Planning K-8 Computer Science through the UDL Framework

Maya Israel
University of Illinois-Urbana Champaign
Champaign, IL
misrael@Illinois.edu

Todd Lash
University of Illinois-Urbana Champaign
Champaign, IL
toddlash@Illinois.edu

Lionel Bergeron
Dept. Affiliation, School/Corp.
City, State, Country
email address@net.com

Meg Ray
Cornell Tech
New York City, New York
Meg.ray@cornell.edu

Abstract

Due to the historic underrepresentation of women, people with disabilities, and people from a broad range of cultural and socioeconomic backgrounds in computer science (CS), there is growing demand to provide CS opportunities in K-12 settings. As a result, school districts are beginning to implement CS For All initiatives. Consequently, there are increasing demands on teachers to find ways of making CS instruction accessible and engaging to a broad range of learners. Unfortunately, few resources exist to guide teachers in how to engage a broad range of learners in CS instruction. This paper describes the rationale for using the UDL framework for designing K-8 CS instruction as well as preliminary findings from a National Science Foundation STEM+C study that examines instructional strategies that support students with disabilities and other struggling learners through Universal Design for Learning (UDL). Secondly, this paper describes the collaborative efforts between the Creative Technology Research Lab (CTRL) and New York City's K-12 CS For All initiative in support of teachers in implementing universally designed CS instruction across grades K-8. Lastly, this paper provides resources developed through these initiatives. Implications for practice and future research directions will be shared.

Keywords

K-12 Computer Science Education

INTRODUCTION

Over the past few years, there has been increasing demand to expand opportunities for all learners to receive computer science (CS) education across K-12 settings. Whereas in the past, CS opportunities were often only available to students from affluent backgrounds, within informal educational settings, or within enrichment programs for students who qualified based on grades or academic status, there is now a call to increase access to a broader range of learners. Consequently, CS education has proliferated within K-12 school settings and is now available to students in greater numbers than ever before. This focus on increasing CS opportunities for a broader range of learners is often referred to as CS For All and focuses on equity from multiple perspectives including socio-economic, culture, gender, and ability. The focus on CS For All stems from the historical underrepresentation of women, people from different cultural backgrounds, and people with disabilities in the CS fields. Equitable access to CS education is imperative, given the prevalence of technology in everyday life and the opportunities that CS skills can provide (Bobb, 2016).

Despite the commitment of the CS education field to increasing equity within CS education, there is still limited guidance for K-12 teachers on how to support a broad range of learners in CS education. This lack of guidance is especially evident when examining resources available to teachers about increasing access and engagement of students with disabilities and other struggling learners (Ladner & Israel, 2016). Several studies, however, have highlighted how the UDL framework can be used to increase access and engagement for students with disabilities. However, Ray, Israel, Lee, and Do (2018) found that teachers often implemented UDL in a narrow manner, by over-emphasizing the role of student choice without considering a more holistic view of UDL. There is, therefore, a great need to expand the interpretation of UDL within the CS education community.

