Middle School CS in Science Standards Crosswalk

Project GUTS has developed a series of modules for Code.org that augment existing lessons in Earth, Life and Physical Science curricula. These Code.org "Computer Science in Science" modules integrate computer science through the use, modification, and creation of computer models and simulations within the context of modern scientific practice.

Prior to creating these modules (and/or adapting existing modules from the Project GUTS curriculum for this purpose), crosswalks between the NRC Framework for K-12 Science Education and CSTA K-12 Computer Science Standards; and between the Next Generation Science Standards (NGSS) and the CSTA K-12 Computer Science Standards were conducted to elucidate the commonalities that could serve as the basis for a set of learning outcomes addressed in the Code.org modules.

The first step in creating the crosswalks was to compare the two Standards in terms of the goals, context, approach, breadth, depth, content, and practices included. The document "1. Overview of the CSTA K-12 CS Standards and NGSS" contains a broad comparison of the Standards. The document acquaints the general audience with the two Standards and serves as a foundation for understanding the crosswalks.

The diagram "2. Computational Science in the CSTA and NGSS" frames Code.org's Computer Science in Science modules in the context of Computational Science, Modeling and Simulation, the Computer Science Teachers Association K-12 Computer Science Standards and the Next Generation Science Standards.

Document "3. Augmenting Practices" describes how Computational Science practices augment traditional scientific and engineering practices. Table 3-2 of the "Framework for K-12 Science Education: Practices, Crosscutting Concepts and Core Ideas" served as a starting point for this document. This document provides an introduction to Computational Science for those unfamiliar with the practice.

The "4. Aligning the Framework and CSTA" document contains a preface and the crosswalk between the NRC Framework's Scientific Practices and the CSTA K-12 Computer Science Standards.

The "5. Aligning NGSS DCI ETS and CSTA" document contains a preface and the crosswalk between the NGSS Disciplinary Core Idea of Engineering, Technology and Applications of Science (ETS) and the CSTA K-12 Computer Science Standards.

Comparison of the CSTA K-12 Computer Science Standards and the Next Generation Science Standards

This "Overview" is a broad comparison of the goals, context, approach, breadth/depth, and content /practices included in the CSTA K-12 Computer Science Standards and the Next Generation Science Standards.

n.b.: This document originated from Achieve Inc. and came pre-filled with Achieve's responses to a series of twenty-two questions. The responses to the questions from the perspective of the CSTA K-12 Computer Science Standards were drawn from the CSTA Standards themselves. Subsequently, upon review by Achieve, it was determined that questions 4, 6, 11, and 12 could be omitted. After the questions were removed, the remaining questions were renumbered.

General Information on Standards Development and Design

Question #1: What process was used to develop the standards, including what research and background materials (NSES, etc.) are the standards documents based on?

CSTA : The CSTA K-12 Computer Science standards were developed in a year-long process led by curriculum committee of the CSTA. Members of the committee were assigned to grade bands based on their experience teaching K-12 Computer Science. Three grouping were made, K-5, 6-8, and 9-12. Each group reviewed the existing K-12 CS standards paying special attention to their assigned grade band, and suggested edits and adapted the standards to reflect changes in the field. All drafts of the report were informed by feedback from many organizations and individuals. In all two rounds of review and three drafts were produced. The standards were published on the CSTA Web site (http://csta.acm.org) as well as in hardcopy form. The Computer Science Standards aim to provide a framework within which state departments of education and school districts can revise their curricula to better educate young people in this important subject area and thus better prepare students for effective citizenship in the 21st century. [CSTA K-12 Computer Science Standards, 2011. Pg 1.]

NGSS : The NGSS were developed in a state-led process. Twenty-six states signed on to be Lead State Partners. The states provided guidance and direction in the development of the NGSS to the 41-member writing team, composed of K–20 educators and experts in both science and engineering. In addition to six reviews by the lead states and their committees, the NGSS were reviewed during development by hundreds of experts during confidential review periods and tens of thousands of members of the general public during two public review periods. The NGSS content and structure are based on the National Research Council's Framework for K–12 Science Education (2012), and an NRC review found that the NGSS were faithful to the Framework. [NGSS Introduction (NGSS Lead States, 2013, Vol. I, p. xvi); http://www.nextgenscience.org/lead-state-partners; http://www.nextgenscience.org/writing-team; http://www.nextgenscience.org/critical-stakeholders; National Research Council

Review of the Next Generation Science Standards (NGSS Lead States, 2013, Vol. I, p. v)]

Question #2: Which part(s) of the standards documentation represent the assessable components?

CSTA: The assessable components are the learning outcomes. The standards for K–12 computer science education are presented in a learning objective-based format that identifies the specific computer science concepts and skills students should achieve at each of the three levels (grades K-6, 6-9, and 9-12). [CSTA K-12 Computer Science Standards, 2011. Pg. 12.]

NGSS: The performance expectations of the NGSS are the assessable components. [NGSS Introduction (NGSS Lead States, 2013, Vol. I, p. xviii)]

Question #3: What parts of the science standards are required of all high school students, and to what extent do these fit the time restrictions of a typical school year?

CSTA: None.

NGSS: The NGSS focuses on a limited number of core ideas in science and engineering that build coherently over time throughout K–12 in an effort to foster a greater depth of understanding on a few fundamental concepts within the constraints of the typical school year (Vol. II, pp. 40, 113–115). These standards are expected of all students, including at the high school level, with opportunity for accelerated students to continue past the requirement of the standards (Vol. II, pp. 25, 31, 114). However, having expectations for all students doesn't mean that all students will take the same courses in high school. There are many different ways to structure different courses (e.g., CTE courses, integrated science, senior project, etc.) that could help different students reach and exceed proficiency on the standards. [Appendix D (NGSS Lead States, 2013, Vol. II, pp.25–39), Appendix E (NGSS Lead States, 2013, Vol. II, pp.40–47), Appendix K (NGSS Lead States, 2013, Vol. II, pp.113–136)]

Nature of Science and Methods of Inquiry in Science

Question #4: What aspects of scientific inquiry and processes (e.g., skills and habits of mind) are expressed in the standards, and how are they related to or integrated with the content?

CSTA: Aspects of scientific inquiry and processes are interwoven with content in the computer modeling and simulation portion of the Computational Thinking strand. Scientific practice includes the use, creation, and analysis of computer models and simulations for STEM inquiry. Acting as computational scientists, students must learn, understand, and use computational thinking, computer science concepts, and computer programming constructs. As users and evaluators of models, students must be able to look "under the hood" and understand the mechanisms, abstractions and algorithms implemented in a model using a computer programming language (as well as evaluate what has been left out of a model). As creators of their own models, students must understand the computational science process and use computational thinking and computer science to design, implement, test and revise their model. As young computational scientists, students must learn to use models as experimental test-beds and conduct experiments that sweep multi-dimensional parameter spaces. The data output from these virtual experiments are subsequently analyzed and interpreted to gain an understanding of the underlying system or phenomenon. Finally, students must be able to analyze their creations and determine to what extent their model represents the real world. Often this entails comparing simulation-generated output and real-world data captured of a similar phenomenon. Within the computational science process, students construct theories, design computational models that embody those theories, and then execute the models with various inputs (simulation) and gather evidence that support or refute their theories. Similarly, teachers charged with preparing students as computational scientists must also learn these concepts and practices in order to teach them. [CSTA K-12 Computer Science Standards, 2011.]

NGSS: The NGSS are written as performance expectations built from the three dimensions described in the NRC Framework (2012), including Science Practices (Vol. II, p. 48). These eight practices are the behaviors that scientists engage in as they investigate and build models and theories about the natural world: Asking questions; Developing and using models; Planning and carrying out investigations; Analyzing and interpreting data; Using mathematics and computational thinking; Constructing explanations; Engaging in argument from evidence; and Obtaining, evaluating, and communicating information. The practices are integrated with the disciplinary core ideas and crosscutting concepts in every NGSS performance expectation, such that students are expected to demonstrate their understanding of the core ideas and crosscutting concepts in the context of the practices. For an example, see HS-ESS1 (Vol. I, pp. 119–121). [Appendix F (NGSS Lead States, 2013, Vol. II, pp.48–78)]

Question #5: In what ways do the standards encourage students to utilize multiple avenues of learning (e.g., learning by doing, direct instruction, reading, etc.) and to apply content material in novel situations?

CSTA: The CSTA standards encourage students to utilize multiple avenues of learning such as “learning by doing” as exemplified by the action verbs “use, build, modify, create, make, interact with, act out, analyze, and evaluate”; and “reflective practice” as exemplified by the verbs “explain, classify, describe, and discuss”. [Computer Science Standards, 2011. Pg 56-63.]

NGSS: The performance expectations are not a curriculum and do not dictate methods of instruction. Instead, they are statements of what students should know and be able to do at the end of each grade band (Vol. I, p. xxiii). The performance expectations can and should be met through a number of different means (e.g. Vol. II, p.101) allowing for multiple avenues of learning for diverse student groups (Vol. II, p. 35) as well as encouraging innovation and creativity in instruction. Importantly, the integration of science and engineering practices into the performance expectations in the NGSS ensures that students will be expected to demonstrate proficiency in many different kinds of skills, thereby increasing the likelihood that instruction will incorporate many different kinds of learning modes. [Appendix D (NGSS Lead States, 2013, Vol. II, pp.25–39); Appendix F (NGSS Lead States, 2013, Vol. II, pp.49–50)]

Connections / Relationships Among Standards at Different Grade Levels

Question #6: In what ways are the standards designed to build from grade level to grade level in science content, depth of content understanding, and the application of scientific inquiry and processes?

CSTA: These standards provide a three-level framework for computer science. The first two levels are aimed at grades K–6 and 6–9 respectively. We expect that the learning outcomes in Level 1 will be addressed in the context of other academic subjects. The learning outcomes in Level 2 may be addressed either through other subjects or in discrete computer science courses. Level 3 is divided into three separate exemplar courses: Computer Science in the Modern World, Computer Science Principles, and Topics in Computer Science. The standards provided in Computer Science in the Modern World reflect learning content that should be mastered by all students; Computer Science Principles and Topics in Computer Science are courses intended for students with special interest in computer science and other computing careers, whether they are college-bound or not. [CSTA K-12 Computer Science Standards, 2011. Pg iii.]

