2017

# TPACK learning activity types for secondary computer science courses

Rebecca Carton
*University of Northern Iowa*

# TPACK learning activity types for secondary computer science courses

**Abstract**
Learning activity types for secondary computer science courses support educators in integrating technology and developing their TPACK (technological, pedagogical, and content knowledge) authentically. The taxonomy of computer science activity types presented in this project report provide seven identified activity types and descriptions aligned with the Computer Science Teachers' Association (CSTA) standards and framework. Included in the taxonomy are possible technologies for each activity type. Along with the CSTA standards and framework, ten peer-reviewed studies published between 2009 and 2015 were selected for analysis in the literature review and as research backing the construction of the taxonomy. Further research and expansion of the computer science learning activity types and technologies was recommended.

# TPACK Learning Activity Types for Secondary Computer Science Courses

A Graduate Project Report

Submitted to the

Division of Instructional Technology

Department of Curriculum and Instruction

In Partial Fulfillment

Of the Requirements for the Degree

Master of Arts

UNIVERSITY OF NORTHERN IOWA

by

Rebecca Carton

August, 2017

Running head: TPACK FOR LEARNING ACTIVITY TYPES

This Report by: Rebecca Carton

Titled: TPACK Learning Activity Types for Secondary Computer Science Courses

has been approved as meeting the research requirement for the

Degree of Master of Arts.

| | |
|---|---|
| _____ | _____ |
| Date Approved | Graduate Faculty Reader |
| _____ | _____ |
| Date Approved | Graduate Faculty Reader |
| _____ | _____ |
| Date Approved | Head, Department of Curriculum and Instruction |

**Abstract**

Learning activity types for secondary computer science courses support educators in integrating technology and developing their TPACK (technological, pedagogical, and content knowledge) authentically. The taxonomy of computer science activity types presented in this project report provide seven identified activity types and descriptions aligned with the Computer Science Teachers' Association (CSTA) standards and framework. Included in the taxonomy are possible technologies for each activity type. Along with the CSTA standards and framework, ten peer-reviewed studies published between 2009 and 2015 were selected for analysis in the literature review and as research backing the construction of the taxonomy. Further research and expansion of the computer science learning activity types and technologies was recommended.

Keywords: *learning activity types, TPACK, technology integration, secondary, computer science*

## Table of Contents

# TPACK Learning Activity Types for Secondary Computer Science Courses

## Introduction

Our technologically advanced society and workplaces now require students to exit high school with a strong 21st-century skill set and an understanding of how to effectively use technologies. To teach students, teachers must possess these skills themselves and have a deep understanding of content knowledge, pedagogical knowledge, and knowledge of how learning can be supported through technology (TPACK). Content knowledge is an educator's understanding and mastery of their subject area, while pedagogical knowledge is the teacher's skills and awareness of how to teach. Teacher expertise is the union of instructional strategies, curriculum, and digital tools, not just a proficiency in technology skills, and determines the productive integration of educational technologies into the classroom (Hofer, M., & Harris, J., 2010, p. 3857). The complex dynamics of classrooms created by different student learning styles, teacher qualifications, and access to technology further emphasizes the importance of teachers having a strong knowledge foundation via the TPACK framework.

No matter how digitally literate students are, if instructors are not comfortable and literate in the technologies they are using, successful integration will be difficult. According to Matherson, Wilson, & Wright many educators who graduated before 2005 lacked the technical knowledge, skills, and experiences necessary because they were not taught with technology nor were they immersed in a "technology-soaked society" (2014, p. 46). Teachers may feel pressured to force technology into their instruction, even when it is not appropriate, supportive of the content standards, or focused on the learning goals. Harris and

Hofer consider that placing this emphasis on technology instead of the students and goals as being "technocentric" and flawed (2009, p.23-25). The authors suggest that emphasis should be placed upon student needs and course objectives first, with technology being an operationalization of TPACK via curriculum-based learning activity types.

