Barriers to Digital Literacy: Learning to Program

Andréa Cartile
Gina Cody School of Engineering, Concordia University
andrea.cartile@concordia.ca

Abstract – There are many challenges associated with teaching and learning computer programming for first year engineering students in non-computer based fields. This paper discusses barriers to acquiring the digital literacy needed to learn end-user programming, or programming as a tool to support activities in a non-computer science domain. The first barrier discussed is the gap in educational curriculum, where the first formal introduction to computer science and programming is found in pre-university preparatory courses. The second barrier is a lack of consensus in approaches to learning programming in online resources. A solution of integrating opportunities to use programming as a tool in existing course curriculum activities is proposed, as a way to improve programming accessibility and allow future engineers to use digital skills to innovate in non-computer based applications.

Keywords: digital literacy, end-user programming, Canadian engineering education

1. INTRODUCTION

Digital skills are increasingly required in today’s job market. A UK report found that 82% of online job postings in 2019 required digital skills [1]. Despite almost 50% of the world’s population having a cellphone with an internet connection [2], the experience of using a digital device is not necessarily equivalent to digital knowledge [3] [4], and there still exists gaps in digital literacy [5]. With 90% of Information Technology (IT) jobs being outside the IT sector [6], most digital skills are required for application in non-IT domains [1]. In Canada, all engineering disciplines report requiring digital skills, ranging from office tools such as word processors and spreadsheets, to domain-specific software use and development [7], requiring both digital literacy and knowledge within the applied domain. This multi-disciplinary and computer data-driven engineering employment landscape illustrates the need to promote digital literacy within engineering curriculum across all engineering disciplines, yet gaps still exist between digital literacy and domain application. This gap is particularly prominent within the topic of computer programming.

The intersection of a non-professional programmer using programming on an applied basis is known as end-user programming, involving “people who write programs, but not as their primary job function”, but rather in support of it [8]. While computer programming courses have been incorporated into course sequences across engineering disciplines in Canadian universities, teaching and learning computer programming continues to pose challenges for both institutions and students alike. Often associated with high failure rates [9], barriers to acquiring introductory programming skills have been examined in literature since the mid-1980’s [10] [11] [12]. Many authors have discussed the different elements of learning to program, from aptitude [13], to informational barriers [14], and areas of difficulty faced when programming [15] [16]. The many facets of literacy required to progress in programming create barriers for new programmers, reducing opportunities for engineers in non-computer-based fields to develop end-user programming proficiencies.

This paper addresses some digital literacy barriers associated with learning to program, within the context of non-computer-based engineering domains in accredited Canadian universities.

2. BACKGROUND

2.1 Digital literacy

While many terms have been used to describe digital literacy [17][18][19][20], Allan Martin [21] presents three levels of digital capabilities, shown in Figure 1.

![Figure 1: Categories of digital literacy recreated from Allan Martin [21]](image)

Martin [21] describes the attainment of each of these levels as being relative to the required skills needed by an individual in a particular domain, and that “Instantiations of digital competence will vary from person to person as their situations vary, and will change over time as new tools and facilities are developed.” Martin describes each digital literacy level within
the context of a senior citizen demographic, which has been adapted in this paper to the context of non-computer-based engineering programs.

As each level of digital literacy is relative to the application domain, Martin [21] describes Level I: Digital Competence as including skills that could range from internet searches, word processor and email use; to website publishing, game use, and database creation and use. Correspondingly, Level II: Digital Usage is the successful application of the Digital Competencies within a specific domain. An engineering example of this could be using a finite element modeling software to understand failure modes in a part. Finally, Level III: Digital Transformation is described by Martin as “when the digital usages which have been developed enable innovation and creativity, and stimulate significant change within the professional or knowledge domain or the personal and social context.”.

While this model contributes toward an understanding of digital learning progression within a domain, it does not address the acquisition of digital skills inherent to other domains as tools for application. In the case of computer programming, it has been traditionally associated as a digital competence foundational to the domains of software and computer engineering, or fields in which it is common to acquire professional programming level capabilities. This level of in-depth knowledge development is not necessarily required for engineers in non-computer-based domains.

2.2 End-user programming

End-user programming, or literacy in computer programming as means of developing digital tools that support every-day personal or professional tasks, is significantly more common than professional programming [22]. Chilana et al. [23] suggest that an end-user programming literacy level is a stepping stone to professional programming proficiency, as depicted in Figure 2.

