Design and Analysis of a Disciplinary Computer Science Course for Pre-service Primary Teachers

Jean-Philippe Pellet(✉), Gabriel Parriaux, and Morgane Chevalier

University of Teacher Education, Lausanne, Switzerland
{jean-philippe.pellet,gabriel.parriaux,morgane.chevalier}@hepl.ch

Abstract. According to new curricula being introduced in Switzerland, primary teachers have to teach concepts related to computer science (CS), but most of them have never been through a CS course themselves. At our university of teacher education, we have introduced a new disciplinary CS course for pre-service teachers, aiming to provide them with basic CS foundations to better grasp, contextualize, and explain the CS topics they will bring to their classrooms. This "experience report" paper describes the structure and design choices of the new disciplinary course. We propose a thematic split of the relevant topics to discuss and highlight strategies to make the course relevant for our audience. The declared and effective learning outcomes are then analyzed, topic by topic, through crossing survey responses and exam data. We also use survey data from a year later, polling the same participants again for relevance of their learnings in the disciplinary course after being in classrooms and conducting activities in CS. Through this, success points and improvement areas of the new course as well as changes to be made for the next occurrences are identified.

Keywords: Teacher education · Computer science education · Disciplinary course

1 Introduction and Context

In the French-speaking part of Switzerland, the K–12 curriculum for digital education was updated in 2021. Next to the already existing axes "media" and "ICT", a new "computer science" axis (CS) was introduced. The new curriculum was presented with the main objective of developing so-called *digital citizenship* and *digital culture* of pupils. Prior to this reform, political changes had brought the topic of digital education in general—and CS in particular—in the spotlight. There seemed to be a general preoccupation in the economic, scientific, and political spheres, relayed by the media, that children in our country would not receive sufficient education in the digital domain. In canton Vaud (one of the French-speaking Swiss cantons), digital education was elected as one of the main projects of the government and an ambitious pilot project was launched

A. Bollin and G. Futschek (Eds.): ISSEP 2022, LNCS 13488, pp. 125–137, 2022.
https://doi.org/10.1007/978-3-031-15851-3_11

to introduce the new curriculum of digital education and a teaching of CS in eleven schools starting from primary level [8].

Our university of teacher education has thus had to adapt its own curriculum to give primary pre-service teachers (hereafter referred to as "students") the competencies needed to teach the content mentioned in the new school curriculum, in particular CS. As those students had never had the opportunity to study CS in their school career, most of them needed education not only on the didactical side of CS, but first and foremost on its disciplinary aspects. Hence, a special course has been set up in our university of teacher education, focusing especially on disciplinary content of CS. Participation is optional, but all students have to go through the assessment at the end of the course. It is followed a semester later by a didactical course (not further discussed here).

In our educational system, universities of teacher education rarely provide disciplinary courses to pre-service teachers. For most of primary school disciplines, students' knowledge acquired during past school years is sufficient to follow the linked didactics courses (or else they join a regular university to study disciplinary content). Because of its novelty, and also to better understand what happened with teachers' knowledge in CS, we framed this new disciplinary teaching with a research setup capable of providing us with the information necessary for its regulation. This paper presents our setup and our main results.

Our main research questions are:

RQ1: How did students view CS before the course and has this view evolved?
RQ2: How did students self-assess their mastery of various subfields of CS before and after the course, and how accurate is this self-assessment?
RQ3: A year later, after trying out CS activities in the classroom, how did student retrospectively view this disciplinary new course?

This paper has the following structure: in Sect. 2, we mention analyses of similar initiatives or approaches to CS education of future teachers. In Sect. 3, we detail the structure of our new course and justify the design decisions. We talk about the source of the data we collected to answer our research questions in Sect. 4 and analyze it in Sect. 5. We conclude finally in Sect. 6.