NGSS: The NGSS focuses on a limited number of core ideas in science and engineering that build coherently over time throughout K–12 (Vol. II, pp. 41–47, “Disciplinary Core Idea Progression charts”), such that by the end of high school all students are expected to have developed an accurate and thorough understanding of each core idea. The depth of understanding appropriate for each grade band was specified by the NRC Framework (2012). Student performance expectations at each grade band form a foundation for the achievement of the next grade band’s associated performance expectation(s). Practices and crosscutting concepts also grow in complexity and sophistication across the grades (Vol. II, pp.49–67, “practices tables”; Vol. II, pp.80–88 “crosscutting concept tables”), allowing for a greater depth of understanding of the core ideas and crosscutting concepts over time, as well as a greater mastery of science and engineering practices. [Appendix E (NGSS Lead States, 2013, Vol. II, pp.40–47), Appendix F (NGSS Lead States, 2013, Vol. II, pp.48–78), Appendix G (NGSS Lead States, 2013, Vol. II, pp.79–95)]

Question #7: If there are any “outlier concepts” within the standards, how might they relate back to or reinforce the other concepts in the standards?

CSTA: The CSTA K-12 CS Standards do not include outliers concepts.

NGSS: The NGSS are standards for all students, and focus on a limited number of core ideas in science (Vol. II, p. 40). These core ideas were derived from the NRC Framework (2012) and met the Framework committee’s criteria for inclusion in expectations for all students. To ensure that the NGSS scope was teachable, and that there can be time in a typical classroom to help all students build depth of understanding in these core ideas, the NGSS does not include concepts that fall outside the direct progression to each core idea. However, the NGSS should not be viewed as a ceiling for instruction. Students who are proficient in the NGSS can and should go beyond to make connections and apply what they've learned to other areas of interest. [Appendix E (NGSS Lead States, 2013, Vol. II, pp.40–47)]

Evaluation of Understanding and Application of Content Knowledge

Question #8: In what ways do the standards encourage students to apply content knowledge or to use content knowledge in novel situations to build and demonstrate depth of understanding?

CSTA: In many strands, students are encouraged to apply content knowledge to build artifacts that demonstrate depth of understanding. Students are asked to apply content knowledge or use content knowledge in novel situations to build or demonstrate depth of understanding. For example, in the CT strand students are directed to "critically examine classical algorithms and implement an original algorithm". In CPP strand, they are asked to "Anticipate future careers and the technologies that will exist". [CSTA K-12 Computer Science Standards, 2011. pp. 56-59.]

NGSS: Decades of science education research have indicated that the best way to help students learn content deeply is to provide opportunities to practice applying content material, particularly in novel situations (e.g. Grabinger and Dunlap, 1995). By building the performance expectations from the three dimensions described in the NRC Framework (2012), the NGSS requires application of a relevant practice of science or engineering with a core disciplinary idea(s) and connects a crosscutting concept(s) with that core idea. Through the repeated application of the science and engineering practices and crosscutting concepts to different core ideas (especially among different disciplines) and through the explicit connections between core ideas in different performance expectations, the students are expected to use these in different and novel contexts, which enhances depth of understanding of all three of the dimensions (Vol. II, p. 49–50, 80–81). In addition, many performance expectations throughout K–12 explicitly describe engineering applications for core ideas. [Appendix F (NGSS Lead States, 2013, Vol. II, pp.48–78), Appendix G (NGSS Lead States, 2013, Vol. II, pp.79–95)]

Question #9: In what ways do the standards require students to combine or synthesize multiple content ideas in order to demonstrate a deeper understanding of a large, broad theme within science or a specific scientific discipline?

CSTA: The CSTA K-12 Standards contain learning outcomes in computational thinking that require students to combine or synthesize multiple content areas and processes to demonstrate a deeper understanding of computational science and how computer models and simulation can be used to create new knowledge and/or solve problems. The Computational Thinking strand synthesizes learning outcomes in problem solving, algorithms, data representation, abstraction, and modeling and simulation, as well as connections to other fields. [CSTA K-12

Computer Science Standards, 2011. Computational Thinking pp. 9-10 and 56-57.]

NGSS: The NGSS is composed of student performance expectations, which are statements of what students should know and be able to do at the end of each grade band (Vol. I, p.xxiii). In the NGSS, performance expectations are grouped together based on how they support or build up to a core idea. In this way, a single performance expectation that requires the student to apply a science or engineering practice to one aspect of a core idea can be combined with the other performance expectations in the group to address multiple facets of the disciplinary core idea, leading to a greater depth of understanding of that core idea. For example, all of the components of the disciplinary core idea "HS-ESS1: Earth's Place in the Universe" listed in the disciplinary core idea foundation box are addressed in total by the six performance expectations HS-ESS1-1 to HS-ESS1-6 (Vol. I, pp. 119–121). By demonstrating proficiency in all the performance expectations of HS-ESS1, the student will have demonstrated a deeper understanding of the broader theme of the core idea. Each of the core ideas also builds in complexity from grade level to grade level, with increasingly more sophisticated performance expectations that address that core idea at each grade level band (Vol. II, pp. 41-47). ["How to Read the Next Generation Science Standards" (NGSS Lead States, 2013, Vol. I, pp. xxii-xxvi), Appendix E (NGSS Lead States, 2013, Vol. II, pp. 40–47)]

Incorporation of Engineering Technology Standards

Question #10: How do the standards define engineering skills and habits of mind, and in what ways are students expected to demonstrate an understanding of these?

CSTA: The CSTA K-12 Computer Science Standards are focused on learning outcomes specific to the discipline of computer science and therefore do not address "Engineering Practices" per se, They do, however, consider aspects of the process of designing, developing, and testing algorithms, models and simulations, and software artifacts. Both engineering and computer science put forth methods for problem solving using an iterative approach. [CSTA K-12 Computer Science Standards, 2011. Pg 4.]

NGSS: Engineering practices are raised to the level of traditional science practices and include behaviors that engineers engage in, such as "defining problems" and "designing solutions" (Vol. II, pp.49,104). There are eight engineering practices defined by the NGSS and the NRC Framework (2012) – most of which have equivalents in science: Defining problems; Developing and using models; Planning and carrying out investigations; Analyzing and interpreting data; Using mathematics and computational thinking; Designing solutions; Engaging in argument from evidence; and Obtaining, evaluating, and communicating information. These practices are incorporated throughout the NGSS with the disciplinary core ideas and crosscutting concepts in performance expectations, such that students are expected to demonstrate engineering design methods as applied to the content of the core ideas. The NGSS also expect students to develop an understanding of some core engineering design principles—the disciplinary core idea ETS1 is devoted to describing engineering as a discipline, and serves as the foundation for engineering-specific performance expectations from K–12. [Appendix F (NGSS Lead States, 2013, Vol. II, pp.48–78), Appendix I (NGSS Lead States, 2013, Vol. II, pp.103–107)]

Question #11: How are students expected to demonstrate increasing levels of proficiency over time in the use of engineering design methods, including how to incorporate ”failure” in the design process?

CSTA: The iterative design process, including testing and debugging, are central to Computer Science as seen in CT-Problem solving 3A-2: Describe a software development process used to solve software problems; CPP-Programming 3A-3: Use various debugging and testing methods to ensure program correctness; and CPP-Programming 3A-4: Apply analysis, design and implementation techniques to solve problems. [CSTA K-12 Computer Science Standards, 2011.]

NGSS: The NGSS includes the core idea of engineering design that requires use of engineering methods and practices that build coherently and grow in complexity and sophistication from grade level to grade level (Vol. II, pp. 49–67; 104–107). Of the three components of this core idea, “optimizing the design solution” requires students to test their designs and to refine the final design, addressing any “failures” (Vol. II, p.104; e.g. HS-ETS1, Vol. I, pp. 129–130). [Appendix F (NGSS Lead States, 2013, Vol. II, pp. 48–78), Appendix I (NGSS Lead States, 2013, Vol. II, pp.103–107)]

Incorporation of Engineering Design and Methods into Science Standards

Question #12: For which students are the engineering-related standards a requirement (e.g., graduation requirements)?

CSTA: None

NGSS: Engineering design is integrated throughout the standards and is required of every student in two ways: 1) with science specific performance expectations that apply engineering practices and 2) with engineering-specific performance expectations that focus on engineering design at the K–2, 3–5, 6–8, and 9–12 grade level bands (Vol. II, pp. 104–107). [Appendix I (NGSS Lead States, 2013, Vol. II, pp.103–107)]

Question #13: How and to what degree are engineering methods and the design process coupled with the science content standards to enhance the learning of both?

CSTA: Engineering methods and the engineering design process are not tightly coupled with the computer science content standards. Computer Science students learn skills that are applicable in many contexts including engineering. Computer science students learn logical reasoning, algorithmic thinking, design and structured problem solving—all concepts and skills that are valuable well beyond the computer science classroom. Students gain awareness of the resources required to implement, test, and deploy a solution and how to deal with real-world constraints. These skills are applicable in many contexts, from science and engineering to the humanities and business, and they have enabled deeper understanding in these and other areas. [CSTA K-12 Computer Science Standards, 2011. Pg 3 and Computing Practice and Programming 3A-4 “Apply analysis, design & implementation techniques to solve problems” and 3A-3 “Use various debugging and testing methods to ensure program correctness”.]

NGSS: Engineering method and design, as both practice and disciplinary content, are coupled with science content in each grade band of the NGSS (Vol. II, p. 104). [Appendix I (NGSS Lead States, 2013, Vol. II, pp.103–107)]

Course Sequencing and Relationships with Courses in other Content Areas

Question #14: In what ways do the standards provide a foundation for AP courses or other advanced course work?

CSTA: The CSTA K-12 CS standards at Levels 1 and 2 provide a foundation for the AP CS A course that can be offered as an option at level 3 (HS). Level 3 provides three suggested exemplar course configurations, one of which is an AP CS course. [CSTA K-12 Computer Science Standards, 2011. Pg 22.]

NGSS: The NGSS performance expectations are specifically designed not to limit the curriculum and to allow students interested in continuing their coursework in science or engineering the opportunity to do so (Vol. II, pp. 113–115). The NGSS performance expectations provide a foundation for rigorous advanced courses in science or engineering. During the NGSS development process, over a hundred university and community college professors, as well as career training program instructors, met together to examine the NGSS expectations to ensure that they would provide a thorough foundation for entry-level courses in their fields. Course models are currently being developed to show how the NGSS standards could specifically lead into advanced study in AP courses. [Appendix K (NGSS Lead States, 2013, Vol. II, pp.113–136), pending AP course models.]

Question #15: How and to what extent do the science standards require the application of knowledge from other content areas as well as enhance learning in these other areas, including English language arts and mathematics?

CSTA: The learning experiences created from the CSTA K-12 Computer Science standards should be relevant to the students and should promote their perceptions of themselves as proactive and empowered problem solvers. They should be designed with a focus on active learning and exploration and can be taught within explicit computer science courses or embedded in other curricular areas such as social science, language arts, mathematics, and science. [CSTA K-12 Computer Science Standards, 2011. Pg. 8.]

