Creating a Solid Foundation for Secondary Education and ICT through Technology Equitable Education

Carl Reilly, Christine Chan and Craig Gelowitz

Software Systems Engineering
Faculty of Engineering and Applied Science
University of Regina, Saskatchewan, Canada

Abstract

The purpose of the research is to investigate the use of software systems as a tool in the Canadian Public School classroom as a means to provide the diverse skills required for today’s digital world. This study investigated ICT diversity within a pedagogical context to implement digital computer competence, specifically Software Systems Engineering, to transform students into effective practitioners of technology literacy. The constructed approach represented in this study to education provides a practical process into critical thinking, digital equity, technology literacy, life-long learning, and prepares elementary and high school students for the diverse and intricate expectations of industry. This research provides a step forward to the goals of preparing a generation of skilled technology literate and diverse critical thinkers for a demanding and ever changing digital world. The focus of this article represents the introduction to Computer Science at a grade 11 level and the results of the implemented research.

Keywords: teaching, computer science, software systems engineering, learning, Information and Communication Technology (ICT), digital equity, technology literacy
1 Introduction

The most efficient, powerful and influential route to private, industry and public prosperity as well as a positive world view is education [1]. Business innovation, national economic performance, and up-to-date technology literate individuals have a direct relationship with education [1]. However, the formation of Information and Communications Technology (ICT) integration in schools is largely unaddressed [3], populated by grossly underqualified technology literate teachers [3], and huge challenges in general public understanding of the topic are required to be overcome if schools wish to come to terms with the fact that their current responsibility of meaningful skill sets for a technology literate society are not being achieved. Universities depend on intelligent and informed students to succeed in the institute’s studies. Industry depends on individuals that have an explicit understanding of the skills required for a specific task. Education has an objective to provide and nurture our young thinkers with the proper skills and tools that can then be turned into a prosperous endeavor as well as a digital equitable standard of living. This research provides solutions and proofs to properly implement an ICT based pedagogy within the Saskatchewan high school curriculum for providing a technology literate society of young thinkers to support the expectations of University, businesses and over all general digital equity.

ICT is no more about computers than education is about books [4]. In that understanding, “ICT becomes Information, Communication, and Tools for Education” [4]. However, that idea is constantly being readdressed and with few positive results [2]. The goal is to go beyond computers as a tool in the classroom, which is the current approach [5], and to provide technology as a practical and applied art, in the same way any other subject is taught. This article will prove that the concept of teaching to understanding computers, that deep understanding that existed in the 1980’s, using an SSE based approach as one would teach and understand Math or Writing or Music, is achievable in this modern world of integrated complex digital technology.

2 State of the Art in Computer Science

Despite the clear understanding that the key measure of a population’s education and competency is observed through literacy, technology literacy is not being addressed. Digital equity is being misinterpreted with a “have-not attitude” in such that if you supply a computer, that is good enough and we can all say that there are computers in a classroom. Regardless of the advancements in education, digital technology is still being applied in antiquated and grossly misunderstood ways such as, computers in the classroom solely for writing documents and playing a game (if you are good in class). Most of the technology is not understood, in contrast as a book, itself, is understood.
A student’s first breath of computer science is in a high school setting as an elective class in grade eleven [5] void of any pre-requisite classes. They must apply for a spot in the course before being considered to participate in the classroom for a graduating credit. Take note, that before this event, most students have only experienced computer science through informal and mostly unstructured and uninformed ways. Almost all of the anticipation of computer science is what the student has seen through movies, indirect contact with technology such as a game[2], etc.

The structure of the computer science course is a number of theory based units comprised of notes, videos and hands-on programming in BASIC[5]. Some of the courses are provided online and the teacher is nothing more than a facilitator of the hour that the students have in the lab. Videos are used to represent the notes that are taken, usually about the history of digital computer technology in the 20th century. To exemplify the workings of a computer, students are often given a picture of the inside of a computer and asked to research the topic and present its tasks. Programming is a separate topic related to a computer, were creating a for-loop is given in a formula rather than associating it into representing the inner workings of a computer. Programming has become a unique and separate idea and represented as what you can do with a computer rather than representing how a digital computer works, internally. The examples for programming are very specific and the worksheets provide the students with a narrow window for easy repetitive functions. Traditional graphic flowcharts were also required before every worksheet was performed. The lack of technology literacy is clearly exemplified in the fact that units and subjects within computer science represented what you do with a computer, rather than associating the process of software systems to control the computer hardware and output. The later has outcomes that represent, clearly, an understanding of technology and how to control it. Much like literacy skills in the 19th and early 20th century, technology literacy needs to be understood and represented properly within education for a digital equitable society as a core subject and tool, no different than language combined with a pen and paper technology is represented. The problem to solve is how to deliver the concepts of our modern and complex digital technology to a group of young adults; what is the best approach to acquiring the expectations of lifelong learning and fulfill ICT expectations.