2 Related Work

CS is relatively recent compared to other sciences, and this is even more true for its teaching. Only recently did it enter compulsory-school curricula in several countries (e.g., New Zealand, 2011 [2]; Estonia, 2012 [16]; the U.K., 2014 [15]; etc.). Most CS core knowledge is as new for the students as for their teachers. Despite this, teachers must be able to carry out a didactic transposition [4]. On the one hand, this professional gesture implies acquiring knowledge in CS (first level of transposition) and, on the other hand, reflecting/planning/proceeding to the transmission of this knowledge to their students (second level of transposition) [4]. In this study, we are interested in this first level of transposition on the part of the pre-service teachers and, as such, many researches have looked into

the CS conception among teachers. For instance, Funke et al. [10] interviewed six primary-school teachers on their opinions towards CS courses at primary schools. Results unsurprisingly showed that teachers need to be trained to access the core CS concepts. This kind of need has also been reported in another similar study [13], in which educators could improve upon teachers' misconception about CS in only three training days. Besides, more and more recommendations to decision-makers in education systems encourage pre-service teachers to take CS courses as part of their teaching degree programs [5] to meet minimum content and knowledge requirements.

Moreover, a survey [7] among 116 secondary-school CS teachers about the integration of CS in primary education showed that essential topics for primary school should be introduced into primary-teacher training to ensure that they do not pass on their misconceptions to students. In this spirit, Repenning et al. [14] designed and experimented with a core CS course among pre-service primary teachers ($n = 600$). Results showed that teaching a mandatory CS class for pre-service teachers helps change the representations of these actors (particularly women) regarding CS and digital technologies. Nevertheless, results still reported a lack of confidence in implementing CS concepts in the classroom.

Another study [17] investigated the lack of confidence of in-service K–12 teachers concerning their self-efficacy in assessing Digital Technologies against the Australian Teacher Professional Standards and various assessment practices. Teachers reported that they need time and support to develop assessment strategies for this new area.

This raises the question of the perceived usefulness of CS knowledge to primary school teachers. Unfortunately, to our knowledge, no study reports the needs and feedback of in-service and pre-service teachers regarding the transfer of core CS knowledge that they have been able to carry out in class with their pupils.

As a result, given state of the art, it seems necessary to offer training to pre-service teachers on the core CS knowledge—ideally, spread out over time and of at least three days. Concrete links with society should be emphasised in such training to enable pre-service teachers to foresee a transfer of this knowledge and a didactic transposition appropriate to the maturity of their pupils.

3 Structure and Content of the New Disciplinary Course

The time slots obtained for this course encompassed 6 half days over the course of one semester, which is equivalent to Prieto et al.'s three full days [13].

3.1 Syllabus

The syllabus of the disciplinary course, according to our institution's policy, should be based on the high-school syllabus of the same topic. However, at the time of course preparation, there was no mandatory CS course in high schools yet;[1] our own syllabus was then based on circulating unofficial drafts.

[1] Such a course is actually due to be introduced in 2022–2023.

There is a general trend in CS education to move away from a coding-centric approach [1,9]. We thus wanted to ensure that we were not only focusing on programming and ended up with the following three main subfields: (a) data representation; (b) algorithms and programming; (c) machines and networks.

As mentioned in the introduction, mandatory CS in schools, in the view of our minister of education, should serve a "digital citizenship" goal rather than a mainly technical goal. According to this point of view, while the basis of data representation and programming should be taught, it is equally important that the societal implications linked to the usage of technology in the general public be exposed and discussed [11]. To embody this perspective in the new course, we discussed societal issues linked to the technical topics in each of the 6 sessions and made them an integral part of the syllabus.

We also strongly felt that we needed a common thread along the course. The bigger part of our audience has no special appetency for technical matters: we could not just place next to each other a series of themes deemed relevant by us without a strong, visible link between them. We thus picked *web search* as a common thread, and arranged the conceptual topics of the syllabus around the exploration of what really happens at various stages of running a web search.

Here is the final six-session syllabus, formulated in terms of the common thread and linked to societal issues:

1. **Data representation**. *"Computers work with 1s and 0s. When you do a web search, the search terms are also 1s and 0s—so are the results you get, be them text or images. Let's find out how we can represent such data with bits."* ⇒ Binary representation of positive integers; representation of text, basics of bitmapped images. *Societal issues:* Energy consumption of storage systems and large communication infrastructure in response to growing usage.
2. **Computer architecture**. *"We now know how our web request will be represented. Let's now look at how these 1s and 0s travel through the electronics inside a computer and how that electronics can be build to process that information."* ⇒ Basic logic gates; high-level view of components such as CPU, storage devices, and sensors. *Societal issues:* History of CS and automated machines; influence of war-time goals (deciphering, ballistic computations) on the development of computers.
3. **Network and cryptography.** *"After exiting our computer, our web search goes through the internet to reach the search provider's servers. How it is relayed by the intermediaries involved? How can I prevent these relays from reading what's in my request?"* ⇒ Packet switching, basic routing and idea of protocol; examples of symmetric ciphers, common attacks, and principles of asymmetric cryptography. *Societal issues:* Disparities in the world for internet access; pros and cons of strong ciphers and end-to-end encryption.
4. **Programming I**. *"Our request has reached the remote server. To be answered, it is processed by the searched company's software. What is software and how do you instruct a machine what to do? Through programming."* ⇒ Using Python's `turtle` module: simple movement, simple loops, simple func-

