**Determining the Feasibility of Providing New Workshops**

**Yang Liu**

**ENGL 301**

 **Table of Content**

* Introduction ------------------------------------------------------------------------- Page 2
* Data Session ------------------------------------------------------------------------- Page 5
* Conclusion ------------------------------------------------------------------------- Page 12
* References ----------------------------------------------------------------------- Page 16

**Introduction**

Computer science is an essential part of society and benefits our lives in many ways. Even though people do not know computer technologies in detail, they connect themselves with computer technologies by using maps on their mobile phones, watching online videos on Netflix and many other ways. Modern businesses also rely on computer science technologies. Business people communicate with emails and transfer using online banking systems. Most companies generate and store any taxing information like bank statements or cash flow tables in the computers instead of manually writing any information down and keeping it in the file storage. Computer science education is more important than ever, and demand for computer science graduates continues to outpace supply. Based on research done at the University of Washington (UW), the number of first-year students who applied to UW in computer science and engineering is about three times that of first-year students in other majors. A consistent result can be found in many other world-famous universities, such as Harvard, Stanford, and Michigan. (Soper)

UBC, as a top-ranking university globally, meets a similar demand of interest in computer science study. The computer science department usually receives many applications from high school graduates and university/college transfer students. The minimum GPA for students to successfully apply to the computer science department is usually an A-range grade. UBC helps students acquire computational thinking and problem-solving skills by delivering professional computer science courses. However, the academic targets do not meet the industrial requirements in many ways. Computer science courses in UBC focus on understanding conceptual ideas and principles, while industries require designing and writing a program in any specific programming language. Students from universities are unskilled in using programming languages because university computer science courses emphasize academic principles instead of programming skills. Even though students can practice when they write their programming assignments or participate in co-op programs, their programming skills are not enough to meet the actual industry requirements. A new graduating student faces many challenges in seeking jobs, and technology companies must lower their employment requirements for incoming new graduating students. Many companies even design training sessions in specific programming languages for new graduating students after they get employed, increasing their cost.

This report tries to seek a way to help UBC computer science students develop qualified coding skills to meet industrial needs. A possible solution to solve the mismatch between the academic targets in computer science educations and the industrial requirements for IT companies is to provide a workshop. In the workshop, students will get practice in coding skills. The CSSS, a formal club for all computer science students, is expected to take the responsibility to hold this workshop. Four areas of inquiries are pursued to determine the feasibility of this workshop.

* What programming languages are commonly required in the industry and essential to teach in the workshop session?
* How willing are students to attend the workshop, and how many hours will they spend?
* How to quantify a student if he/she is well-trained in a specific programming language? What are the standard requirements for students to complete the workshops?

Two surveys sources are used in this report to answer these questions. The participants for the two surveys are current UBC CS students and past UBC CS students, respectively. A general conclusion regarding this workshop's pros and cons will be given at the end of this report.

**Data Session**

* **Feedback from Previous Students:**

The finding is based on the survey assigned to 23 previous students.



As shown in Figure 1, over 40% of participants choose Java as the most common programming language in their career life. The second common language in use is JavaScript, where 22% of participants use JavaScript. Other languages like C/C++, python roughly divided the remaining participants. However, the distribution varies as the participants' job changes. All the participants who work as Frontend developers mentioned that JavaScript is the primary programming language in their life. For software testers and backend developers, Java is still the most common language.



35% of participants think they are "very familiar" with their commonly used languages in the language practice evaluation part. They can do almost all the work by themselves in their career life without seeking help from others. They know the efficiency of their code and continually optimize the efficiency. (Code efficiency refers to a good time complexity and space complexity in their code.) 48% of participants think they are "familiar" with their commonly used languages. Like those "very familiar" participants, they finish their tasks most of the time and sometimes with some help. They can guarantee the complexity in their code with limited exceptions. The remaining participants choose "neutral." and no participant chooses "unfamiliar." or "very unfamiliar" in the survey.