This project addresses the issue of teachers using ineffective technocentric views when designing instruction or lacking TPACK knowledge to effectively integrate technology into their classrooms by creating a taxonomy of learning activity types which are supportive of the computer science standards. Specifically, this taxonomy project creates activity types and possible technologies that can support the 2011 Computer Science Teachers Association (CSTA) Standards and the 2016 K-12 Computer Science Framework. Computer science in a secondary classroom is the subject of this taxonomy. To this date, and to my own awareness, there are no learning activity types developed for computer science courses. Being a secondary STEM teacher who teaches multiple computer science courses, it is fitting that I chose this project which is activity types for computer science.

It is the my hope that this taxonomy will be used by educators to expand their repertoire of digital resources and provide instructional activities which support students' curriculum-based learning needs. Modification and expansion of this computer science activity types taxonomy by educational faculty, staff, and researchers are welcomed and encouraged to meet the individual teacher preferences and student population. Due to technologies rapidly improving and evolving, this project is not a complete list of all possible technologies which meet the CSTA standards and practices but should be seen as a starting point for educators, instructional coaches, researchers, and other individuals in the education community. A table of commonly used terms in this paper and their definition is provided

below for the benefit of the reader.

| Term | Definition | More Information |
|---|---|---|
| TPACK | An educator's content knowledge, pedagogical knowledge, and knowledge of how learning can be supported through technology. TPACK stands for Technological Pedagogical Content Knowledge. | TPACK.org |
| Learning Activity Type | Educational activities categorized by the action students are performing | Learning Activity Types Web Site |
| Taxonomy | Taxonomies created for this project are organized collections of learning activities aligned with the CSTA computer science standards and framework. Included in these taxonomies are brief descriptions of the activities along with possible technologies that can be used for the activity. | "Grounded" Technology Integration: Instructional Planning Using Curriculum-Based Activity Type Taxonomies |
| CSTA Standards | The 2011 Computer Science Teachers Association (CSTA) Standards for grades K-12 identifies the specific skills and knowledge students need for computer science courses. | CSTA Computer Science Standards |
| CSTA Framework | The 2016 Computer Science Teachers Association (CSTA) Framework defines the core concepts and practices students should experience and build upon in K-12 computer science courses. | CSTA Computer Science Framework |

## Literature Review

Research for this project focused on curriculum-based technology integration with the TPACK (Technological Pedagogical and Content Knowledge) framework and learning activity types. Content is one component of the TPACK knowledge that educators need to possess, so to be able to research TPACK studies and identify learning types I needed first to know and understand the computer sciences standards and practices. Although there is not a national set of standards that is currently adopted for every state, the Computer Science Teachers' Association (CSTA) and Association for Computing Machinery (ACM) constructed a set of computer standards which are widely used or referenced for the creation of state standards. The 2011 CSTA K–12 Computer Science Standards focus on "abstraction, automation, analysis, and computational thinking" for grades K-12 while outlining the skills and knowledge students need to thrive in our global information economy (p. 7).

In addition to the CSTA and ACM 2016 computer science standards, the 2016 K-12 Computer Science Framework also guided research for this learning activity types taxonomy. This framework is a collaboration among CSTA, ACM, Code.org, Cyber Innovation Center, National Math and Science Initiative, state governments, and school districts. The goal of this Computer Science Framework is to provide a guide for schools and states to design computer science curriculum and standards which provide opportunities for all students to succeed.

Five core concepts (Computing Systems, Networks and the Internet, Data and Analysis, Algorithms and Programming, and Impacts of Computing) are identified in the Computer Science Framework as the major content areas in the field of computer science.

Seven core practices in the framework" describe the behaviors and ways of thinking that computationally literate students use to fully engage in today's data-rich and interconnected world" (Fostering an Inclusive Computing Culture, Collaborating Around Computing, Recognizing and Defining Computational Problems, Developing and Using Abstractions, Creating Computational Artifacts, Testing and Refining Computational Artifacts, Communicating About Computing) ("K–12 Computer Science Framework," 2016, p. 67). Since learning activity types are based on what students actively do, the focus for the creation of this taxonomy project was placed on these seven core practices. .

