *ACM Inroads*
- *Conference on Computing Education Practices* (CEP)
- *Journal of Computer Science in Colleges*
- *Western Canadian Computing Education Conference* (WCCE)
- *Workshop in Primary and Secondary Computing Education* (WiPSCE)

All of the above venues were accessed through the ACM Digital Library which is considered to be a high-quality "principal source" for systematic literature reviews [49]. In addition to *ACM Inroads* and the *Journal of Computer Science in Colleges*, we included the following two research-oriented journals:

- *ACM Transactions on Computing Education* (TOCE); through the ACM Digital Library
- *Computer Science Education* (CSE); through Taylor & Francis

Acknowledging the fact that some facets of Computing Education Research overlap with Engineering Education Research, we used IEEE Xplore, a "supplementary source" for systematic literature reviews [49], to search the following three venues:

- *ASEE/IEEE Frontiers in Education Conference* (FIE)
- *IEEE Global Engineering Education Conference* (EDUCON)
- *IEEE Transactions on Education* (TOE)

Finally, complementing the papers from WiPSCE, we considered the *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (ISSEP), a conference with a focus on best practice studies, country reports, and contests. Given that the ISSEP proceedings are published by Springer and given that their system, a "supplementary source" [49], does not fully support the nested query we used for the ACM Digital Library (see Table 3), two researchers independently inspected all conference proceedings and checked the eligibility criteria outlined in Section 2.4.3; this resulted in a total of two ISSEP papers included in the first round of analysis.

As a group we decided to not include journals whose primary focus is on the integration of technology and education (such as Computers & Education), on studying the use of computers from a psychological perspective (such as Computers in Human Behavior), or on general aspects of teaching and learning (such as Journal of the Learning Sciences). Similarly, we decided against including journals dedicated to teacher training in general (such as Teaching and Teacher Education).

2.4.3 *Eligibility Criteria.*

The inclusion criteria were developed by the six members involved in the SLR and are listed below:

- The paper should describe a coherent sequence of instructional units, such as a course or a workshop, addressing multiple aspects of CE.
- The paper should target the population we are interested in; that is CS undergraduates, tutors, TAs, mentors, Faculty, and pre-service CS teachers.
- The paper should have been published between 1st January 2010 and 10th May 2023.
- The course or module described should be about learning and teaching Computing or about learning and teaching within Computing contexts.

Correspondingly, a paper was excluded if:

- The paper addressed an intervention focused on an isolated topic in CE.
- The paper did not provide details of the CE course's contents.
- The paper did not include the population considered in this study.
- The paper was published before the 1st of January 2010 and after the 10th of May 2023.
- The paper described the content but the focus was not on Computing contexts or was not about learning and teaching Computing.

2.4.4 *Search strategy.*

To identify the search terms, we followed a simplified version of PICO [11, 14] by focusing on the following aspects: Population and Intervention-Outcome, and collaboratively generated the search terms. For instance, for the term "course", we consider relevant terms such as "module" and "workshop". The full search string is provided below. The search queries were conducted by one of the authors in the second week of May 2023. The number of articles initially retrieved from each database is listed in the following Table 4 as well as the included papers.

Table 3: Search Terms

Abstract: ("computer science" OR "computing" OR "informatics" OR "programming") AND ("tutors" OR "teaching assistants" OR "tas" OR "faculty" OR "lecturers" OR "professors" OR "prospective teachers" OR "pre-service teachers" OR "preservice teachers" OR "PhD" OR "majors" OR "major" OR "mentor" OR "mentors" OR "undergraduate" OR "undergraduates") AND ("training" OR "course" OR "workshop" OR "seminar" OR "curriculum" OR "curricula" OR "professional development" OR "module" OR "teacher education" OR "pedagogy" OR "pedagogical")

2.4.5 *Selection Process – Conducting the review.*

The literature review was conducted in May-July 2023. Using the search criteria above we reviewed 2143 papers in total. To make sure that everyone was applying the criteria appropriately and to identify any potential issues, all six members reviewed the abstract of 501 papers and applied the inclusion and exclusion criteria. The members met again after the review and further discussed and refined the criteria accordingly.

The review process included 2 rounds. During the first one, each member was assigned a set of papers to assess by reading the title and abstract. Each paper was categorised into the exclusion category or proceeded to the second round for further screening. From the 2143 papers retrieved, only 90 papers proceeded to the second round. A second reviewer randomly reviewed 100 papers (20 papers from each of the other five reviewers) that were excluded for reliability purposes.

During the second round, the 90 papers were further divided among the members and each member read the allocated papers fully and further decided whether the paper is excluded or included. If the reviewer was uncertain, they indicated their evaluation as "maybe" and then a second reviewer was assigned to this paper to further help with the process. Independent of whether the paper was included or excluded, at this stage the reviewers also applied the snowballing method to collect cited papers that may be relevant to the aims of this working group. From the snowball method, we selected 15 additional papers to examine.

The overall review process is depicted in Figure 1. As a final outcome, we retained in total 25 papers, as shown in Table 4. The references to all such papers are reported in Table 8, grouped by targeted population (see section 3.3).

2.4.6 *Data Collection Process and Synthesis.*

For all the included papers, we collected and reported in a spreadsheet the following information:

Table 4: Distribution by publisher of the references processed at subsequent stages of the literature review (see also figure 1)

Databases	Retrieved	Included 1st Round	Included 2nd Round	Snowballing Retrieval	Snowballing Included	Total Included
ACM	1472	62	15	3	0	15
IEEE	430	24	4	1	0	4
Taylor & Francis	24	2	2	0	0	2
Springer	217	2	1	1	1	2
Other databases				10	2	2
Total	2143	90	22	15	3	25

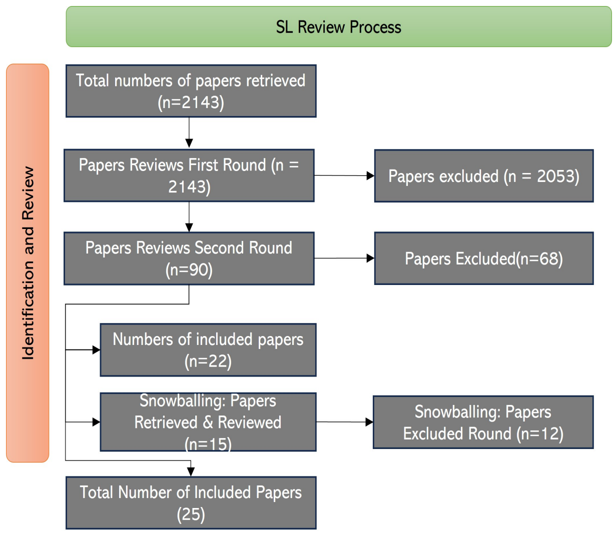
SL Review Process
Identification and Review
Total numbers of papers retrieved (n=2143)
Papers Reviews First Round (n = 2143)
Papers excluded (n = 2053)
Papers Reviews Second Round (n=90)
Papers Excluded(n=68)
Numbers of included papers (n=22)
Snowballing: Papers Retrieved & Reviewed (n=15)
Snowballing: Papers Excluded Round (n=12)
Total Number of Included Papers (25)

Figure 1: Review Process

- the aims/objectives of the course described
- the course's contents
- the target population
- the assessment method if reported as well as the teaching methods

After we collected all the information, two of the members involved in the literature review process were assigned one or more of the papers' targeted populations in an attempt to condense all the aims/objectives of their training as well as the contents of the corresponding course into high-level categories. The results, in terms of the identified high-level categories, are summarised in section 3.3.

2.4.7 Collection of Informal Resources. For the collection of informal resources, we proceeded as follows: each of the working group members (all 11) suggested online resources from courses we were familiar with or from our colleagues who run courses relevant to this working group's interest. In total, we collected information for 21 courses relevant to Computing Education.

2.4.8 Analysis of Informal Resources. Two of the working group members collected and added in a spreadsheet the same information we collected from the reviewed papers, namely:

- the aims/objectives of the course
- the course's contents
- the target population
- the teaching and assessment methods if reported

Similar to the analysis performed on the reviewed papers, once all the information was collected, the two members synthesised the high-level aims/objectives and contents from one or more of the targeted populations of these courses. The references to these less formally collected resources (documented online) are reported in Table 9, grouped by targeted population (see section 3.3).

3 RESULTS

In this section, we report on the findings of each of the three tracks of investigation outlined so far. More specifically, in section 3.1 we present the arguments contributed by a variety of stakeholders in support of a CE course. Then, section 3.2 is about the results of the Delphi study. Finally, in section 3.3 we summarise the insights drawn from the literature review as well as from the other less formal resources that we considered.

3.1 Interview Results

We interviewed 14 stakeholders. The details of the participants are summarised in Table 5.

From the data collected, our inductive analysis identified 44 codes, which we coalesced into the 18 clear themes which are now presented below. Some closely linked to the roles we specified in our initial argument prompts to the interviewees, others broader or overarching (addressed to a wider audience), still others relating to the current perception of CE in terms of policy and CE research. The emergent themes are structured according to the original argument categories from our draft argument summary. The codes for interviewees defined in Table 5 are italicised for clarity in the running text.

3.1.1 Industry and Software Engineers.

There were four main themes of discussion from our participants in terms of the arguments for CE courses for industry and software engineers (SEs): a. SEs involved in formal/informal teaching activities; b. communication skills; c. the need to create instructional materials of various kinds; and d. coaching and mentoring.

SEs involved in formal/informal teaching activities. The first was in terms of industrial involvement in schools. Our participants recognised that employers want to work with schools and universities. Sometimes their motivation for doing so is to encourage learners to join the profession and to show what their classroom learning can lead to in terms of their future career prospects. They

Table 5: Participants for Arguments Interviews

ID	Position	Background and Rationale
CS-1	Head of CS Department	Institution with major teaching focus, with a strong CE group in the past.
CS-2	Professor in CE	Balanced institutional research and teaching focus. Long enough in post to understand departmental politics.
CS-3	Professor in CE	R1 institution. Been programme lead and introduced numerous new courses, so has good understanding of departmental politics. Leads a CE research group.
CS-4	Professor in CS	R1 university. Heavily involved in nation CS education decision-making, with industry perspectives.
UL	Professor of Practice for Inclusive Education	Business background now at university leadership level. Ex-secondary head teacher, involved at national level in education research.
UTE	Senior Lecturer (Associate Professor) of secondary teacher education	CS background, teacher for over 10 years, recently in post, with perspective on relevance to incoming teacher education students
NTCS	Teacher leader of national CS teacher association	Long-time CS school teacher (35+ years), now involved in policy of school CS education at local and national levels.
NTL-1	Top-tier staff member in national education improvement and inspection agency	Heavily involved in schools inspections, particularly in respect of CS, with a PhD in CE.
NTL-2	Head of Digital Education, national government level	Involved in development of national education policy involving all forms of digital education including CS; previously a CS academic.
I-1	Head of Competence Development, large software company	Teacher and CE researcher for 18 years before going into industry where role requires deep understanding of educational processes
I-2	Software engineer in global financial services company	30 year veteran of the software industry with a visiting professor role at local university involving both teaching and research
SS-1	Director of Education and Public Benefit, national scientific society	Involved in CS education policy at multiple levels. Executive lead on national CS teacher professional development programme prior to current role.
SS-2	Head of Education, national scientific society	Involvement in national curriculum development in digital and CS context over a 30-year period.
CER	CE researcher with primary school focus	Involved with national teacher professional development programme and particularly in relation to primary school CS education.

may also visit schools/universities with the motivation of helping learners to understand a new or complex technology and the challenges that arise from its development and use. Whatever the reason, there are real benefits in bringing people from the industry in to help teach, as often (as in the UK) there is a shortage of experienced CS teachers at the school level. However, problems can arise when people from the industry come in and try to teach – they may not know how to break down concepts enough for their audience to understand or may not be aware of the existing knowledge level and experience of learners as a starting point for their discussions. This is where learning about CE, in industry, might be of benefit. As one participant commented:

> *"Teachers are incredibly good at communication, simplifying, translating the complex to the understandable, structuring things in a way that takes people through a series of discoveries or (...) knowing when not to tell the answer and ask questions to let that discovery happen."* (*SS-1*)

Another participant observed the following about teaching in an industry setting and the value of knowing about educational underpinnings:

> *"it's not so much a question of person A teaching person B. It's question of person A and B discovering and generating new knowledge together as opposed to the old-fashioned transfer approach or model of learning. It's much more exploratory and generative in that sense. And I think, having a grounding in some of the teaching models and the learning models and also a grounding on how learning works, and what the challenges and limitations and constraints are, I think, will be very useful for software engineers."* (*I-2*)

Communication skills. Our participants also recognised that there is a real need for people from schools and industry to be able to speak to each other and *"formulate a genuine understanding of what the other person is experiencing and how they see things, to properly collaborate and understand each other's world."* (*NTCS*)

Our industrial participants also recognised that most software developers need science communication skills e.g., when peer programming, presenting in sprints, or when presenting to clients or the general public. They could see the advantage of having the interaction skills as well as the technical know-how, *"both of these being in the head of the same person"* (*I-1*):

> *"Our marketing department wants to make our experts visible and for our experts to tell why a specific technology is of use to your company. (...) we need engineers capable of telling how solutions can boost finances (...) No-one is going to buy anything from us if we fail to communicate what kind of added value we can create for them, not only how the code works, but what is its added value. . . "* (*I-1*)

They could see the benefit of having good science communication skills in general.

Need to create instructional materials of various kinds. The third strand of discussion focused on the fact that there is a need for courses, educators and learning materials outside of schools and academia due to the changing nature of the computing profession. As one participant noted:

> *"I think we've all seen <company name> and their tutorial on how to use one of their tools. And it's like laughably bad. And so I think it's a real value."* (*CS-3*)

Similarly, another participant focused on documentation and connected it to teaching:

> *"And so when, if you're just trying to explain something, or you're writing documentation essentially what you're doing it teaching. Writing documentation is teaching the people who come after you exactly what you've done."* (*CS-2*)

There was also a recognition that the nature of training has changed in large organisations and that most software engineers no longer attend formal training sessions. It is more optimal for industry that software engineers train themselves or coach their colleagues, on the job, in a 'just in time' fashion.

Coaching/Mentoring. However, when it came to specifically focusing on CE and teaching skills one participant described how coaching junior developers might easily end up in people emulating what they perceived as "teaching". One participant commented:

> *"We don't usually have time for really preparing teaching, (...) and can't play university teacher in addition to their work. We try to make things work in a pedagogically effective way and the means of transferring knowledge to colleagues is definitely not in the form of (self-made) lectures. Creating a slide set is out of the question"* (*I-1*)

Their organization preferred using outside courses and materials in training their professionals and supporting internal coaching. We note that the external training organizations might also be possible employers for students with Computing education experience.

As part of coaching and mentoring, *I-2* emphasised the role of feedback:

> *"You not only need to learn how to be a coach, and in some sense you need to learn how to be coached as well. This idea of giving and receiving feedback are two different skills, but you need to learn them both. It's not something that's intrinsic or intuitive, giving or receiving feedback. It's something that you need both practice and training on how to do well."* (*I-2*)

There may be some myth-busting to be done about what CE involves if we are to make a strong argument for the need for CE skills in industry, at least, based upon this particular case, it is something to be aware of. There may be misconceptions about the benefits of learning about CE or indeed how we teach CS currently in schools and universities. Mostly these days, whilst our students learn theory, we also focus on practical learning and exercises, not just lecturing. We encourage our students to learn by doing, and in this respect learning is the same in industry and university.

3.1.2 University Teachers (TAs, Tutors, Academics).

We grouped university teaching roles together for the purpose of analysis as most of our interviewees focused on each of these interchangeably in their discussions rather than in the order that we presented them in the arguments document. This seemed a natural grouping as all these roles are interrelated and take place in the same location and context at a university.

Two different threads of thought came out of the interviews. On the one hand, the participants focused on why we should run courses on CE for the considered roles, and on the other hand, they expressed concern for the barriers we might face in trying to do this.

Enhancing the teaching quality and retention. Regarding the former — why we should run courses on CE — participants' arguments focused on the following theme: Enhanced teaching quality and retention via building on a. Assessment and Pedagogy, b. Subject-specific education skills, and c. Reflective practice and continuous professional development.

Overall, participants felt that universities need to formalise the training of unqualified teachers to help maintain their quality and standards: *"We need to improve student experience and retention"* (*CER*). Becoming a TA or tutor in CS is traditionally an informal arrangement with a temporary contract and generally to obtain such a role a teaching interview is not conducted. We tend to rely on technical ability of students/graduates taking these roles rather than ask for any prior teaching experience.

Assessment and Pedagogy. Most of these roles (TAs, Tutors, PhD students) are also required to assess students and have a high degree of responsibility in our laboratory teaching. It was felt therefore that people in these roles need to understand the basics of education and also the context of CE with respect to these — some theory and practises applicable, useful, not applicable etc.

> *"It is essential they understand underpinning concepts of education...and culturally relevant pedagogy so they are specifically aware of unconscious bias in our context."* (*CER*)

Assessment and feedback is something against which university CS is often bench-marked and ranked alongside peer institutions in national league tables. Participants felt there was a need to ensure that TAs, tutors and PhD students (unqualified or inexperienced teachers in the main) were trained in how to assess in an unbiased and fair way. CS1 has a high drop-out rate in many countries and some of our respondents felt that training in pedagogic methods pertaining to good CE for teaching staff would help to enthuse and encourage students and most likely help university CS departments to improve their students' learning experience and therefore retention.