Background

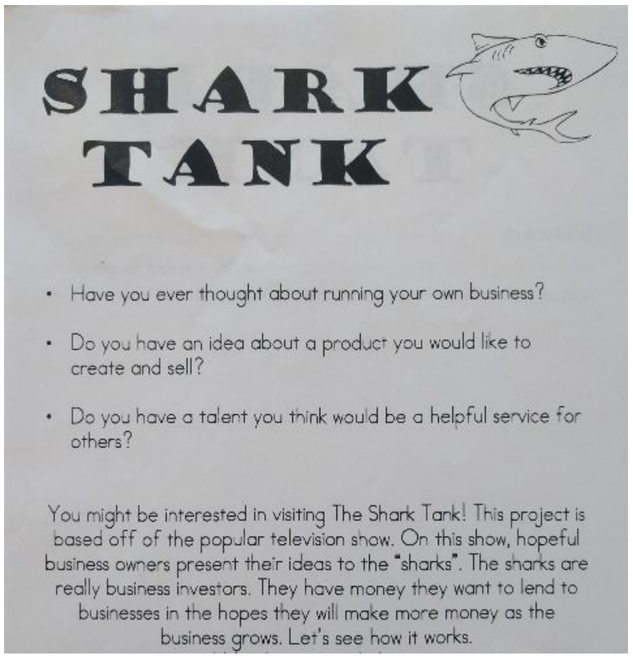
The need to better articulate how UDL can be implemented in the context of CS education has led to an NSF-funded project as well as a collaborative effort between the Creative Technology Research Lab (CTRL) at the University of Illinois, Cornell Tech, and the New York City Department of Education's Computer Science For All Initiative. The first project, Project TACTIC (Teaching All Computational Thinking through Inclusion and Collaboration; https://ctrl.education.illinois.edu/TACTICal), investigates pedagogical approaches, including UDL, that support students with disabilities as well as students at risk for academic failure in elementary and middle school CS activities. This project includes school sites both in the Midwest and the Northeast that have CS For All initiatives. The second project, UDL in CS Ed, seeks to provide both instructional supports and coaching to teachers who are implementing CS education in academically diverse K-8 classrooms in New York City Public Schools. Through these projects, we seek to both understand how to support teachers in teaching CS through the UDL framework and provide resources and supports for teachers in implementing UDL-based CS instruction. Thus, there is an interconnection between the STEM+C research and practical implications of that research in a large urban district that can implement findings as well as inform the research efforts.

UDL IN THE CONTEXT OF CS FOR ALL

Interest in and access to a wide range of CS experiences for learners in K-12 (e.g., Grover, Pea, & Cooper, 2015; Lee, Martin, & Apone, 2014) has grown over the last several years; however, the research base on CS/CT instructional support for students with disabilities is currently meager (Snodgrass, Israel, & Reese, 2016). As CS becomes more a part of the general curriculum, how do we ensure access, participation and progress for all students? At the elementary level, the majority of CS practices are shaped by a "low floor, high ceiling" principle (Grover, Pea, & Cooper, 2015). CS activities often begin with a concrete level of experience or focus on engagement, creativity, and collaboration (i.e., low floor). CS activities must also conform to the "high ceiling" condition as well, allowing the student to move beyond entry level learning experiences. This allows students considerable and continual opportunities for achievement.

If students with disabilities and other struggling learners are to be successful in CS activities, it is essential that they receive the supports allowing them to be able to not only access and engage in introductory computing experiences (i.e., low floor) but also continue to advance and expand their computing skills (i.e., high ceiling) over time. Due to CS education's relative nascence, the opportunity exists to embrace this new subject by considering the largest variation of learners from the beginning. By adopting the UDL framework from the onset, and acknowledging that curriculum itself is often the barrier, we can help to assure that all students have opportunities to advance beyond the ground floor and make real progress in their CS education.

Unfortunately, many teachers new to CS education are also new to UDL. This, "twice new" condition, coupled with a paucity of research-based CS pedagogy, means that teachers often struggle to provide quality CS instruction that meets the needs of all learners. Many CS teachers, previously unaware of UDL, struggle with how individual needs fit into a whole-class framework. Furthermore, even those teachers who believe it to be important, often do not know where to begin in terms of implementing CS instruction utilizing the UDL framework. Though UDL may make sense to them conceptually, it is often hard for teachers to see how to apply it to CS activities. For example, in one of our studies (Ray, Israel, Lee, & Do, in press), ten teachers taught CS education within academically diverse classroom contexts. Of these ten teachers, only six attempted to use the UDL framework. When examining their UDL implementation, however, it was found that the teachers had an extremely limited interpretation of UDL. In fact, they only equated UDL with two instructional practices. The first instructional practice provided students with choice. However, almost all the options for choice were limited to options that elicited increased engagement, recruitment of interest, and sustained effort and persistence. Options for choice did not include how to express their understanding or how content would be represented. The second instructional practice that the teachers implemented was representing content through flexible problem-solving protocols. These problem-solving protocols often provided students with guidance about directions as well as flexibility in options within projects. Figure 1 provides an artifact of a project planning guide that a teacher produced to recruit and guide student interest through the process of designing a digital project that aligns to their own ideas and interests.