tion definitions, without or with one parameter. No variables at this point. *Societal issues:* Open source/free software, licensing (not limited to software).

5. **Programming II.** *"The previous examples have showed us how to give instructions to a computer; this session will make more explicit the way data is referenced and handled in programming languages through variables that can represent values that are still unknown when the software is written."* Same programming environment: variables, `if` statements and conditions with variables, simple lists (definition and iteration). *Societal issues:* Data collection, profiling, recommendation algorithms, and third-party cookies.
6. **More algorithms and AI.** *"CS is more than web searches. Let's examine graphs, which allow us to model many problems, and a related algorithm, which is used in our GPS but not only there. Finally, let's talk about AI: what it is, what it isn't, and a little bit of how it works."* ⇒ Without programming: concept of graph, tracing of Dijkstra's algorithm, applications. AI: high-level principles of a rule-based classification algorithm. *Societal issues:* Importance of training data for AI systems and awareness of bias-reproducing systems.

There were many other topics (technical or societal) we had deemed worthy of interest that did not end up making it into the syllabus for time constraints. Many steps in the web-search-processing story are still missing. The goal of the common thread is to arrange the selected topics in a tractable sequence rather than provide a full explanation of the chosen phenomenon.

3.2 Operational Planning

The course ended up being given entirely remotely due to COVID-19. We prepared 4 to 6 videos of between 5 and 15 min for each of our 6 sessions, for a total never surpassing 60 min. Our aim was to keep the video time at a maximum of 60% of what the actual lecture time would have been.

We made a creative use of the Label element in Moodle to structure the subsections, link each of them with a short list of expected learning outcomes, and added to each of them practical exercises, the solutions to which were available to students and explained in details, sometimes with more videos.

Societal issues cannot readily be linked to practical exercises. In order not to limit ourselves to videos on these issues, questions were asked at the end of each videos, in a "food for thought" way—with no correct or incorrect answer. Arguments serving these discussions were provided in the "solutions" part.

In addition to the material on the Moodle page, we opened for each session a Zoom room for 120 min, creating several breakout rooms which corresponded to the session's subsections. This always included a room to discuss the open questions related to the societal issues. Each breakout room was staffed with one instructor and the students could thus freely navigate between them according to the concepts they were stuck with or wanted to discuss.

As a last means of interactions, a traditional Moodle forum was made available for public, asynchronous questions and answers about the course.

3.3 Evaluation

Evaluation was done during an open-book 90 min Moodle quiz comprising 28 multiple-choice (MC) questions (which were corrected automatically) and 4 open-text questions (which were corrected manually).

The MC questions were "rich" in the sense that they were not only text, but embedded images and (with the help of HTML `iframe`s) interactive logic diagram or code editors. The open-text questions asked the students to describe, with a few sentences of their own writing, their understanding of the societal issues discussed in the course and short analysis of a small related example situation. Typically, such topics cannot adequately be assessed with MC questions.

We insisted on the open-book policy, convinced that, especially in the field of CS, any evaluation requiring students to learn certain things by heart was assessing capabilities related to the lower levels of the cognitive domains of Bloom's taxonomy (Comprehension and Understanding), and we were trying as much as possible to focus on the higher levels (more specifically, Application and Analysis for technical topics, and Analysis and Evaluation for societal issues).

4 Data Collection and Methodology

To answer our research questions, we used data from the following sources:

- A survey given before the beginning of the new course, which contained question about age, gender, previous education, and included a section meant to capture their representation of CS and their declared a priori mastery of the main subtopics of the course (see details below);
- A post-course survey, with the same questions about their representation of CS and their declared a posteriori mastery of the same subtopics;
- Grades from the examination described above, given at the end of the course, with a detailed split of the points among the same subtopics whose declared mastery was asked about in the surveys;
- A survey given a year later to the same students, asking if they viewed what they had learned in the new course as useful for the following didactics course and for the classroom activities they had conducted in the meantime.