Some interviewees felt that there is also a need for people in these roles to have some training on how to spot plagiarism and what constitutes plagiarism in CE, particularly in programming. Our participants involved directly in CE at universities felt this was particularly important in light of recent developments in the usage of Generative AI applications for writing or debugging code.

Subject-specific education skills. When discussing CS academics and whether they would benefit from courses in CE, our respondents highlighted that they felt that CS academics need subject-specific education skills so that they understand the misconceptions that students face, which concepts students find difficult and the existing knowledge level of their audience.

Reflective practice and continuous professional development. They also highlighted that academics need to be reflective practitioners. One respondent noted that most academics do an initial teaching course (that is generic, not CS-focused) at the beginning of their career to pass probation and then teach forever after, without having to renew or refresh their pedagogic skills. This respondent (*CER*) felt that CE courses should not be something that is a tick-box exercise and that academics need to be continuously learning and developing their teaching skills, as well as their research skills. In alignment with this, another participant noted:

> *"Ongoing professional development. I think that it's important to refresh those skills and to actually renew them....a proper centre for learning and teaching, where academics are kind of encouraged and become eventually enthused by engaging and participating in things. This would be an amazing yeah."* (*UL*)

<u>Barriers</u>. Reflecting on the barriers, the participants' arguments centre around a. lack of supporting mechanisms, b. low levels of engagement c. academic shortage and packed undergraduate programme.

Lack of supporting mechanisms. Some universities use undergraduate students as TAs for technical laboratory classes in CS, and this is becoming increasingly more common. Other institutions do not have such mechanisms in place for student support as yet, and therefore, one participant felt that using arguments about assessment or plagiarism concerns for training of undergraduate TAs in CE would not be particularly persuasive in their institution.

Low levels of engagement. Several of our academic respondents stated that they already run training courses for TAs in CE in their institution, and one noted that reading education literature was the least popular part of the training class for their TAs.

> *"[our tutor training] class has some education literature in there, at the start. That is the least popular part of that class. So it's the part that students are first to complain about because they just want the practical part. 'Give me the nuts and bolts of how to make it happen, because I have to grade this class this week.' Which is unfortunate. ... The students don't want to work and so anything that we require, say read a research paper, is not appreciated - because this is just TA training. It's not seen as a real course."* (*CS-2*)

This respondent stated that for the most part TAs just wanted to get to the practical part of the training because they were concerned that they had to grade work soon. This respondent suggested that their course was not taken seriously by TAs because it was only a short course or workshop style event rather than a formal substantive CE course. They posited that a more substantial course would be more successful, especially if it was run as an elective for all students. Other respondents noted that they had run successful CE research classes where graduates read research papers on CE and their students had vibrant discussions and were fully engaged. So, it may be a matter when considering running such courses as to the level of maturity of the audience. Some respondents stressed that they agreed PhD students need to be trained how to teach, and they felt a dedicated course on CE would be very beneficial for them. However, one respondent noted that often CS departments did not have the capacity to teach these courses in more depth as not many CS academics can actually teach CE classes.

Academic shortage and packed undergraduate programmes. Many of those who could teach CE classes in their institution were on *"the teaching stream and already had huge teaching demands elsewhere which makes things tricky for the teaching load"* (*CS-3*). Others highlighted that there would be difficulty fitting such a course into their current undergraduate programme. They stated that their specialist programme content is almost entirely fixed (compulsory) and even in pathway programmes (that offer some choice) it might be difficult to get such a course accepted as *"when you start having discussion with the department as a whole – everyone wants a thing in there"* (*CS-2*), so there would be lots of competing arguments for content and courses other than CE. In terms of putting a CE course in the earlier years of a CS undergraduate programme, this respondent also stated, *"we don't have optional modules at levels 1 and 2 of our programme"* (*CS-2*) and this might be the case for many undergraduate programme structures. Another respondent outlined that their university was a meritocracy in terms of determining which courses to run. They felt that if someone was willing to put in the effort to create such a course, their department would not stand in their way, however, there was a caveat, *"as long as it does not dramatically impact other required courses you would be allowed to teach that class"* (*CS-3*). So, it seems it is still not very straightforward to get a CE course onto current CS Undergraduate programmes even where there is support and academic freedom. One of our respondents felt that proposing a CE course would make sense to some colleagues, but others would ask what makes such a course CE rather than just a standard Education course. They envisaged that if we started with defining programme level outcomes then these would be required to be fulfilled by modules or more likely parts of programmes, this was one way that those interested in getting CE on UG CS programmes, could perhaps go about it (*CS-2*).

In the participants' view, courses on CE and CE Research might be more likely to be accepted within undergraduate CS programmes where there is already an acceptance of HCI. Moreover, the CE community should provide more persuasive arguments that there is a subject-specific pedagogy for CS:

> *"If you can get across the idea there was a subject-specific pedagogy, that matters, because I think that is part of the problem. I don't think the university necessarily believes in subject-specific pedagogy."* (*CS-2*)

It was felt that it would be a huge contribution if we (this working group) could identify what core elements we think everyone should be learning and if we could start to characterise a subject-specific pedagogy.

3.1.3 School Teachers and Future Educators.

Shortage of teachers with CK and/or PCK. Interview participants recognised that in some countries there is still a shortage of CS teachers in the school system and that many who do teach CS may well not have a CS degree. This amounts to what some view as a *"desperate"* shortage of deep knowledge at the school level (*UL*). Teachers who do not have a CS background, whilst being able to teach, may lack, as another respondent put it *"a subject specialist's ability to lift from the page and expand into something that is relevant and have context"* and this lack of subject specialism is a problem that they think *"is not going to go away"* (*SS-1*). They felt the proposal for Education courses in Undergraduate CS programmes was exciting because *"we might be able to learn from subject specialists who have got this additional superpower around education and to be able to communicate more effectively with educators"* (*SS-1*).

They felt there was also a potential shortage of teachers with both CS and Education background at other levels of education too, and this issue was wider than just schools:

> *"We were talking about teacher capacity and retention, well, this isn't just teachers. This is FE [Further Education, like community college]. Tutors and lecturers. This is actually HE [Higher Education]."* (*SS-1*)

They referred to an example of universities filling their vacancies in CS with hourly paid temporary teachers *"who are not fully committed to students and may not be able to provide the tutorial material students need"* (*SS-1*).

One of our interviewees involved with teacher training felt that the introduction of CE in undergraduate CS programmes would make graduates of such programmes a great resource for school teaching *"because some of these people may want to be a primary teacher which would be fantastic, and even if they don't, you will have more people in secondaries who could work with the primary partners and be a fantastic resource in the same way that they can enhance what's going on in the secondary classroom"* (*NTCS*).

Teaching as a career path. Some participants thought there would be positive knock-on effects if CS undergraduates attended these courses and then wanted to take up teaching as a career. Others felt this was a good idea, but they were clear that such a course should make sure that the students understand CE methodologies in particular, and education in general as *"they generally have no understanding or experience of teaching or why we teach the way we do"* (*UL*). This participant viewed the need for discipline-specific teaching skills to be taught more generally in undergraduate programmes across their institution as important and felt that institutions *"need to provide more instructional collaborative development of teaching skills"* and that a CE course in undergraduate CS programmes *"could be a really interesting prototype of this kind of education, slash communication with the world, advocating for the importance of your disciplinary area. As there are major topics that the general public probably do need some help with, like security"* (*UL*).

Useful skills for industry. One of our academic respondents felt that being able to get a CE course into the undergraduate CS programme might be easier if it *'ticked a box'* for course accrediting bodies (such as IET, BCS), and if it could be demonstrated that the skills taught in these particular courses were shown to be those needed by a software engineer in industry (*CS-2*). As an example of how this could be demonstrated, the need for software engineers to create effective documentation was pointed out: The ability to create succinct and clear documentation for users and other developers is, indeed, *"basically teaching through a document"* (*CS-2*).

3.1.4 Undergraduate Students.

Self-regulation and Metacognition. A starting point for the discussion around the arguments about benefits for all students relates to the benefits for their own CS learning. As *SS-2*, who has been involved in aspects of national CE programmes for over 3 decades, put it:

> *"[Computing] people don't reflect. They don't do the meta-cognition, they don't reflect on how they learned it. So you ask them. It's like trying to get a squirrel to explain centre of mass to you. They demonstrate it all the time, but they can't articulate it, which is partly communicating what this is. But there's also [the influence on] your own ability to learn if you're not conscious of how you learn."*

SS-2's reference to meta-cognition was picked up by other participants. The Education Endowment Foundation (EEF), a UK body that identifies evidence-based educational innovation in the school system, rates the development of meta-cognition in young people as the most effective intervention, particularly in relation to the low cost of adoption. *UL* cited meta-cognition as key, noting:

> *"If I'm given feedback [as a learner], I need to know what to do with that feedback... if a TA gives me feedback, what do I do? No one else can be responsible for that, apart from the learner, so metacognition is at the heart of self-assessment."*

UL noted further that *"often we teach UGs and we don't reveal why we are teaching these things, and why this way. We don't use the vocabulary of Education." NTL-1* noted the focus in their national education organisation on life-long learning, and how meta-cognition was a part of this, providing essential transferable skills. More pragmatically, given the nature of the interviews, *CS-2* suggested that the importance of meta-cognition is so great, given the EEF findings, that it could be a lever, a Trojan Horse, to encourage the adoption of wider CE studies.

Skills relevant to industry. Participants talked about the importance of teaching in connection with communication skills, particularly in relation to industry-based activity. *CS-2* noted:

> *"I constitute just about everything as if it's a teaching thing. Everything becomes better if you think of it like that. And so when, if you're just trying to explain something, or you're writing documentation, essentially what you're doing is teaching."*

I-1 spoke to the importance for their company in terms of the ability to share ideas with customers:

> *"our marketing [department] wants to make our experts visible and for our experts to say why a specific technology is of use to your company… basically we need engineers that are capable of telling to e.g. amusement park staff how our solutions can help them boost their finances. No one is going to buy anything from us if we fail to communicate what kind of added value we can create for them."*

UL noted that training courses were no longer run face-to-face, but via online resources in a teach-yourself mode, saying that *"some resources are of poor quality — often not created by people who understand CE and who are not in industry, not in the training team and not in learning and teaching."* In *CS-3*'s opinion, *"we've all seen <insert company name> and their tutorial on how to use one of their tools: And it's laughably bad."* And hence *"it's a real value [to consider] training skills for industry, and how to do this better."*

CS-1 and *CS-2* made connections to Human-Computer Interaction. For example, *CS-1* said that aspects of a CE course could be seen as a requirement to be able to understand the user. *UL* added:

> *"We develop systems for people so we need to understand how they learn, where there are misconceptions, how we can help them to overcome barriers to learning and using technology."*

Away from the industry focus, *UL* also noted the universal societal need to understand the technology around us, and how CS graduates had a crucial role to play in fostering that understanding:

> *"We've been stuck in the technical aspects of our profession — but we don't make products for the sake of it. So it makes no sense to not be educating people about that. We should take advantage of AI, an opportune moment…[to explain a society-changing technology that is poorly understood and generating concern across our populations.]"*

Employability. Employability, relevant to all students, was noted as a driver for including CE material, capturing many of the points made above. *CS-1* advised that *"the tactic you would have to use here is employability — it is a big focus here. The employers might be interested in those sorts of skills. We are heavily driven by employability. From that point of view, it almost sounds like it is something for our first and second year courses [mandatory for all students]." CS-2* noted that their organisation is keen for their graduates to go into jobs as that is an important league table indicator, and that by giving them a flavour of what education and being a teacher is like, more of those who might otherwise not get a job would consider going into teaching.

3.1.5 Ecosystem, Policy and Leadership in CE, CE Advocacy.

Having CE courses in the UG programme helps to emphasise the "coming of age" aspect of the area, although it was acknowledged by *CS-4* and *NTL-2* that this may take a number of years to have the desired impact – *"we should start now, with small steps"* (*CS-2*). Both *UTE* and *CS-4* highlighted the wider policy prominence of digital skills and CS education reform across various jurisdictions, settings and contexts (especially at K-12), alongside a renewed post-COVID focus on high-quality, relevant and authentic learning, teaching and assessment. Having said this, respondents noted that some universities as a whole do not believe in subject-specific pedagogy, especially in how they support early-career academics to develop their learning and teaching in their first permanent posts.

A wider point was emphasised by *NTL-1* (from a national schools improvement agency) and *NTL-2* (working in government), that while our arguments had currency and traction due to ongoing reforms and wider initiatives, they were missing an over-arching summary of the global importance of CE. Drawing on both of their local universities' civic missions, it seemed obvious that any CS department should be contributing to CE, given its growing status. From this point of view, then CE should clearly be seen as a mainstream topic, with appropriate research and teaching, like other topic areas such as systems, HCI, machine learning, and so on, given that CE's influence on the world is becoming similar to the influence of any other topic in CS. This overlap with other topic areas such as HCI was clearly made; for example: *"if it involves a human, then all of a sudden, it's no longer interesting. When I think that actually makes it more interesting..."* (*CS-2*); so the acceptance of HCI makes space for the acceptance of CE.

Incorporating educational aspects into professional accreditation of CS degrees was also considered in the interviews. *SS-1*, one of the staff interviewed from the science society for CS in their country, noted that *"we've definitely changed over the last few years, to recognise the importance of the educator."* In relation to professional accreditation of CS degrees, they observed that *"if you stand back and think about contributing to the discipline, your professional community, the whole education piece is directly relevant."* In discussion, it was clear that a CS school teacher could be just as likely as an engineer to have professional accreditation based on their respective professional backgrounds. Both *UTE* and *CS-2* noted this as a potential driver for adoption in institutions like theirs, with *UTE* acknowledging how it could support both initial teacher education and in-service training for CS educators. Referring to knowledge of CE, he said *"I think you'd have to get that into the [science society's] accreditation, and then, yes, it would immediately [be considered seriously]."* The inclusion of CE within professional accreditation regimes was highlighted by both *CS-1* and *CS-2*, a department head and a long-standing full professor respectively, as another persuasive element for adoption. Although not all CS departments aim for accreditation, both of these academics' departments recognised accreditation of their programmes as important to their students' prospects. Emphasising this point, *SS-1*, from a scientific society that does offer personal and programme accreditation, noted that education was already a criterion for personal recognition at chartered engineer and fellowship status, although not yet at undergraduate programme level. However, she pointed out that "education is as relevant as software engineering or anything else. So, the Society itself has recognised the educators and how important they are".

However, some of these high-level themes and priorities do not manifest at the institutional level, particularly in the UK due to the prominence of research over learning and teaching, and the dominance of university league tables and performance measures such as the Research Excellence Framework (REF)[2], a national research impact evaluation of UK higher education institutions that takes place roughly every seven years. The REF has huge significance for UK universities, especially in the allocation of strategic research funding. Thus, echoing comments from *CS-4* and *NTL-2*, if a research area does not neatly fit into one of the REF units of assessment, this has a detrimental impact on how it is perceived, resourced and supported within an institution. A number of interviewees (for example, both *CS-2* and *CS-4*) highlighted the highly variable attitudes to CE, not always defined by university status – some R1s very supportive, others not, with some smaller colleges that are more focused on teaching not necessarily welcoming of a CE person as they might "not fit" into the established academic career/role structures and hierarchies. But *CS-4* and *UTE* acknowledged the importance of the breadth of academic staff expertise and role, especially the impact of teaching fellows and teaching-focused pathways that were complementary (and equitable) with research pathways. It is possible to be stuck in a rut: since we do not employ people to teach or do this due to perceived REF rules or constraints, it means there are no strong drivers for change. These blocks in the wider ecosystem, while potentially UK-specific, were voiced by academics inside the system, with limited actionability for how to shift and change prevailing policy at both the institutional and national level. However, *CS-4*, *NTL-2* and *UTE* all suggested potential ways in which to alleviate some of these institutional blocks, which could support influencing reform and change at national level, including: developing innovative co-delivery models (e.g. online/blended/distance learning) between multiple institutions to bootstrap activity and build critical mass in this area; developing local, regional and national communities of practice for early-career academics in CS and CE to share and disseminate best practice and co-produce future developments for the area; as well as fostering and promoting disciplinary leadership and advocacy by engaging with national academies, professional bodies, learned societies, as well as industry.

3.2 Content for CS Education Courses

In this section, we present the results of the Delphi study. To reiterate, the aim of this line of investigation was to collect informed opinions about the content that should be embedded in CS undergraduates' departments. The first subsection is about the outcome of the first round of the Delphi process and the second subsection reports the results of the second and third rounds. Additional details on the agreement rates for the categories and themes emerging from the Delphi process can be found in Appendix A.

3.2.1 Delphi first round. In the first round of the Delphi, the participants, apart from some demographic and background questions, were asked to answer an open-ended question regarding their perspectives on CE topics that should be integrated into undergraduate CS departments. Table 6 presents the high-level topics/themes and the more concrete topics suggested by the participants and organised under the corresponding theme. In the following paragraphs we expand a little on this material and report representative related excerpts drawn from the survey.

[2]see: https://www.ref.ac.uk/about-the-ref/what-is-the-ref/

Learning Strategies. Strategies that are employed by learners to enhance learning. As can be seen from Table 6, participants referred to strategies like metacognition, self-explanations and reflective thinking.