SHARK TANK

- Have you ever thought about running your own business?
- Do you have an idea about a product you would like to create and sell?
- Do you have a talent you think would be a helpful service for others?

You might be interested in visiting The Shark Tank! This project is based off of the popular television show. On this show, hopeful business owners present their ideas to the "sharks". The sharks are really business investors. They have money they want to lend to businesses in the hopes they will make more money as the business grows. Let's see how it works.

Figure 1. Teacher Created Project Planning Guide

Our research has also shown that there are specific challenges and opportunities to CS education for a broad range of learners and that both must be addressed in order to provide students with engaging and accessible CS activities. For example, K-8 CS activities often include ill-defined problems that require students to engage in complex multi-step problem solving. This open exploration or semi-open exploration can result in students engaging in creative problem solving that may be highly motivating. At the same time, these ill-defined problems and open exploration can also be coupled with inaccessible curricula, low expectations from teachers about who can engage in complex problem solving within the context of CS, and lack of well-defined instructional strategies and practices (Ladner & Israel, 2016). Thus, in many cases, students with disabilities and other struggling learners are being set up for failure because of lack of access and instructional supports.

Supporting Teachers in Implementing UDL within CS for All

Our research has indicated that teachers needed extensive support in implementing UDL-based CS education. For teachers to implement UDL-based CS instruction, they must have expertise both in CS education and in UDL. Unfortunately, given how novel CS education is in most school districts, many teachers who teach CS are new to both CS and UDL. Therefore, any support that teachers receive in implementing UDL within the context of CS education must take into account that they are often novices in both areas. Support would ideally involve a comprehensive approach that includes professional development, instructional coaching, and access to curriculum that is aligned with the UDL framework.

Professional Development (PD)

A major component of supporting teachers in implementing UDL-based CS instruction is providing PD that focuses both on CS education and applying the UDL framework to CS instruction. If teachers are expected to implement CS activities in the 4th grade, for example, they must first learn about the available CS education curricula and have opportunities to practice using those curricula. Additionally, if teachers have not had experience with the programming platforms used alongside those curricula, they should have opportunities to learn how to use those programming platforms themselves so that they are comfortable introducing it to their students. At the same time, if teachers are going to be expected to teach CS through the UDL framework, they must also learn strategies for teaching that curriculum through the UDL framework. Because most curricula are not aligned to the UDL framework, PD must provide teachers with the time to consider ways of integrating UDL into the CS curricula available to them.

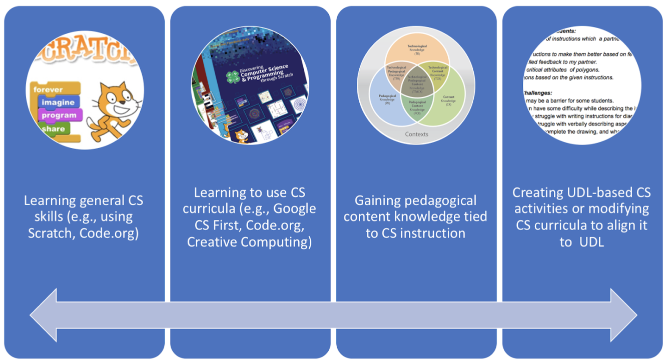

Learning general CS skills (e.g., using Scratch, Code.org)
Learning to use CS curricula (e.g., Google CS First, Code.org, Creative Computing)
Gaining pedagogical content knowledge tied to CS instruction
Creating UDL-based CS activities or modifying CS curricula to align it to UDL

Figure 2. Continuum of UDL-based PD

As Figure 2 showcases, PD must include four interrelated components of CS education: (a) CS-specific skills such as learning how to use the Scratch, JavaScript, or other CS education programming language; (b) CS curricula that are used in schools such as Creative Computing (http://scratched.gse.harvard.edu/guide/) or Code.org that likely are not aligned with the UDL framework (c) pedagogical content knowledge (Mishra & Koehler, 2006) tied to CS/CT instruction that provide context for how the technologies used in CS education can be integrated into instruction; and (d) applying the information about curricula, technology, and programming environments to developing UDL-based CS activities or modifying existing activities to align instruction with the UDL framework.