The grade records were anonymized and uniquely identified by some Moodle-generated ID. The same ID was automatically filled out in the pre- and post-course surveys, enabling us to automatically link the survey responses while keeping them anonymous.

4.1 Common Pre- and Post-Course Survey Questions

To address *RQ1*, both pre- and post-course surveys included questions to depict the students' view of CS before and after the course. They were asked to rate these statements on a Likert scale ranging from 1 (disagree completely) to 6 (agree completely): "To me, computer science..."[2] [12]:

[2] Students were also asked to answer an free-text question on how they would describe CS. Size constraints do not allow the inclusion of the analysis of those results here.

1. is mainly applied mathematics
2. does not really have permanent components and is constantly evolving
3. changes rapidly but rests on stable notions
4. has theoretical foundations
5. is mainly about learning how to use office software
6. is primarily about practical knowledge rather than concepts and notions
7. is the major science of the 21st century

To address *RQ2*, students were also asked to rate their mastery of the following subtopics on a scale ranging from 1 (no mastery) to 5 (excellent mastery): 1. binary data representation; 2. CPU and computer architecture; 3. cryptography; 4. programming; 5. algorithms and AI; and 6. computer networks.

The survey enabled us to observe how each student changed their view of CS and how their declared mastery evolved. Moreover, we could determine the correlation between the declared post-course mastery and the actual exam results for first 5 subfields listed above (subfield 6 was made optional and was absent from the final exam).

4.2 Year-After Survey

In the year after the disciplinary course, student have followed a didactics course and have had the opportunity to conduct a CS-related activity in a classroom. Since the new course was supposed to provide the foundations for the didactics course, we were interested in asking students' opinion through these two questions to address *RQ3*: (a) How useful did you find the disciplinary course for the activity you conducted? (rated on a 7-point Likert scale), and (b) Retrospectively, how adequate did you find the level of the disciplinary course? (rated on a 5-point scale: way too hard/too hard/adequate/too easy/way too easy).

We were especially interested in the year-after (rather than the right-after) opinion since it would be difficult for students to evaluate the adequacy of such a course without practical experience with actual classroom activities.

5 Analysis and Discussion

357 students were registered for the course and obtained a grade. Out of them, 284 filled the pre-course survey; 130 filled the post-course survey (114 filled them both); and 117 filled the year-after survey. The Demographics subsection thus rests on the 284-sample dataset; the analysis covering pre/post comparisons and the year-after opinion use the 114- and 117-sample datasets, respectively.

5.1 Demographics

About 65% of the respondents are between 18 and 22 years old; 22% are between 23 and 30; 8% between 31 and 40, and the rest 5% are older. 85% are female.

The highest degree of 85% of the respondents is a high-school degree. About 12% have a college degree; about 3% have another degree (professional or other).

87% of all students passed the exam on the first attempt.