Learning & Instructional Theories. This category covers a body of learning and instructional theories that are general, meaning they are applicable in educational settings independently of a specific subject. Participants referred, for instance, to the theories of constructivism and cognitive load theory which have been studied and employed in CE research. For instance, one of the participants explained:

> *"Cognitive Load – looking at what we are trying to get a student to learn and how the exercises we do help or hinder this process. Constructivism – how we construct our own knowledge and that each person's knowledge will be unique and affect how and what we learn."*

Cognitive Processes in Learning Programming. Processes relevant to cognition and learning CS. Examples listed from our participants were, for instance, mental models, notional machines, schemas and plans. These were regarded as important both for the learner as well from a teacher's perspective. As one of our participants mentioned:

> *"I found an understanding of the cognitive processes involved in learning programming concepts or a specific programming language were not just useful for me in teaching these but also influenced the way I approached problems myself and helped me catch some mistakes early."*

Non-Cognitive or Affective Dimensions of Learning. This category includes aspects impacting learning but not bound to cognition. For instance, our participants highlighted the important role of motivation, self-efficacy, a sense of belonging and a growth mindset. On this, one of our participants reported:

> *"Motivation, interest and general attitudes towards Computing (including definitions of constructs such as interest, motivation, self-efficacy, and participation, as well as the gender divide within Computing). This gives students a broader knowledge of how those who do not study computing tend to feel about it and how this can be affected."*

Broad Teaching Methods and Approaches. By this we refer to teaching methods/approaches with a wider scope, which is not limited to computing contexts. For example, methods like blended learning, Socratic questioning, peer learning and semantic waves were some of the examples our participants mentioned. As previously, not only did our participants consider benefits from the learners' side, but also for other groups such as mentors and tutors. One of the participants highlighted that:

> *"Broader teaching and discussion around pedagogy is useful when engaging in peer mentoring activities, either group work, or as a tutor/demonstrator for other students, but also helped me engage more effectively as a student with teaching staff since I could more quickly recognise their approach."*

Table 6: Results from the first round of the Delphi

	High-Level Categories	Concrete Examples	Frequency
1.	Learning Strategies	Metacognition & Self-regulation, Reflective Thinking, Self-explanations	21.3%
2.	Learning & Instructional Theories	Constructivism/Constructionism, Sociocultural Perspectives, Multiple Conceptions Theory, Critical Consciousness, Cognitive Load Theory, Theory of Multimedia Development, Nudge Theory, Learning Styles, Direct Instruction	23.4%
3.	Cognitive Processes in Learning Programming	Mental Models, Notional Machines, Memory, Cognitive Load Theory, Schemas & Plans	25.5%
4.	Non-Cognitive or Affective Dimensions of Learning	Growth Mindset, Self-efficacy, Motivation, Engagement, Attitudes, Identity, Sense of Belonging	21.3%
5.	Broad Teaching Methods and Approaches	Blended Learning, Socratic qQ Pedagogy, Taxonomies and Hierarchies of Knowledge and Understanding, Summative and Formative Feedback	19.1%
6.	Teaching Methods, Approaches to Teaching Programming	Pair Programming, PRIMM, Program Comprehension (e.g., Block Model), Programming Problem-solving, Reasoning in Programming, Strategies for debugging, Levels of Abstraction	27.7%
7.	Activities and Tools for Learning Programming	Worked Examples, Parsons Puzzles, Explain in Plain English, Tracing Code, Code Reviews	10.6%
8.	Learning Difficulties in Computing and Programming	Threshold Concepts, Misconceptions, Problem solving & Reasoning, Levels of Abstraction, Thinking Computationally	25.5%
9.	Diversity & Inclusion	Microaggression, Implicit Bias, Stereotype Threat, Stereotypes, Unconscious Bias, Gender Differences, Accessibility, Sociological Theories (e.g., Bourdieu), Neurodiversity	21.3%
10.	Society and Ethics in Computing	Societal Impact of Technology and Computing, Ethics Issues Involved in Teaching relevant to TAs as well as teachers/faculty (e.g., conflict of interest), Responsible Computing Practices, Accessibility	6.4%
11.	Course Planning	Curriculum Design, Policy and CS Curricula, Course Design, Planning and Delivering a Lesson or a Session	19.1%
12.	Course Management	Behaviour and Classroom Management	4.3%
13.	Assessment	Exam design, Formative and Summative Assessment Design, Tools for Assessment (e.g, automatic)	14.9%
14.	Grading	Providing Feedback, Formative and Summative, Grading and Rubrics	8.5%
15.	Engaging with Students	Responding to Students, Generating Discussions, Engagement	4.3%
16.	Technology in Education	Developing Educational Software/Technology, Accessibility, Online Learning Platforms, Learning Management Systems, LLM-based Chatbots, User-generated Content (e.g., internet forums, StackExchange, youtube, etc.)	8.5%
17.	History and Nature of Computing and Computing Education		6.4%
18.	Research in Computing Education	Qualitative and Quantitative Research Designs and Use in CE, Research Methods (such as surveys, interviews, focus groups, experimental methods and statistics, and application in CE), CE research methodology (e.g., phenomenography, phenomenology), Reading & Understanding Academic Papers, Practice Academic Writing, Practical Eexperience of the whole Research Cycle, Current Trends of research in CE, Communicating Research Ffindings, Reflecting on Research Findings	25.5%
19.	Professional Competencies	Communication Skills, Intercultural Skills, Teamwork and Collaboration, Conflict Resolution, Argumentation, Explaining/Communicating Ideas Clearly (e.g., programming concepts, problem solutions), Teaching Peers, Mentoring & Coaching, Giving Feedback, Leading & Managing a Team, User-centred Approaches like Participatory Design, Delivering a Presentation, Time Management	42.6%

Teaching Methods, Approaches to Teaching Programming. Methods and approaches specific to teaching programming such as PRIMM, program comprehension, and strategies for debugging:

> *"Finally, instruction on effective strategies for debugging and reasoning about code, and especially for helping others do the same, would not only be an excellent way to introduce students to education-related topics but would also be helpful even to those on a purely technical track, as those skills are central to industry roles as well."*

Activities and Tools for Learning Programming. Specific activities that have been used to facilitate learning in programming such as worked examples and Parsons Puzzles:

> *"General how learning CS works should be taught to all students to help them more effectively learn CS. Such as how learning how to code trace helps you learn how to explain in plain English what code is doing, which helps you with communication skills and grappling with code in general."*

Learning Difficulties in Computing and Programming. Some of the participants' suggestions refer to the problems and challenges learners face when learning Computing and Programming, such as misconceptions and more general difficulties during problem-solving:

> *"CE could improve their skills in computational reasoning, which is necessary for computer scientists working in all areas (private companies, teachers, researchers). By better analyzing inside CE courses which are the key ingredients and the difficulties in teaching/learning CS topics, students could have an advantage in the Master courses and more in general for Long-Life Learning."*

Diversity & Inclusion. This category reflects the participants' responses relevant to issues of inclusion and diversity such as implicit bias, stereotypes and accessibility. In highlighting the importance of these topics, one of our participants emphasised:

> *"CS graduates should also know about gender differences, computing identity, and computing-related stereotypes, which are topics covered by CE, are rarely touched upon in curricula, and would help to improve the work culture in the CS industry."*

Society and Ethics in Computing. Here the focus is on ethics, whether relevant to teaching or regarding technology and the impact of technology on society:

> *"It is crucial to raise awareness about ethical considerations, responsible computing practices, and the societal impact of technology."*

Course Planning. Design topics, ranging from planning an individual lesson, a unit or a course, to designing a more comprehensive curriculum:

> *"Curriculum design is a big one for me – being able to understand why an exercise is included and what students should be learning from that exercise has been really important for me when teaching".*

Course Management. This category highlights topics relevant to classroom and behavioural management:

> *"If students are doing a student ambassador module - where they are going into schools, then they need to also learn about behaviour management."*

Assessment. Assessment brings together topics referring to different aspects of assessment such as designing an exam, formative and summative assessment, as well as tools that can be employed to facilitate this process. One of our participants mentioned that:

> *"Aspects of summative, formative and continuous assessment/evaluation, both in general (e.g., the use of rubrics or of techniques of formative assessment like live quizzes to break the lecture or checklist for qualitative observations) and specifically (e.g. pros and cons of using automated assessment of programming exercises, how to create good cloze quizzes for programming, and so on) is important because those techniques can be applied in a variety of situations."*

Grading. Topics relevant to grading processes and feedback:

> *"Exam setting and grading: students need to be taught different techniques that go into exam setting and grading process so that they can effectively prepare exams that test their students' knowledge, and also perform fair grading and reach an appreciable outcome."*

Engaging with Students. The subject of this category is the interaction between learners and teachers or tutors, such as prompting and facilitating discussions or engaging a student:

> *"Contemporary issues – [...], classroom engagement – can be included as final year options or project topics."*

Technology in Education. This area brings together aspects relevant to developing technological educational tools, online learning platforms and learning management systems:

> *"Since technology is increasingly used in education, it is essential to cover subjects such as educational technology, online learning platforms, learning management systems, and instructional design. These topics will help students use digital tools to enhance their teaching and learning experiences."*

History and Nature of Computing and Computing Education. This is a broad theme focusing on the evolution of CE and Computing as scientific fields as well as of policies and curricula:

> *"History of computing education within universities and schools, including a history of policy and curricula. This gives students some contextual understanding into the field of computing education, how long it existed and how many students learn to compute globally."*

Research in Computing Education. This last category groups together a set of topics and skills relevant to those interested in conducting research both in the field of CE as well as in industrial settings:

> *"Providing students with knowledge of research methodologies, data collection techniques, and analysis approaches specific to computing education research will enable them to conduct thorough and valuable research."*

Professional Competencies[3]*.* This category bring together topics relating to skills that computing undergraduates as well as those involved with teaching CS should develop. For instance, one of our industry participants highlighted:

[3]Each of the competencies identified within this theme and listed in Table 6 was examined for agreement. Details about the rates of agreement are reported in the tables of Appendix A.

"Articulating effectively a technical solution, software design or even an idea to other engineers is an important aspect of an engineer's work. [...] another important aspect is effective collaboration and letting go of egos to push the team forward."

3.2.2 Results of the second and third round. In the second round of the Delphi, the participants were asked to rate, using a 5-point Likert scale, the importance of the CE topics and professional skills identified in the first round. Each participant was asked to rate the importance of each topic for each of the (future) roles of undergraduate students, i.e. tutors/teaching assistants (TA), teachers/faculty (TT), researchers (R), industry (I), and general undergraduate students (UG). Following the Delphi method, the results from the second round were given to the participants for a third round, where they were asked to rate the importance of the different topics and skills again if they wanted to change anything after having read the results and arguments from the second round. The final results after the third round are presented in Tables Q1–Q32 of Appendix A, where disagree means 1–2 and agree 4–5 on the Likert scale.

According to the Delphi findings, there was agreement (> 60%) that *Society and Ethics in Computing*, *Communication Skills*, *Teamwork and Collaboration*, *Delivering a Presentation*, *Explaining ideas*, and *Time Management*, are important for all CS undergraduates, regardless of role. The topic of *Learning Strategies* reached an agreement for all groups apart from undergraduates aiming for an industry career. The topic of *Code Reviews* reached an agreement for tutors, undergraduates aiming for industry, and general undergraduates. The topics of *Intercultural Skills*, *Giving Feedback*, and *Diversity and Inclusion* reached an agreement for all groups apart from general undergraduates (not TAs or industry). The topics of *Cognitive Processes*, *Non-cognitive dimensions of learning*, *Broad Teaching Methods"*, and *Teaching Methods and Approaches* reached an agreement for all apart from general undergraduates and those aiming for a career in industry. The topics of *Mentoring and Coaching*, *Team Leadership* and *Conflict Resolution* reached an agreement for all apart from research students and general undergraduates. The topics of *Course Management*, *Assessment*, *Grading*, *Engaging with Students*, *Teaching Peers* and *Activities for Learning Programming* reached an agreement for teaching assistants and those interested in a teaching career. *Research skills*, *Argumentation* and *Learning and Instructional theories* reached an agreement only for those interested in a teaching career and research students. Finally, the topics of *Applying User-centered Approaches* like participatory design, *Course Planning*, and *Technology in Education* reached an agreement for those interested in teaching careers, while the topic of *History and Nature of Computing and CE* reached an agreement only for research students.

In addition to the agreement results documented in the Appendix, we were interested in having group-specific results. By group-specific results, we mean focusing on responses from four Delphi participant groups: TAs/Tutors (TA), PhD students (PhD), Industry, and CE academics (CSE) and see whether there is an agreement between participants from each group about the importance of topics to the role(s) related to this participant group. For example, we were interested to see agreement percentages (Agree% and Disagree%) among TA Delphi participants to the importance of each topic as related to the TA role. The same is true for PhD (importance for the R role) and Industry participants (importance for the I role). For CSE participants, we believe that it would be helpful to do the same, but for all roles (i.e., TA, TT, R, I, and UG). Results from this group-specific analysis are included in Table 7.

As we see from Table 7, there is an agreement determined by having 60% or more of Agree (the topic is important for the respective role) or Disagree (topic is unimportant for the respective role) among the TA participants on all topics except for: *Society and Ethics in Computing Course Planning, Course Management, Technology in Education, History and Nature of Computing and CE, Research Skills, Argumentation, Team Leadership and Management, and Delivering a Presentation.*

For PhD participants, there is an agreement on all topics except for: *Activities/Tools for Learning Programming, Course Planning, Course Management, Assessment, Grading, Engaging with Students, Technology in Education, Code Reviews, Conflict Resolution, and Mentoring and Coaching.*

For Industry participants, there is an agreement on all topics except for: *Broad Teaching methods and Approaches, Teaching Methods and Approaches to Teaching Computing/Programming, Activities/Tools for Learning Programming, Assessment, Grading, Technology in Education, and Team Leadership and Management.*

In Table 7, we see that CSE participants agreed on all topics for the TA role except *Learning and Instructional Theories, Course Planning, History and Nature of Computing and CE, Research Skills, Argumentation, and Applying User-centred approaches* like participatory design. For the TT role, CSE participants agreed on all topics except *Code reviews* and *Argumentation.* For the R role, CSE participants agreed on all topics except for: *Course Planning, Course Management, Code Reviews, Conflict Resolution, and Teaching Peers.* For the I role, CSE participants agreed on all topics except *Learning Strategies, Learning and Instructional Theories, Cognitive Processes in Learning programming, Non-cognitive or affective dimensions of learning, Activities/Tools for learning programming, Learning Difficulties in Programming, Course Management, Assessment, Grading, Engaging with Students, Technology in Education, History and Nature of Computing and CE, Research Skills, Code Reviews, Argumentation, Conflict Resolution, Teaching peers, and Applying User-centered approaches like participatory design.* Finally, for the UG role, CSE participants agreed on the following topics: *Diversity and Inclusion, Society and Ethics in Computing, Communication Skills, Teamwork and Collaboration, Explaining ideas, Intercultural skills, Conflict Resolution, Teaching peers, Giving feedback, Delivering a presentation, Time management.*

3.3 Related CS Education Courses

Regarding the formal resources collected through the systematic literature review and the considered informal resources, the following tables demonstrate the percentage of papers and informal resources per targeted population.

Table 8 shows the number of papers grouped by the targeted population as well as the papers referring to these.