Regarding the percentage of code ability from schoolwork, no participants think their code ability is fully contributed by schoolwork like academic projects or assignments. The number of participants who agree that most or half of their code skills benefit from schoolwork is close, with 39% and 44% percentages, respectively. 13% of participants agree that a minority of their code ability comes from their schoolwork. 4% of participants think schoolwork contributes nothing to their coding skills related to their most common language in their current job.



As shown in Figure 4, the most common learning source is writing projects individually. Collaborating projects with colleagues and video tutorials from YouTube, Edx, and Coursera share similar percentages with 26% and 22%. Less than 10% of participants choose to improve their coding skills by reading code dictionary books.



What coding ability should computer science students enhance most before graduating? As shown in Figure 5, most participants think something other than coding format, code efficiency, database, and debugging ability should be improved before leaving school. Besides, the ability to write efficient codes (i.e., good time and space complexity) is the most important to be considered as an IT jobseeker.

* **Feedbacks from UBC current students:**

The second data source is a survey for current UBC students in the computer science department. Thirty-six participants take this study anonymously.



Based on students' career plans, web developers and software engineers are the most popular among all careers, with 25% and 22%, respectively. Jobs related to data analysis like data scientist, business analyst, and machine learning engineers are 17%, 11% and 8% of all. Students who plan to work as network architects, systems analysts, software testers and product managers roughly have the same percentage, around 4%. None of the students plan to do a non-technical job after they graduate from university.



Figure 2 shows the results regarding the students' expectations about the most common programming languages in their future career life. The self-evaluation about their coding skills in their chosen languages is shown in Figure 3. Students who think Python will be the most common language in their future life occupies 33% of all. However, only less than 10% of these students believe they can perfectly handle Python when programming. The majority of these students are unconfident in programming using Python, and they either only know the basic structure and syntax of Python or part of them. The coding efficiency is not guaranteed, and time/space complexity needs to be optimized by others. Some students even think they know nothing about Python. The second common language in expectation is JavaScript, and students are generally familiar with JavaScript. 60% of students who plan to use JavaScript in the future can write projects individually with some help from others. C family languages (C/C++) and Java have similar percentages, around 20%, but students' coding skills in Java are generally better than their coding skills in C family languages. Figure 3 shows that 60% of students feel either professional or proficient in using Java. None of the students think they are unfamiliar with Java. In other words, they all know the basic syntax and common-used data structure in Java. Regarding future C family language users, 70 % of students think they are neutral or unfamiliar with C family languages. Only about 30 % of students feel confident in using C/C++.





Many students expect to have tutorials about the code syntax and some packages/libraries for specific programming languages if CSSS decides to hold a workshop for students, which takes 33% of all. Besides, assigning projects to students and providing essential help in finishing the project is highly expected from the workshop with roughly 25%. 28% of students hope the workshop can guide students in solving the problems in LeetCode. LeetCode is a commonly used website for companies, where many algorithm design problems are selected to examine students' coding abilities when interviewing. Other parts can be found in Figure 4. The time that students are willing to spend in the workshop varies from 2 hours to 16 hours, and most of the students prefer to spend 4 - 6 hours in the workshop each week. 6 - 8 hours and less than 4 hours are also common expectations among students.

**Conclusion**

* **The Essentiality of the Workshop**

Academic scholars have discussed computer science educations in many ways. Computer science educations based on traditional lecture format teaching have been widely criticized. The conventional teaching method in computer science somehow leads to "inert" knowledge, and students fail to associate their in-school learning with the out-school context. New teaching approaches like project-based, problem-based and constructive learning are adapted to advance computer science education. These new approaches highlight the importance of practices and collaboration in computer science education. (Schilling and Klamma 377)

Moreover, the conflict between the academic targets in university and actual industrial needs is also an issue to be considered in computer science education. The industries prefer the universities to train students in the most recent technologies, while the universities' concerns are always the long-term educations. Even though some universities make some effort to meet the industrial need by providing project courses or internships, these efforts usually begin late and are not enough. (Tvedt et al. 102)

UBC also faces the same challenges as all other top universities. Based on the data collected from previous UBC CS students, 44% of the students agree that schoolwork only contributes to half of their coding skills now. Based on the data collected from current UBC CS students, 50% of students feel neutral or even unfamiliar with the potential programming language in their future jobs. In addition, many courses in UBC are traditional lecture format and emphasize teaching knowledge. It is necessary to have a workshop that provides opportunities for students to associate their knowledge with out-school context, practice their coding and collaborating abilities, and know what their future employers expect from them.