NGSS: The NGSS were designed to align and keep pace with the CCSS-M/ELA, and the performance expectations are explicitly connected to the specific CCSS standards (Vol. I, p. xxvi; Vol. II, pp. 50, 137, 158). These connections highlight how the performance expectations require mathematic principles to more deeply understand the core ideas as well as the role of writing, reasoning, and communication in understanding and applying the core ideas via the practices (Vol. II, pp. 27–28, 50, 137–138, 158). In addition, these connections also provide suggestions for where science skills and knowledge could be built simultaneously with mathematics or ELA skills and knowledge. [Appendix L (NGSS Lead States, 2013, Vol. II, pp.137–157), Appendix M (NGSS Lead States, 2013, Vol. II, pp.158–169)]

Preparing Students for College, Career, and Citizenship

Question #16: In what ways do the standards help students develop the technical knowledge requirements and the collaboration, communication, and problem-solving skills desired by employers (e.g., in preparation for Career and Technical Education [CTE] programs or direct employment out of high school)?

CSTA: 4.2.2 Collaboration of the CSTA standards states “Computer science is an intrinsically collaborative discipline. Significant progress is rarely made in computer science by one person working alone. Typically, computing projects involve large teams of computing professionals working together to design, code, test, debug, describe, and maintain software over time. New programming methodologies such as pair programming emphasize the importance of working together. Additionally, development teams working with discipline-specific experts ensure the computational solutions are appropriate, effective, and efficient. Developing collaboration skills is thus an important part of these K–12 national computer science standards. In elementary school, students can begin to work cooperatively with fellow students and teachers using technology. They learn to gather information and communicate with others using a variety of traditional and mobile communication devices. They also learn to use online resources and participate in collaborative problem solving activities. These collaborative activities continue into middle school, where students apply multimedia and productivity tools for group learning exercises. In secondary school, students enhance their collaborative abilities by participating in teams to solve software problems that are relevant to their daily lives. Skills learned at this level can include teamwork, constructive criticism, project planning and management, and team communication, all of which are considered necessary 21st Century skills (see Partnership for 21st Century Skills at p21.org). “ [CSTA K-12 Computer Science Standards, 2011. Pg 10.]

NGSS: The NGSS integrates science and engineering practices throughout the K–12 standards, and describes explicit connections to the CCSS-M/ELA — thereby expecting students to develop habits, skills, and knowledge specifically applicable to many technical fields or preparation program (Vol. II, pp. 11–14, 17–20, 25–30, 138). To be proficient in the NGSS, students will need to develop the means to communicate effectively and the critical thinking and problem solving skills necessary for employment in rapidly changing job market. [Appendix C (NGSS Lead States, 2013, Vol. II, pp.11–24), Appendix D (NGSS Lead States, 2013, Vol. II, pp.25–39)]

Question #17: How do the standards reflect the ways in which science and engineering are currently practiced in society as well as how these disciplines impact society and address societal needs and concerns?

CSTA: The ethical use of computers and networks is a fundamental aspect of computer science at all levels and should be seen as an essential element of both learning and practice. As soon as students begin using the Internet, they should learn the norms for its ethical use. Principles of personal privacy, network security, software licenses, and copyrights must be taught at an appropriate level in order to prepare students to become responsible citizens in the modern world. Students should be able to make informed and ethical choices among various types of software such as proprietary and open source and understand the importance of adhering to the licensing or use agreements. Students should also be able to evaluate the reliability and accuracy of information they receive from the Internet. Computers and networks are a multicultural phenomenon that effect society at all levels. It is essential that K–12 students understand the impact of computers on international communication. They should learn the difference between appropriate and inappropriate social networking behaviors. They should also appreciate the role of adaptive technology in the lives of people with various disabilities. Computing, like all technologies, has a profound impact on any culture into which it is placed. The distribution of computing resources in a global economy raises issues of equity, access, and power. Social and economic values influence the design and development of computing innovations. Students should be prepared to evaluate the various positive and negative impacts of computers on society and to identify the extent to which issues of access (who has access, who does not, and who makes the decisions about access) impact our lives. [CSTA K-12 Computer Science Standards, 2011. Pg 11.]

NGSS: By integrating science and engineering practices with core ideas and by describing connections to the CCSS-M/ELA, the NGSS better reflect the interconnection of science, engineering, and math in industry (Vol. II, pp. 17–20, 49–50, 103–104, 138). The inclusion of engineering and science practices reflects the emphasis on investigation and innovation in technical fields. The NRC Framework (2012) disciplinary core idea of ETS2: "Links among engineering, technology, science, and society" is included in the NGSS as an overarching, cross-disciplinary idea, and the specific components of this idea are explicitly stated within the foundation boxes where they apply to individual performance expectations in each of the scientific disciplines and across grade bands (e.g., HS-ESS1; Vol. I, pp. 120–121). The standards that specifically address the interrelationship among science, engineering, and human society help students develop the understanding that technological advances can have a profound impact on society and the environment (Vol. II, pp. 108–111, including the "Science, Technology, Society, and the Environment Connections Matrix"). This highlights the importance of technology in developing scientific understanding and the importance of science on driving technological innovation. [Appendix C (NGSS Lead States, 2013, Vol. II, pp.11–24), Appendix J (NGSS Lead States, 2013, Vol. II, pp.108–112)]

Question #18: To what extent do the science standards explicitly support underserved student populations and those populations that traditionally have not succeeded in science and engineering (e.g., females, minorities, English language learners, etc.)?

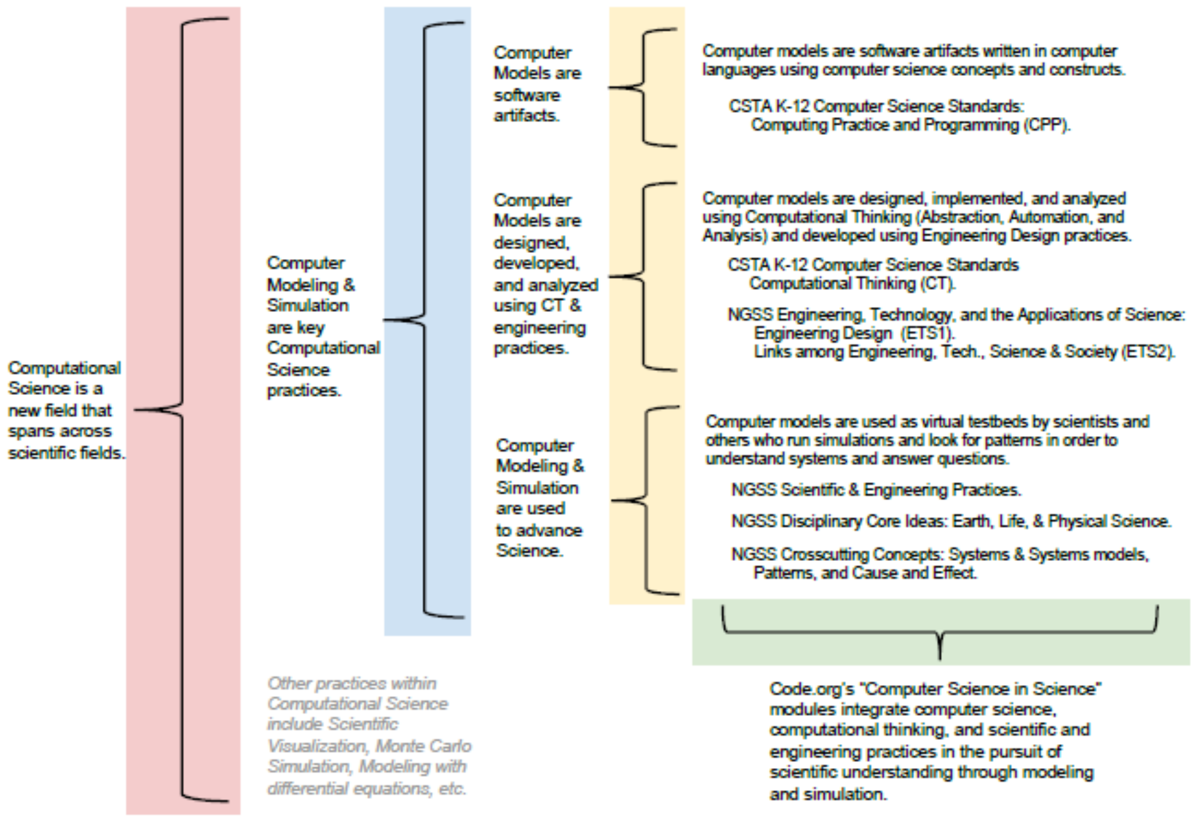
CSTA: Computer science applies to virtually every aspect of life, so that it can be easily tied to myriad student interests. For example, students who are fascinated with specific technologies such as cell phones may have an innate passion for visual design, digital entertainment, or helping society. K–12 computer science teachers can thus nurture students' interests, passions, and sense of engagement with the world around them by offering opportunities for solving computational problems relevant to their own life experiences. Excellence in computer science education relies on equitable practices that maximize the learning potential of all students. Computer science learning opportunities must be shaped in ways that connect the canon of computer science content provided in the curricular standards to the lived experiences of diverse students. The equitable practices in computer science education that connect students with the curriculum include:

- All students should have access to rigorous and culturally meaningful computer science and be held to high expectations for interacting with the curriculum.
- Diverse experiences, beliefs, and ways of knowing computer science should be acknowledged, incorporated, and celebrated in the classroom.
- The integration of different interpretations, strategies, and solutions that are computationally sound enhance classroom discussions and deepen understandings.
- The resources needed for teaching and learning computer science should be equitably allocated across groups of students, classrooms, and schools.
- Classroom learning communities should foster an environment in which all students are listened to, respected, and viewed as valuable contributors to the learning process.
- Ongoing teacher reflection about belief systems, assumptions, and biases support the development of equitable teaching practices.

Pedagogically, computer programming has the same relation to studying computer science as playing an instrument does to studying music or painting does to studying art. In each case, even a small amount of hands-on experience adds immensely to life-long appreciation and understanding, even if the student does not continue programming, playing, or painting as an adult. Although becoming an expert programmer, violinist, or oil painter demands much time and talent, we still want to expose every student to the joys of being creative. The goal for teaching computer science should be to get as many students as possible enthusiastically engaged with every assignment. We can provide students with the tools to design and write programs that control their cell phones or robots, create physics and biology simulations, or compose music. Students will want to learn to use conditional statements, loops, parameters, and other fundamental concepts just to make these exciting things happen. In a fast-paced field such as computer science, we are all challenged to keep up with our peers and our students. Technology changes rapidly, and students are sometimes more likely than teachers to be familiar with the latest incarnations. No teacher should be apprehensive of learning from her or his students. Real learning involves everyone in the room living with a sense of wonder and anticipation. We know that teaching computer science involves some unique challenges and that none of us has all of the answers. The CSTA Source Web Repository at http://csta.acm.org/WebRepository/WebRepository.html provides a comprehensive collection of resources for teachers. These resources have been found to be helpful in our attempts to better interest, engage, and motivate our students. Not all of them will be completely applicable to every classroom, but we believe that many contain useful and varied suggestions that may inspire both students and teachers alike. [CSTA K-12 Computer Science Standards, 2011. Pg 4-5.]