Table 7: Group-specific results

Delphi participant group		TA	PhD	I	CSE				
Target group		TA	R	I	TA	TT	R	I	UG
Q1: Learning Strategies	Agree	**100.0**	**77.8**	**66.7**	**84.6**	**92.3**	**92.3**	46.2	53.8
	Disagree	0.0	0.0	0.0	0.0	0.0	0.0	23.1	15.4
Q2: Learning & Instructional Theories	Agree	**71.4**	**77.8**	33.3	38.5	**76.9**	**92.3**	7.7	0.0
	Disagree	14.3	0.0	**66.7**	7.7	0.0	7.7	38.5	53.8
Q3: Cognitive processes in learning programming	Agree	**85.7**	**66.7**	**66.7**	**84.6**	**92.3**	**92.3**	30.8	30.8
	Disagree	0.0	0.0	33.3	0.0	7.7	0.0	53.8	7.7
Q4: Non-cognitive or affective dimensions of learning	Agree	**71.4**	**88.9**	**100.0**	**76.9**	**92.3**	**100.0**	53.8	53.8
	Disagree	0.0	11.1	0.0	0.0	0.0	0.0	23.1	7.7
Q5: Broad Teaching Methods & Approaches	Agree	**71.4**	**77.8**	33.3	**76.9**	**92.3**	**69.2**	30.8	23.1
	Disagree	0.0	11.1	33.3	7.7	7.7	7.7	**61.5**	53.8
Q6: Teaching Methods & Approaches to teaching Computing/Programming	Agree	**100.0**	**88.9**	33.3	**100.0**	**100.0**	**76.9**	30.8	38.5
	Disagree	0.0	11.1	33.3	0.0	0.0	0.0	**61.5**	23.1
Q7: Activities/Tools for leaning programming	Agree	**100.0**	55.6	0.0	**92.3**	**100.0**	**61.5**	38.5	46.2
	Disagree	0.0	33.3	33.3	0.0	0.0	0.0	46.2	23.1
Q8: Learning Difficulties in Programming	Agree	**100.0**	**66.7**	**66.7**	**100.0**	**100.0**	**84.6**	30.8	46.2
	Disagree	0.0	22.2	0.0	0.0	0.0	0.0	38.5	23.1
Q9: Diversity & Inclusion	Agree	**71.4**	**88.9**	**66.7**	**100.0**	**100.0**	**76.9**	**76.9**	**69.2**
	Disagree	14.3	0.0	0.0	0.0	0.0	0.0	0.0	7.7
Q10: Society and Ethics in Computing	Agree	57.1	**88.9**	**66.7**	**61.5**	**76.9**	**92.3**	**76.9**	**61.5**
	Disagree	14.3	0.0	0.0	15.4	7.7	0.0	15.4	0.0
Q11: Course Planning	Agree	42.9	44.4	33.3	53.8	**92.3**	38.5	23.1	23.1
	Disagree	0.0	22.2	**66.7**	7.7	0.0	15.4	**61.5**	**61.5**
Q12: Course Management	Agree	57.1	33.3	0.0	**84.6**	**100.0**	38.5	30.8	15.4
	Disagree	14.3	22.2	**66.7**	0.0	0.0	46.2	53.8	**69.2**
Q13: Assessment	Agree	**71.4**	55.6	0.0	**76.9**	**100.0**	**84.6**	30.8	30.8
	Disagree	0.0	22.2	33.3	7.7	0.0	7.7	53.8	38.5
Q14: Grading	Agree	**85.7**	44.4	33.3	**92.3**	**100.0**	**61.5**	30.8	53.8
	Disagree	0.0	22.2	0.0	0.0	0.0	23.1	53.8	23.1
Q15: Engaging with students	Agree	**100.0**	44.4	**66.7**	**100.0**	**100.0**	**84.6**	38.5	53.8
	Disagree	0.0	11.1	33.3	0.0	0.0	15.4	38.5	38.5
Q16: Technology in Education	Agree	42.9	55.6	33.3	**76.9**	**69.2**	**69.2**	46.2	53.8
	Disagree	14.3	11.1	33.3	15.4	7.7	0.0	23.1	46.2
Q17: History and Nature of Computing and CE	Agree	14.3	**100.0**	33.3	23.1	**61.5**	**69.2**	30.8	38.5
	Disagree	42.9	0.0	**66.7**	15.4	0.0	7.7	38.5	23.1
Q18: Research Skills	Agree	14.3	**100.0**	**66.7**	38.5	**69.2**	**100.0**	38.5	38.5
	Disagree	14.3	0.0	33.3	23.1	7.7	0.0	46.2	15.4
Q19: Communication Skills	Agree	**100.0**	**88.9**	**100.0**	**100.0**	**100.0**	**76.9**	**84.6**	**84.6**
	Disagree	0.0	0.0	0.0	0.0	0.0	15.4	0.0	7.7
Q20: Teamwork and Collaboration	Agree	**85.7**	**88.9**	**66.7**	**84.6**	**92.3**	**76.9**	**69.2**	**76.9**
	Disagree	14.3	0.0	0.0	7.7	7.7	0.0	15.4	0.0
Q21: Code Reviews	Agree	**85.7**	33.3	**66.7**	**69.2**	53.8	30.8	46.2	53.8
	Disagree	0.0	22.2	0.0	0.0	7.7	23.1	15.4	0.0
Q22: Argumentation	Agree	57.1	**100.0**	**66.7**	30.8	46.2	**61.5**	23.1	38.5
	Disagree	14.3	0.0	33.3	23.1	7.7	7.7	23.1	7.7
Q23: Explaining Ideas	Agree	**100.0**	**100.0**	**66.7**	**100.0**	**92.3**	**92.3**	**76.9**	**76.9**
	Disagree	0.0	0.0	33.3	0.0	0.0	7.7	7.7	15.4
Q24: Intercultural Skills	Agree	**71.4**	**66.7**	**66.7**	**84.6**	**92.3**	**76.9**	**76.9**	**69.2**
	Disagree	14.3	0.0	0.0	0.0	0.0	7.7	7.7	7.7
Q25: Conflict Resolution	Agree	**71.4**	44.4	**100.0**	**100.0**	**92.3**	46.2	53.8	**61.5**
	Disagree	0.0	22.2	0.0	0.0	0.0	30.8	30.8	7.7
Q26: Teaching Peers	Agree	**71.4**	**77.8**	**100.0**	**69.2**	**76.9**	53.8	46.2	**61.5**
	Disagree	0.0	22.2	0.0	7.7	7.7	15.4	23.1	15.4
Q27: Mentoring and Coaching	Agree	**85.7**	44.4	**100.0**	**84.6**	**84.6**	**69.2**	**69.2**	30.8
	Disagree	0.0	11.1	0.0	0.0	0.0	15.4	0.0	23.1
Q28: Team Leadership and Management	Agree	42.9	**77.8**	33.3	**92.3**	**84.6**	**84.6**	**61.5**	53.8
	Disagree	42.9	0.0	0.0	0.0	15.4	0.0	7.7	15.4
Q29: Giving Feedback	Agree	**100.0**	**77.8**	**100.0**	**100.0**	**100.0**	**61.5**	**61.5**	**61.5**
	Disagree	0.0	11.1	0.0	0.0	0.0	23.1	23.1	15.4
Q30: Delivering a Presentation	Agree	57.1	**100.0**	**66.7**	**76.9**	**84.6**	**92.3**	**61.5**	**61.5**
	Disagree	0.0	0.0	33.3	15.4	0.0	0.0	0.0	7.7
Q31: Time Management	Agree	**85.7**	**100.0**	**66.7**	**92.3**	**92.3**	**84.6**	**92.3**	**84.6**
	Disagree	0.0	0.0	33.3	0.0	7.7	0.0	0.0	0.0
Q32: Applying User-centered approaches like participatory design	Agree	**85.7**	**77.8**	**66.7**	46.2	**61.5**	**69.2**	53.8	38.5
	Disagree	14.3	22.2	33.3	30.8	15.4	7.7	46.2	46.2

Table 8: Papers included per group

Group	Number of papers		References
Tutors/TAs/Mentors	11	(44%)	[30, 40, 67, 68, 74, 78, 83, 88, 89, 98, 121]
Faculty Training & Development	2	(8%)	[72, 92]
MSc students	1	(4%)	[115]
Undergraduates/Graduates (not TAs)	4	(16%)	[5, 8, 87, 112]
Undergraduates related to teacher training	2	(8%)	[81, 128]
Teacher Training	5	(20%)	[13, 50, 52, 71, 95]

With respect to teacher training, we note that most of the relevant papers matching our search terms did focus on addressing CE content in in-service teacher (re-)training, introducing Computational Thinking to pre-service teachers in other disciplines, or addressing isolated aspects of CE in single-intervention studies. In contrast, reports on the overall design of CE courses and their evaluation in the venues surveyed were scarce, likely reflecting the fact that most countries either had well-established teacher training programmes in place already or were in "bootstrapping" phases during the time frame considered for our literature review.

Table 9 shows the number of informal resources grouped by the targeted population as well as the online material referring to these. Note that we did not include informal resources accompanying formal teacher training: As discussed in Section 1, such training is composed of a variety of courses, including general education, and reporting only on the CE part of those would present a limited view and skew our results. For the intent and purposes of our working group, the relevant information could be extracted from the textbooks used.

Table 9: Informal resources per group

Group	Number of resources		References
Tutors/TAs/Mentors	9	(43%)	[1, 2, 54, 75, 76, 82, 90, 120, 122]
Faculty Training & Development	0	(0%)	
MSc students	3	(14%)	[32, 66, 77]
Undergraduates/Graduates (not TAs)	7	(33%)	[39, 60, 61, 69, 109, 110, 126]
Undergraduates related to teacher training	1	(5%)	[4]
PhD students	1	(5%)	[36]

3.3.1 High-Level Aims and Contents per group.

In the following paragraphs, we outline the courses' high-level aims and contents collected and listed together for each of our groups of interest.

Tutors/TAs/Mentors. Use of TAs is one way to help with scaling of CS courses. The aims of TA training were to improve the teaching effectiveness of TAs, to build community, and to promote inclusivity. Topics included:

- Basic terminology in CE.
- Learning theory basics: Cognitive load, constructive alignment.
- Pedagogy and teaching methods: Class structure, classroom management
- Small group discussions and role-play teaching scenarios.
- Active learning and group work.
- Observations of teaching, and demonstration and feedback on teaching.
- Diagnostic teaching.
- Assessment, grading and feedback, and rubrics.
- Teaching scenarios, instructional design for TA-led interventions.
- Inclusive teaching, equity and diversity, microaggressions, stereotypes, and biases
- Professionalism and Ethics, and Roles and Responsibilities.

Faculty. CE courses aimed to equip faculty with an understanding of pedagogical frameworks and theories and learner-centred issues. Specifically, the aim was to help faculty understand theories of intelligence (growth vs fixed mindset) emphasising the importance of promoting a growth mindset as well as issues relating to recruitment and retention of women and underrepresented groups and promoting interactions among and with computing students.

High-level topics included:

- Production of meaningful content for students.
- Collaborative learning and inclusive pedagogy strategies.
- Classroom management for diverse populations.
- Stereotype threat and unconscious bias.
- Growth and fixed mindset.
- Knowledge of multiple CS-specific teaching approaches.

MSc Students. Courses received by these students are mostly focused on preparing them for becoming academics. However, these courses could also serve students pursuing a PhD in CE. Two main aims drive how CE is approached to MSc Students. On the one hand, there is an interest in CE research, focusing on research methods, e.g. dealing with the congruence that the elements of a CS research study must have. On the other hand, these students are taught about the teaching CS, providing detailed instruction and practice. Thus, the focus is on fundamental concepts concerning how students learn CS and on how the subjects are taught. These aims are reflected in the following list of topics:

- Knowledge of some historical, epistemological and ethical aspects of CS as a scientific discipline and of the motivations underlying the necessity of its teaching.
- Ability to organize laboratories, classroom equipment, and to integrate students' devices as useful tools for learning.
- Equity and diversity.
- Knowledge of research methodology: the techniques and their applicability.

- Knowledge of the main cognitive difficulties in learning CS (with particular focus on programming), and of possible strategies to adopt to overcome them.
- Ability to formulate and manage learning paths consistent with national standards and curricula related to CS in schools of all levels.
- Understanding of pedagogical aspects and learning theories in the context of CS teaching.

Undergraduates/Graduates (not TAs). This category includes courses that introduce core topics in CE Research to potential CS Ed researchers and courses aimed at students who want to better understand their own education or who are considering becoming a teacher, as well as those who want to put evidence-based teaching techniques into practice in their work as peer mentors, teaching in a summer camp or local schools, developing curricula, advising CS teachers or educational organizations, future outreach efforts, or other classroom environments not specifically tied to a role as a TA. Topics include:

- Cognitive science, pedagogy and learning theories.
- Curricular design activities and real-world practice.
- Learning objectives and assessment.
- Instructional design, designing courses and assignments
- Motivation and affect.
- Environments for learning programming.
- How novices learn to program.
- How people learn CS concepts.
- Computing for other disciplines.
- CS curriculum and curriculum design.
- Learning programming.
- K–12 CE and outreach.
- Programming paradigms.

Courses focused on enabling students to do CS Ed research also included:

- research methodologies used in CE
- statistical methods
- qualitative methods
- how to read and critically examine CS Ed Research papers
- how to plan and carry out a CS Ed research project

Undergraduates aiming for a teaching career or on a teaching track. The primary objective of CE courses offered to this category was to equip students thoroughly for teaching and managing their own classes through highlighting key aspects of pedagogy and teaching methods in CS, tools and learning environments, learner-centered issues and providing them with field experience. High levels topics include:

- Field-related activities.
- Interacting with students and teachers, lesson plans, and differentiating instruction.
- Feedback.
- Equity.
- Computational Activities (e.g., online CS resources, block-based programming, computational tools and robotics)
- Pedagogical Activities to learning CS.

PhD Students. Courses offered to these students focus their attention on gaining knowledge, competencies and skills needed to participate and set up research in the area of CE. High levels topics of these courses include:

- Learning theory: Constructivism.
- Active learning, e.g., flipped classroom and project-based learning.
- Online learning, MOOCs.
- Learner-centered design of instruction.
- Teachers' beliefs and motivational orientations.
- Technology in education.
- Employability and professional skills.
- Participation.

Teacher training. We analyzed the papers found through our literature review and merged the topics identified with the core curriculum suggested by Ragonis and Hazzan [52, 93] resulting in the following list:

- Assessment.
- CE Research.
- Classroom climate.
- Course planning.
- Definition and history of the discipline.
- Diversity.
- Fundamentals of Learning Sciences.
- Goals of and curricular requirements for teaching CS in schools.
- Hands-on activities with feedback.
- Intervention design.
- Learning techniques and methods (e.g., unplugged activities and peer evaluation).
- Methodological aspects of teaching specific topics.
- Projects.
- Setting and reflecting upon competence-oriented goals.
- Subject-matter specifics.
- Teaching methodologies.
- Tools and learning environments in CS.
- Topics of and challenges to CE in schools.
- Transferring general educational principles to CE.

3.3.2 Concluding remarks. As seen above, we have collected a large range of topics covered in the considered CE interventions, either documented in the literature or identified through less formal means. Based on this material, we can observe that some areas of interest are deemed to have a broader scope than others, i.e. occur in the lists of diverse target groups. This is the case, in particular, for the matters related to *instructional design*, general *learning theory* and *inclusive teaching*. However, also *learning environments*, *assessment*, *providing feedback* and *CE research* issues seem to play some "transversal" role.

Perhaps unsurprisingly, instructional design, learning theory and inclusive teaching reached appreciable rates of agreement also in the Delphi process. On the other hand, while *metacognition* has been highlighted in the interviews and reported as a meaningful learning strategy by some Delphi participants, there is scarcely any clear reference to how to develop metacognitive skills in the identified resources. Similarly, the role of developing *abstraction* appear to be downplayed, whereas being able to reason at different levels of abstraction has been deemed as important by Delphi participants in connection with both the approaches to teaching and the difficulties faced by learners.

4 SAMPLE COURSES AND CONTEXTS

Through the literature review and the Delphi study, we have identified a number of high-level topics that should be considered for inclusion in a course in computing education, depending on the target group(s) for the course. Synthesising this work, we now present exemplar curricular outlines for two of the groups we focused on in this study.

4.1 TA/Tutor Training Course

Figure 2 gives an overview of a possible course for TAs/tutors, with a partial order of the topics to be included. Note that this is only meant as an example of how to use the results of this working group. Different courses for TAs may emphasize different aspects of the topics and also choose a different ordering of the topics, depending on their local context.

To arrive at the course structure in Figure 2, we used the following 3 sources presented in the results section:

(1) Tables Q1–Q32 in Appendix A
(2) Table 7
(3) Literature Review in Section 3.3.1

We included topics that had reached an agreement in more than one of the above Delphi sources (1–2[4]) or had reached an agreement in one of these and were also reported in the literature.

This resulted in a set of CE topics, represented with white boxes in Figure 2, and soft skills/general professional competencies, represented with coloured boxes in Figure 2. Please note that these skills/competencies can be practised during activities that address different topics — the figure provides only one of the multiple possibilities. After identifying the topics, we discussed a possible ordering based on the experiences from our local contexts and from the work done in this working group.

As seen from Figure 2, this example course consists of four main modules: 1) society, ethics and computing, 2) a theoretical part including both general learning theories and more CS-specific topics, 3) practical aspects of teaching CS, and 4) assessment and grading theory and practice. These modules may be taught in sequence, but could also be interleaved to achieve a tighter integration between theory and practice. The soft skills (coloured boxes) may be added to the different CE topics in various ways, where Figure 2 depicts one way of doing this. Within an undergraduate programme, one single course cannot be responsible for all the soft skills considered to be important. As seen in the figure, for this particular example course we decided not to include teamwork & collaboration, team leadership & management, time management and teaching peers, but leave that to other courses in the programme.

[4]Note that Table 7 is seen as two merged tables: one referring to CSE ratings and the other one to the specific groups of our Delphi participants encompassing the roles of TA, PhD, and Industry.

4.2 Industry-Focused General Undergraduate Course

Using the same process as above, a second sample course structure for general undergraduates focusing on topics relevant to working in the industry was created, with the result shown in Figure 3. Again, this course consists of four modules, where the first two related to society and ethics, and related to learning computing are similar to the ones for TA/tutor training, but without the focus on the general learning theories. Furthermore, instead of theory and professional competencies related to working as a TA/tutor, the general undergraduate course focuses on professional competencies that were found to be relevant for work in industry, including research skills.

Considering the sample courses derived from the intersection of the literature review and the Delphi study, in connection with the arguments presented in section 3.2, it is noteworthy that a good alignment exists. To reiterate, our interview participants underscored the significance of Computing Education (CE) courses for students aspiring to pursue a career in industry: Software Engineering (SE) entails both formal and informal teaching activities, the development of instructional materials, and demands essential communication skills, coaching, and mentoring. While the course outline depicted in Figure 3 may not directly engage these students in teaching methods, it does afford them the opportunity to comprehend various facets associated with teaching and learning. This includes aspects related to cognition and affect, issues surrounding inclusion and diversity, and the acquisition of skills vital for effective communication, mentoring, coaching, articulating ideas, providing feedback, and delivering presentations.