3 The Research and Problem Solving

Much of the research for this article has been implemented, observed and practiced over a four-year period between the years 2011 and 2015 within a computer science classroom in Regina, Saskatchewan, Canada. Grades 11 and 12, as imposed by government curriculum, were the representation of the research outcomes. The problem at hand is threefold: 1. The expectations of the learners and curriculum being met and to prepare the learners with a skillset for lifelong learning and independence. 2. The expectations and requirements of industry
(ICT) to provide the expected standard of marketable skillsets and knowledge.

3. The method of delivering the knowledge and evaluation required to measure the competence and skills expected of both the student and industry.

To address the first item, a survey was performed at the beginning of the semester to provide that data and understanding of the students’ intrinsic representation of the course. The survey asked the student what they thought computer science was, what their knowledge of technology came from, with an example of technology literacy they were asked to rate their competency of the topic, what they thought computer science should be, and what their expectations of a computer science class should provide them.

The results of the survey provided the necessary feedback to represent the changes required to focus the needed outcomes and objectives of all three points. With the new focus articulated from the survey results, a new process of representing the knowledge as an outcome and as an evaluation method was planned as a number of measurable-themed units within a semester.

The feedback provided by the students via the surveys reflected that the majority of the student population had no understanding of how digital technology and software drove each other; the ideas were understood as two distinctly separate concepts. All students (and educators who took the survey) also associated technology with education, intelligence, and competence. While the majority of survey participants could not read source code, two percent of them stated they could read the supplied source code method (in Java). Those that could read the source code could only read it line by line, as a list of items rather than as a whole understanding of what the method was trying to achieve. Almost all survey participants could read an algorithm and understand the objective of what the algorithm produced. No-one in the survey could write an algorithm or could explain what an algorithm was.

The same survey was used at the end of every semester. Tracking of attendance, grades, and assignments were also kept throughout the semester.

The first solution was to extract what the students expected from the course and meld that with what is expected from an ICT based expectation. That result was developed as a practical skill set that represented a reading and writing source code process using game development as the platform for learning. Students wanted to know about technology and how it works, while ICT based expectations represented a skill set that could identify technological innovation by overcoming obstacles throughout that process. In order to implement those basic requirements, it was clear that the process of implementing the learned skills was as important as understanding the technology which was being used and manipulated. As a result, a Software Systems Engineering approach was used to plan and implement the expectations. To that means, the approach to learning would exemplify the process of actively constructing software first before any discussion of the concepts, from which a student led solution could be achieved, building confidence along the path to learning. The newly learned skills and confidence would allow the understanding of how the technology/hardware interacts with each other, as the student understood them to be as they see their
Surveys of the previous course content prior to the research and changes reflected that the students could recreate a specific task, but could not solve a problem in computer science if the outcome to that specific practiced task was changed and no direction was given (i.e., no step by step worksheet were provided). Restating that, in a practical way, the majority of students could recite what a for loop was, however, could not implement the statement as a tool, but rather as a specific solution that has already been exampled by a worksheet—such as a hammer is used only for sinking nails and nothing beyond that. No student could link what a for loop was in respect to the inner workings of a digital computer.

From the surveys, an understanding of what the outcomes and objectives were, were extracted and a plan for presenting the information was created from those results based on this set of teaching philosophies: 1) People do judge a book by its cover. 2) People form an opinion about everything based on the signals that the subject conveys. And 3) Present the subject in a creative, professional manner as an active skill to relate and represent the required outcomes.

As a plan, a Software Systems Engineering approach was utilized in the delivery and objectives for this change. To represent that statement, the course was renamed to CS: Foundations of Software Systems and would approach technology as a literate subject first and throughout. Supplementing the vocabulary would be a foundational skill based approach where digital hardware would be presented to the students and a software objective would be provided linking the concept to a student owned physical representation and solution based on games.

The approach used to deliver the information was based on the foundations of a simplified software systems lab approach. In such that the topical concepts are performed first within a controlled lab experience before any real discussion of the mechanics of the concept was presented. That process allowed the student to actively understand and link the concepts to actions before contemplating any discussions. When discussions are presented, students have a firm understanding of the concept with concrete constructive and intrinsic connections. Discussions are often more compelling and filled with self-confidence as students have reported that their questions are “in the moment” and are fully justified with concrete, self-taught examples. That is the fundamental difference in traditional presentation of such new concepts where one would discuss the concepts and then allow the students a directed and constructive way of interacting with the information led by step by step instructions. That lends to a somewhat blind approach to using the tools and provides a very strict path of application. With the traditional approach, students do not understand the vocabulary and they are struggling to understand the concepts at the same time they are implementing the tools. The failure to associate an instruction as a means to represent how a computer works is missed. That standard provides an unstable foundation from
which to understand any new concept. Also worth noting, traditionally, the act of teaching computer science was approached by making the computer the centre of interest, rather than as a student that will use the computer as a tool.