Technology tools for education should support the curriculum standards, typically be close to last in the educator's planning process, and address students' learning needs and objectives. The TPACK framework outlines the knowledge educators need to effectively integrate educational technologies and provides a "common language" for teachers, curriculum specialists, administrators, and IT coordinators (Harris & Hofer, 2014, p. 2309). It is not enough for teachers to be literate in technology, they must also have a strong foundation in the content area as well as the pedagogical choices that are most effective. According to Baran, Chuang, and Thompson, "teachers who have [an understanding of] TPACK, act with an intuitive understanding of the complex interplay between the three basic components of knowledge" (2011, p. 371). This knowledge comes from experience, professional development, and an awareness of the content standards, student needs, and technology integration as a supporting role.

Mishra, Harris, and Koehler (2009) report that there is a "mismatch between educational technology leaders' visions for technology integration and how most practitioners use digital tools" (p. 393). Educators tend to overuse and gravitate towards

presentation software, learner-friendly websites, and classroom management tools. This "routinization" of using the same technologies and activities at a great frequency can result in lost opportunities for engaging and productive learning (Hofer & Harris, 2015, p. 7-2). TPACK knowledge should guide educational technologies selected by teachers, which support the curriculum standards, student inquiry, and collaboration.

However, "TPACK is not easily applied, learned, or taught, it is professional knowledge developed over time" (Harris & Hofer, 2009, p. 100). Thus, teachers need to develop their TPACK knowledge actively. Using curriculum-specific learning activity types in their instructional planning is one method to increase teacher TPACK knowledge. Learning activity taxonomies provide a collection of technologies aligned with curriculum goals, supporting student needs, and for a wide variety of teaching approaches. Through these taxonomies, more teachers can quickly and efficiently implement a broader range of educational technologies.

Blanchard et al. (2010) define the selection of learning activities which match content goals, student needs, and pedagogy as a "grounded" approach and state that these taxonomies assist educators in achieving this method (p. 603-604). For teachers to use these technologies and properly integrate them into their teaching, they must first understand how to use and work these digital tools. Therefore, it is important for educators to receive continuous professional development to overcome shortcomings in technology skills and integration (Matherson, Wilson, & Wright, 2014, p. 48).

Teachers are continuously challenged with providing instruction which meets the diverse needs of learners, aligns to content goals and standards, and integrates instructional technologies. Selecting these technologies requires teachers to "make deliberate and critical

choices which match the pedagogically congruent learning activities and goals" (Hofer & Harris, 2010, p. 3862). To accomplish these decisions and assess their technology-integrated lesson plans, teachers apply their TPACK knowledge and experience. Learning activity types taxonomies, such as the computer science taxonomy in this project report and the taxonomies for other subject areas (e.g., math, literacy, social studies, . . . ) listed in Hofer and Harris's 2011 wiki (http://activitytypes.wmwikis.net), present educators with a foundation for their instructional planning and personal growth of TPACK knowledge.

## Project Description

Careers in computer science and information technology continue to grow as technologies are continually advancing and integrating into more workplaces. The developed skills and acquired knowledge such as problem-solving, logical thinking, communication, working with diverse individuals, creativity, technology literacy, and perseverance gained from studying computer science are not only useful for computer science careers, but other jobs, schooling, and everyday life. Although computer science teaches students 21st-century skills and prepares them for the workforce, the views and implementations of many teachers, schools, and politicians are still in progress.

As a secondary STEM (science, technology, engineering, mathematics) educator who teaches computer science courses, I have found that it is easy to fall into the mistake of using technologies just because they are available, but not necessarily supporting the course goals and standards. According to Harris, J., & Hofer, M. (2009), "*technocentric* (selecting digital tools first in the instructional planning) instruction rarely helps students meet content standard goals since the standards were not the focus of the planning" (p. 107). Therefore, creating this taxonomy project which focuses on learning activity types and technologies that align with the CSTA computer science framework and standards are helpful in reducing technocentric instruction.

The process of creating this taxonomy was iterative and involved researching, seeking feedback, and reflection. My experience as a STEM educator along with the information collected through scholarly articles, websites, and experts contributed to the development of this project. The current production is limited by my knowledge and the feedback I received from the STEAM (science, technology, engineering, art, mathematics) Curriculum Director,

Chris Like, at Bettendorf high school and University of Northern Iowa, instructional technology graduate professor, Dr. Leigh Zeitz. The feedback I received from these two individuals was to use the CSTA standards and framework and 21st-century skills to guide the creation of my project as to my knowledge there currently is not published secondary computer science learning activity types which I could use as a guide. I will be refining this taxonomy as I use it for teaching my STEM classes and as I receive feedback from my students.