Similarly, interview participants stressed the pertinence of CE courses for students aiming to assume a Teaching Assistant (TA) role. These courses, the participants argued, could serve to enhance teaching quality and retention by equipping students with training in assessment techniques, pedagogy, and subject-specific educational skills. As Figure 2 illustrates, TAs are presented with a valuable opportunity to immerse themselves in a multifaceted learning experience. This includes delving into topics related to assessment such as grading and feedback, as well as pedagogical, addressing various teaching methodologies and approaches, and learning and instructional theories which span a wide spectrum, ranging from broader topics to subject-specific nuances.

5 DISCUSSION AND CONCLUSIONS

At the outset of this study, we stated three research questions:

RQ1: What are arguments in support of introducing CE in undergraduate CS programmes?
RQ2: What are topics that different stakeholders, including, but not limited to, instructors, researchers, and employers consider both important and fitting for an undergraduate audience?
RQ3: What are current practices in teaching CE to undergraduate audiences?

We will now explore these questions in light of the results from the three major activities of our working group, leading on to discussion of the example course outlines that have been presented, and concluding with our overarching observations and expectations for future work.

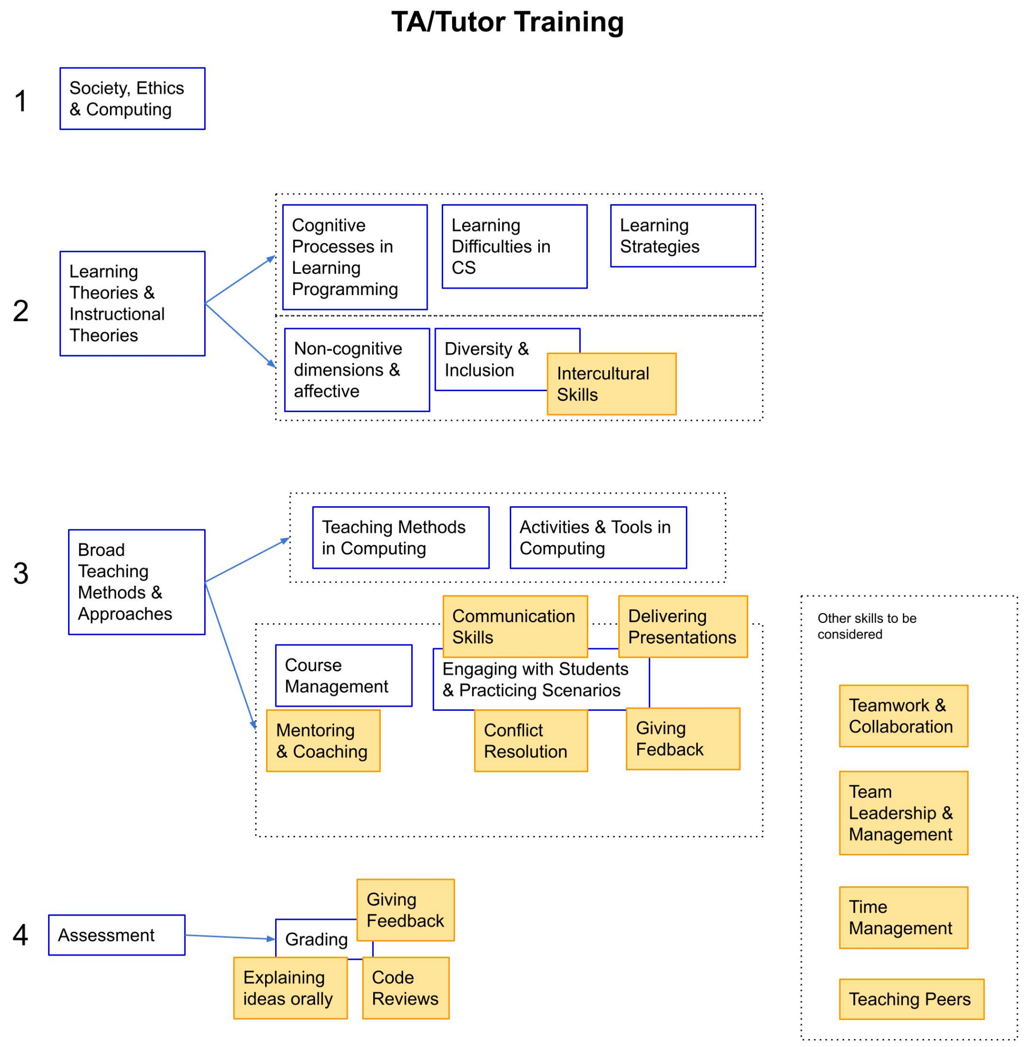
TA/Tutor Training
1
Society, Ethics & Computing
2
Learning Theories & Instructional Theories
Cognitive Processes in Learning Programming
Learning Difficulties in CS
Learning Strategies
Non-cognitive dimensions & affective
Diversity & Inclusion
Intercultural Skills
3
Broad Teaching Methods & Approaches
Teaching Methods in Computing
Activities & Tools in Computing
Communication Skills
Delivering Presentations
Course Management
Engaging with Students & Practicing Scenarios
Mentoring & Coaching
Conflict Resolution
Giving Fedback
4
Assessment
Grading
Giving Feedback
Explaining ideas orally
Code Reviews
Other skills to be considered
Teamwork & Collaboration
Team Leadership & Management
Time Management
Teaching Peers

Figure 2: High-level contents for one possible Computing Education course targeted at TAs/tutors

5.1 RQ1 – Arguments

In our draft arguments, we had seven categories representing seven lines of argument. Spread between these, we had a further 28 statements that fleshed out these categories and lines of argument. The thematic analysis of our interviews with stakeholders resulted in 18 themes which matched well to our arguments. The discussion will center on the comments of two groups of respondents – the academics, and the rest. The academics understand the internal workings of university departments deeply, and so are in a good position to advise on blocks or opportunities at that level, whereas the other stakeholders were able to give us perspectives and opportunities to reflect on our own, probably biased, thinking.

We discuss the academics' advice here, and the other stakeholder's perspectives are considered as part of an overall discussion in Section 5.5. While we had mainly presented positive arguments for the inclusion of CE courses, our academic respondents noted that our arguments would not be universally applicable depending on the department. For example, some programmes predominantly consist of a fixed set of modules or courses, and it would be much harder to add in a new CE course to such a programme compared to one where there was student choice among a large range of optional modules or courses. At the same time, some departments make little use of undergraduate TAs/tutors besides using them for grading and so would not immediately warm to our undergraduate TA/tutor argument. Potentially, however, a twin argument could be made based on the value of such teaching support alongside the necessary mechanism to prepare for it, and on teaching loads of CE academics already being too high. Some of these arguments could be addressed by considering limited CE input placed within another important course. For example, the industry-facing course outline we have proposed here serves both to introduce a little CE material

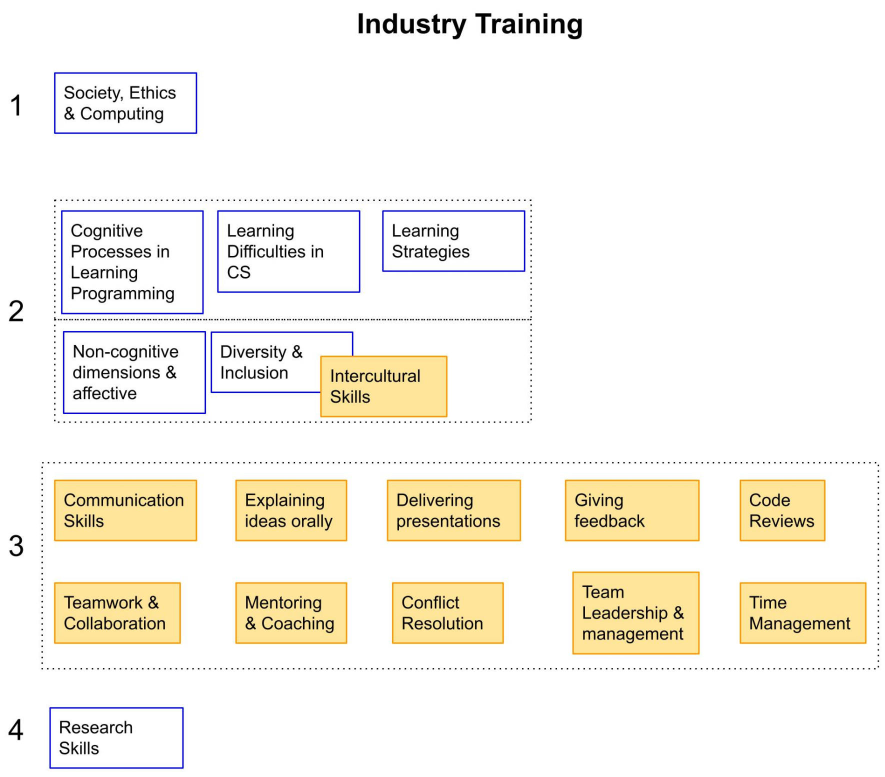

Figure 3: High-level contents for one possible Computing Education course targeted at undergraduates aiming for industry

while also covering essential soft-skills material appropriate for software engineering students.

The academics were also able to point us towards persuasive arguments that could be effective in the longer term. A particular strand here is the inclusion of CE within degree accreditation processes. Not all institutions attempt to accredit their degrees, of course, but for those that do, CE skills and competences are seen as an important aspect of their students' employability. We were surprised to find that the accrediting body included in our stakeholder group held strong views about this, already recognising educational contribution towards their fellowship awards, and were open to considering CE aspects in degree programme accreditation. We note that – although not an official accreditation mechanism – the ACM's decennial curriculum report is a strong persuader about what should be included in a computing department's curricula and, at the time of this writing, the drafts for the ACM Computing Curricula 2023 make no mention of CE as a knowledge area. We hope that we do not need to wait until the ACM Computing Curricula 2033 before CE is acknowledged in this way.

5.2 RQ2 – Stakeholder CE Topics

The Delphi process produced 19 high-level categories for consideration in CE courses in undergraduate CS programmes. Along with the examples given, this is a clear demonstration of how much there is that could be covered in a programme of CE courses.

While reviewing the broad range of data presented in sect. 2.3, we were reminded of Kafai et al.'s [59] structuring of CT courses into three shells of activity – and we wonder whether there is a similar high-level structuring activity here also. For example, there are cognitive and non-cognitive aspects to learning, both general and discipline-specific. These represent core aspects of how learning works, and what can aid and abet successful learning in CS. Then there are teaching mechanisms and approaches used to support learning – again these have both general and discipline-specific aspects. Also, there are aspects that are less close to the core, or even overlap with other areas, for example, historical aspects, society and ethics, and the broad range of professional competencies. Finally, one might argue about whether the professional competencies lie at the core as well, but split apart from those directly related to how learning works.

This breadth and potential structuring allows for both early adoption and steady growth in the longer term of CE provision – with more in-depth topics being available for continuation at graduate level as well. For the early adoption, however, we could provide just one course that captures key elements of learning and of teaching delivery, as appropriate for the context. We hope that this *foot in the door* would, as staffing and attitudes improve, lead to expansion towards a larger programme. Such a structuring, or taxonomy, of the content is left for future work.

The value of the wider stakeholder group was evident in enabling us to reflect on our own views and attitudes. This was apparent when we explored academics' ratings of the importance of the proposed high-level categories for particular target groups against the ratings of those groups towards themselves. That is, for example, comparing what an academic thinks about the importance of a topic to industry against what an industry person says about its importance to industry. We uncovered a number of significant differences here, highlighted in Table 7.

5.3 RQ3 – Current Practices

Our structured reviews resulted in 25 courses described in academic publications and 21 courses presented informally, e.g., on web pages. By far the largest proportion of these courses is focused on tutors, teaching assistants, and mentors. This is perhaps unsurprising as such courses service an urgent practical need in a department. By comparison, the number of courses for non-TA undergraduate students is only half this (11, compared to 20); we note that several of them are exemplars for teacher training courses on undergraduate level and thus representative of a larger body of such courses. We do note a higher proportion of such courses in the informal courses review (7 compared to 9) – but speculate that those who have committed to this working group are also likely to be those who would know informally of courses like this.

Considering the different kinds of course outlines that we identified from these roughly 50 courses, we note the somewhat limited theoretical foundations in the TA training outline. In this, we remember argument interviewee CS-2's comments about how little the TAs engaged with theoretical foundations in their course – where there was no credit for the course and they were keenest to simply learn what was necessary to get the job done – in particular, the development of a rubric and grading practices generally. Underneath this characteristic may be a recognition that – more often than not – TAs are employed with a strong focus on grading (as opposed to teaching) and that TA work thus is not always appreciated other than as a source of income. Also, some courses are taught in a compact way, e.g., as one-week onboarding courses prior to the start of the academic term. More detail on the variation in TA courses is given in [78].

A combination of a credit-bearing course, with appropriate motivation and history of success among students, and subsequent TA work that allows for putting the CE content learned into teaching practice would certainly raise interest in the study of computing education for its own sake, and help to address this avoidance of underlying theory.

We see common components across most of the course outlines, and recognise that in the longer term, this perhaps gives weight to the idea of a common introductory course, or common materials, that can then be followed by specialisations that go into more depth in the particular area. The common introduction provides an opportunity to lay out the wider CE area more fully, giving learners a clear picture of their options while preparing them for more immediate and focused CE activity.

5.4 Sample Courses

The results of the three studies were used to produce two sample courses — see Figures 2 and 3. We see these figures as a way to start both a discussion among all stakeholders and an internal reflection within our research community. For example, the industry-facing sample course in Figure 3 features a high proportion of so-called soft-skills material. In a paper about CE courses, this may seem surprising – but the key is that this course is framed from the perspective of what our expert panel agreed that a software engineering person needs. However, the understanding provided by the CE elements is well suited to develop their soft-skills ability in a industry-specific context. We proposed other arguments for the importance of CE content for industry in our prompt list for interview, such as improved ability to develop documentation and user guides, and work on a help desk, and these are captured in elements of the third component of the sample course. All of this aligns well with Begel and Simon's landmark paper [7] about the task loading of novice software developers who have just graduated and are in the first six months of their first job: over 50% of their time typically was spent on communication and documentation.

An examination of the sample course in Figure 2 prompts consideration of developmental stages and learning frameworks, e.g. [96, 97, 127], and similarly for the course outlines presented in Section 3.3. In both cases, significant exploration of the *content* that computing instructors should be delivering is not evident – instead, the material covered is mainly learning theories and strategies, pedagogy/teaching techniques, and logistics. That is, educational considerations about which subject matter topic should be presented does not stand out. Of course, we cannot expect one, or even a few, CE courses to cover everything that should be taught. But understanding, for example, developmental sequences for introductory computing *is* important. Repeating SS-2's quote in Section 3.1 *"[Computing] people do not reflect. They do not do the meta-cognition, they do not reflect on how they learnt it. So you ask them. It's like trying to get a squirrel to explain centre of mass to you. They demonstrate it all the time, but they cannot articulate it…"*. That is to say, typical instructors in computing are unlikely to have sat through introductory computing classes that made evident a really clear developmental sequence; most are likely to have largely taught themselves. The issue here is that while CE research has explored these aspects, they are not obvious, or well-known, to typical educators and hence should be included in the kinds of courses we espouse. An example of the kind of computing content being referred to here is Schulte's Block Model [102], which is a framework that breaks down the skill of code comprehension (an essential component of the content a novice must learn) into a number of components. Note this is computing content, not learning theory, nor teaching technique.

Having made this observation about the apparent lack of such content, we went back to the TA model in Figure 2 and identified the "Teaching Methods in Computing" component, and then flipped to Table 6 and identified the high-level category "Teaching Methods, Approaches to Teaching Programming". In the concrete examples that had been provided by the Delphi participants, we saw a mix of the kind of content noted here (Program Comprehension — e.g. Block Model — Programming Problem-solving, Reasoning in Programming, and Levels of Abstraction) and also teaching methods or techniques (Pair Programming and PRIMM).

In both cases, the sample courses have provided an opportunity for thoughtful reflection on our understanding of CE and related courses.

5.5 Conclusions and Next Steps

With this work, we have come a step closer to satisfying Denning's three charges – (1) providing a definitional description of CS as a research field on its own, (2) devising a teaching paradigm for CS aligned with standards and quality criteria in other established fields, and (3) outlining how CS could be taught in undergraduate education – but as applied to the curricular embedding of CE, rather than CS. We argued in the introduction that the first of these was already satisfied. For the second, the devising of a teaching paradigm for CE, we have drawn on the literature and identified a number of course structures to suit different targets, and developed two samples of our own. For the third, outlining how CE could be taught, we have sought the opinions of stakeholders via the Delphi process, and also the literature and course materials, in order to identify a wide range of content for CE courses.

This report can be thought of as a kind of manifesto for CE courses in undergraduate programmes, or as source material for such a manifesto, to be used to persuade a university department to adopt CE courses. As such, the original arguments list used to prompt interviewees was ordered by how persuasive we thought the arguments would be with a typical departmental leadership team, most persuasive first. Hence, the potential for CE courses to improve the quality of undergraduate TAs/tutors is at the top, as improved course outcomes and student satisfaction seemed to us a strong motivation for any department. This was followed by arguments pertaining to most or all undergraduates, before moving to advocating for the likelihood of increased teacher recruitment and quality. At the bottom, we noted the likely enhancement to the ecosystem for CE within academia, placing this last because it seemed the most narrowly-focused.