NGSS: The NGSS describe performance expectations for all students, raising the expectations for students who might not otherwise take much science in high school. The NGSS also make connections across the school curricula, including to mathematics and English Language Arts. In addition, the NGSS practices converge with the math and ELA practices. These connections are beneficial for students from non-dominant groups who are pressed for instructional time to develop literacy and numeracy at the cost of other subjects, including science. The NGSS integrate science and engineering practices in every performance expectation, providing students an opportunity to demonstrate their understanding in multiple, diverse ways and providing a justification for multiple, diverse modes of instruction. The NGSS, by emphasizing engineering, recognizes the contributions of non-dominant cultures and groups to science and engineering. Engineering also has the potential to be inclusive of students who have traditionally been marginalized in the science classroom and who do not see science as being relevant to their lives or future (Vol. II, pp. 27–30). By solving problems through engineering in local contexts, students view science as relevant to their lives and future, and engage in science in socially relevant and transformative ways. Engagement in any of the scientific and engineering practices involves both critical thinking and communication skills (Vol. II, p. 50). Because the NGSS is required of all students, these skills will help ESL learners to practice language skills. Finally, the integration of practices with crosscutting concepts require students to think deeply about material and to make connections among big ideas that cut across disciplines, which provides opportunities for learning that has not traditionally been available to disadvantaged or less privileged learners (Vol. II, pp. 80–81). The following case studies are provided to detail how the NGSS can be used to benefit diverse groups of students: (1) Economically Disadvantaged, (2) Race and Ethnicity, (3) Students with Disabilities, (4) English Language Learners, (5) Girls, (6) Alternative Education, (7) Gifted and Talented Students (www.nextgenscience.org/appendix-d-case-studies). [Appendix D (NGSS Lead States, 2013, Vol. II, pp.25–39); www.nextgenscience.org/appendix-d-case-studies

Relationship between Computational Science, Computer Modeling and Simulation, the Computer Science Teachers Association K-12 Computer Science Standards (CSTA), and the Next Generation Science Standards (NGSS).



Computational Science is a new field that spans across scientific fields.

Computer Modeling & Simulation are key Computational Science practices.

Computer Models are software artifacts.

Computer models are software artifacts written in computer languages using computer science concepts and constructs.

CSTA K-12 Computer Science Standards: Computing Practice and Programming (CPP).

Computer Models are designed, developed, and analyzed using CT & engineering practices.

Computer models are designed, implemented, and analyzed using Computational Thinking (Abstraction, Automation, and Analysis) and developed using Engineering Design practices.

CSTA K-12 Computer Science Standards Computational Thinking (CT).

NGSS Engineering, Technology, and the Applications of Science: Engineering Design (ETS1). Links among Engineering, Tech., Science & Society (ETS2).

Computer Modeling & Simulation are used to advance Science.

Computer models are used as virtual testbeds by scientists and others who run simulations and look for patterns in order to understand systems and answer questions.

NGSS Scientific & Engineering Practices.

NGSS Disciplinary Core Ideas: Earth, Life, & Physical Science.

NGSS Crosscutting Concepts: Systems & Systems models, Patterns, and Cause and Effect.

Other practices within Computational Science include Scientific Visualization, Monte Carlo Simulation, Modeling with differential equations, etc.

Code.org's "Computer Science in Science" modules integrate computer science, computational thinking, and scientific and engineering practices in the pursuit of scientific understanding through modeling and simulation.

Augmenting Practices in Science and Engineering with those from Computational Science

By Irene Lee 5/12/2014

This document is derived from table 3-2 of the "Framework for K-12 Science Education" and includes practices that distinguish Computational Science from Science and Engineering.

	Science	Computational Science	Engineering
1. Asking Questions and Defining Problems	Science begins with a question about a phenomenon, such as “Why is the sky blue?” or “What causes cancer?” and seeks to develop theories that can provide explanatory answers to such questions. A basic practice of the scientist is formulating empirically answerable questions about phenomena, establishing what is already known, and determining what questions have yet to be satisfactorily answered.	Computational Science begins with a question, such as “How do birds form flocks?” or a problem and seeks to develop a computer model with which to test theories or design solutions. In Computational Science, part of the question is whether or not it is suitable for modeling using computational methods, and if so, which method? The practice of creating models necessarily involves reflection on the essential mechanisms at work in the real-world system or problem domain. When computer models are constructed that correspond reasonably well with real-world problems, the model can be used to collect data that promote understanding of the real-world problem. Furthermore, reflection upon the limitations and inaccuracies of the model offer opportunities to consider deep questions about essential mechanisms at play in the real-world systems.	Engineering begins with a problem, need or desire that suggests an engineering problem that needs to be solved. A societal problem such as reducing the nation’s dependence on fossil fuels may engender a variety of engineering problems, such as designing more efficient transportation systems, or alternative power generation devices such as improved solar cells. Engineers ask questions to define the engineering problem, determine criteria for a successful solution, and identify constraints.
	The distinction between science and engineering may not make sense in the age of Computational Science.		

	Science	Computational Science	Engineering
2. Developing and Using Models	Science often involves the construction and use of a wide variety of models and simulations to help develop explanations about natural phenomena. Models make it possible to go beyond observables and imagine a world not yet seen. Models enable predictions of the form “if . . . then . . . therefore” to be made in order to test hypothetical explanations.	Computational Science involves the construction and use of computer models to help develop explanations about the natural and artificial world. In the simplest cases, computer models support prediction. Models of more complex or interactive systems offer opportunities to gather quantitative data which points to qualitative outcomes. Models are used as experimental test beds with which to run simulations by changing parameters and rules. Rather than simply testing strengths and limitations of designs (as in engineering), computer modeling and simulation can be used to test theories, illuminate core dynamics within a system, discover new questions, understand the landscape of outcomes, and build intuition about complex systems.	Engineering makes use of models and simulations to analyze existing systems so as to see where flaws might occur or to test possible solutions to a new problem. Engineers also call on models of various sorts to test proposed systems and to recognize the strengths and limitations of their designs.
	Computer models are used to: [J. Epstein, 2008]1. Explain (very distinct from predict); 2. Guide data collection; 3. Illuminate core dynamics; 4. Suggest dynamical analogies; 5. Discover new questions; 6. Promote a scientific habit of mind; 7. Bound (bracket) outcomes to plausible ranges; 8. Illuminate core uncertainties; 9. Offer crisis options in near-real time; 10. Demonstrate tradeoffs / suggest efficiencies; 11. Challenge the robustness of prevailing theory through perturbations; 12. Expose prevailing wisdom as incompatible with available data; 13. Train practitioners; 14. Discipline the policy dialogue; 15. Educate the general public; and 16. Reveal the apparently simple to be complex.		

	Science	Computational Science	Engineering
3. Planning and Carrying Out Investigations	Scientific investigation may be conducted in the field or the laboratory. A major practice of scientists is planning and carrying out a systematic investigation, which requires the identification of what is to be recorded and, if applicable, what are to be treated as the dependent and independent variables (control of variables). Observations and data collected from such work are used to test existing theories and explanations or to revise and develop new ones.	The investigations that can be carried out virtually using a computer model can be unlike traditional science or engineering experiments. Increases in computational power have enabled Computational Scientists to “sweep” the parameter spaces of all possible combinations of inputs and collect outcome data from each run. Analysis of these data can reveal “landscapes” of possible outcomes and help scientists better understand the behavior of the system modeled. Some models are stochastic in nature, and multiple runs with each set of input parameters can be run quickly and efficiently. The computer’s ability to efficiently generate pseudorandom inputs allows modelers to effectively explore these systems as well.	Engineers use investigation both to gain data essential for specifying design criteria or parameters and to test their designs. Like scientists, engineers must identify relevant variables, decide how they will be measured, and collect data for analysis. Their investigations help them to identify how effective, efficient, and durable their designs may be under a range of conditions.
	How is experimental design different when using computer models? 1) The space of variables may be larger and multidimensional - sweep space of variables 2) Stochasticity within models requires that multiple runs be performed at each setting to get a sense of the variability of outcome. 3) The goal of the experimentation may be different. Computational experiments may be generative - for example, a researcher may investigate if a set of simple rules can generate a phenomenon seen in nature. Planning and carrying out scientific investigations using computer models also involves parameterizing the model by selecting relevant variables and determining experimental design; planning the data collection and analysis and considering what constitutes “proof” when using data output from models; simulating and collecting data; and using the computational model as a test bed for running experiments.		

	Science	Computational Science	Engineering
4. Analyzing and Interpreting Data	Scientific investigations produce data that must be analyzed in order to derive meaning. Because data usually do not speak for themselves, scientists use a range of tools including tabulation, graphical interpretation, visualization, and statistical analysis—to identify the significant features and patterns in the data. Sources of error are identified and the degree of certainty calculated. Modern technology makes the collection of large data sets much easier, thus providing many secondary sources for analysis.	Computational Science investigations can produce large amounts of output data that, when analyzed, can give scientists insights into the nature of systems. Computational Scientists use various computational and mathematical analysis techniques to identify salient features and patterns in data. Once data are produced from simulations (multiple runs of the model with different input parameters), regression and a variety of machine learning techniques can be used to determine the correlations between inputs and outputs.	Engineers analyze data collected in the tests of their designs and investigations; this allows them to compare different solutions and determine how well each one meets specific design criteria—that is, which design best solves the problem within the given constraints. Like scientists, engineers require a range of tools to identify the major patterns and interpret the results.
	Does science proceed from observation to models and theory that account for data, OR visa versa? “On this point, many non-modelers, and indeed many modelers, harbor a naïve inductivism that might be paraphrased as follows: 'Science proceeds from observation, and then models are constructed to 'account for' the data.' … This can be very productive, but it is not the rule in science, where theory often precedes data collection. Maxwell's electromagnetic theory is a prime example. From his equations the existence of radio waves was deduced. Only then were they sought…and found! General relativity predicted the deflection of light by gravity, which was only later confirmed by experiment. Without models, in other words, it is not always clear what data to collect!” [J. Epstein, 2008]		