* **The Design to Workshop**

The design of this workshop combines some insights from previous studies and the findings from two source data.

1. **A project-based workshop is strongly recommended.**

In recent years, constructivist learning has been introduced to many university courses as case studies to advance computer science education. Constructivism suggests that allowing learners to construct knowledge by themselves is an efficient way to learn. Constructivist learning emphasizes an active learning process instead of passively transmitting knowledge from instructors. A common implication of constructivist learning in computer science education is providing a project-oriented course. (Hadjerrouit 102) Moreover, our data also suggest that writing project is one of the most common ways for students to practice coding skills. Over 60% of past UBC CS students state that writing projects individually or in team format helps improve their coding abilities and is the major learning source.

1. **Collaboration should be encouraged when working on a project.**

A collaboration involving social interaction and negotiation is another crucial aspect of constructivist learning. The collaboration includes group formation to finish the project, communication between teams, and between instructors and students. Instead of providing students with solutions, instructors should take a role to guide them in searching relevant (helpful) information, evaluating their projects and providing feedback to improve. (Hadjerrouit 102) The collaboration benefits students in sharing the knowledge and thereby actively learning. (Jan Schilling) Some consistent conclusions can be found in our data as well. 26% of previous UBC CS students in the survey suggest that collaborating projects with classmates best help them train their coding skills.

1. **The workshop should help establish connections between students and local industries.**

Connections between students and local industries during the workshop help students understand the needs of industries. The workshop can invite local industries to involve in workshop programs by assigning the project to students, and the project can be part of tasks to be finished in the companies.

This connection intends to benefit both local industries and students. Students also learn necessary industrial skills and reflect upon their own experiences. The industry needs, including exposing students to new technologies, teamwork, large-scale development, and management activities, are addressed by connecting with students. (Tvedt et al. 102) In addition, any new concepts and methods in computer science education are also transferred from universities into company practices. (Jan Schilling) In work done by Ellis et al. (120), the characteristics of successful collaborations between universities and companies are discussed. The common characteristics of companies include a solid initiative to seek professional training from university, a relatively strong background (enough finance and big size).

1. **A Technical Lab is preferred to a conceptual lecture.**

As a top-ranking university, UBC has provided enough academic concepts and principles for its students in the CS department. This workshop intends to enhance students' coding skills and other industrial skills to help students seek jobs. Therefore, a technical lab regarding any techniques and software engineering methods is preferred to a regular lecture. Based on the survey from current UBC students, students basically know what common programming languages are but have a bad performance in some of them. For example, students are generally unfamiliar with python and C family languages. The workshop can assign projects in these programming languages to practice their coding skills. The technical lab can provide technique teaching based on the project objectives and the project plan.

**Reference**

Ellis, Heidi J. C., et al. “Characteristics of Successful Collaborations to Produce Educated Software. Engineering Professionals.” Computer Science Education, vol. 12, no. 1–2, 2002, pp. 119–40. Crossref, doi:10.1076/csed.12.1.119.8217.

Hadjerrouit, S. “Learner-Centered Web-Based Instruction in Software Engineering.” IEEE Transactions on Education, vol. 48, no. 1, 2005, pp. 99–104. Crossref, doi:10.1109/te.2004.832871.

Soper, Taylor. “Analysis: The Exploding Demand for Computer Science Education, and Why America Needs to Keep Up.” GeekWire, 21 Feb. 2015, [www.geekwire.com/2014/analysis-examining-computer-science-education-explosion](http://www.geekwire.com/2014/analysis-examining-computer-science-education-explosion).

Tvedt, John D., et al. “The Software Factory: An Undergraduate Computer Science Curriculum.” Computer Science Education, vol. 12, no. 1–2, 2002, pp. 91–117. Crossref, doi:10.1076/csed.12.1.91.8213.