However, and to our surprise, the external support we have received suggests that CE should be recognised on a much wider scale as an essential component of the academic computing infrastructure, both in research and teaching, and also for industry. NTL-1, from a national schools education agency, said *"For me, it seems like everyone would do something like this in an undergraduate course … Your uni would be undertaking effective approaches to learning, teaching, to deliver computer science education. I just thought that was a given."* SS-1, from a national scientific society involved in accreditation, noted that *"if you stand back and think about contributing to the discipline, your professional community, all those sort of things that I talked about in relation to [professional accreditation], then the whole education piece is directly relevant."* (for example, see recent work on CS degree accreditation in the UK [24, 57]). SS-2, from the same society, added *"I welcome this because I think the future of all this is the braided career – and people will move between teaching, academic, and industry."* From an industry perspective, SS-1 added *"obviously, teaching is about translating what you know into consumable chunks [which is like] people communicating effectively with their board. Or with that senior leadership team – where we know there is often a disconnect between the IT team and the leadership of an organisation."*. SS-2 suggested that this might be an effective way of gaining real cultural change in the IT industry: *"because the IT industry is so not diverse and has such an issue particularly around education, then developing those skills you'd get as part of becoming an educator might actually also make the cultural change happen – because of understanding that people are different and learn in different ways. People communicate in different ways. People thrive in different ways. I think [the IT industry] is awakening to the different difficulties – but they're not actually doing anything different."* These quotes all suggest that our well-positioned external interviewees' place a high value on CE.

And hence, in our external stakeholders' eyes, the last argument we made, about the ecosystem, was one of the most important, whereas we had downplayed it. Rather than finding arguments that enable us to get CE in via *the side door*, we should be embracing and promoting CE as at least as important as other areas. Those external to us are seeing the importance of CE, so we, that is, the larger discipline of CS, should be also. We speculate that inertia alone is keeping us both small and hidden, behind historic academic, educational and financial walls: from an academic point of view, we are not universally accepted within either CS or Education departments; educationally, in many CS departments, successful CE academics' research is tolerated but not embraced amid an expectation of high teaching loads compared to other computing academics, or else CE is not accepted at all; and financially, funding for CE research is highly variable – for example, while the US seems to have a reasonable funding regime, the UK has almost none.

Hence, our next steps may call on a level of evangelism. One route is to show the success of adopting course models such as we have shown. Indeed, two of the authors already teach TA/mentor courses and another, the broader introduction to CE style of course we espouse; and three of the authors are working together to introduce instances of this kind of course in the coming year, including theoretical foundations, practical elements, and opportunities for research preparation and activity. With some experience under our belts, and course outlines and materials on offer, we may then offer a multi-institutional, multi-national experience for, say, 10–15 academics to work together over a two-year period to deliver these courses, via the UKICER conference's RIPPA programme [29]. These activities will enable us to address Clements's research-based evaluation phases 6–10 [19], introduced in section 1, and any findings can be fed into our on-going understanding of CE courses, as encouraged by Clements' curriculum development framework. Furthermore, as suggested by some of our interviewees, identifying core elements everyone should be learning and starting to characterise a subject-specific pedagogy would significantly contribute to the acceptance of CE in the context(s) of undergraduate CS programmes. Perhaps these types of changes will be increasingly feasible and tractable in a world in which we are trying to better understand the emerging impact and ramifications of generative AI [41, 91] as part of a post-COVID "new normal" [23, 27], where there is a renewed focus on high-quality learning, teaching and assessment, and specifically what this means for effective pedagogy and practice in CS and CE [106, 107]. Ultimately, we hope that an open international community of practice will form around this work.

ACKNOWLEDGMENTS

We thank the interviewees who discussed our arguments with us, and the Delphi participants for the repeated contributions to the development of our content list. Their comments have given us rich insights into the whole area, going beyond our own. And we thank the ITiCSE Steering Committee, Organising Committee and Working Group chairs for their support of the ITiCSE working groups scheme, which is so valuable to our community.

REFERENCES

[1] Alliance for Identity Inclusive Computing Education. 2023. *AIICE Teaching Assistant Professional Development Course.* Alliance for Identity Inclusive Computing Education. https://sites.google.com/mtholyoke.edu/aiice-ta-pd-info-site

[2] Ruth Anderson and Justin Hsia. 2023. *CSE General TA Training.* University of Washington. https://courses.cs.washington.edu/courses/ta-training/

[3] Robert B. Archibald and David H. Feldman. 2010. *Why Does College Cost So Much?* Oxford University Press, New York, NY. https://doi.org/10.1093/acprof:oso/9780199744503.001.0001

[4] Michael Ball and Nick Weaver. 2023. *Designing Computer Science Education.* University of California Berkeley. https://cs302.org/sp23/

[5] Michael Barrow, Shelby Thomas, and Christine Alvarado. 2016. ERSP: A structured CS research program for early-college students. In *ITiCSE '16: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, Janet Carter and Yván Túpac (Eds.). ACM, New York, NY, 148–153. https://doi.org/10.1145/2899415.2899436

[6] Brett A. Becker, Steven Bradley, Joseph Maguire, Michaela Black, Tom Crick, Mohammed Saqr, Sue Sentance, and Keith Quille. 2023. Computing Education Research in the UK & Ireland. In *Past, Present and Future of Computing Education Research: A Global Perspective*, Mikko Apiola, Sonsoles López-Pernas, and Mohammed Saqr (Eds.). Springer, 421–479. https://doi.org/10.1007/978-3-031-25336-2_19

[7] Andrew Begel and Beth Simon. 2008. Novice Software Developers, All over Again. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08)*. Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/1404520.1404522

[8] Tim Bell and Lynn Lambert. 2011. Teaching computer science majors about teaching computer science. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, SIGCSE 2011*, Thomas J. Cortina, Ellen Lowenfeld Walker, Laurie A. Smith King, and David R. Musicant (Eds.). ACM Press, New York, NY, 541–546. https://doi.org/10.1145/1953163.1953317

[9] Anders Berglund, Päivi Kinnunen, and Lauri Malmi. 2007. A Doctoral Course in Research Methods in Computing Education Research How Should We Teach It?. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88* (Koli National Park, Finland) *(Koli Calling '07)*. Australian Computer Society, Inc., AUS, 175–178.

[10] John D. Bransford, Ann L. Brown, and Rodney R. Cocking. 2000. *How People Learn: Brain, Mind, Experience, and School – Expanded Edition.* National Academy Press, Washington, D.C. https://doi.org/10.17226/9853 National Research Council: Committee on Developments in the Science of Learning.

[11] David Brown. 2020. A Review of the PubMed PICO Tool: Using Evidence-Based Practice in Health Education. *Health Promotion Practice* 21, 2 (July 2020), 496–498. https://doi.org/10.1177/1524839919893361

[12] Paul Bruno and Colleen M. Lewis. 2022. Computer Science Trends and Trade-offs in California High Schools. *Educational Administration Quarterly* 58, 3 (2022), 386–418. https://doi.org/10.1177/0013161X211054801

[13] Deirdre Butler and Margaret Leahy. 2021. Developing preservice teachers' understanding of computational thinking: A constructionist approach. *British Journal of Educational Technology* 52, 3 (May 2021), 1060–1077. https://doi.org/10.1111/bjet.13090

[14] Patrina N. Y. Caldwell, Trish Bennett, and Craig Mellis. 2012. Easy guide to searching for evidence for the busy clinician. *Journal of Paediatrics and Child Health* 48, 12 (Dec. 2012), 1095–1100. https://doi.org/10.1111/j.1440-1754.2012.02503.x

[15] Michel Callon. 1984. Some Elements of a Sociology of Translation: Domestication of the Scallops and the Fishermen of St Brieuc Bay. *The Sociological Review* 32, 1_suppl (1984), 196–233. https://doi.org/10.1111/j.1467-954X.1984.tb00113.x

[16] Michael Carroll. 2018. Understanding curriculum: An Actor Network Theory approach. *Studies in Self-Access Learning Journal* 9, 3 (2018), 247–261. https://doi.org/10.37237/090302

[17] Michael E. Caspersen, Judith Gal-Ezer, Andrew McGettrick, and Enrico Nardelli. 62. Informatics as a Fundamental Discipline for the 21st Century. *Commun. ACM* 4, 2019 (April 62), 58–63. https://doi.org/10.1145/3310330

[18] Mark J. Clayton. 1997. Delphi: A Technique to Harness Expert Opinion for Critical Decision-Making Tasks in Education. *Educational Psychology* 17, 4 (1997), 373–386. https://doi.org/10.1080/0144341970170401

[19] Douglas H. Clements. 2007. Curriculum Research: Toward a Framework for "Research-Based Curricula". *Journal for Research in Mathematics Education* 38, 1 (2007), 35–70. https://doi.org/10.2307/30034927

[20] Gülşah Coşkun Yaşar and Berna Aslan. 2021. Curriculum Theory: A Review Study. *International Journal of Curriculum and Instructional Studies (IJOCIS)* 11, 2 (2021), 237–260. https://doi.org/10.31704/ijocis.2021.012

[21] Juliet M. Corbin and Anselm Strauss. 1990. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology* 13 (1990), 3–21. https://doi.org/10.1007/BF00988593

[22] James Cowan, Dan Goldhaber, Kyle Hayes, and Roddy Theobald. 2016. Missing Elements in the Discussion of Teacher Shortages. *Educational Researcher* 45, 8 (2016), 460–462. https://doi.org/10.3102/0013189X16679145

[23] Tom Crick. 2021. COVID-19 and Digital Education: A Catalyst for Change? *ITNOW* 63, 1 (2021). https://doi.org/10.1093/itnow/bwab005

[24] Tom Crick, James H. Davenport, Paul Hanna, Alastair Irons, and Tom Prickett. 2020. Computer Science Degree Accreditation in the UK: A Post-Shadbolt Review Update. In *Proceedings of Computing Education Practice (CEP'20)*. ACM, Article 6, 4 pages. https://doi.org/10.1145/3372356.3372362

[25] Tom Crick, James H. Davenport, and Alan Hayes. 2015. Innovative Pedagogical Practices in the Craft of Computing. Advance HE. https://www.advance-he.ac.uk/knowledge-hub/innovative-pedagogical-practices-craft-computing.

[26] Tom Crick, James H. Davenport, Alan Hayes, and Tom Prickett. 2023. Teaching Programming Competencies: A Role for Craft Computing?. In *Proceedings of United Kingdom and Ireland Computing Education Research Conference (UKICER'23)*. ACM. https://doi.org/10.1145/3610969.3611140

[27] Tom Crick, Cathryn Knight, Richard Watermeyer, and Janet Goodall. 2021. The International Impact of COVID-19 and "Emergency Remote Teaching" on Computer Science Education Practitioners. In *Proceedings of IEEE Global Engineering Education Conference (EDUCON'21)*. IEEE, 1048–1055. https://doi.org/10.1109/EDUCON46332.2021.9453846

[28] Quintin Cutts, Maria Kallia, Ruth Anderson, Tom Crick, Marie Devlin, Mohammed Farghally, Claudio Mirolo, Ragnhild Kobro Runde, Otto Seppälä, Jaime Urquiza-Fuentes, and Jan Vahrenhold. 2023. Considering Computing Education in Undergraduate Computer Science Programmes. In *Proceedings of 28th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'23)*. ACM. https://doi.org/10.1145/3587103.3594210

[29] Quintin Cutts, Joseph Maguire, Sally Fincher, and Jack Parkinson. 2021. Forming Community in Computing Science Education with Research in Practice Project Activities. In *Proceedings of the 2021 Conference on United Kingdom & Ireland Computing Education Research* (Glasgow, United Kingdom) *(UKICER '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 3 pages. https://doi.org/10.1145/3481282.3481285

[30] Holger Danielsiek, Jan Vahrenhold, Peter Hubwieser, Johannes Krugel, Johannes Magenheim, Laura Ohrndorf, Daniel Ossenschmidt, and Niclas Schaper. 2017. Undergraduate Teaching Assistants in Computer Science: Teaching-Related Beliefs, Tasks, and Competences. In *2017 IEEE Global Engineering Education Conference (EDUCON)*, Christos Douligeris, Michael E. Auer, Cleo Sgouropoulou, Christian M. Stracke, and Michalis Xenos (Eds.). IEEE Computer Society, Los Alamitos, CA, 12–17. https://doi.org/10.1109/EDUCON.2017.7942927

[31] James H. Davenport, Tom Crick, and Rachid Hourizi. 2020. The Institute of Coding: A University-Industry Collaboration to Address the UK's Digital Skills Crisis. In *Proceedings of IEEE Global Engineering Education Conference (EDUCON'20)*. IEEE, 1400–1408. https://doi.org/10.1109/EDUCON45650.2020.9125272

[32] Renzo Davoli and Michael Lodi. 2022. *Teaching Tools for Informatics.* Università degli Studi di Bologna. https://www.unibo.it/en/teaching/course-unit-catalogue/course-unit/2022/479040/

[33] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Markus Hendrik Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer, Berlin. https://doi.org/10.1007/978-3-540-77974-2

[34] Leigh Ann Delyser, Joanna Goode, Mark Guzdial, Yasmin Kafai, and Aman Yadav. 2018. *Priming the Computer Science Teacher Pump: Integrating Computer Science Education into Schools of Education.* Technical Report. CSforALL.

[35] Peter J. Denning, Douglas Earl Comer, David Gries, Micheal C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. 1989. Computing as a Discipline. *Commun. ACM* 32, 1 (Jan. 1989), 9–23. https://doi.org/10.1145/63238.63239

[36] Monica Divitini and Birgit Rognebakke Krogstie. 2018. *Research in Computing Education.* Norwegian University of Science and Technology. https://www.ntnu.edu/studies/courses/IT8007/2018/1

[37] Brian Dorn and Mark Guzdial. 2010. Learning on the Job: Characterizing the Programming Knowledge and Learning Strategies of Web Designers. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Atlanta, Georgia, USA) *(CHI '10)*. Association for Computing Machinery, New York, NY, USA, 703––712. https://doi.org/10.1145/1753326.1753430

[38] Thomas Eckhardt (Ed.). 2021. *The Education System in the Federal Republic of Germany 2018/2019.* Secretariat of the Standing Conference of the Ministers of Education and Cultural Affairs of the Länder in the Federal Republic of Germany, Bonn. `https://www.kmk.org/fileadmin/Dateien/pdf/Eurydice/Bildungswesen-engl-pdfs/dossier_en_ebook.pdf`.

[39] John Edwards. 2023. *Computing Education Research.* Utah State University. https://www.coursicle.com/usu/courses/CS/5750/

[40] Francisco J. Estrada and Anya Tafliovich. 2017. Bridging the Gap Between Desired and Actual Qualifications of Teaching Assistants: An Experience Report. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in*

Computer Science Education, ITiCSE 2017, Renzo Davoli, Michael Goldweber, Guido Rößling, and Irene Polycarpou (Eds.). ACM, New York, NY, 134–139. https://doi.org/10.1145/3059009.3059023

[41] Yogesh K. Dwivedi et al. 2023. "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management* 71, 102642 (2023). https://doi.org/10.1016/j.ijinfomgt.2023.102642

[42] Peter J. Fensham. 2004. *Defining an Identity: The Evolution of Science Education as a Field of Research.* Springer Science+Business Media, Dordrecht, NL. https://doi.org/10.1007/978-94-010-0175-5

[43] Sally A. Fincher and Anthony V. Robins (Eds.). 2019. *The Cambridge Handbook of Computing Education Research.* Cambridge University Press, Cambridge. https://doi.org/10.1017/9781108654555

[44] Terry J. Frederick. 1975. Computer Science Education for Students Training to Be Secondary Teachers. *SIGCUE Outlook* 9, SI (jan 1975), 10–14. https://doi.org/10.1145/952845.952849

[45] Judith Gal-Ezer and David Harel. 1998. What (Else) Should CS Educators Know? *Commun. ACM* 41, 9 (sep 1998), 77–84. https://doi.org/10.1145/285070.285085

[46] Judith Gal-Ezer and Chris Stephenson. 2014. A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States. *ACM Transactions on Computing Education* 14, 2, Article 8 (2014), 18 pages. https://doi.org/10.1145/2602483

[47] Jacob Eli Goodman and Joseph O'Rourke (Eds.). 2004. *Handbook of Discrete and Computational Geometry* (2nd ed.). CRC Press, Boca Raton, FL. https://doi.org/10.1201/9781420035315

[48] Shuchi Grover (Ed.). 2020. *Computer Science in K–12: An A-To-Z Handbook on Teaching Programming.* Edfinity, Austin, TX.

[49] Michael Gusenbauer and Neal R. Haddaway. 2020. Which academic search systems are suitable for systematic reviews or meta-analyses? Evaluating retrieval qualities of Google Scholar, PubMed, and 26 other resources. *Research Synthesis Methods* 11, 2 (March 2020), 181–217. https://doi.org/10.1002/jrsm.1378

[50] Mark Guzdial, Barbara Ericson, Tom Mcklin, and Shelly Engelman. 2014. Georgia computes! An intervention in a US state, with formal and informal education in a policy context. *ACM Transactions on Computing Education (TOCE)* 14, 2, Article 13 (June 2014), 29 pages. https://doi.org/10.1145/2602488

[51] Philip Hallinger. 2011. Leadership for learning: lessons from 40 years of empirical research. *Journal of Educational Administration* 49, 2 (2011), 125–142. https://doi.org/10.1108/09578231111116699

[52] Orit Hazzan, Tami Lapidot, and Noa Ragonis. 2020. *Guide to Teaching Computer Science: An Activity-Based Approach.* Springer, London. https://doi.org/10.1007/978-3-030-39360-1

[53] Dede Heidt and James Poirot. 1988. Implementing a University Level Computer Education Course for Preservice Teachers. *SIGCUE Outlook* 20, 1 (sep 1988), 141–145. https://doi.org/10.1145/382236.382866

[54] Victor Huang and Armando Fox. 2022. *Teaching Techniques for Computer Science.* University of California Berkeley. https://bcourses.berkeley.edu/courses/1516384/

[55] Peter Hubwieser. 2007. *Didaktik der Informatik: Grundlagen, Konzepte, Beispiele* (3rd ed.). Springer, Berlin. https://doi.org/10.1007/978-3-540-72478-0 In German..