	Science	Computational Science	Engineering
5. Using Mathematics and Computational Thinking	In science, mathematics and computation are fundamental tools for representing physical variables and their relationships. They are used for a range of tasks, such as constructing simulations, statistically analyzing data, and recognizing, expressing, and applying quantitative relationships. Mathematical and computational approaches enable predictions of the behavior of physical systems, along with the testing of such predictions. Moreover, statistical techniques are invaluable for assessing the significance of patterns or correlations.	In Computational Science, computational thinking (abstraction, automation, and analysis) is intrinsic to computer modeling and simulation. Abstraction is used to reduce a problem to essential elements and their relationships. Abstraction can result in a general instance that can represent all other instances. Automation is used when designing algorithms to process information and when using a computer as a labor saving device that executes repetitive tasks quickly and efficiently. Computer models use algorithms and automation as their "engines". Analysis is the validation of whether or not the abstractions made were appropriate to the questions being asked. Validation occurs at the fine-grained level of the mechanisms responsible model low-level interactions as well as at the highest levels of whether the results of the model match observation.	In engineering, mathematical and computational representations of established relationships and principles are an integral part of design. For example, structural engineers create mathematically-based analyses of designs to calculate whether they can stand up to the expected stresses of use and if they can be completed within acceptable budgets. Moreover, simulations of designs provide an effective test bed for the development of designs and their improvement.
	Computational thinking describes a set of human thinking skills, habits and approaches that are integral to solving complex problems using a computer. Computational thinking skills involve understanding and formulating a problem in such as way that its "solution" can be systematically and efficiently produced through a set of computational steps or algorithms to be carried out by a computer. Some, such as Dave Moursand (2009), suggest that the underlying idea in computational thinking is developing models and simulation of problems that one is trying to study and solve.		

	Science	Computational Science	Engineering
6. Constructing Explanations and Designing Solutions	The goal of science is the construction of theories that can provide explanatory accounts of features of the world. A theory becomes accepted when it has been shown to be superior to other explanations, in the breadth of phenomena it accounts for, and its explanatory coherence and parsimony. Scientific explanations are explicit applications of theory to a specific situation or phenomenon, perhaps with the intermediary of a theory-based model for the system under study. The goal for students is to construct logically coherent explanations of phenomena that incorporate their current understanding of science, or a model that represents it, and are consistent with the available evidence.	As with Science, the goal of Computational Science is the construction of theories that can provide explanatory accounts of features of the world. A theory becomes accepted when it has been shown to be superior to other explanations, in the breadth of phenomena it accounts for, and its explanatory coherence and parsimony. Scientific explanations are reinforced when a theory-based model for the system under study produces outcomes similar to those observed in the real world. The goal for students is to construct models of phenomena that incorporate their current understanding of science, use them as experimental test beds and determine if running the model produces outcomes consistent with the available evidence.	Engineering design, a systematic process for solving engineering problems, is based on scientific knowledge and models of the material world. Each proposed solution results from a process of balancing competing criteria of desired functions, technological feasibility, cost, safety, esthetics, and compliance with legal requirements. There is usually no single best solution but rather a range of solutions. Which one is the optimal choice depends on the criteria used for making evaluations.
	“Many simple models: the Lotka-Volterra ecosystem model, Hooke's Law, or the Kermack-McKendrick epidemic equations (compartmental models) continue to form the conceptual foundations of their respective fields. They are universally taught knowing that these models approximate nature, but nonetheless are useful in developing basic intuitions. This is because they capture qualitative behaviors of overarching interest, such as predator-prey cycles, or the nonlinear threshold nature of epidemics and the notion of herd immunity.” [J.Epstein, 2008]		

	Science	Computational Science	Engineering
8. Obtaining, Evaluating, and Communicating Information	Science cannot advance if scientists are unable to communicate their findings clearly and persuasively or to learn about the findings of others. A major practice of science is thus the communication of ideas and the results of inquiry—orally, in writing, with the use of tables, diagrams, graphs, and equations, and by engaging in extended discussions with scientific peers. Science requires the ability to derive meaning from scientific texts (such as papers, the Internet, symposia, and lectures), to evaluate the scientific validity of the information thus acquired, and to integrate that information.	Computational Science combines features of both science and engineering practices including obtaining, evaluating, and communicating information. Computer models are frequently designed on platforms or written in computer languages that offer easy access to visualization tools and other interface elements that promote inter-disciplinary use of the models. The visualizations and, in some cases, the automated and animated runs of the model, allow the models to be used as powerful communication tools.	Engineers cannot produce new or improved technologies if the advantages of their designs are not communicated clearly and persuasively. Engineers need to be able to express their ideas, orally and in writing, with the use of tables, graphs, drawings, or models and by engaging in extended discussions with peers. Moreover, as with scientists, they need to be able to derive meaning from colleagues' texts, evaluate the information, and apply it usefully. In engineering and science alike, new technologies are now routinely available that extend the possibilities for communication and collaboration.

Notes:

“Computational Science is a field of applied computer science, that is, the application of computer science to solve problems across a range of disciplines. In the book Introduction to Computational Science [3], the authors offer the following definition:

> *“the field of computational science combines computer simulation, scientific visualization, mathematical modeling, computer programming and data structures, networking, database design, symbolic computation, and high performance computing with various disciplines.” Computer science, in contrast, is largely focused on the theory, design, and implementation of algorithms for manipulating data and information...The needs of scientists and engineers for computation have long driven research and innovation in computing. As computers increase in their problem-solving power, computational science has grown in both breadth and importance. It is a discipline in its own right [2].*
>
> *An amazing assortment of sub-fields have arisen under the umbrella of computational science, including computational biology, computational chemistry, computational mechanics, computational archeology, computational finance, computational sociology and computational forensics.*
>
> *Some fundamental concepts of computational science are germane to every computer scientist (e.g., modeling and simulation), and computational science topics are extremely valuable components of an undergraduate program in computer science. Students who take courses in this area have an opportunity to apply these techniques in a wide range of application areas, such as molecular and fluid dynamics, celestial mechanics, economics, biology, geology, medicine, and social network analysis.*
>
> *Modeling and simulation of real world systems represent essential knowledge for computer scientists and provide a foundation for computational sciences. Any introduction to modeling and simulation would either include or presume an introduction to computing. In addition, a general set of modeling and simulation techniques, data visualization methods, and software testing and evaluation mechanisms are also important.”*

Excerpted from “Computer Science Curricula 2013 – Curriculum Guidelines for Undergraduate Degree Programs in Computer Science”, December 20, 2013. The Joint Task Force on Computing Curricula; Association of Computing Machinery (ACM) and IEEE Computer Society. ISBN: 978-1-4503-2309-3 / DOI: 10.1145/2534860 / Web link: http://dx.doi.org/10.1145/2534860

Aligning the Framework for K-12 Science Education and CSTA K-12 Standards through the Scientific Practice of Modeling and Simulation

Irene A. Lee

Background

To address the problems of the 21st century that affect us all such as climate change, loss of biodiversity, energy consumption and virulent disease (Emmott et al, 2006), students need to understand large, complex systems. Computational Science, at the intersection of computer science, mathematics and science, is seen as the third leg of science. In addition to theoretical and experimental/lab/field-based science, Computational Science is intrinsic to the work of the modern scientist. Increases in computational power have enabled scientists and researchers across disciplines to design and conduct experiments on models of systems that are too big, too expensive or too dangerous to experiment with in the real world. Using a computer model an experimental test bed, scientists are able to run multiple "What if" scenarios quickly and collect and analyze large amounts of data utilizing the computational power computers afford. New fields that explicitly integrate the use of computation include computational biology, computation physics, computational social science, and computational chemistry, to name a few.

The spread of diseases (and interventions to prevent them) offers a powerful example of the critical use of complex models to solve daunting scientific and human problems. In the August 2009 issue of Nature Josh Epstein states:

> *As the world braces for an autumn wave of swine flu (H1N1), the relatively new technique of agent-based computational modeling is playing a central part in mapping the disease's possible spread, and designing policies for its mitigation...Classical epidemic modeling, which began in the 1920s, was built on differential equations. These models assume that the population is perfectly mixed, with people moving from the susceptible pool, to the infected one, to the recovered (or dead) one. Within these pools, everyone is identical, and no one adapts their behavior. But such models are ill suited to capturing complex social networks and the direct contacts between individuals, who adapt their behaviors—perhaps irrationally—based on disease prevalence. Agent-based models (ABMs) embrace this complexity. ABMs are artificial societies: every single person (or 'agent') is represented as a distinct software individual.*

Computer modeling and simulation are important tools in the computational scientist's toolkit. Computer modeling and simulation are used to test theories, illuminate core dynamics within a system, discover new questions, understand the landscape of outcomes, and build intuition about complex systems. As explicit representations of scientists' abstraction and assumptions, they can serve as artifacts around which to focus a dialogue, train practitioners, and educate the general public. But all of these benefits do not come about without human creativity and ingenuity. Computational thinking describes a set of human thinking skills, habits and approaches that are integral to solving complex problems using a computer. Computational thinking skills involve understanding and formulating a problem in such as way that its "solution" can be systematically and efficiently produced through a set of computational steps or algorithms to be carried out by a computer. Some, such as Dave Moursand (2009), suggest that the underlying idea in computational thinking is developing models and simulation of problems that one is trying to study and solve.

The three pillars of computational thinking: abstraction, automation, and analysis, are intrinsic to computer modeling and simulation. Abstraction is the stripping down of a problem to its bare essentials and capturing common characteristics that can be used to represent all other instances. A computer model is an abstraction of a real-world phenomenon or scenario and time is abstracted allowing scientists to run simulated experiments faster than the analogous experiments in real-life. Automation entails writing algorithms to process information and using a computer as a labor saving device that executes repetitive tasks quickly and efficiently. Computer models use algorithms and iterations as their "engines". Analysis is the validation of whether or not the abstractions made were correct. In the context of modeling and simulation one might ask "Were the right assumptions made when narrowing down the problem to its bare essentials?" and "Were important factors left out of the model?" Thus, in terms of modeling and simulation, computational thinking is used on many levels of a model. At a high level, the "problem" at hand is that of describing/encapsulating a phenomena or scenario in the form of a model and the "solution" is a resulting model that mimics the real-world to the required degree (or in required ways) such that it can be used as an experimental test bed and/or learning tool. At a lower level, computational thinking may be the development of algorithms that encapsulate the behavior of a component of the model or system.

The Need

As described in the Computational Thinking in the Next Generation Science Standards document, "Currently, in most science classrooms, the use of computers is relegated to simulations or data entry. The opportunities afforded to science through the use of computer science concepts are immense. Instead of a student simply manipulating conditions, students who are able to construct their own simulations will display a clear grasp of the scientific concepts expected in the NGSS. More importantly, this ability more closely aligns to how scientists do their work today. There are many concepts in science that are difficult or even impossible to test or manipulate. Students prepared using this type of [computer science] instruction have a whole world of opportunities opened to them that will stimulate interest and mastery of material."