## Outcome

Currently, there are no nationally adopted computer science standards and according to the Computer Science Teachers Association (CSTA), "roughly two-thirds of the fifty states do not have computer science standards for secondary school education" (2011). Through research, feedback, and nine years teaching experience in STEM courses I was able to produce a taxonomy of computer science learning activity types based on the CSTA K-12 standards and framework practices. This taxonomy is only a starting point for the expansion of identifying computer science learning activity types and I welcome modification, additions, and suggestions. The learning activity types included in this taxonomy are open to a teacher's interpretation and implementation in their classroom based on the specific needs of their students.

### Computer Science Learning Activity Types Taxonomy

The activity types presented in the taxonomy below are derived from the K–12 Computer Science Framework's seven practices (fostering an inclusive computing culture, collaborating, recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, testing and refining, and communicating about computing) (2016). These practices promote and guide quality computer science education for all students. The CSTA K–12 Computer Science Standards, practices focus on engaging students in planning, designing, and creating computational artifacts, approach problems systematically and in creative ways, and participate in real-world computer science issues (2011). Many of the words are drawn from the framework and standards and are provided as action words that state what the student is doing.

As there as seven practices in the framework (see *figure 1*), I have identified seven learning activity types (LAT's) which align with these computer science practices and the CSTA K–12 Computer Science Standards. These learning activity types for computer science are titled *Inclusion, Collaborate, Interpret, Abstract, Develop, Improve, and Communicate*. For each learning activity type, a brief description of the activity is provided along with possible technologies which support the activity type. This taxonomy is by no means a comprehensive and complete list of LATs and technologies but instead presented as an invitation for others to use, add to, and modify. With technologies rapidly improving, becoming outdated, and being modified daily, there is an inherent risk that some of the possible technologies listed below are no longer relevant. I do not necessarily endorse the specific websites, software, and digital tools listed.

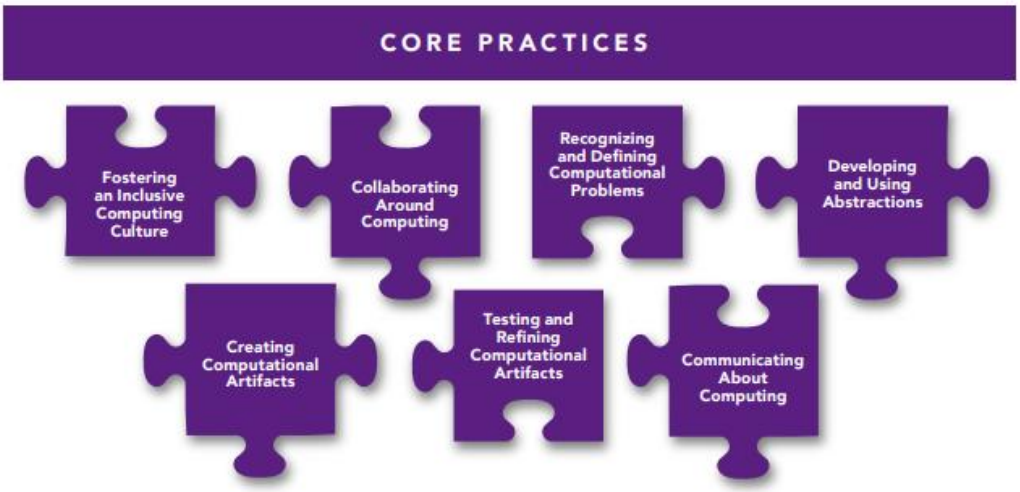


*Figure 1:* CSTA 2016 Computer Science Framework's Seven Core Practices

**The "Inclusion" Activity Types**

Computing in the real-world typically is a team effort, where individuals of diverse backgrounds come together to create, modify, and maintain computational products. To provide students with different perspectives, educators must provide learning environments and activities which are inclusive and collaborative.