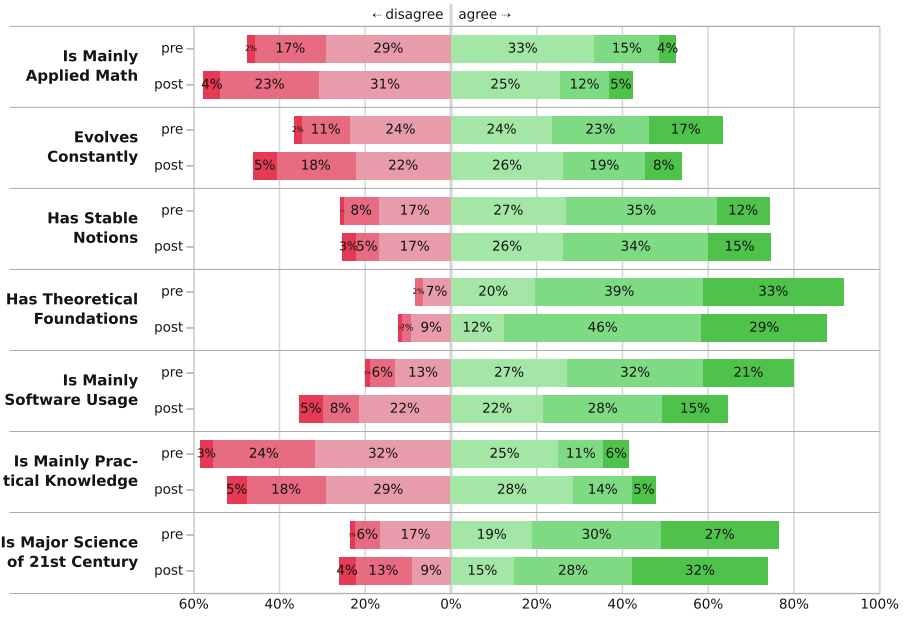

Fig. 1. Shift of opinions on what CS is according to the students, following the 7 questions described in Sect. 4.1, rated on a 6-level Likert scale. Red bars indicate (center-to-left) "somewhat disagree", "disagree", and "strongly disagree"; green bars indicate (center-to-right) "somewhat agree", "agree", "strongly agree". (Color figure online)

5.2 Representation of Computer Science

We now analyze students' views of CS through their compared (pre- and post-course) opinion on the 6 assertions listed in Sect. 4.1, shown on Fig. 1.

Students are almost evenly split on whether CS is mainly applied mathematics. After the course, they tend to reject this assertion more (even if the shift is on the verge of being significant at the .05 threshold: Mann–Whitney's $U = 20568$, $p = .0537$). The diversity of the subtopics and the discussion of the societal issues were meant to favor such a shift and a conceptual separation of math and CS.

Although CS is still viewed by the majority as constantly evolving with no permanent components, there is a significant shift ($U = 21749$, $p = .0029$) occurring. In hindsight, we should have phrased this question differently, as it mixes two dimensions (having permanent components and evolving constantly, both of which can be argued to be true) into a single statement.

More than 75% of respondents agree that CS has stable notions (this does not change between pre and post). Even more agree that CS has theoretical foundations. We were surprised to see more students disagreeing on this after the course. Even though the difference is not statistically significant, we ideally would have liked to see the opposite shift: we believe this is due to our very

practical approach which tried to include as little theory (and as little math) as possible.

We were pleased to see significantly more disagreement on CS being mainly software usage ($U = 21507$, $p = .0055$), even though a majority still agrees. There was no significant shift on the two remaining questions of our surveys.

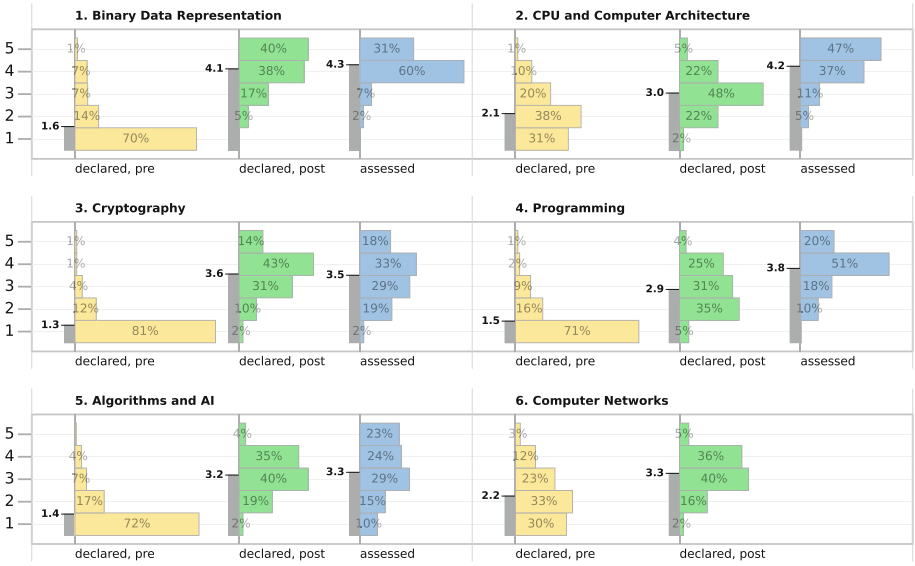

Fig. 2. Declared pre- and post-course mastery of CS subfields taken from surveys, compared to the performance on the final exam.

5.3 Declared and Assessed Mastery Levels of Subtopics

The results shown on Fig. 2 show, for each of the 6 subfields, a distribution of declared mastery on a scale ranging from 1 (no mastery) to 5 (excellent mastery). Pre-course data is shown first (yellow), then post-course data (green), and last (blue), we show the scores obtained by averaging over the grades of the relevant questions in the final exam and rescaling to reach the same 1-to-5 range. Subtopic 6 was made optional and no exam question was asked on it. The vertical grey bars show the mean of each distribution.