The NGSS are performance standards for students, goals that reflect what students should know and be able to do. New to the standards is the call for science education to change in three fundamental ways: 1) science content and practice are to be intertwined; 2) scientific practice includes the use, creation, and analysis of computer models and simulations for STEM inquiry and in the engineering design cycle; and 3) scientific practice includes computational thinking. In other words, students as science learners need to act as modern computational scientists. Furthermore, to act as computational scientists, students must learn, understand, and use computational thinking, computer science concepts, and computer programming constructs. As users and evaluators of models, students must be able to look "under the hood" and understand the mechanisms, abstractions and algorithms implemented in a model using a computer programming language (as well as evaluate what has been left out of a model). As creators of their own models, students must understand the computational science process and use computational thinking and computer science to design, implement, test and revise their model. As young computational scientists, students must learn to use models as experimental test-beds and conduct experiments that sweep multi-dimensional parameter spaces. The data output from these virtual experiments are subsequently analyzed and interpreted to gain an understanding of the underlying system or phenomenon. Finally, students must be able to analyze their creations and determine to what extent their model represents the real world. Often this entails comparing simulation-generated output and real-world data captured of a similar phenomenon.

Within the computational science process, students construct theories, design computational models that embody those theories, and then execute the models with various inputs (simulation) and gather evidence that support or refute their theories. Similarly, teachers charged with preparing students as computational scientists must also learn these concepts and practices in order to teach them.

Approach

To communicate the importance of computer science and illustrate how computer science concepts can be integrated into the science classroom, we have developed a crosswalk between middle and high school NGSS Framework and CSTA K-12 Computer Science Standards. The crosswalk between the NGSS Framework and the CSTA K-12 CS Standards hinges on the aforementioned work of the computational scientist. Irrespective of whether the computational scientist was first trained as a computer scientist or as a scientist in a discipline other than computer science, the "computational scientist" is a computational thinking-enabled STEM professional [Malyn-Smith and Lee, 2012] who works at the intersection of science, computer science and mathematics. As defined in the "Profile of a Computational Thinking Enabled STEM Professional in America's Workplaces" a computational thinking-enabled STEM professional uses skills, habits and approaches integral to solving problems using a computer (e.g. abstraction, automation, and analysis) as he/she engages in a creative process to solve problems, automate systems, or improve understanding by defining, modeling, qualifying and refining systems, processes, or mechanism generally through the use of computers [EDC, 2011]. This professional may work as a scientist seeking answers or as an engineer solving problems.

The NGSS framework explicitly references engineering practice at the nexus of computer science and science and describes the work as "engineers apply science to design solution to problems and the result is technology". The description of "computational science" as the scientific practice at the nexus of computer science and science, however, is less clear.   From a computational science perspective, computational scientists create and use technology tools (e.g. computer models and simulation) to look for patterns in seeking to answer questions, the result is scientific knowledge. In this crosswalk we will focus on this perspective.

The computational science cycle is a progression that was first developed and used in the Adventures in Supercomputing program, a Sandia National Laboratory funded educational program that engaged high school students in the practice of computational science, in 2000-2003. Subsequently it has been used in the Supercomputing Challenge, Project GUTS: Growing Up Thinking Scientifically, and the New Mexico Computer Science for All programs with middle and high school students over the past ten years. The "computational science cycle" was validated as representative of processes used in computational modeling and simulation by scores of computational scientists working at the Los Alamos National Laboratory and Sandia National Laboratories and is aligned with the activities and tasks of the computational thinking STEM professional [Malyn-Smith and Lee, 2012]. The following diagram illustrates the cycle.

The computational science cycle was chosen as an intermediary scaffold between the NGSS and the CSTA Standards because it exemplifies a common trajectory through the practices within the realm of computational science and because it captures a subset of the activities and tasks of the computational thinking STEM professional that is appropriate for middle and high school students to understand. It describes process of designing, implementing, using and analyzing a computer model. Using the Computational Science Cycle as a roadmap, computational thinking and computer programming are contextualized within answering a question and/or solving a real-world problem using computational modeling and simulation.

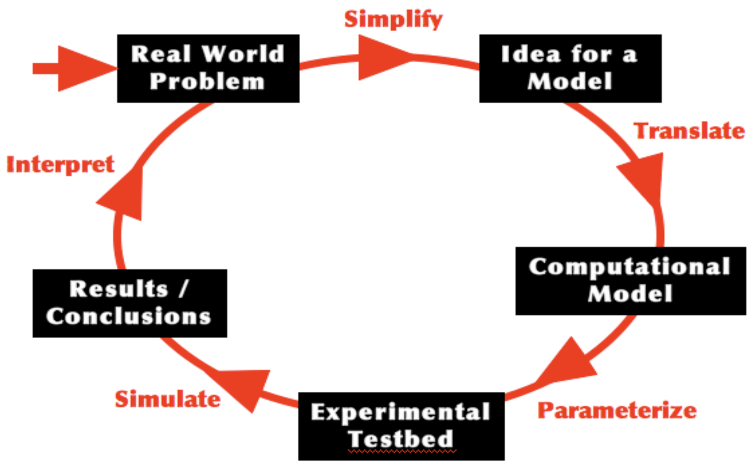


Figure 1. The Computational Science Cycle

From the perspective of an educator, the stages in the "Computational Science Process" include:

Stage 1: Selection of a real-world problem or scientific phenomenon to study. We will discuss what makes a problem or phenomenon suitable for studying using computational methods. We will discuss the simplifications made in models through abstraction. We will ask: What real-world issue you are interested in investigating? What are measurable aspects of the problem? and guide participants in constructing questions that could be answered through modeling and simulation.

Stage 2: Simplify the scope of the model using abstraction. We will ask: What aspects of the problem are important to model? and What is happening at different scales of observation in the complex system? The scope of the problem will be narrowed to one that can be modeled given the software and computing resources available. We will diagram the model components and the simulation loop.

Stage 3: From the description and diagram of the model, we will move to the translation of the idea into a computational model. At this stage we will introduce fundamental concepts in CS through hands-on activity, and we will develop computer programs while building simple models prior to building student designed models.

Stage 4: Parameterize the model. We will discuss relevant variables and parameter and experimental design. We will discuss data collection and analysis and what constitutes proof when using data output from models.

Stage 5: Simulate and collect data. Use the computational model as a test bed for running experiments. In some cases this will involve writing another program that runs the model repeatedly over a set of input values; called a parameter sweep.

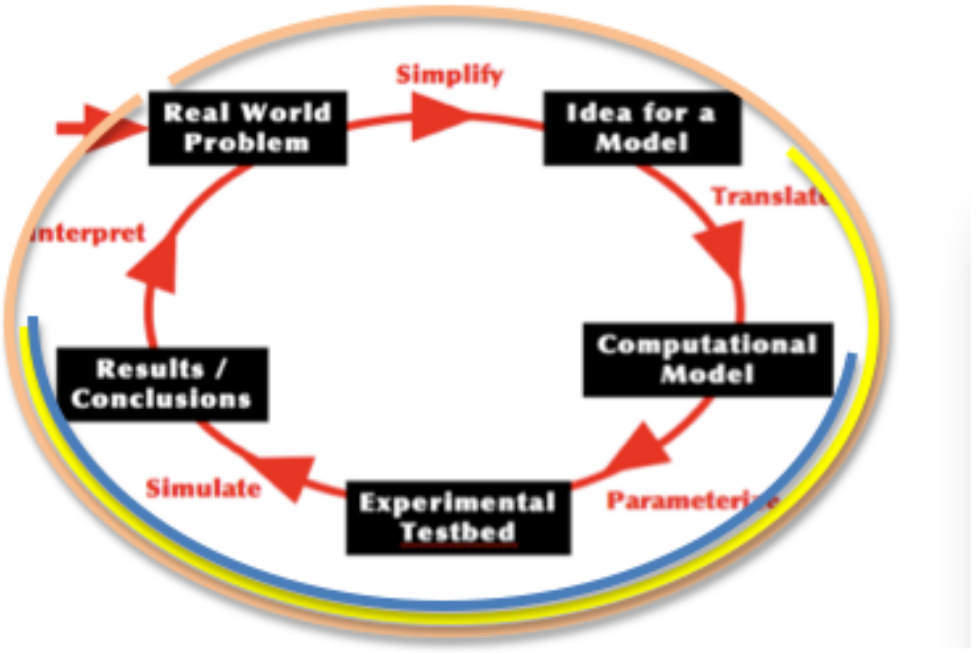
Stage 6: Analyze / Interpret. We will review what constitutes proof when using data output from models. We will discuss the limitations of the computer model, what assumptions were made, and what the model tell us, if anything, about the real world. We will mention exploratory uses of models when no theory exists.

Repeat: We will explain that the "Computational Science Process" is an iterative or repeated process. In evaluating the model one might find verification errors (e.g., bugs in code) or validation errors (e.g. when comparing model behavior to real world data there are differences that suggest that the wrong assumptions or simplifications were made.) In either case, at many points throughout the processes it may be necessary to loop back to an earlier stage or begin the whole computational cycle anew.

An alignment between the Computational Science Cycle and the CSTA K-12 Computer Science Standards was constructed by the Santa Fe Institute in 2012 during the development of the NM-Computer Science for All program. CSTA K-12 Computer Science Standards state: "The use of modeling and simulation, visualization, and management of massive data sets has fostered the emergence of a new field that bridges science, technology, engineering and math—computational science. This field integrates many aspects of computer science such as the design of algorithms and graphics with their application in the sciences."

Bridging the two standards using the Computational Science cycle contextualizes computer science within modern scientific practice. Yet, following the stages in the Computational Science cycle mechanically is not the intent of the cycle diagram. As in the Framework's Scientific and Engineering Practices, stages in the Computational Science cycle may proceed along different paths. The Use-Modify-Create trajectory for student engagement with models guides students through discrete portions of cycle as they learn with and about models. In the Use phase, students are given a completed model. They may run experiments using the computer model as an experimental test bed by setting different parameters, executing the model, making observations, collecting data, and trying to infer the rules of the model. It is important to note that without understanding of the model itself, its underlying assumptions, abstractions, and mechanisms, the model as a black box cannot be used to make predictions about real-world phenomena.