Table 1: "Inclusion" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Seek/Analyze Diverse Input | Students seek out and analyze the perspectives of others with diverse backgrounds | Social networking sites, blogs (e.g. Edublogs), wiki (e.g. Wikispaces), online discussion forum (e.g. TodaysMeet), messaging (e.g. Google Hangouts) |
| Evaluate Accessibility | Students evaluate the accessibility of a product or computational artifact | Web Accessibility Initiative, Wave Web Accessibility Evaluation Tool, Usability testing (i.e. Optimal Workshop), peer-review (i.e. NowComment) |
| Identify Bias | Students test for potential bias of a product or computational artifact | Usability testing (i.e. Optimal Workshop), peer-review (i.e. NowComment, publishing online (e.g. GitHub), webquest (e.g. Evaluating Sources) |
| Employ Self-advocacy | Students employ self-advocacy strategies | LMS (e.g. Schoology, Edmodo), Google Classroom, email, messaging (e.g. Remind), Q&A platform (e.g. Piazza) |
| Advocate for Others | Students advocate for the diverse needs of their peers | Online discussion forum (e.g. TodaysMeet), email, messaging (e.g. Remind), Q&A platform (e.g. Piazza) |

**The "Collaborate" Activity Types**

Working in teams (or pairs) rather than individually provides different experiences, perspectives, ideas, and feedback for the development and creation of computation artifacts. Being able to work collaboratively and work through conflict are skills which employers value as it is required for many careers. Therefore, collaborative computing and tools assist computer science students in creating quality computational artifacts.

Table 2: "Collaborate" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Perform Team Role | Students perform a team role and use methods for whole team inclusion | Google apps, blogs (e.g. Edublogs), wiki (e.g. Wikispaces), collaborative tools (e.g. Evernote) |
| Increase Team Productivity | Students evaluate team dynamics and use multiple strategies to increase productivity | Online project spaces (e.g. Padlet, Prezi, TitanPad), Google Hangout communication, knowledge sharing tools (e.g. Diigo) |
| Improve Workflow | Students control and evaluate workflow | Digital agendas and timelines (e.g. Google Calendar), project management tools (e.g. Bitrix24, Asana) |
| Give/Receive Feedback | Students give and receive feedback on their computing and projects | Pair programming (e.g. CodeStudio), screen sharing (e.g. ScreenHero), Google Docs, online feedback (e.g. Peergrade) |
| Select/Evaluate Tools | Students select and evaluate collaboration tools | Interactive whiteboard, online forums, wiki (e.g. Wikispaces), blogs (e.g. Edublogs) |

**The "Interpret" Activity Types**

Identifying whether a problem can be solved using a computational approach is a skill that comes from experience and time. Being able to address an issue through computation requires being able to define the problem and then break the larger problem into parts to be analyzed. Students need multiple opportunities to identify, interpret, and solve problems which can be solved with computation to build this skill set.

Table 3: "Interpret" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Identify Problems | Students identify real-world problems which can be solved computationally | Online coding challenges (e.g. CodeEval), development platforms/communities (e.g. GitHub, StackOverflow) |
| Decompose Problems | Students decompose real-world problems into more manageable subproblems | Mindmapping/brainstorming tools (e.g. Popplet, Coggle, MindMup), interactive whiteboard, online whiteboard (e.g. Realtime Board) |
| Evaluate Problems | Students evaluate problems to determine if they can be solved computationally | Online project spaces (e.g. Padlet, Prezi, TitanPad), Coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ) |
| Discuss Problems | Students discuss and ask clarifying questions about a problem's ability to be solved with a computational approach | Online discussion forum (e.g. TodaysMeet), interactive whiteboard, Q&A platform (e.g. Piazza) |

**The "Abstract" Activity Types**

Abstraction is a fundamental concept in programming as it simplifies the complexity and development of computational artifacts. For students to be able to form abstractions they must be able to identify patterns and common features of problems. Students must be able to create systems of modules, standalone parts of a program, through subdividing the main program.