The declared pre-course level was almost the same for subtopics 1, 3, and 5, and the means of the declared post-course levels are very close to the final grades. Some students slightly overestimated their understanding of binary data representation and slightly underestimated that of algorithms of AI. On the two subtopics 2 and 4, students significantly underestimated their understanding with respect to our exam questions. These are the two topics where our course exercises included synthesis questions—small logic circuits in an interactive tool

for subtopic 2 and short Python programs with `turtle` for subtopic 4. These were experienced as difficult by students. On our final exam, these subtopics were addressed with multiple-choice question, which definitely made them easier to achieve than if they had also been synthesis questions.

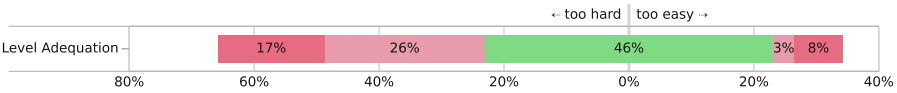

Fig. 3. Perceived adequation of the level of the course a year later.

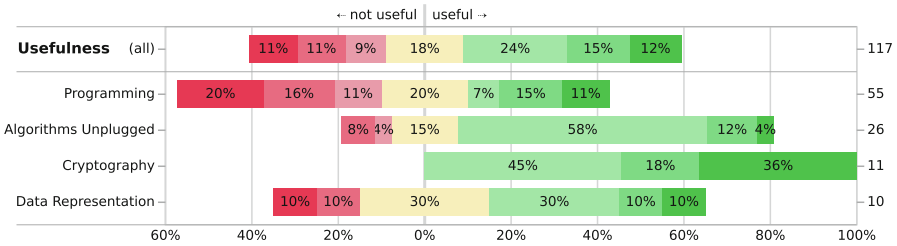

Fig. 4. Perceived usefulness of the course, a year later, split according to conducted classroom activity.

5.4 Year-After Opinion

A year later, 43% of the students who expressed their opinion said the course had been too hard, 11% too easy, and 46% adequate, as shown in Fig. 3. The imbalance between the two extremes has since then made us reconsider the inclusion of some more involved notions, especially in programming (Python's lists) and algorithms (some properties of graphs).

About half of the students found the disciplinary course had been useful to them (Fig. 4, first row). We were nevertheless surprised that almost a third said it had not been useful, so we split the analysis according to the type of activity the students had conducted in the classroom (filtering out activity types with fewer than 5 instances). The next rows in the same figure show the significant differences between them. The negative ratings mostly come from students who have conducted programming activities, while those having dealt with cryptography-related activities had unanimously found the course useful.

The main difference between these two subtopics is how closely what we do in the new course is related to what they can actually conduct as activity in the classroom. The programming activities are quite different: the course

uses Python, but they will use robots or Scratch Jr in classrooms—which is quite different, even though underlying concepts may coincide. For cryptography, we begin with Caesar's cipher and, although we also discuss more advanced polyalphabetic ciphers and several attack types, they can directly conduct a classroom activity based on Caesar's cipher, making the link immediately clear.

While research shows that being correctly educated about the disciplinary content is crucial for teachers, it also highlights the fact that it is not necessary to go far beyond the level of corresponding knowledge that they teach in their class. The more one goes beyond a minimal base of disciplinary knowledge as taught at a given level, the less added value this disciplinary knowledge brings to teaching [6]. But we also know from Bruner [3] that a knowledge of individual concepts is not sufficient. There must be an understanding of the way concepts are organized together, of the underlying principles that support them.

Let us be reminded that a subsequent mandatory didactics course exists to precisely present classroom activities. We have no clear way of actually knowing that the basics of programming they acquired in the new course did not make them more effective programming teachers. But seeing how closeness to classroom activities is beneficial to perceived usefulness has made us change the design of future occurrences of the course so as to always start with a motivating example very closely linked to an activity they will be able to conduct.

6 Conclusion

We have described the primary-teacher-education context in which we deemed necessary to introduce a new introductory course to CS. We have outlined its specificity and its syllabus, highlighting our common-theme approach and the discussion of the broader societal issues. We explained the all-remote modalities related to the COVID-19 situation.