When modifying a model, students can look at the rules of the model and change the rules or add new ones. Because computer models written in computationally-rich environments enable users to view and manipulate underlying code, students can view the rules behind them, inspect how the rules were translated into code, uncover the assumptions of the person who made the model, understand that rules can be changed, and see how these changes may affect how the model works. Ultimately, when creating a model, students design the model and determine its rules. They engage in the iterative design, implement and test process as they refine their model. They may experience the entire cycle as they run experiments using the computer model as an experimental test bed by setting different parameters, executing the model, making observations, collecting data, and finally, analyzing the collected data and determining to what extent their model reflects the real world.



The Use-Modify-Create trajectory mapped onto the Computational Science Cycle (Use stage in blue; Modify stage in yellow; Create stage in pink)

In the crosswalk that follows, the mapping the NGSS Scientific and Engineering Practices to the stages of the computational science cycle is shown in the left hand and middle columns of Table 1. The mapping of the Computational Science Cycle to the CSTA K-12 Computer Science Standards is shown in the middle and right hand column of Table 1.

* Additional learning objectives from the CSTA K-12 Computer Science Standards (such as responsible use of technology) were not included in this mapping because they were not explicitly addressed in the Computational Science Cycle. They may be included in the future.

NGSS Scientific and Engineering Practice	**Stages of the Computational Science Cycle:**	**CSTA K-12 Computer Science Standard (CT = Computational Thinking; CPP = Computer Programming and Practice)**
1. Asking questions / Defining problems. Begin with a question about a phenomenon and seek to develop theories that can provide explanatory answers to such questions. Formulate empirically answerable questions.	1. Select a real-world problem to study. Discuss what makes a problem suitable for studying using computational methods. Discuss the simplifications made in models through abstraction. We will ask: What real-world issue you are interested in investigating? What are measurable aspects of the problem? and guide participants in constructing questions that could be answered through modeling and simulation.	CT-Modeling and simulation 2-10: Evaluate the kinds of problems that can be solved using modeling and simulation.
		CT-Modeling and simulation 1:6-4 Describe how a simulation can be used to solve a problem.
		CT-Modeling and simulation 3A-8 Use modeling and simulation to represent and understand natural phenomena.
2. Developing and using models. Construct and use a wide variety of models and simulations to help develop explanations about natural phenomena. Models make it possible to go beyond observables and imagine a world not yet seen. Models enable predictions of the form "if… then… therefore…" to be made in order to test hypothetical explanations.	2. Simplify the scope of the model using abstraction. We will ask: What aspects of the problem are important to model? and What is happening at different scales of observation in the complex system? The scope of the problem will be narrowed to one that can be modeled given the software and computing resources available. We will diagram the model components and the simulation loop.	CT-Abstraction 3A-9: Discuss the value of abstraction to manage problem complexity.
		CT-Abstraction 3B-10: Decompose a problem by defining new functions and classes.
		CT-Data representation 2-8: Use visual representation of problem state, structure and data.
		CT-Modeling and simulation 2-9: Interact with content-specific models and simulations to support learning and research.
5. Using mathematics and computational thinking. In science, mathematics and computation are fundamental tools for representing physical variables and their relationships. They are used for a range of tasks, such as constructing simulations, statistically analyzing data, and recognizing, expressing, and applying quantitative relationships. Mathematical and computational approaches enable predictions of the behavior of physical systems, along with the testing of such predictions.	3. From the description and diagram of the model, we will move to the translation of the idea into a computational model. Computational thinking describes a set of human thinking skills, habits and approaches that are integral to solving complex problems using a computer. Computational thinking skills involve understanding and formulating a problem in such as way that its "solution" can be systematically and efficiently produced through a set of computational steps or algorithms to be carried out by a computer. The three pillars of computational thinking, abstraction, automation, and analysis, are intrinsic to computer modeling and simulation.	CT-Modeling and simulation 3A-8: Use modeling and simulation to represent and understand phenomena.
		CT-Abstraction 2-12: Use abstraction to decompose a problem into sub problems.
		CT-Data representation 3B-6: Compare and contrast simple data structures and their uses.
		CT-Data representation 3A-12: Describe how mathematical and statistical functions, sets, and logic are used in computation.
		CT-Algorithms 1:6-2: Develop a simple understanding of algorithms using computer-free exercises.

5. (cont.) Statistical techniques are invaluable for assessing the significance of patterns or correlations. (Note: computational thinking is intrinsic to developing computer models and thus practice #5 sits within practice #2.)	3. (cont.) Abstraction is the stripping down of a problem to its bare essentials and capturing common characteristics that can be used to represent all other instances. A computer model is an abstraction of a real-world phenomenon or scenario and time is abstracted allowing scientists to run simulated experiments faster than the analogous experiments in real-life. Automation entails writing algorithms to process information and using a computer as a labor saving device that executes repetitive tasks quickly and efficiently. Computer models use algorithms and iterations as their "engines". Analysis is the validation of whether or not the abstractions made were correct. In the context of modeling and simulation one might ask "Were the right assumptions made when narrowing down the problem to its bare essentials?" and "Were important factors left out of the model?" Thus, in terms of modeling and simulation, computational thinking is used on many levels of a model. At a high level, the "problem" at hand is that of describing/encapsulating a phenomena or scenario in the form of a model and the "solution" is a resulting model that mimics the real-world to the required degree (or in required ways) such that it can be used as an experimental test bed and/or learning tool. At a lower level, computational thinking may be the development of algorithms that encapsulate the behavior of a component of the model or system.	CT-Algorithms 2-4: Evaluate ways that different algorithms may be used to solve the same problem.
		CT-Algorithms 3A-3: Explain how sequence, selection, iteration and recursion are the building blocks of algorithms.
		CPP-Programming 2-5: Implement a problem solution in a programming environment using looping behavior, conditional statements, logic, expressions, variables and functions.
		CPP-Programming 3A-3: Use various debugging and testing methods to ensure program correctness.
		CPP-Programming 3A-4: Apply analysis, design and implementation techniques to solve problems.
		CT-Connections to other fields 2-15: Provide examples of interdisciplinary applications of computational thinking.

3. Planning and carrying out investigations. Scientific investigations may be conducted in the field or laboratory. A major practice of scientists is planning and carrying out a systematic investigation, which requires the identification of what is to be recorded and what are to be treated as the dependent and independent variables.	4. (Planning investigations) Parameterize the model. We will discuss relevant variables and parameter and experimental design. We will discuss data collection and analysis and what constitutes proof when using data output from models.	CT-Modeling and simulation 3B-8: Use models and simulation to help formulate, refine, and test scientific hypotheses.
		CPP-Data collection /3A-11: Describe techniques for locating and collecting small-and large-scale data sets.
	5. (Carrying out investigations) Simulate and collect data. Use the computational model as a test bed for running experiments. In some cases this will involve writing another program that runs the model repeatedly over a set of input values; called a parameter sweep.	CPP-Data collection and analysis 3B-8: Deploy various data collection techniques for different types of problems.
4. Analyzing and interpreting data. Observations and data collected from investigations are used to test existing theories and explanations or to revise and develop new explanations. Scientists use a range of tools to identify the significant features and patterns in the data.	6. Analyze / Interpret: We will review what constitutes proof when using data output from models. We will discuss the limitations of the computer model, what assumptions were made, and what the model tell us, if anything, about the real world. We will mention exploratory uses of models when no theory exists. We compare outcomes with what is known about the real world—to see if they "make sense.	CPP- Data collection and analysis 2-9: Collect and analyze data that are output from multiple runs of a computer program.
		CT-Modeling and simulation 3B-9: Analyze data and identify patterns through modeling and simulation.
		CT-Modeling and simulation 2-11: Analyze the degree to which a computer model accurately represents the real world.
		CPP- Data collection and analysis 3B-7: Use data analysis to enhance understanding of complex natural and human systems.
6. Constructing explanations 7. Engaging in argument from evidence 8. Obtaining, evaluating, and communicating information.	These three practices occur throughout the Computational Science Cycle, especially during the comparison of model generated outcomes with what is known about the real world. We are engaging in argument from evidence and constructing explanations while evaluating and communicating information. The "Computational Science Process" is an iterative or repeated process. In evaluating the model one might find verification errors (e.g., bugs in code) or validation errors (e.g. when comparing model behavior to real world data there are difference that suggest that the wrong assumptions or simplifications were made). In either case, at many points throughout the processes it may be necessary to loop back to an earlier stage or begin the whole computational cycle anew.	

Aligning the NGSS Disciplinary Core Idea of Engineering, Technology, and Applications of Science and the CSTA K-12 Computer Science Standards

Irene A. Lee

Background

Chapter 3 of the Framework for K-12 Science Education describes how student's understanding of engineering practices is to develop in the classroom as they use engineering practices to acquire and apply scientific knowledge. These objectives were integrated in the NGSS within Crosscutting Concepts (under Connections to Engineering, Technology and Applications of Science) and as a standalone Disciplinary Core Idea. The Connections to Engineering, Technology and Applications are linked back to the DCI of ETS by the themes "Interdependence of Science, Engineering and Technology" and "Influence of Science, Engineering, and Technology on Society and the Natural World". The CSTA K-12 Computer Science Standards are focused on learning outcomes specific to the discipline of computer science and therefore do not address "Engineering Practices" per se, They do, however, consider aspects of the process of designing, developing, and testing algorithms, models and simulations, and software artifacts.

Another area of commonality between the NGSS Disciplinary Core Idea of Engineering, Technology, and Applications of Science and the CSTA K-12 Computer Science Standards appear in the areas of responsible use and impacts of technology. In the section addressing Community, Global and Ethical Impacts. The CSTA K-12 Computer Science Standards state "the ethical use of computers and networks is a fundamental aspect of computer science at all levels and should be seen as an essential element of both learning and practice." [CSTA K-12 Computer Science Standards, 2011. Pg 11] Similarly, the NGSS disciplinary core idea "Links among engineering, technology, science and society" addresses the interrelationship among science, engineering and society and states that "students should develop an understanding that technological advances can have profound impact on society and the environment" (Vol. II, pp.108–111, including the "Science, Technology, Society, and the Environment Connections Matrix").