![Figure 2: Non-programmer, to end-user, to professional programmer scale, adapted from Chilana et al. [23]](image)

An overlap of challenges within the process of acquiring the digital skills to transition from non-programmer to both end-user and professional programmer is supported by Ko et al., stating that “Despite their differences in priorities from professional developers, end-user programmers face many of the same software engineering challenges.” [24].

For engineering students, two major barriers exist between the intersection of digital capabilities and end-user programming proficiencies. The first barrier is in acquiring Level 1 Digital Competencies needed to learn programming, and the second barrier is the opportunity to develop Level 2 Digital Usage and Level 3 Digital Transformation within the end-user programming context of their specialized engineering domain.

3. ACQUERING DIGITAL COMPETENCE

3.1 Secondary education level

Entrance requirements to non-computer-related engineering programs are relatively uniform across accredited Canadian institutions [25], and include three primary entrance routes: a technical certificate; a CEGEP DEC (Québec); or a high-school diploma. Out of the 44 accredited engineering institutions, 41 have admissions routes for all three categories, and all contain requirements of either Grade 11, 12, or Québec CEGEP-level math, physics, chemistry, and sometimes biology. However, while some institutions recommend them, none of the non-computer based programs require any computer science prerequisites.

A solution to this problem, however, may not be to impose prerequisites, as computer science is also absent in high-school curricula. In Québec, while computer science has been developed and integrated into the Diversified Basic Education (DBE) Adult general education system [26], as well as several stand-alone initiatives such as the English Montreal School Board STEAM pilot project [27], the topic of computer science is only formally introduced into the curriculum at the CEGEP-level [28] (a two-year pre-university preparation equivalent to grades 12 and the first year of university in other Canadian provinces [29]) as an optional science elective. As accredited Canadian university engineering undergraduate curricula have an introductory programming or computer science course within the first or second year, it is likely that the first time many students will see this topic in a formal educational setting is in their first year of university.

3.2 Post-secondary education

Van Dijk and van Deursen [30] identify several approaches to developing digital literacy skills. These include learning-by-doing, asking others for help, learning at work, learning through formal education (such as post-secondary institutions) and learning through informal education (such as public community centers). While 94% of Canadians had home internet access in 2019 [31], suggesting that most first-year engineering students have some internet operational skills [32] and experience using computers to access the internet, a lack of formal computer science education suggests that students arriving into their first year of engineering have varying degrees of digital competencies, with any competencies having been acquired through self-teaching. The many students with lower levels of
digital literacy exposure face a steep learning curve to familiarize themselves with the various computer science and programming subjects needed to complete the introductory programming class typically taken in the first or second year of the engineering degree. The constraints of a 13-week course result in many students approaching self-teaching with the help of internet resources.

### 3.3 Online resource use

While the use of online resources as complementary study tools has become commonplace among university students [33], finding the right resources has its own set of challenges. A key word search for “learn computer programming” in Google yields over 600 million results, and “learn how to code” yields 6.38 billion results, covering a wide range of information in different digital formats. The volume of information, combined with the number of different approaches to teaching and learning programming, can create further barriers to digital literacy development. Students looking to acquire the basics of computer science would likely not be aware of the concept of end-user programming, or the difference between programming platforms and interfaces. The curriculum gap in the secondary education system is not being adequately bridged by online resources, which have their own set of accessibility barriers towards developing digital literacy.

### 3.4 Assumed level of digital literacy in online resources

One major challenge in navigating this myriad of online information is finding a resource that is compatible with a students’ current-state digital literacy. As digital literacy is inherently complex, a combination of different resources is likely required to learn a new topic. However, the lower the digital literacy starting point, the more challenging the navigation of these resources. For example, the online programming resource “W3Schools” contains many programming-related tutorials, including Java [34]. The very first introductory page states that “It is not necessary to have any prior programming experience.”, but is immediately followed by a series of instructions that assume the user has familiarity with the Windows Command Prompt interface. The intended user of this platform must therefore already know what a command prompt is, and will already need a sufficient computer science background to effectively use this resource. The student without sufficient digital literacy will then either need to either amass other resources addressing the computer science topics, and risk preventing their progress, or recognize that some resources assume a higher level of digital literacy than others and look elsewhere. This barrier is similar to what students might experience in a programming course that introduces the topic of programming by opening a command-line interface and returning “Hello World”; the background information required to move forward from this point is significant, and the knowledge areas are not easily identifiable by a student attempting to catch up on literacy deficits.