Embedded Instructional Coaching

In addition to professional development, instructional coaching has been widely cited as a form of professional development that improves teachers' pedagogical content knowledge and allows them to learn within the context of classroom implementation (Knight, 2009; Darling-Hammond, 2006). As teachers take the leap to adding both CS content and a UDL approach to their classrooms, a coach or master teacher can utilize a cycle of observation and reflection to give timely feedback as well as model UDL strategies through co-planning and co-teaching (see Figure 3). In our projects, we have found that teachers report an increased level of confidence in CS instruction and willingness to implement CS in the classroom when they participate in instructional coaching. Teachers report that the supportive, non-judgmental relationship with a coach is a motivator for them to move out of their comfort zone and try new strategies. This also allows the coach to see how UDL is being implemented in the CS classroom and to address potential missed opportunities to increase access for students.

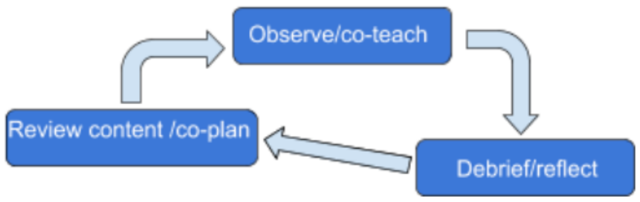


Figure 3. UDL-based CS Education Coaching Cycle

CS Curricula Aligned to the UDL Framework

Because current K-12 CS curricula are not developed through the UDL framework, teacher support often involves co-constructing UDL-based instructional experiences with teachers to help them move towards UDL-based CS instruction. Common starting points used in our projects have included showing examples of how the UDL framework can be used to add more flexibility to existing curricula. Although this is not an ideal starting point, as it involves retrofitting existing instructional materials, it is often an initial launching point. Figure 4, for example, provides a screenshot of a tool that provides examples of ways of addressing representation, action/expression, and engagement within CS activities. This tool is continually revised and as additional ideas are developed, they are added to the table. The complete table can also be accessed at https://goo.gl/uzhqqP.

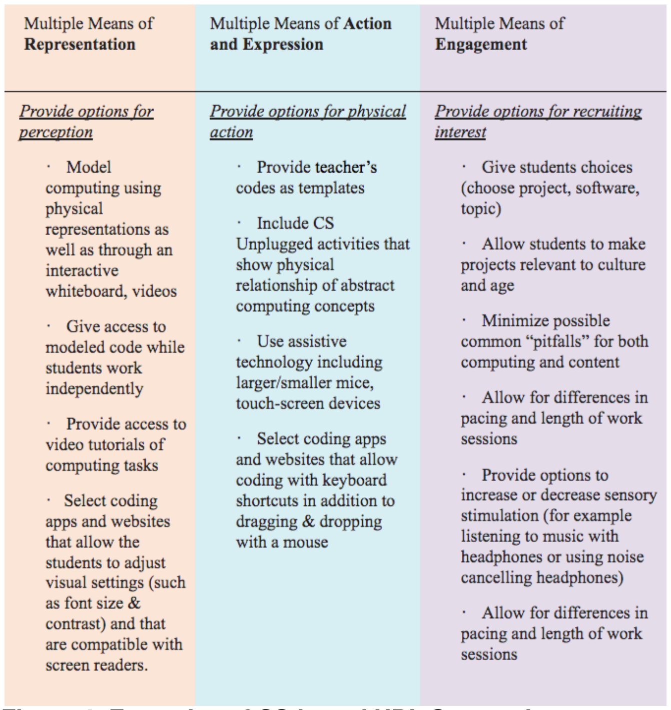
By encouraging teachers to make changes to curricula that they are currently using, they begin to understand how adding flexibility into instruction and proactively addressing the barriers created by existing curricula can increase engagement and learning for their students.