[56] C. Jinshong Hwang, Gerald Kulm, and Grayson H. Wheatley. 1981. Computing Education for Secondary School Teachers: A Cooperative Effort between Computer Scientist and Educators. In *Proceedings of the Twelfth SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA) *(SIGCSE '81)*. Association for Computing Machinery, New York, NY, USA, 257–261. https://doi.org/10.1145/800037.800998

[57] Alastair Irons, Tom Crick, James H. Davenport, Paul Hanna, and Tom Prickett. 2021. Increasing the Value of Professional Body Computer Science Degree Accreditation. In *Proceedings of 52nd ACM Technical Symposium on Computer Science Education (SIGCSE'21).* ACM. https://doi.org/10.1145/3408877.3439678

[58] Lisa C. Kaczmarczyk. 2015. The Case for Teaching Computer Science Education Research to Undergraduates. *ACM Inroads* 6, 1 (feb 2015), 32–33. https://doi.org/10.1145/2700443

[59] Yasmin Kafai, Chris Proctor, and Debora Lui. 2019. From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 101–109. https://doi.org/10.1145/3291279.3339400

[60] Ayaan Kazerouni. 2023. *Computing Education Research and Practice.* California Polytechnic State University. https://ayaankazerouni.github.io/courses/csc513/

[61] Ayaan Kazerouni. 2023. *Teaching Computing.* California Polytechnic State University. https://ayaankazerouni.github.io/courses/csc313/

[62] Maria Knobelsdorf, Johannes Magenheim, Torsten Brinda, Dieter Engbring, Ludger Humbert, Arno Pasternak, Ulrik Schroeder, Marco Thomas, and Jan Vahrenhold. 2015. Computer Science Education in North-Rhine Westphalia, Germany – A Case Study. *ACM Transactions on Computing Education* 15, 2 (April 2015), 9:1–9:22. https://doi.org/10.1145/2716313

[63] Tami Lapidot and Orit Hazzan. 2003. Methods of Teaching a Computer Science Course for Prospective Teachers. *SIGCSE Bull.* 35, 4 (dec 2003), 29–34. https://doi.org/10.1145/960492.960520

[64] Bruno Latour. 2005. *Reassembling the Social: An Introduction to Actor-Network-Theory.* Oxford University Press, Oxford, UK. https://sciencespo.hal.science/hal-02057191

[65] Diana Cheng-Man Lau. 2001. Analysing the curriculum development process: three models. *Pedagogy, Culture & Society* 9, 1 (2001), 29–44. https://doi.org/10.1080/14681360100200107

[66] Violetta Lonati and Anna Chiara Giovanna Morpurgo. 2023. *Computing Education.* Università degli Studi di Milano. https://www.unimi.it/en/education/degree-programme-courses/2023/computing-education/

[67] Madeleine Lorås. 2020. From teaching assistants to learning assistants–lessons learned from learning assistant training at Excited. *Læring om læring* 4, 1 (2020), 7 pages.

[68] Ding Luo, Matthew W. Shuman, and Donald Heer. 2010. Work in progress – Leadership training for new EECS graduate teaching assistants. In *2010 IEEE Frontiers in Education Conference (FIE).* IEEE Computer Society, Los Alamitos, CA, S3D–1–S3D–2. https://doi.org/10.1109/FIE.2010.5673214

[69] Andrew Luxton-Reilly and Paul Denny. 2023. *Computing Education.* University of Auckland. https://courseoutline.auckland.ac.nz/dco/course/COMPSCI/747/1233/

[70] R. Adam Manley. 2013. The policy Delphi: a method for identifying intended and unintended consequences of educational policy. *Policy Futures in Education* 11, 6 (2013), 755–768. https://doi.org/10.2304/pfie.2013.11.6.755

[71] Melanie Margaritis, Johannes Magenheim, Peter Hubwieser, Marc Berges, Laura Ohrndorf, and Sigrid Schubert. 2015. Development of a competency model for computer science teachers at secondary school level. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, Danilo Garbi Zutin (Ed.). Curran Associates, Red Hook, NY, 211–220. https://doi.org/10.1109/EDUCON.2015.7095973

[72] Sharon Mason and Elissa Weeden. 2022. Chronicling the Development of a Growth Mindset Community of Practice for Computing Faculty: Lessons Learned and Looking Forward. In *2022 IEEE Frontiers in Education Conference (FIE).* IEEE, Curran Associates, Red Hook, NY, 9 pages. https://doi.org/10.1109/FIE56618.2022.9962535

[73] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. 2019. Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW, Article 72 (nov 2019), 23 pages. https://doi.org/10.1145/3359174

[74] Mia Minnes. 2022. Designing TA Training for Computer Science Graduate Students: Remote and Self-paced Options for A Supported Introduction to Reflective Teaching. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education – Volume 1*, Judithe Sheard, Leen-Kiat Soh, and Brian Dorn (Eds.). ACM Press, New York, NY, 752–758. https://doi.org/10.1145/3478431.3499342

[75] Mia Minnes. 2022. *Teaching Methods in Computer Science.* University of California San Diego. https://cseweb.ucsd.edu/~minnes/cse599/

[76] Mia Minnes. 2023. *Tutor Training in Computer Science.* University of California San Diego. https://canvas.ucsd.edu/courses/44390/

[77] Claudio Mirolo. 2022. *Computer Science Education.* Università degli Studi di Udine. https://www.dmif.uniud.it/en/master/computer-science/study-plan/computer-science-education/

[78] Diba Mirza, Phillip T. Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review. In *Proceedings of the 2019 ACM Conference on International Computing Education Research, ICER 2019*, Robert McCartney, Andrew Petersen, Anthony V. Robins, and Adon Moskal (Eds.). ACM Press, New York, NY, 31–40. https://doi.org/10.1145/3291279.3339422

[79] Peter Moulton and David Moursund. 1975. A Summer Master's Degree Program in Computer Education. *SIGCUE Outlook* 9, SI (jan 1975), 31–36. https://doi.org/10.1145/952845.952855

[80] David Moursund. 1978. Computer Science Education for Preservice Elementary School Teachers. *SIGCUE Outlook* 12, 4 (oct 1978), 3–10. https://doi.org/10.1145/964047.964048

[81] Chrystalla Mouza, Scott Sheridan, Nancy C. Lavigne, and Lori Pollock. 2023. Preparing undergraduate students to support K–12 computer science teaching through school-university partnerships: reflections from the field. *Computer Science Education* 33, 1 (2023), 3–28. https://doi.org/10.1080/08993408.2021.1970435

[82] Felix Muzny. 2023. *TA Training Modules.* Northeastern University. https://cic.northeastern.edu/ta-training/

[83] Felix Muzny and Michael D. Shah. 2023. Teaching Assistant Training: An Adjustable Curriculum for Computing Disciplines. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, Judithe Sheard, Leen-Kiat Soh, and Brian Dorn (Eds.). ACM Press, New York, NY, 430–436. https://doi.org/10.1145/3545945.3569866

[84] Brad A. Myers, Amy J. Ko, Thomas D. LaToza, and YoungSeok Yoon. 2019. *Human-Centered Methods to Boost Productivity.* Apress, Berkeley, CA, 147–157. https://doi.org/10.1007/978-1-4842-4221-6_13

[85] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, Roger Chou, Julie Glanville, Jeremy M. Grimshaw, Asbjørn Hróbjartsson, Manoj M. Lalu, Tianjing Li, Elizabeth W. Loder, Evan Mayo-Wilson, Steve McDonald, Luke A. McGuinness, Lesley A. Stewart, James Thomas, Andrea C. Tricco, Vivian A. Welch, Penny Whiting, and David Moher. 2021. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *The British Medical Journal* 373, 8286 (2021), 1115–1117. https://doi.org/10.1136/bmj.n71

[86] J. L. Poirot. 1976. A Course Description for Teacher Education in Computer Science. In *Proceedings of the ACM SIGCSE-SIGCUE Technical Symposium on Computer Science and Education (SIGCSE '76).* Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/800107.803446

[87] Lori Pollock, Chrystalla Mouza, James Atlas, and Terry Harvey. 2015. Field experiences in teaching computer science: Course organization and reflections. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, Adrienne Decker, Kurt Eiselt, Carl Alphonce, and Jodi Tims (Eds.). ACM Press, New York, NY, 374–379. https://doi.org/10.1145/2676723.2677286

[88] Heather Pon-Barry, Becky Wai-Ling Packard, and Audrey St. John. 2017. Expanding capacity and promoting inclusion in introductory computer science: a focus on near-peer mentor preparation and code review. *Computer Science Education* 27, 1 (2017), 54–77. https://doi.org/10.1080/08993408.2017.1333270

[89] Heather Pon-Barry, Audrey St. John, Becky Wai-Ling Packard, and Barbara Rotundo. 2019. A flexible curriculum for promoting inclusion through peer mentorship. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, Elizabeth K. Hawthorne, Manuel A. Pérez-Quiñones, Sarah Heckmann, and Jian Zhang (Eds.). ACM Press, New York, NY, 1116–1122. https://doi.org/10.1145/3287324.3287434

[90] Heather Pon-Barry, Audrey St. John, Becky Wai-Ling Packard, and Barbara Rotundo. 2023. *MaGE (Megas and Gigas Educate) Training Course.* Mount Holyoke College. https://sites.google.com/a/mtholyoke.edu/mage-training/

[91] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michael E. Caspersen, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. Transformed by Transformers: Navigating the AI Coding Revolution for Computing Education: An ITiCSE Working Group Conducted by Humans. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023).* 561–562. https://doi.org/10.1145/3587103.3594206

[92] Beth A. Quinn, Wendy M. DuBow, and Jamie Huber Ward. 2018. Broadening participation in computing via professional development for community college CS/IT faculty. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Tiffany Barnes, Daniel D. Garcia, Elizabeth K. Hawthorne, and Manuel A. Pérez-Quiñones (Eds.). ACM Press, New York, NY, 789–793. https://doi.org/10.1145/3159450.3159546

[93] Noa Ragonis and Orit Hazzan. 2008. Disciplinary-Pedagogical Teacher Preparation for Pre-Service Computer Science Teachers: Rational and Implementation. In *Proceedings of the 3rd International Conference on Informatics in Secondary Schools - Evolution and Perspectives: Informatics Education - Supporting Computational Thinking* (Torun, Poland) *(ISSEP '08).* Springer-Verlag, Berlin, Heidelberg, 253–264. https://doi.org/10.1007/978-3-540-69924-8_23

[94] Stuart Reges, John McGrory, and Jeff Smith. 1988. The Effective Use of Undergraduates to Staff Large Introductory CS Courses. In *Proceedings of the Nineteenth SIGCSE Technical Symposium on Computer Science Education* (Atlanta, Georgia, USA) *(SIGCSE '88).* Association for Computing Machinery, New York, NY, USA, 22–25. https://doi.org/10.1145/52964.52971

[95] Alexander Repenning, Anna Lamprou, and Ashok R. Basawapatna. 2021. Computing effect sizes of a science-first-then-didactics computational thinking module for preservice elementary school teachers. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, Pamela Cutter, Alvaro Monge, and Jusy Sheard (Eds.). ACM Press, New York, NY, 274–280. https://doi.org/10.1145/3408877.3432446

[96] Kathryn M. Rich, T. Andrew Binkowski, Carla Strickland, and Diana Franklin. 2018. Decomposition: A K–8 Computational Thinking Learning Trajectory. In *Proceedings of the 2018 ACM Conference on International Computing Education Research, ICER 2018*, Lauri Malmi, Ari Korhonen, Robert McCartney, and Andrew Petersen (Eds.). ACM Press, New York, NY, 124–132. https://doi.org/10.1145/3230977.3230973

[97] Kathryn M. Rich, Carla Strickland, T. Andrew Binkowski, Cheryl Moran, and Diana Franklin. 2017. K–8 Learning Trajectories Derived from Research Literature: Sequence, Repetition, Conditionals. In *Proceedings of the 2017 ACM Conference on International Computing Education Research, ICER 2017*, Josh Tenenberg, Donald Chinn, Judy Sheard, and Lauri Malmi (Eds.). ACM Press, New York, NY, 182–190. https://doi.org/10.1145/3105726.3106166

[98] Emma Riese and Viggo Kann. 2022. Training teaching assistants by offering an introductory course. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education – Volume 1*, Judithe Sheard, Leen-Kiat Soh, and Brian Dorn (Eds.). ACM Press, New York, NY, 745–751. https://doi.org/10.1145/3478431.3499270

[99] Judy Robertson. 2019. *Towards a Sustainable Solution for the Shortage of Computing Teachers in Scotland.* Commissioned report. University of Edinburgh, Edinburgh.

[100] Mehram Sahami, Alex Aiken, and Julie Zelinski. 2010. Expanding the Frontiers of Computer Science: Designing a Curriculum to Reflect a Diverse Field. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education, SIGCSE 2010*, Gary Lewandowski, Steven A. Wolfman, Thomas J. Cortina, and Ellen Lowenfeld Walker (Eds.). ACM Press, New York, NY, 47–51. https://doi.org/10.1145/1734263.1734279

[101] Mehran Sahami, Andrea Danyluk, Sally Fincher, Kathleen Fisher, Dan Grossman, Elizabeth Hawthorne, Randy Katz, Rich LeBlanc, Dave Reed, Steve Roach, Ernesto Cuadros-Vargas, Ronald Dodge, Robert France, Amruth Kumar, Brian Robinson, Remzi Seker, and Alfred Thompson. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM Press, New York, NY. https://doi.org/10.1145/2534860

[102] Carsten Schulte. 2008. Block Model: An Educational Model of Program Comprehension as a Tool for a Scholarly Approach to Teaching. In *Proceedings of the Fourth International Workshop on Computing Education Research* (Sydney, Australia) *(ICER '08).* Association for Computing Machinery, New York, NY, USA, 149–160. https://doi.org/10.1145/1404520.1404535

[103] Sue Sentance, Erik Barendsen, Nichol R. Howard, and Carsten Schulte (Eds.). 2023. *Computer Science Education: Perspectives on Teaching and Learning in School* (2nd ed.). Bloomsbury Academic, London.

[104] Sue Sentance and Andrew Csizmadia. 2017. Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies* 22, 2 (March 2017), 469–495. https://doi.org/10.1007/s10639-016-9482-0

[105] Lee S. Shulman. 1986. Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher* 15, 2 (Feb. 1986), 4–14. https://doi.org/10.3102/0013189X015002004

[106] Angela A. Siegel, Mark Zarb, Bedour Alshaigy, Jeremiah Blanchard, Tom Crick, Richard Glassey, John R. Holt, Celine Latulipe, Charles Riedesel, Mali Senapathi, Simon, and David Williams. 2021. Teaching through a Global Pandemic: Educational Landscapes Before, During and After COVID-19. In *Proceedings of ITiCSE-WGR'21.* 1–25. https://doi.org/10.1145/3502870.3506565

[107] Angela A. Siegel, Mark Zarb, Emma Anderson, Brent Crane, Alice Gao, Celine Latulipe, Ellie Lovellette, Fiona McNeill, and Debbie Meharg. 2022. The Impact of COVID-19 on the CS Student Learning Experience: How the Pandemic Has Shaped the Educational Landscape. In *Proceedings of ITiCSE-WGR'22.* 165–190. https://doi.org/10.1145/3571785.3574126

[108] Simon. 2015. *Emergence of computing education as a research discipline.* Ph. D. Dissertation. School of Science, Aalto University. http://urn.fi/URN:ISBN:978-952-60-6416-1

[109] Gerald Soosairaj. 2023. *Intro to Computing Education Research.* University of California San Diego. https://sites.google.com/ucsd.edu/cse291-cer-spring2023/

[110] Kristin Stephens-Martinez. 2022. *Computing Education Research.* Duke University. https://sites.duke.edu/compsci290cerfa2022/

[111] Margaret-Anne Storey, Christoph Treude, Arie van Deursen, and Li-Te Cheng. 2010. The Impact of Social Media on Software Engineering Practices and Tools. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (Santa Fe, New Mexico, USA) *(FoSER '10).* Association for Computing Machinery, New York, NY, USA, 359––364. https://doi.org/10.1145/1882362.1882435

[112] Kelvin Sung, Karen Gourd, Ann McMahon, Kulsoom Mansoor, and Riley Gaggero. 2018. A collaborative course for learning how to teach summer Java coding camps. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Tiffany Barnes, Daniel D. Garcia, Elizabeth K. Hawthorne, and Manuel A. Pérez-Quiñones (Eds.). ACM Press, New York, NY, 515–520. https://doi.org/10.1145/3159450.3159610

[113] Leib Sutcher, Linda Darling-Hammond, and Desiree Carver-Thomas. 2016. *A Coming Crisis in Teaching? Teacher Supply, Demand, and Shortages in the U.S.* Technical Report. Learning Policy Institute, Palo Alto, CA. https://doi.org/10.54300/247.242.