We have analyzed data from pre- and post-course surveys, from the final exam, and from a year-later survey, after the students had conducted CS activities in actual classroom. Data shows that they view CS differently on a subset of statements on which we asked them to express agreement; notably, that CS is not about software usage and does have permanent components, although it evolves constantly. We noted that our theory- and math-poor approach has not reinforced much the impression that CS has theoretical foundations, although the course was also meant to convey the idea that CS is science indeed.

Declared mastery and exam questions show that topics treated with exercises involving creating programs or small logic circuit (where synthesis is needed) reduce the perceived mastery compared to other topics like data representation and cryptography (where exercises rather test understanding and analysis than synthesis).

Finally, the year-after survey showed greatly different perceived usefulness of the new course depending on the type of conducted classroom activity, even though all such activities were conceptually linked to the course. Perceived usefulness was maximal when the course not only conceptually coincided with the

conducted activity, but also directly treated (and expanded on) scenarios that could form a direct basis for that activity.

These results have enabled us to make data-driven adjustment to the new course so as to better highlight some fundamental aspects of CS as well as increase the perceived usefulness of the course.

References

1. Astrachan, O., Briggs, A.: The CS principles project. ACM Inroads **3**(2), 38–42 (2012)
2. Bell, T., Andreae, P., Robins, A.: A case study of the introduction of computer science in NZ schools. ACM Trans. Comput. Educ. (TOCE) **14**(2), 1–31 (2014)
3. Bruner, J.S.: The Process of Education. Harvard University Press, Cambridge (2009)
4. Chevallard, Y.: On didactic transposition theory: some introductory notes. In: Proceedings of the International Symposium on Selected Domains of Research and Development in Mathematics Education. pp. 51–62. Comenius University Bratislava, Czechoslovakia (1989)
5. Computer science teachers association, code.org advocacy coalition: state of computer science education (2018). https://code.org/files/2018_state_of_cs.pdf
6. Darling-Hammond, L.: Teacher quality and student achievement. Educ. Policy Anal. Arch. **8**, 1 (2000)
7. Dengel, A.: Opinions of CS teachers in secondary school education about CS in primary school education. In: Proceedings of the 12th Workshop on Primary and Secondary Computing Education, pp. 97–98 (2017)
8. El-Hamamsy, L., et al.: A computer science and robotics integration model for primary school: evaluation of a large-scale in-service K-4 teacher-training program. Educ. Inf. Technol. **26**(3), 2445–2475 (2020). https://doi.org/10.1007/s10639-020-10355-5
9. Fincher, S.A., Robins, A.V.: The Cambridge Handbook of Computing Education Research. Cambridge University Press, Cambridge (2019)
10. Funke, A., Geldreich, K., Hubwieser, P.: Primary school teachers' opinions about early computer science education. In: Proceedings of the 16th Koli Calling International Conference on Computing Education Research, pp. 135–139 (2016)
11. Paoletti, F.: Épistémologie et technologie de l'informatique. Revue de l'Enseignement Public et Informatique **71**, 175–182 (1993)
12. Parriaux, G., Pellet, J.-P.: Computer science in the eyes of its teachers in French-speaking Switzerland. In: Brodnik, A., Tort, F. (eds.) ISSEP 2016. LNCS, vol. 9973, pp. 179–190. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46747-4_15
13. Prieto-Rodriguez, E., Berretta, R.: Digital technology teachers' perceptions of computer science: it is not all about programming. In: 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, pp. 1–5. IEEE (2014)
14. Repenning, A., Lamprou, A., Petralito, S., Basawapatna, A.: Making computer science education mandatory: exploring a demographic shift in Switzerland. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, pp. 422–428 (2019)
15. Sentance, S., Csizmadia, A.: Computing in the curriculum: challenges and strategies from a teacher's perspective. Educ. Inf. Technol. **22**(2), 469–495 (2017)

16. Shin, S., Bae, Y.: Study on the implications about curriculum design through the analysis of software education policy in Estonia. J. Korean Assoc. Inf. Educ. **19**(3), 361–372 (2015)
17. Vivian, R., Falkner, K.: A survey of Australian teachers' self-efficacy and assessment approaches for the K-12 digital technologies curriculum. In: Proceedings of the 13th Workshop in Primary and Secondary Computing Education, pp. 1–10 (2018)