Table 4: "Abstract" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Identify/Extract Patterns | Students identify and extract patterns which are opportunities for abstraction | Mindmapping/brainstorming tools (e.g. Popplet, Coggle, MindMup), interactive whiteboard, online whiteboard (e.g. Realtime Board) |
| Simplify Complex Code | Students substitute parts of a code for a single segment which uses variables to account for any differences | Coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ), Mindmapping/ brainstorming tools (e.g. Popplet, Coggle, MindMup), Online diagram tools (e.g. draw.io, Google Drawings) |
| Assess/Use Existing Functionalities | Students assess and use existing functions, libraries, and application programming interfaces (APIs) | Coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ), development platforms/communities (e.g. GitHub, StackOverflow) |
| Design/Create Modules | Students design and create systems of interacting modules and abstractions | Coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ), online diagram tools (e.g. draw.io, Google Drawings) |
| Model/Simulate | Students represent patterns, | Presentation tools (e.g. Prezi, Google |

| | | |
|---|---|---|
| | processes, or phenomena through models and simulations | Slides), video creation (e.g. WeVideo, PowToon), Online diagram tools (e.g. draw.io, Google Drawings) |

**The "Develop" Activity Types**

The production of computational artifacts challenges students to express themselves or to solve problems, using and refining their programming skills. Planning, developing, and refining are processes which are used both in the real-world and classroom to create efficient and quality computational artifacts. These artifacts can be original student creations or a modification/combination of existing ones. Computer programs, robotic systems, mobile and web applications, simulations, animations, and games are all examples of computational artifacts.

Table 5: "Develop" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Plan/Design Artifacts | Students plan and design computational artifacts. | Mindmapping/brainstorming tools (e.g. Popplet, Coggle, MindMup), Online diagram tools (e.g. draw.io, Google Drawings) |
| Reflect/Modify Development | Students reflect and modify development to reach end goals. | Screen sharing (e.g. ScreenHero), Google Docs, online feedback (e.g. Peergrade) |
| Create Artifacts | Students create computational artifacts to solve problems, express themselves, or complete tasks. | Coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ), Google Docs |

| | | |
|---|---|---|
| Modify Existing Artifacts | Modify, improve, and customize existing artifacts. | Online coding challenges (e.g. CodeEval), development platforms/communities (e.g. GitHub, StackOverflow) |

**The "Improve" Activity Types**

Computer programmers must be able to troubleshoot, debug (identify and correct program errors), test, and refine computational artifacts to enhance their reliability and performance. This process must be iterative and take into consideration the ever-changing needs of end users. Students must act as computer programmers within the classroom, continuously testing and refining their products.

Table 6: "Improve" Activity Types

| **Activity Type** | **Brief Description** | **Possible Technologies** |
|---|---|---|
| Test Artifacts | Students systematically test computational artifacts to determine if criteria and constraints are met. | LMS (e.g. Schoology, Edmodo), Q&A platform (e.g. Piazza), online feedback (e.g. Peergrade), coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ) |
| Debug/Troubleshoot | Students troubleshoot computer systems and systematically debug errors in computational artifacts. | Development platforms/communities (e.g. GitHub, StackOverflow), coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ), Q&A platform (e.g. Piazza) |
| Refine Artifacts | Students evaluate and refine artifacts to enhance their performance and reliability. | Online feedback (e.g. Peergrade), coding learning environments (e.g. CodeStudio, Codecademy, CodeBender, BlueJ) |

**The "Communicate" Activity Types**

Being able to use clear communication with a diverse audience allows individuals to express themselves, collaborate with others, document their work, and explain their thinking. The ability to effectively communicate is a valuable skill for school, work, and everyday life. Computer science students demonstrate and refine their communication skills through completion, collaboration, and presentation of computational artifacts.

Table 7: "Communicate" Activity Types

| Activity Type | Brief Description | Possible Technologies |
|---|---|---|
| Justify with Data Sets | Students communicate an idea through selecting, organizing, and interpreting large data sets. | Mindmapping/brainstorming tools (e.g. Popplet, Coggle, MindMup), online diagram tools (e.g. draw.io, Google Drawings), presentation tools (e.g. Prezi, Google Slides) |
| Document/ Explain | Students use appropriate terminology and documentation to explain their artifacts and processes. | Online terminology glossary (e.g. Java Glossary), online diagram tools (e.g. draw.io, Google Drawings), presentation tools (e.g. Prezi, Google Slides), code documentation generator (e.g. Doxygen, Javadoc) |
| Articulate Ideas Responsibly | Students adhere to copyright laws and give proper attribution to any work borrowed. | Copyright information and checking tools (e.g. Copyright Genie, Fair Use Evaluator), search engines (e.g. Creative Commons, Google), citation generator (e.g. EasyBib, BibMe) |