NGSS Core Disciplinary Ideas: Engineering, Technology and Applications of Science. **ETS1: Engineering Design** **ETS2: Links among Engineering, Technology, Science and Society.**	**CSTA K-12 CS Standards** **(CPP = Computer Programming and Practice)** **(CCD = Computers and Communication Devices)** **(COL = Collaboration)** **(CGE = Community, Global and Ethical Impacts)**
CORE IDEA ETS1: ENGINEERING DESIGN How do engineers solve problems?	Seen in CSTA Computational Thinking Strand.
ETS1.A. Defining and Delimiting an Engineering Problem What is a design for? What are the criteria and constraints of a successful solution? The engineering design process begins with the identification of a problem to solve and the specification of clear goals, or criteria, that the final expected end-user of a technology or process, address such things as how the product or system will function (what job it will perform and how), its durability, and its cost. Criteria should be quantifiable whenever possible and stated so that one can tell if a given design meets them. Engineers must contend with a variety of limitations, or constraints, when they engage in design. Constraints, which frame the salient conditions under which the problem must be solved, may be physical, economic, legal, political, social, ethical, aesthetic, or related to time and place. In terms of quantitative measurements, constraints may include limits on cost, size, weight, or performance, for example. And although constraints place restrictions on a design, not all of them are permanent or absolute.	CT-Modeling and simulation 2-10: Evaluate the kinds of problems that can be solved using modeling and simulation.
Grade Band Endpoints for ETS1.A **By the end of grade 8.** The more precisely a design task's criteria and constraints can be defined, the more likely it is that the designed solution will be successful. Specification of constraints includes consideration of scientific principles and other relevant knowledge that are likely to limit possible solutions (e.g., familiarity with the local climate may rule out certain plants for the school garden).	CT-Modeling and simulation 2-10: Evaluate the kinds of problems that can be solved using modeling and simulation.
Grade Band Endpoints for ETS1.A **By the end of grade 12.** Design criteria and constraints, which typically reflect the needs of the end-user of a technology or process, address such things as the product's or system's function (what job it will perform and how), its durability, and limits on its size and cost. Criteria and constraints also include satisfying any requirements set by society, such as taking issues of risk mitigation into account, and they should be quantified to the extent possible and stated in such a way that one can tell if a given design meets them.	CT-Modeling and simulation 2-10: Evaluate the kinds of problems that can be solved using modeling and simulation.

ETS1.B: Developing Possible Solutions **What is the process for developing potential design solutions?**	**Seen in CSTA Strands Computing Practice and Programming, and Computational Thinking.**
Grade Band Endpoints for ETS1.B **By the end of grade 8**. A solution needs to be tested, and then modified on the basis of the test results, in order to improve it. There are systematic processes for evaluating solutions with respect to how well they meet the criteria and constraints of a problem. Sometimes parts of different solutions can be combined to create a solution that is better than any of its predecessors. In any case, it is important to be able to communicate and explain solutions to others. Models of all kinds are important for testing solutions, and computers are valuable tools for simulating systems. Simulations are useful for predicting what would happen if various parameters of the model were changed, as well as for making improvements to the model based on peer and leader (e.g., teacher) feedback.	CT-Abstraction 1:6-5 Make a list of sub-problems to consider while addressing a larger problem.
	CPP-Programming 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
	CT-Abstraction 2-12. Use abstraction to decompose a problem into sub-problems.
	CT-Problem Solving 1:6. Understand and use the basic steps in algorithmic problem solving.
	CT-Problem Solving 2-1. Use the basic steps in algorithmic problem solving to design solutions.
Grade Band Endpoints for ETS1.B **By the end of grade 12**. Complicated problems may need to be broken down into simpler components in order to develop and test solutions. When evaluating solutions, it is important to take into account a range of constraints, including cost, safety, reliability, and aesthetics, and to consider social, cultural, and environmental impacts. Testing should lead to improvements in the design through an iterative procedure. Both physical models and computers can be used in various ways to aid in the engineering design process. Physical models, or prototypes, are helpful in testing product ideas or the properties of different materials. Computers are useful for a variety of purposes, such as in representing a design in 3-D through CAD software; in troubleshooting to identify and describe a design problem; in running simulations to test different ways of solving a problem or to see which one is most efficient or economical; and in making a persuasive presentation to a client about how a given design will meet his or her needs.	CT-Abstraction 3B-10. Decompose a problem by defining new functions and classes.
	CT-Abstraction 3A-9: Discuss the value of abstraction to manage problem complexity.
	CT-Problem solving 3A-1. Use predefined functions and parameters, classes, and methods to divide a complex problem into simpler parts.
	CT-Problem solving 3A-2. Describe a software development process used to solve software problems.
	CPP-Programming 3A-3. Use various debugging and testing methods to ensure program correctness.
	CPP-Programming 3A-4. Apply analysis, design and implementation techniques to solve problems.

ETS1.C: Optimizing the Design Solution How can the various proposed design solutions be compared and improved?	**CSTA Strands Computational Thinking, and Computing Practice and Programming.**
Grade Band Endpoints for ETS1.C **By the end of grade 8.** There are systematic processes for evaluating solutions with respect to how well they meet the criteria and constraints of a problem. Comparing different designs could involve running them through the same kinds of tests and systematically recording the results to determine which design performs best. Although one design may not perform the best across all tests, identifying the characteristics of the design that performed the best in each test can provide useful information for the redesign process—that is, some of those characteristics may be incorporated into the new design. This iterative process of testing the most promising solutions and modifying what is proposed on the basis of the test results leads to greater refinement and ultimately to an optimal solution. Once such a suitable solution is determined, it is important to describe that solution, explain how it was developed, and describe the features that make it successful.	CT-Algorithms 2-4. Evaluate ways that different algorithms may be used to solve the same problem.
Grade Band Endpoints for ETS1.C **By the end of grade 12.** The aim of engineering is not simply to find a solution to a problem but to design the best solution under the given constraints and criteria. Optimization can be complex, however, for a design problem with numerous desired qualities or outcomes. Criteria may need to be broken down into simpler ones that can be approached systematically, and decisions about the priority of certain criteria over others (trade-offs) may be needed. The comparison of multiple designs can be aided by a trade-off matrix. Sometimes a numerical weighting system can help evaluate a design against multiple criteria. When evaluating solutions, all relevant considerations, including cost, safety, reliability, and aesthetic, social, cultural, and environmental impacts, should be included. Testing should lead to design improvements through an iterative process, and computer simulations are one useful way of running such tests.	CPP-Programming 3A-3. Use various debugging and testing methods to ensure program correctness.
	CPP-Programming 3A-4. Apply analysis, design and implementation techniques to solve problems.
	CT-Algorithms 3A-4. Compare techniques for analyzing massive data collections.
	CT-Algorithms 3B-4. Evaluate algorithms by their efficiency, correctness and clarity.
	CT-Data representation 3B-6. Compare and contrast simple data structures and their uses.
	CT-Problem solving 3A-2. Describe a software development process used to solve software problems.

ETS2.A: Interdependence of Science, Engineering, and Technology.	Seen in CSTA Strands Computational Thinking and Community, Global and Ethical Impacts.
Grade Band Endpoints for ETS2.A **By the end of grade 8**. Engineering advances have led to important discoveries in virtually every field of science, and scientific discoveries have led to the development of entire industries and engineered systems. In order to design better technologies, new science may need to be explored (e.g., materials research prompted by desire for better batteries or solar cells, biological questions raised by medical problems). Technologies in turn extend the measurement, exploration, modeling, and computational capacity of scientific investigations.	CT-Connections to other fields 1:6-6 Understanding the connections between computer science and other fields.
	CT-Connections to other fields 2-14. Provide examples of interdisciplinary applications of computational thinking.
	CT-Modeling and Simulation 2-9. Interact with content-specific models and simulations to support learning and research.
	CT-Modeling and Simulation 2-10. Evaluate what kinds of problems can be solved using modeling and simulation.
	CT-Modeling and Simulation 2-11. Analyze the degree to which a computer model accurately represents the real world.
Grade Band Endpoints for ETS2.A **By the end of grade 12.** Science and engineering complement each other in the cycle known as research and development (R&D). Many R&D projects may involve scientists, engineers, and others with wide ranges of expertise. For example, developing a means for safely and securely disposing of nuclear waste will require the participation of engineers with specialties in nuclear engineering, transportation, construction, and safety; it is likely to require as well the contributions of scientists and other professionals from such diverse fields as physics, geology, economics, psychology, and sociology.	CPP-Programming 2-5. Implement problem solutions using a programming language including: looping behavior, conditional statements, logic, expressions, variables, and functions.
	CPP-Programming 3A-3. Use various debugging and testing methods to ensure program correctness.
	CPP-Programming 3A-4. Apply analysis, design and implementation techniques to solve problems.
	CT-Modeling and Simulation 3A-8. Use modeling and simulation to represent and understand natural phenomena.
	CT-Modeling and Simulation 3B-8. Use models and simulations to help formulate, refine and test scientific hypotheses.
	CT-Modeling and Simulation 3B-9. Analyze data and identify patterns through modeling and simulation.

ETS2.B: Influence of Engineering, Technology and Science on Society and the Natural World.	**Seen in CSTA Strand Community, Global and Ethical Impacts.**
Grade Band Endpoints for ETS2.B **By the end of grade 8**. All human activity draws on natural resources and has both short- and long-term consequences, positive as well as negative, for the health of both people and the natural environment. The uses of technologies and any limitations on their use are driven by individual or societal needs, desires, and values; by the findings of scientific research; and by differences in such factors as climate, natural resources, and economic conditions. Thus technology use varies from region to region and over time. Technologies that are beneficial for a certain purpose may later be seen to have impacts (e.g., health-related, environmental) that were not foreseen. In such cases, new regulations on use or new technologies (to mitigate the impacts or eliminate them) may be required.	CGE-Impacts of Technology 1:6-2. Identify the impacts of technology on personal life and society.
	CGE-Impacts of Technology 2-2. Demonstrate knowledge of changes in information technologies over time and the effects those changes may have on education, the workplace, and society.
	CGE-Impacts of Technology 2-3. Analyze the positive and negative impacts of computing on human culture.
Grade Band Endpoints for ETS2.B **By the end of grade 12.** Modern civilization depends on major technological systems, including those related to agriculture, health, water, energy, transportation, manufacturing, construction, and communications. Engineers continuously modify these technological systems by applying scientific knowledge and engineering design practices to increase benefits while decreasing costs and risks. Widespread adoption of technological innovations often depends on market forces or other societal demands, but it may also be subject to evaluation by scientists and engineers and to eventual government regulation. New technologies can have deep impacts on society and the environment, including some that were not anticipated or that may build up over time to a level that requires attention or mitigation. Analysis of costs, environmental impacts, and risks, as well as of expected benefits, is a critical aspect of decisions about technology use.	CGE-Impacts of Technology 3A-3. Discuss the impact of computing technology on business and commerce.
	CGE-Impacts of Technology 3A-3. Describe the role that adaptive technology can play in the lives of people with special needs.
	CGE-Impacts of Technology 3A-4. Compare the positive and negative impacts of computing on culture.
	CGE-Impacts of Technology 3B-2. Analyze the beneficial and harmful effects of computing innovations.
	CGE-Impacts of Technology 3B-3. Summarize how financial markets, transactions, and predictions have been transformed by automation.
	CGE-Impacts of Technology 3B-4. Summarize how computation has revolutionized the way people build real and virtual organizations and infrastructures.