[114] Harriet G. Taylor and James L. Poirot. 1984. A Proposed Computer Education Curriculum for Secondary School Teachers. In *Proceedings of the Fifteenth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '84).* Association for Computing Machinery, New York, NY, USA, 115–118. https://doi.org/10.1145/800039.808633

[115] Matti Tedre. 2013. Methodology education in computing: Towards a congruent design approach. In *The 44th ACM Technical Symposium on Computer Science Education, SIGCSE '13*, Tracy Camp, Paul T. Tymann, J. D. Dougherty, and Kris Nagel (Eds.). ACM Press, New York, NY, 159–164. https://doi.org/10.1145/2445196.2445246

[116] Matti Tedre, Simon, and Lauri Malmi. 2018. Changing aims of computing education: a historical survey. *Computer Science Education* 28, 2 (2018), 158–186.

https://doi.org/10.1080/08993408.2018.1486624

[117] The White House: Office of the Press Secretary. 2016. Fact Sheet: New Progress and Momentum in Support of President Obama's Computer Science for All Initiative. https://obamawhitehouse.archives.gov/the-press-office/2016/09/14/fact-sheet-new-progress-and-momentum-support-president-obamas-computer

[118] James E. Tomayko. 1998. Forging a Discipline: An Outline History of Software Engineering Education. *Annals of Software Engineering* 6, 1–4 (March 1998), 3–18. https://doi.org/10.1023/A:1018953214201

[119] Murray Turoff. 1970. The Design of a Policy Delphi. *Technological Forecasting and Social Change* 2 (1970), 149–171. https://doi.org/10.1016/0040-1625(70)90161-7

[120] Luther Tychonievich. 2021. *TA Practicum*. University of Virginia. https://www.cs.virginia.edu/luther/2910/F2021/

[121] Martin Ukrop, Valdemar Švábenský, and Imrich Nagy. 2020. Teaching lab: Training novice computer science teachers. In *ITiCSE '20: Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, Andrew Luxton-Reilly and Monica Divitini (Eds.). ACM, New York, NY, 561. https://doi.org/10.1145/3341525.3393967

[122] Jan Vahrenhold. 2020. *KETTI – Competence Development of Student Teaching Assistants in Computer Science*. University of Münster. https://www.uni-muenster.de/Ketti/en/

[123] Maarten van Veen, Fred Mulder, and Karel Lemmen. 2004. What is Lacking in Curriculum Schemes for Computing/Informatics?. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Leeds, United Kingdom) *(ITiCSE '04)*. Association for Computing Machinery, New York, NY, USA, 186–190. https://doi.org/10.1145/1007996.1008046

[124] Colin C. Venters, Rafael Capilla, Stefanie Betz, Birgit Penzenstadler, Tom Crick, Steve Crouch, Elisa Yumi Nakagawa, Christoph Becker, and Carlos Carrillo. 2018. Software Sustainability: Research and Practice from a Software Architecture Viewpoint. *Journal of Systems and Software* 138 (2018), 174–188. https://doi.org/10.1016/j.jss.2017.12.026

[125] Colin C. Venters, Rafael Capilla, Elisa Yumi Nakagawa, Stefanie Betz, Birgit Penzenstadler, Tom Crick, and Ian Brooks. 2023. Sustainable Software Engineering: Reflections on Advances in Research and Practice. *Information and Software Technology* 164 (2023), 107316. https://doi.org/10.1016/j.infsof.2023.107316

[126] Brett Wortzman and Kevin Lin. 2022. *Equitable and Inclusive CS Pedagogy*. University of Washington. https://courses.cs.washington.edu/courses/cse492t/

[127] Benjamin Xie, Dastyni Loksa, Greg L. Nelson, Matthew J. Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Amy J. Ko. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 29, 2-3 (2019), 205–253. https://doi.org/10.1080/08993408.2019.1565235 arXiv:https://doi.org/10.1080/08993408.2019.1565235

[128] Stelios Xinogalos. 2022. Designing, Deploying and Evaluating an Undergraduate Course on the "Didactics of Informatics". In *International Conference on Technology and Innovation in Learning, Teaching and Education (Communications in Computer and Information Science, Vol. 1720)*, Arsénio Reis, João Barroso, Paulo Martins, Athanassios Jimoyiannis, Ray Yueh-Min Huang, and Roberto Henriques (Eds.). Springer, Cham, 83–99. https://doi.org/10.1007/978-3-031-22918-3_7

A DELPHI RESULTS

The tables in this appendix report the final results (after the third round) of the Delphi process in terms of rate of agreement on the importance of the 32 topics identified in sect. 3.2. Participants were asked to rate the importance of these topics for each of the (future) roles of undergraduate students: tutors/teaching assistants (TA), teachers/faculty (TT), researchers (R), industry (I), and general undergraduate students (UG) on a 5-point Likert scale from 1 (not important) to 5 (very important). The tables below show the percentage of overall agreement/disagreement on the importance of each theme (ratings of 1–2 were counted as "disagree" and ratings of 4–5 were counted as "agree") as well as the Mean, Median and Mode.

A.1 Categories listed in Table 6 apart from "Professional Competencies"

Q1: Learning Strategies
(e.g., Metacognition & self-regulation, Reflective Thinking, Self-learning, Self-explanations, Argumentation)

	TA	TT	R	I	UG
Disagree	2.7	0.0	0.0	16.2	10.8
Agree	81.1	89.2	86.5	51.4	64.9
Mean	4.3	4.4	4.4	3.6	3.9
Median	5	5	5	4	4
Mode	5	5	5	3	5

Q2: Learning & Instructional Theories
(e.g., constructivism/constructionism, sociocultural & sociological perspectives, multiple conceptions theory, critical consciousness, cognitive load theory, theory of multimedia development, nudge theory, learning styles direct instruction)

	TA	TT	R	I	UG
Disagree	8.1	2.7	5.4	37.8	45.9
Agree	51.4	83.8	78.4	16.2	21.6
Mean	3.8	4.4	4.3	2.7	2.6
Median	4	5	5	3	3
Mode	3	5	5	3	3

Q3: Cognitive processes in learning programming
(e.g., mental models, notional machines, cognitive load theory, memory, schemas & plans)

	TA	TT	R	I	UG
Disagree	2.7	5.4	5.4	43.2	21.6
Agree	78.4	86.5	70.3	37.8	45.9
Mean	4.3	4.5	4.2	2.9	3.5
Median	5	5	5	3	3
Mode	5	5	5	2	3

Q4: Non-cognitive or affective dimensions of learning
(e.g., growth mindset, motivation, attitudes towards CS, CS identity, sense of belonging)

	TA	TT	R	I	UG
Disagree	2.7	5.4	5.4	16.2	13.5
Agree	78.4	83.8	73.0	51.4	56.8
Mean	4.3	4.6	4.1	3.6	3.8
Median	5	5	5	4	4
Mode	5	5	5	5	5

Q5: Broad Teaching Methods & Approaches
(blended learning, Socratic questioning, peer learning, semantic waves, culturally relevant and equitable pedagogy, taxonomies and hierarchies of knowledge and understanding, summative and formative feedback)

	TA	TT	R	I	UG
Disagree	8.1	5.4	13.5	48.6	45.9
Agree	67.6	91.9	64.9	35.1	29.7
Mean	4.1	4.6	3.8	2.8	2.9
Median	4	5	4	3	3
Mode	5	5	5	2	2

Q6: Teaching Methods & Approaches to teaching Computing/Programming
(pair programming, PRIMM, program comprehension-Block Model, programming problem solving, reasoning in programming, strategies for debugging, levels of abstraction)

	TA	TT	R	I	UG
Disagree	0.0	0.0	8.1	40.5	24.3
Agree	89.2	97.3	64.9	40.5	51.4
Mean	4.5	4.7	4.0	3.2	3.5
Median	5	5	4	3	4
Mode	5	5	5	2	5

Q7: Activities/Tools for learning programming
(worked examples, Parsons puzzles, explain in plain English, tracing code)

	TA	TT	R	I	UG
Disagree	2.7	2.7	13.5	40.5	18.9
Agree	89.2	97.3	51.4	29.7	51.4
Mean	4.4	4.7	3.6	2.8	3.6
Median	5	5	4	3	4
Mode	5	5	3	3	5

Q8: Learning Difficulties in Programming
(e.g., threshold concepts, misconceptions, problem solving & reasoning, levels of Abstraction, thinking computationally)

	TA	TT	R	I	UG
Disagree	8.1	2.7	5.4	24.3	8.1
Agree	91.9	91.9	70.3	48.6	59.5
Mean	4.5	4.8	4.2	3.4	3.9
Median	5	5	5	3	4
Mode	5	5	5	3	5

Q9: Diversity & Inclusion
(e.g., Microaggression, Implicit Bias, Stereotype Threat, Stereotypes, Unconscious Bias, Gender Differences, Accessibility, Sociological Theories like Bourdieu, Neurodiversity)

	TA	TT	R	I	UG
Disagree	8.1	2.7	0.0	10.8	10.8
Agree	83.8	89.2	78.4	64.9	54.1
Mean	4.3	4.6	4.4	3.9	3.8
Median	5	5	5	4	4
Mode	5	5	5	5	5

Q10: Society and Ethics in Computing
(e.g., Societal impact of Technology and Computing, Ethics issues involved in teaching like conflict of interest, Responsible computing practices, Accessibility)

	TA	TT	R	I	UG
Disagree	13.5	8.1	0.0	13.5	2.7
Agree	62.2	81.1	81.1	67.6	70.3
Mean	3.9	4.4	4.3	3.9	4.1
Median	4	5	4	4	4
Mode	5	5	5	5	5

Q11: Course Planning
(Curriculum design, Policy and CS Curricula, Course Design, Planning and delivering a lesson or a session)

	TA	TT	R	I	UG
Disagree	8.1	0.0	35.1	67.6	73.0
Agree	54.1	91.9	35.1	13.5	18.9
Mean	3.8	4.8	3.0	2.3	2.2
Median	4	5	3	2	2
Mode	3	5	3	2	1

Q12: Course Management
(Behavioral Management, Classroom Management)

	TA	TT	R	I	UG
Disagree	5.4	5.4	45.9	70.3	73.0
Agree	75.7	86.5	27.0	16.2	13.5
Mean	4.2	4.6	2.7	2.1	2.1
Median	4	5	3	2	2
Mode	5	5	2	1	1

Q13: Assessment
(Exam design, Formative and Summative assessment design, Tools for assessment)

	TA	TT	R	I	UG
Disagree	10.8	2.7	27.0	59.5	48.6
Agree	67.6	97.3	54.1	21.6	18.9
Mean	4.0	4.8	3.5	2.4	2.4
Median	4	5	4	2	3
Mode	5	5	5	1	1

Q14: Grading
(Providing Formative and summative feedback, Grading and Rubrics)

	TA	TT	R	I	UG
Disagree	5.4	2.7	32.4	59.5	43.2
Agree	78.4	94.6	40.5	21.6	37.8
Mean	4.2	4.8	3.1	2.5	2.8
Median	5	5	3	2	3
Mode	5	5	3	2	4

Q15: Engaging with students
(e.g., Responding to students, Generating discussions, Student engagement)

	TA	TT	R	I	UG
Disagree	0.0	0.0	24.3	37.8	29.7
Agree	97.3	100.0	51.4	35.1	45.9
Mean	4.8	4.9	3.5	3.1	3.4
Median	5	5	4	3	3
Mode	5	5	5	3	5

Q16: Technology in Education
(Developing educational software/technology, accessibility, online learning platforms, learning management systems, chatbots, user-generated content (e.g., internet forums))

	TA	TT	R	I	UG
Disagree	18.9	10.8	5.4	24.3	27.0
Agree	56.8	75.7	56.8	51.4	43.2
Mean	3.6	4.1	3.9	3.5	3.4
Median	4	4	4	4	3
Mode	4	5	5	5	3

Q17: History and Nature of Computing and CE

	TA	TT	R	I	UG
Disagree	32.4	16.2	21.6	54.1	29.7
Agree	24.3	54.1	64.9	21.6	37.8
Mean	2.9	3.5	3.8	2.4	3.1
Median	3	4	4	2	3
Mode	3	3	5	1	3

Q18: Research Skills
(e.g., qualitative and quantitative designs, research methods, research methodology (e.g., phenomenography, phenomenology), reading & understanding academic papers, practice academic writing, practical experience of the whole research cycle, current trends of research in CE, communicating research findings, reflecting on research findings)

	TA	TT	R	I	UG
Disagree	29.7	10.8	0.0	37.8	13.5
Agree	32.4	64.9	100.0	37.8	51.4
Mean	3.2	3.9	4.9	2.9	3.7
Median	3	4	5	3	4
Mode	3	5	5	3	3

A.2 Topics related to "Professional Competencies"

Q19: Communication skills

	TA	TT	R	I	UG
Disagree	0.0	0.0	5.4	2.7	2.7
Agree	97.3	97.3	81.1	86.5	81.1
Mean	4.8	4.9	4.3	4.4	4.2
Median	5	5	5	5	4
Mode	5	5	5	5	5

Q20: Teamwork and Collaboration

	TA	TT	R	I	UG
Disagree	8.1	8.1	2.7	5.4	5.4
Agree	83.8	86.5	81.1	78.4	78.4
Mean	4.2	4.3	4.2	4.3	4.3
Median	4	5	4	5	5
Mode	5	5	5	5	5

Q21: Code Reviews

	TA	TT	R	I	UG
Disagree	13.5	13.5	35.1	18.9	8.1
Agree	64.9	56.8	27.0	62.2	62.2
Mean	3.8	3.7	2.9	3.9	3.9
Median	4	4	3	4	4
Mode	5	5	3	5	5

Q22: Argumentation

	TA	TT	R	I	UG
Disagree	16.2	13.5	2.7	21.6	8.1
Agree	56.8	64.9	83.8	51.4	59.5
Mean	3.8	3.9	4.4	3.6	3.8
Median	4	4	5	4	4
Mode	5	5	5	5	5

Q23: Explaining ideas
(e.g., solution to a problem)

	TA	TT	R	I	UG
Disagree	0.0	0.0	2.7	5.4	5.4
Agree	97.3	94.6	94.6	70.3	86.5
Mean	4.8	4.8	4.6	4.2	4.4
Median	5	5	5	5	5
Mode	5	5	5	5	5

Q24: Intercultural Skills

	TA	TT	R	I	UG
Disagree	5.4	5.4	2.7	5.4	5.4
Agree	81.1	86.5	64.9	67.6	59.5
Mean	4.2	4.3	3.9	4.0	3.8
Median	4	5	4	4	4
Mode	5	5	4	5	3

Q25: Conflict Resolution

	TA	TT	R	I	UG
Disagree	5.4	5.4	27.0	21.6	13.5
Agree	83.8	83.8	43.2	64.9	51.4
Mean	4.3	4.4	3.4	3.8	3.6
Median	4	5	3	4	4
Mode	5	5	3	5	3

Q26: Teaching Peers

	TA	TT	R	I	UG
Disagree	10.8	13.5	16.2	16.2	16.2
Agree	67.6	67.6	51.4	56.8	51.4
Mean	4.1	4.1	3.6	3.6	3.5
Median	4.5	5	4	4	4
Mode	5	5	3	4	3

Q27: Mentoring and Coaching

	TA	TT	R	I	UG
Disagree	2.7	0.0	13.5	13.5	27.0
Agree	78.4	81.1	51.4	67.6	32.4
Mean	4.3	4.4	3.6	3.9	3.2
Median	5	5	4	4	3
Mode	5	5	3	5	3

Q28: Team Leadership & Management

	TA	TT	R	I	UG
Disagree	21.6	18.9	5.4	10.8	21.6
Agree	70.3	75.7	59.5	62.2	48.6
Mean	3.6	3.9	3.8	3.9	3.5
Median	4	4	4	4	3
Mode	4	5	5	5	3

Q29: Giving Feedback

	TA	TT	R	I	UG
Disagree	0.0	0.0	21.6	18.9	24.3
Agree	97.3	100.0	62.2	67.6	48.6
Mean	4.8	4.8	3.7	3.8	3.5
Median	5	5	4	4	3
Mode	5	5	4	5	5

Q30: Delivering a Presentation

	TA	TT	R	I	UG
Disagree	10.8	0.0	2.7	5.4	5.4
Agree	75.7	89.2	89.2	73.0	67.6
Mean	4.2	4.7	4.5	4.1	4.0
Median	5	5	5	4	4
Mode	5	5	5	5	5

Q31: Time Management

	TA	TT	R	I	UG
Disagree	0.0	2.7	5.4	5.4	2.7
Agree	86.5	86.5	86.5	81.1	81.1
Mean	4.4	4.4	4.4	4.3	4.4
Median	5	5	5	5	5
Mode	5	5	5	5	5

Q32: Applying User-centered approaches like participatory design

	TA	TT	R	I	UG
Disagree	24.3	18.9	24.3	27.0	37.8
Agree	56.8	62.2	54.1	48.6	35.1
Mean	3.4	3.6	3.5	3.4	3.0
Median	4	4	4	3	3
Mode	4	4	4	5	2