## Conclusions and Recommendations

Creating this project challenged me to form connections among the CSTA computer science practices and standards and the current technologies available. Through researching the works of Harris, J., Hofer, M., and others I have gained insight into the disadvantages of technocentric instruction, operationalizing TPACK through curriculum-based learning activity types (LATs), and the current state of implementing computer science courses and standards nationwide. Exploring the possible technologies which support these LATs expanded my repertoire of digital tools I can use with my secondary STEM students. Through searching the internet and testing new digital tools, I was able to not only construct this taxonomy but collect new educational technologies that I can use in my classroom.

This taxonomy is meant to be a brief starting resource for educators of all levels of experience in teaching computer science courses which strive to focus on learning objectives and standards first in instructional planning, and then select appropriate technologies for these goals. The rate that technologies are created, improved, and are replaced demands that this taxonomy is not a static artifact, but continually grow and be modified by the STEM education community. Referencing and developing this taxonomy for my classroom use benefits me as a 21st-century educator and helps to support the diverse needs of my students. As each classroom and student population are unique and diverse, it is important for teachers to try out numerous digital tools to determine which ones fit their students' needs and the teacher's pedagogy. I hope that other teachers will find this project useful and time-saving for their instructional planning.

## References

Baran, E., Chuang, H. H., & Thompson, A. (2011). TPACK: An emerging research and development tool for teacher educators. *Turkish Online Journal of Educational Technology*, *10*(4), 370-377.

Blanchard, M., Grandgenett, N., Schmidt, D., van Olphen, M., Harris, J., Hofer, M., & Young, C. (2010). "Grounded" technology integration: Instructional planning using curriculum-based activity type taxonomies. *Journal of Technology and Teacher Education*, *18*(4), 573-605.

CSTA K–12 Computer Science Standards. (2011). Retrieved from http://csta.hosting.acm.org/csta/csta/Curriculum/sub/K12Standards.html

Harris, J., & Hofer, M. (2009). “Grounded” technology integration: Planning with curriculum-based learning activity types. *Learning & Leading With Technology*, *37*(2), 22-25.

Harris, J., & Hofer, M. (2009). Instructional planning activity types as vehicles for curriculum based TPACK development. In Maddux, C. D. (Ed), *Research highlights in technology and teacher education 2009* (pp. 99–108). Chesapeake, VA: AACE.

Harris, J., & Hofer, M. (2014). The construct is in the eye of the beholder: School districts’ appropriations and reconceptualizations of TPACK. In *Society for Information Technology & Teacher Education International Conference* (pp. 2306-2313). Chesapeake, VA: AACE.

Hofer, M., & Harris, J. (2010). Differentiating TPACK development: Using learning activity types with inservice and preservice teachers. In C. D. Maddux, D. Gibson, & B. Dodge (Eds.), *Research highlights in technology and teacher education 2010* (pp.

295–302). Chesapeake, VA: Society for Information Technology and Teacher Education (SITE).

Hofer, M., & Harris, J. (2011). *Learning activity types wiki*. Available: College of William & Mary, School of Education, http://activitytypes.wmwikis.net

Hofer, M., & Harris, J. (2015). Developing TPACK with learning activity types. In M. Hofer, L. Bell & G. Bull (Eds.), *Practitioner's guide to technology, pedagogy, and content knowledge (TPACK): Rich media cases of teacher knowledge* (pp. 7-1 to 7-14). Waynesboro, NC: AACE.

K–12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org

Matherson, L., Wilson, E., & Wright, V. (2014). Need TPACK? Embrace Sustained Professional Development. *Delta Kappa Gamma Bulletin*, *81*(1), 45-52.

Mishra, P., Harris, J.B., & Koehler, M. (2009). Teachers' technological pedagogical content knowledge and learning activity types: Curriculum-based technology integration reframed. *Journal of Research on Technology in Education*, *41*(4), 393-416.