### 3.5 Knowledge areas required to program

Another major issue in navigating online resources is a lack of consensus regarding how to go about learning to program. Online programming resources can be divided into two major categories: the first being learning platforms, that typically include a combination of text, video, and an online programming interface for practice such as Khan Academy [35], Codecademy [36], and Coursera [37]; and the second being text-based articles detailing recommended approaches on how to learn to program [38] [39] [40]. Both categories of resources differ in their recommendations for the foundational knowledge areas needed to learn programming, and provide different approaches to learning to code. A sampling of online resources is shown in Table 1, and lists their corresponding top-level categories of programming learning approaches. These free online resources were identified using Google keyword searches such as “learning how to program”, “learn programming”, and “coding for beginners”, and were selected as illustrative examples.

| Web Resource | Learning Recommendations |
|--------------|--------------------------|
| "I Want to Learn Programming but I Don’t Know where to Start" [38] | 1: Computer Science Fundamentals, 2: Propositional Logic (optional), 3: Java Programming, 4: Algorithms and Data Structures, 5: Android development, Helpful Resources |
| How to learn programming? [41] | 1. Get familiar with a programming language, 2. Learn Data Structures and Algorithms, 3. Get your hands dirty with competitive programming, 4. Make a project, 5. Explore |
| The Best Ways to Teach Yourself to Code [39] | Ask yourself: Why do you want to learn how to code?, Choose the right programming language, Try out some online courses, Focus on learning computational thinking, Get a book, Check out some interactive tutorials or coding games, Try a kid’s toy, Teach your favorite devices (and assistants) new tricks, Watch videos about coding |

| Table 1: Examples of online programming resources and their corresponding approaches to teaching and learning |

**Note:** The resources listed in Table 1 are illustrative examples and may not be representative of the full breadth of online programming resources available.
The starting point differs significantly within the first introduction of each resource, starting from motivation, to an introduction to computer science, and some starting with a formal language introduction such as Computer programming: Intro to JS: Drawing & Animation [35]. While online resources are not usually subject to a peer review process, many provide interactive learning platforms, including gamified platforms [35], and are often broken down into short learning clips, making them an attractive way for students to approach the filling of educational gaps. However, the overwhelming available online resources and diverse approaches to teaching and learning programming contribute to the barriers faced by students already constrained by the structure of a 13-week university programming course. The inaccessibility of this information likely contributes to higher failure rates in introductory programming courses, but more importantly, reduces the likelihood that a student will be able overcome these digital literacy acquisition barriers.

4. DIGITAL USAGE AND OPPORTUNITIES FOR INNOVATION

The computer science and programming educational gap within the secondary-level curriculum, combined with barriers in navigating online resources, creates an information deficit that cannot adequately be met by one or two isolated programming courses within a university engineering curriculum. Learning programming as a siloed subject is also counter-productive to supporting the end-user application learning needs of non-computer-based engineers. Fundamental to end-user programming is “programming in context” Chilana et al [23], whereby programming is used as a tool to support domain-specific applications.

While there are many different introductory end-user programming teaching and learning initiatives that are attempting to bridge the digital literacy gap [42], limiting programming exposure to an introductory programming course does not provide engineering students a sufficient platform to explore areas of opportunity for application. A solution to this problem would be to incorporating programming assignments, projects, and labs within existing courses throughout the engineering degree. Not only would this provide students with many opportunities to learn and practice programming, but would also provide a much broader sense of how programming can be applied to existing problems, and add a range of programming tools to a student’s transferrable skills repertoire. This integration of digital tool development in an application would better reflect the opportunities for end-user digital tool development beyond the classroom, and would provide a better chance of developing Level II digital usage experience. With experience and opportunities to solve problems digitally, this would lead to a higher likelihood of developing Level III innovation and creativity as applied to an engineering domain.

4. CONCLUSIONS

Digital literacy is an integral part of today’s job market. The most common form of digital engagement is at the end-user level, whereby digital skills are applied to support the execution of domain-specific tasks and activities. While end-user programming prevalence far exceeds that of professional programming, several barriers exist in acquiring these digital skills. A significant gap in teaching and learning computer science and programming exists in the secondary-level curriculum, creating a digital literacy deficit for engineering students embarking in a first-year programming course. Internet resources are difficult to navigate and assume many different levels of digital literacy, which contribute to the barriers in towards bridging this digital literacy gap. A solution that would better support the development of end-user programming literacy would be to incorporate opportunities for programming application in existing courses within the engineering curriculum, to develop a complementary set of programming tools and best prepare students for an engineering career.