Another approach our project has taken is to develop and test UDL-based CS instructional units and toolkits for the teachers. These resources are developed by collaborative teams of researchers, curriculum developers and teachers and are currently being field tested. The units use the UDL-IRN instructional process (UDL, 2011) and include: (a) student outcomes, (b) an explanation of the barriers that we anticipate the students will face with the content, software, and activities, and (c) instructional approaches to address those barriers. For example, we have found that students are often motivated to create computational programs that demonstrate their mathematical understanding. They enjoy creating their own story problems, illustrating those stories in Scratch, and interacting with their peers' Scratch-based story problems. However, we observed that students often struggle with project planning for complex multi-step computing activities. Thus, new instructional materials proactively include project planning pages to help students plan out their ideas. These planning documents can then be used to communicate ideas with the teachers and peers if the students need additional support.

In addition to these units, there are current efforts to develop tools for curriculum and lesson writing. These tools provide teachers with strategies for considering UDL integration alongside the engaging CS activities, software, and hardware that are available to them. For example, if teachers are going to use the BBC micro:bit (http://microbit.org/), which is a small, programmable device with LED lights, programmable buttons, and sensors, they may need guidance in how to use these devices in a manner consistent with the UDL framework. The tools being created guide teachers in considering features of the micro:bit that can be engaging, methods that it can be used to offer students flexible opportunities for action and expression (e.g., use of sound, lights, and sensors), and ways of representing content (e.g., using the emulator to represent content in a manner wherein students can test their programs immediately.) Thus, when teachers can see that they can use available resources alongside the micro:bit, they can build-in ways of representing content that is more accessible and provide opportunities for students to demonstrate their learning in more flexible and engaging ways.

UDL in Practice: First Steps for Teachers

How do teachers begin integrating UDL into their practice? Our research has shown that starting small is important. This is especially true for teachers who are new to both CS and UDL. Many of the teachers in both our STEM+C study as well as in our implementation work in the NYC DOE have described feeling overwhelmed by the new CS content as well as by teaching this new content to a broad range of academically diverse learners. By integrating UDL gradually, the teachers are able to build an experience base and increase self-confidence, which they can then apply to increasingly larger projects. Focusing on barriers to learning early in the process is also advantageous. In doing so, teachers can isolate which checkpoints they need to begin with, while relieving themselves of the burden of trying to meet all the checkpoints in the entire UDL framework at once. As teachers grow they can reflect on their successes with UDL in other contexts, such as math, and apply those ideas to CS. For example, if multiple means of representation in mathematics means that you provide students with options for manipulatives and the use of video to reinforce learning, these same strategies can be used in CS education by using "unplugged" activities (those which do not require a computer) and video examples. Finally, cultivating the mindset that UDL is a dynamic process is tremendously freeing and allows teachers the freedom to take small steps in honing their UDL craft. Much like CS, UDL is all about iteration. Practitioners utilizing UDL engage themselves in a cycle of planning, implementation and incremental improvement.



Multiple Means of **Representation**	Multiple Means of **Action and Expression**	Multiple Means of **Engagement**
Provide options for perception	*Provide options for physical action*	*Provide options for recruiting interest*
· Model computing using physical representations as well as through an interactive whiteboard, videos	· Provide teacher's codes as templates	· Give students choices (choose project, software, topic)
· Give access to modeled code while students work independently	· Include CS Unplugged activities that show physical relationship of abstract computing concepts	· Allow students to make projects relevant to culture and age
· Provide access to video tutorials of computing tasks	· Use assistive technology including larger/smaller mice, touch-screen devices	· Minimize possible common "pitfalls" for both computing and content
· Select coding apps and websites that allow the students to adjust visual settings (such as font size & contrast) and that are compatible with screen readers.	· Select coding apps and websites that allow coding with keyboard shortcuts in addition to dragging & dropping with a mouse	· Allow for differences in pacing and length of work sessions
		· Provide options to increase or decrease sensory stimulation (for example listening to music with headphones or using noise cancelling headphones)
		· Allow for differences in pacing and length of work sessions