Interesting videos throughout the new course were introduced and discussed: topics such as the dark web, proper searching techniques, icons of past and current technology representing game makers, technology innovators, upcoming speculation of trends, how parts of a computer are made, etc.—ideas for these are generally student driven and change from semester to semester. Such videos would also represent industry’s needs for what the students use and expect along the diverse skills required, such as mastering technology to produce musical scores, acting, digitizing, logic, user interfacing, IT, etc. Students become aware that all interests can be and are required in the technology literate world. This is the first time a student is experiencing a technology course focused on solution based skills that represent their interests; the focus is on technology literacy. Programming is simply a means to provide the connection to the fundamental processes of technology and how to understand the process which our world is being created around us.

Supplementing the objectives are rubrics. Rubrics were designed to provide a path from which students would know exactly what the outcomes are and why they are being studied as well as to judge their progress of the outcomes and objectives for the specific topic/concept.

The tool used for implementing the course was Greenfoot. Java represented the most diverse platform target for very little investment. The concepts of targeting and factoring solutions into manageable algorithms which would represent methods in their source code were easily achieved with the Greenfoot environment. Students recognized the source code easily since it looked like what they seen on TV, the structure of each document could be easily read and analyzed, accessible open source examples were easy to obtain and read. It is worth noting that being able to read others’ source code is a fundamental skill set and not piracy, which was a hurdle to overcome in the lab. Without being able to read and understand the source code, implementing the source code into their own objects was extremely difficult. Each lab was supplied with a starter skeleton package along with a compiled objective (the end result). Students were allowed to include whatever they could conjure up into their source code, as long as the fundamental outcome specified in the lab was satisfied. The results were anything but amazing. Students represented their interests and personalities in the games they made as if they were writing poetry.

Emphasis on test writing was not weighted as much as the project work was weighted. The written testing included one mid-term and one final to help the students judge themselves on comprehension and were to seek help and guidance.

The written testing consisted of three parts: 1) A multi-choice question section, 2) A short answer section focusing their understanding of how technology worked as it reflected on certain topics and projects, and 3) A short algorithm example. The bulk of the students’ final mark was weighted on a practical two-week project that finished with a competition between student teams. The
objective of the final is to represent the total sum of the outcomes and objectives as a team of two that will provide and share the skills learned in the course: read and write source code, implement algorithms, understand the process involved in the structure of source code and technology.

The majority of knowledge assessment is through the students’ physical work throughout the course, much like a wood working class represents a student’s knowledge. The final projects represent a foundational understanding of software systems via the ability to read technical writing, understand the process of starting with a set of criteria and bringing the project to fruition while staying focused, reading and writing algorithms as well as translating them into source code, experimentation, and re-evaluation. The final project is graded against a supplied rubric, specific to the final project, as well as a team evaluation where the student would be given a rubric from which to separately and privately evaluate their partner. The competition, itself, was not marked and instead was used to showcase the students’ achievements and a prize to the winners was awarded.

4 Conclusion

Populations for the computer science course in the five years prior to this research were low, averaging 10 to 15 students per class and allowed for two individual one hour classes a day. Figure 1, below, describes the number of students who enrolled and the numbers that passed computer science in the 2011 year. Traditional computer science flow charting, problem solving, and worksheets were implemented for that year. Computer Science 30 (Grade 12) is also shown to describe the number of returning students.

![2011 High School CS Student Population Enrollment and Passed](image)

Figure 1 - Circa 2011 Computer Science population trends.

In contrast, Figure 2 provides Computer Science population trends throughout the four years of the research where the SSE platform was implemented:
Figure 2 shows a dramatic increase in student interest in Computer Science for each section. The anomaly seen in 2012a, where the number of students graduating (blue) is higher than the number of students at the beginning (red), represents additional students entering the course after the semester had begun. Importantly, what can be demonstrated by the graph is the number of students who pass the class. Thus, representing their knowledge of computer science as a process and is proven to be understood by the majority of students in the class. The end numbers for year 2015 are not displayed as that data had not been correlated at that time. The maximum number of students possible in the classroom/lab is 32 for each class. As displayed in the graph for year 2014, there were too many students wanting to initially participate in the course, however, those students had to choose another study since the school was unprepared for the higher numbers.

Regardless of the numbers, Figure 2 clearly shows that the change in representing technology clearly shows that students can confidently discuss technology and understand the process of interacting with technology at a controlling level—the mystical magic representing their technology was demystified. To represent that statement, figure 3 is provided, displaying the class averages for the years displayed, above:
Creating a solid foundation for secondary education and ICT

The representation of knowledge concerning computer science in a modern world and also fulfilling the expectations needed for ICT in Canada was best provided using a software systems approach.

Learning computer science from a student centred approach as demonstrated in this study; using their ideas, language and representations, rather than teaching a student from a traditional computer centric approach has been proven to be the most effective way of preparing students with a skillset that represents the world with lasting understanding for their future.

References


Received: February 15, 2017; Published: March 31, 2017