Mäkinen [43] discusses digital empowerment as “used in the sense of enablement – enabling people to do what is important to them, and enabling people to grow as competent subjects who have control over their lives and surroundings.” In the context of engineering, developing sufficient digital literacy to use programming as a tool for everyday work contributes to this digital empowerment.
Providing opportunities for students to acquire a digital literacy that can lead to applied innovation is advantageous in an increasingly digital job market. Digital skills can provide future engineers with the ability to develop innovative tools and solutions within many fields of application, and the opportunity to develop these digital skills should be available and accessible to students in any discipline.

Bibliography

[1] Burning Glass Technologies, "No Longer Optional: Employer Demand for Digital Skills," Department for Digital, Culture, Media and Sport (DCMS), 2019.

[2] GSM Association, "The Mobile Economy 2018," GSMA Intelligence, 2018.

[3] R. H. Kay and S. Loverock, "Assessing emotions related to learning new software: The computer emotion scale," Computers in Human Behavior, vol. 24, p. 1605–1623, 2008.

[4] K. Rex and R. M. Roth, "The Relationship of Computer Experience and Computer Self-Efficacy to Performance in Introductory Computer Literacy Courses," Journal of Research on Computing in Education, vol. 31, no. 1, pp. 14-24, 1998.

[5] R. Craig, "America’s Skills Gap: Why It’s Real, And Why It Matters," Progressive Policy Institute - Radically Pragmatic, Washington, DC, 2019.

[6] Burning Glass Technologies, "Beyond Tech: The Rising Demand for IT Skills in Non-Tech Industries," Oracle Academy, 2019.

[7] Government of Canada, "Essential skills profiles," 2019. [Online]. Available: https://www.canada.ca/en/employment-social-development/programs/essential-skills/profiles.html. [Accessed Feb 2020].

[8] B. A. Myers, A. J. Ko and M. M. Burnett, "Invited Research Overview: End-User Programming," in CHI 2006, Montreal, 2006.

[9] C. Watson and F. W. Li, "Failure Rates in Introductory Programming Revisited," in Proceedings of the 2014 conference on Innovation technology in computer science education (ITICSE '14), New York, 2014.

[10] J. R. Anderson, R. Farrell and R. Sauer, "Learning to Program in LISP," Cognitive Science, vol. 8, no. 2, pp. 87-129, 1984.

[11] E. Soloway and J. C. Spohrer, Studying the Novice Programmer, Michigan: Psychology Press, Taylor & Francis Group, 1989.

[12] E. W. Dijkstra, "On the cruelty of really teaching computing science," Communications of the ACM, vol. 32, no. 12, pp. 1398-1404, 1989.

[13] T. Jenkins, "On the Difficulty of Learning to Program," in Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences., Loughborough, Leicestershire, 2002.

[14] A. J. Ko, B. A. Myers and H. H. Aung, "Six Learning Barriers in End-User Programming Systems," in IEEE Symposium on Visual Languages-Human Centric Computing, Rome, Italy, 2004.

[15] B. d. Boulou, "Some difficulties of learning to program," Journal of Educational Computing Research, vol. 2, no. 1, pp. 57-73, 1986.

[16] C. Kelleher and R. Pausch, "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers," ACM Computing Surveys (CSUR), vol. 37, no. 2, pp. 83-137, 2005.

[17] E. E. Gallardo-Echenique, J. M. de Oliveira, L. Marqués-Molias and F. Esteve-Mon, "Digital Competence in the Knowledge Society," MERLOT Journal of Online Learning and Teaching, vol. 11, no. 1, pp. 1-16, 2015.

[18] E. Hargittai, "Survey Measures of Web-Oriented Digital Literacy," Social Science Computer Review, vol. 23, no. 3, pp. 371-379, 2005.

[19] A. Scheerder, A. van Deursen and J. van Dijk, "Determinants of Internet skills, uses and outcomes. Asystematic review of the second- and third-level digital divide," Telematics and Informatics, vol. 34, pp. 1607-1624, 2017.

[20] A. D. Ritzhaupt, F. Liu, K. Dawson and A. E. Barron, "Differences in Student Information and Communication Technology Literacy Based on Socio-Economic Status, Ethnicity, and Gender," Journal of Research on Technology in Education, vol. 45, no. 4, pp. 291-307, 2013.

[21] A. Martin, "Digital Literacy for the Third Age: Sustaining Identity in an Uncertain World," eLearning Papers, vol. 12, 2009.

[22] C. Scaffidi, M. Shaw and B. Myers, "Estimating the Numbers of End Users and End User Programmers," in Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC’05), 2005.