Figure 4. Examples of CS-based UDL Suggestions

UDL in Practice: The CS for All Initiative in New York City

To contextualize the integration of UDL into K-8 CS education, it is helpful to consider the three-pronged approach that the New York City Department of Education (NYC DOE) is using to address UDL-based CS For All. The NYC DOE's CS For All initiative began in 2015. Since that time, they have worked to develop a Blueprint for CS education in NYC. The purpose of this Blueprint is to support schools, professional development partners, and other stakeholders in planning K-12 CS instruction. Therefore, it was critical to proactively consider UDL within the context of the Blueprint. For example, the Blueprint's Educator Resource page provides teachers with resources and examples on how to begin implementing UDL and evaluating content and curriculum from a UDL lens (https://blueprint.cs4all.nyc/resources/12/). These resources, along with the learner perspectives provided in the Blueprint (https://blueprint.cs4all.nyc/perspectives/), allow for multiple entry points for CS education for teachers and students by clearly demonstrating a progression of understanding for CS concepts and practices.

Along with the district-wide efforts articulated in the Blueprint, the NYC DOE is also focused on curriculum development. This work is done by “Blueprint fellows” who are tasked with developing meaningful units based on the UDL principles and guidelines. The Blueprint fellows also evaluate lessons and units that have already been developed from a UDL lens. Lastly, there is concerted effort to provide teachers and curriculum writers with professional development (PD) focused on aligning CS education opportunities with the UDL framework. The PD includes workshops focused on developing teachers’ capacity not only in CS-specific tools and resources, but also on pedagogical approaches aligned with UDL. These workshops and PD opportunities, thus, focus on engaging and supporting all learners by implementing pedagogical approaches consistent with the UDL framework.

CONCLUSIONS

This paper focuses on strategies for supporting teachers in providing K-8 CS education through the UDL framework. It describes the rationale for using the UDL framework within CS education as well as practical strategies that have arisen from both an NSF STEM+C project and a collaborative effort with the NYC DOE’s CS For All initiative. These efforts have solidified the need to support teachers in better meeting the needs of ALL within CS For All. These efforts are also now being implemented in research studies examining student outcomes including effects on persistence, collaborative problem solving, and understanding of computational thinking concepts. The UDL framework provides the mechanisms for addressing the broadest range of students engaged in CS education. Future research must be conducted on each of the components described in this paper: UDL-based professional development, coaching, and curriculum design.

ACKNOWLEDGMENTS

Funding for this research was provided by the National Science Foundation (award #1639837). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the NSF.

REFERENCES

Bobb, K. (2016). Broadening Participation in Computing: A Critical Perspective. *ACM Inroads*, *7*(4), 49-51.

Darling-Hammond, L. (2006). Constructing 21st-century teacher education. *Journal of Teacher Education*, *57*(3), 300-314.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. Computer Science Education, 25(2), 199–237. http://doi.org/10.1080/08993408.2015.1033142

Knight, J. (2009). Instructional Coaching. In *Coaching: Approaches and Perspectives* (pp. 29–55). Thousand Oaks, CA: Corwin Press.

Ladner, R., & Israel, M. (2016). "For all" in "computer science for all". Communications of the ACM, 59(9), 26-28.

Lee, I., Martin, F., & Apone, K. (2014). Integrating Computational Thinking Across the K – 8 Curriculum. ACM Inroads, 5(4), 64–71. http://doi.org/10.1145/2684721.2684736

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record*, *108*(6), 1017.

Ray, M., Israel, M., Lee, C., & Do, V. (in press). A cross-case analysis of instructional strategies to support participation of K-8 students with disabilities in CS for All. *In Proceedings of the Association for Computing Machinery (ACM) Technical Symposium on Computer Science Education.* SIGCSE ACM.

Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. Computers and Education, 100(September), 1–17. http://doi.org/10.1016/j.compedu.2016.04.011

UDL-IRN (2011) UDL in the Instructional Process. Version 1.0. Lawrence, KS: Author.