[23] P. K. Chilana, C. Alcock, S. Dembla, A. Ho, A. Hurst, B. Armstrong and P. J. Guo, "Perceptions of Non-CS Majors in Intro Programming: The Rise of the Conversational Programmer," in 2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC), 2015.

[24] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, M. Burnett, M. Erwig, C. Scaffidi, J. Lawrence, H. Lieberman, B. Myers, M. B. Rosson, G. Rothermel, M. Shaw and S. Wiedenbeck, "The state of the art in end-user software engineering," ACM Computing Surveys, vol. 43, no. 3, pp. 1-44, 2011.

[25] Engineers Canada, "Accredited Engineering Programs in Canada by Program," 2020. [Online]. Available: https://engineerscanada.ca/accreditation/accredited-programs. [Accessed Jan 2020].

[26] Direction de l’Éducation des adultes et de la formation professionnelle, Program of Study: Computer Science, Québec: Ministère de l’Éducation et de l’Enseignement supérieur, 2020.

[27] English Montreal School Board, "STEAM: Science, Technology, Engineering, Arts and Mathematics," 2020. [Online]. Available: https://www.emsb.qc.ca/emsb/schools/initiatives/steam. [Accessed Jan 2020].
[28] Ministère de l’Éducation et de l’Enseignement supérieur, "Sciences informatiques et mathématiques (200.C0): Programme d’études préuniversitaires," Gouvernement du Québec, 2019.
[29] Ministère de l’Éducation et de l’Enseignement supérieur, "Trousse d'informations sur l'offre éducative du Québec," Gouvernement du Québec, 2018.
[30] J. A. G. M. van Dijk and A. J. A. M. van Deursen, "Chapter 6 - Solutions: Learning Digital Skills," in Digital Skills Unlocking the Information Society, New York, Palgrave Macmillan, 2014, pp. 113-138.
[31] Statistics Canada, "Canadian Internet Use Survey," 29 10 2019. [Online]. Available: https://www150.statcan.gc.ca/n1/daily-quotidien/191029/dq191029a-eng.htm. [Accessed 01 2020].
[32] A. van Deursen and J. van Dijk, "Internet skills and the digital divide," New media & society, vol. 13, no. 6, pp. 893-911, 2011.
[33] M. W. Liberatore, D. W. Marr, A. M. Herring and J. D. Way, "Student-Created Homework Problems Based on YouTube Videos," Chemical Engineering Education, vol. 47, no. 2, pp. 122-132, 2013.
[34] W3Schools, "Java Introduction," 2020. [Online]. Available: https://www.w3schools.com/java/java_intro.asp. [Accessed Jan 2020].
[35] Khan Academy, "Computer programming," 2020. [Online]. Available: https://www.khanacademy.org/computing/computer-programming. [Accessed Jan 2020].
[36] Codecademy, "Learn How to Code," 2020. [Online]. Available: https://www.codecademy.com/learn/learn-how-to-code. [Accessed Jan 2020].
[37] University of Toronto, "Learn to Program: the Fundamentals," 2020. [Online]. Available: https://www.coursera.org/learn/learn-to-program. [Accessed Feb 2020].
[38] R. Silber, "'I Want to Learn Programming but I Don’t Know where to Start',' 11 Dec 2017. [Online]. Available: https://codeburst.io/i-want-to-learn-programming-but-i-dont-know-where-to-start-80dd2d55e1fd. [Accessed Dec 2019].
[39] M. Pinola and G. Sin, "The Best Ways to Teach Yourself to Code," 30 July 2019. [Online]. Available: https://lifehacker.com/top-10-ways-to-teach-yourself-to-code-1684250889. [Accessed Dec 2019].
[40] E. Trautman, "Why Learning to Code is So Damn Hard," 4 Feb 2015. [Online]. Available: https://www.thinkful.com/blog/why-learning-to-code-is-so-damn-hard/. [Accessed Dec 2019].
[41] H. Srivastava, "How to learn programming?," 2020. [Online]. Available: https://hackr.io/blog/how-to-learn-programming. [Accessed Jan 2020].
[42] B. R. Barricelli, F. Cassano, D. Fogli and A. Piccinno, "End-user development, end-user programming and end-user software engineering: A systematic mapping study," The Journal of Systems and Software, vol. 149, p. 101–137, 2019.
[43] M. Mäkinen, "Digital Empowerment as a Process for Enhancing Citizens' Participation," E-Learning and Digital Media, vol. 3, no. 3, pp. 381-395, 2006.