Computer Science Curricular Guidance for Associate-Degree Transfer Programs

Cara Tang
Portland Community College
12000 SW 49th Ave
Portland, OR, U.S.A.
cara.tang@pcc.edu

Cindy Tucker
Bluegrass Community
and Technical College
500 Newtown Pike
Lexington, KY, U.S.A.
cindy.tucker@kctcs.edu

Christian Servin
El Paso Community College
919 Hunter Dr.
El Paso, TX, U.S.A.
cservin1@epcc.edu

Markus Geissler
Cosumnes River College
8401 Center Parkway
Sacramento, CA, U.S.A.
markus.geissler@crc.losrios.edu

ABSTRACT

After two years of intense curriculum development effort, the ACM CCECC (Committee for Computing Education in Community Colleges) published Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity, known as CSTransfer2017. Based on Computer Science Curricula 2013 (CS2013), this guidance was specially designed to aid in the smooth transfer from associate degrees to baccalaureate degrees. The curriculum contains 17 of CS2013's 18 knowledge areas, and a variety of knowledge units appropriate in the first two years of a computer science degree. The guidance comprises over 200 learning outcomes, 64 of which are infused with cybersecurity, along with a three-tiered assessment rubric using measurable verbs from Bloom's Revised Taxonomy. In addition to the CSTransfer2017 task group consisting of 20 community college educators, input from both two- and four-year educators was collected via surveys administered to a global audience, as well as two rounds of public review and comment on drafts of the guidance. Examples of degree and certificate programs that align with CSTransfer2017 are part of a growing repository hosted on the CCECC website, ccecc.acm.org. These program examples demonstrate the adaptability of this competency-based curriculum approach to a variety of computing programs. The CCECC invites institutions to highlight their computer science degree program by submitting a program example at ccecc.acm.org/correlations.

SIGCSE '18, February 21–24, 2018, Baltimore , MD, USA.

ACM ISBN 978-1-4503-5103-4/18/02…$15.00.
https://doi.org/10.1145/3159450.3159536

CCS CONCEPTS

• Social and professional topics → Model curricula • Social and professional topics → Computer science education

KEYWORDS

Curriculum Guidelines; CSTransfer2017; CS2013; Community College; Two-Year College; Transfer Programs; Computer Science Education.

ACM Reference format:

C. Tang, C. Tucker, C. Servin, and M. Geissler. 2018. Commputer Science Curriculum Guidance for Associate-Degree Transfer Programs. In *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159536

1 INTRODUCTION

In 2015, under the auspices of the ACM Education Board the ACM CCECC (Committee for Computing Education in Community Colleges) began updating the ACM *Computing Curricula 2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science* [3]. Several driving factors motivated the update: 1) the dated 2009 guidance; 2) the release of ACM *Computer Science Curricula 2013* (CS2013) [1]; 3) tremendous need expressed by community college educators during a SIGCSE 2015 BoF session [11]; and 4) a top U.S. priority to build a highly capable cybersecurity workforce including computer science professionals [7].

To this end, the CCECC established a task force of community college educators to review CS2013 and identify foundational material in CS2013 that is appropriate for the first two years of a computer science baccalaureate degree. To further inform the guidance, the CCECC administered surveys to a global audience of computer science educators, both 2- and 4-year, to solicit input related to CS2013 knowledge areas (KAs), knowledge units (KUs), and cybersecurity topics which are appropriate for associate-degree computer science transfer programs, reflecting the growing importance of cybersecurity in all aspects of computing. The CCECC and the task force hosted

community engagement workshops at SIGCSE 2016 [10] and SIGCSE 2017 [5, 8], and collected input via a poster session as ITiCSE 2016 [9]. The guidance has been through two rounds of public review and comment on draft versions. In early 2017, the guidance, called CSTransfer2017, was endorsed by the ACM Education Board, and published in June as *Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity* [2].

2 OVERVIEW OF THE GUIDELINES

2.1 Structure and Differences from CS2013

CSTransfer2017 is aligned with CS2013 to facilitate smooth transfer with 4-year university partners. The guidance maintains the knowledge area (KA) and knowledge unit (KU) structure of CS2013, with learning outcomes for each knowledge unit. To facilitate integration into competency-based curriculum typical at many two-year schools, learning outcomes are expressed using Bloom's Revised Taxonomy [4] rather than the custom system used in CS2013. In addition, all learning outcomes are accompanied by a three-tiered assessment rubric providing meaningful evaluation metrics. The lists of topics from CS2013 are absent, reflecting the emphasis on what students can do over what students know.

Included in the associate-degree guidance are 17 of CS2013's 18 knowledge areas. Table 1 shows the 17 KAs and the percentage of the guidance each KA makes up, based on the number of learning outcomes in the KA. The only knowledge area not included is intelligent systems, which consists of upper division content judged not appropriate in the first two years. Two KAs, software development fundamentals (SDF) and discrete structures (DS), are included in their entirety. For the remaining KAs, those knowledge units and learning outcomes determined appropriate for the first two years of a computer science program are included. The information assurance and security (IAS) knowledge area has been renamed Cybersecurity (CYB) to reflect the terminology used in the community, as well as the forthcoming ACM undergraduate curriculum guidelines for cybersecurity education [6]. As in CS2013, the Platform-based Development KA does not have any learning outcomes associated with it, but represents the reality that the design and development of software applications resides on specific software platforms in which platform-specific constraints may apply.

A detailed mapping of CSTransfer2017 learning outcomes to CS2013 can be found on the CCECC website at ccecc.acm.org/guidance/computer-science-2017/classification-mappings. Mappings of CSTransfer2017 to other curricular guidelines, such as Advanced Placement – Computer Science A (AP-CS A), are also included.

The guidance presented in CSTransfer2017 is intended for use by both 2-year and 4-year computer science programs, and should be especially helpful in articulation discussions.

2.2 Learning Outcomes

The guidance consists of a total of 214 learning outcomes (LOs),

Table 1: Knowledge Areas in CSTransfer2017

Knowledge Area	Percentage
Algorithms and Complexity (AL)	7.7%
Architecture and Organization (AR)	5.2%
Computational Science (CN)	0.5%
Cybersecurity (CYB)	10.3%
Discrete Structures (DS)	20.6%
Graphics and Visualization (GV)	1.0%
Human-Computer Interaction (HCI)	2.6%
Information Management (IM)	3.1%
Networking and Communications (NC)	2.6%
Operating Systems (OS)	5.2%
Parallel and Distributed Computing (PD)	1.5%
Platform-based Development	0%
Programming Languages (PL)	7.7%
Software Development Fundamentals (SDF)	15.5%
Software Engineering (SE)	7.7%
Systems Fundamentals (SF)	2.6%
Social Issues and Professional Practice (SP)	6.2%

64 with infused cybersecurity as applied to computer science. The LOs in the guidance fall across the upper five levels of Bloom's taxonomy: *understanding*, *applying*, *analyzing*, *evaluating*, and *creating*. There are no LOs at the lowest level of Bloom's Revised Taxonomy, *remembering*. Figure 1 shows the distribution of Bloom's levels in the 200+ learning outcomes. The emphasis is on student performance, with half of the learning outcomes at the *applying* level, 22% at the *analyzing* level, and 7% at the *evaluating* and *creating* levels. Only 21% of LOs are at the comprehension level of *understanding*.

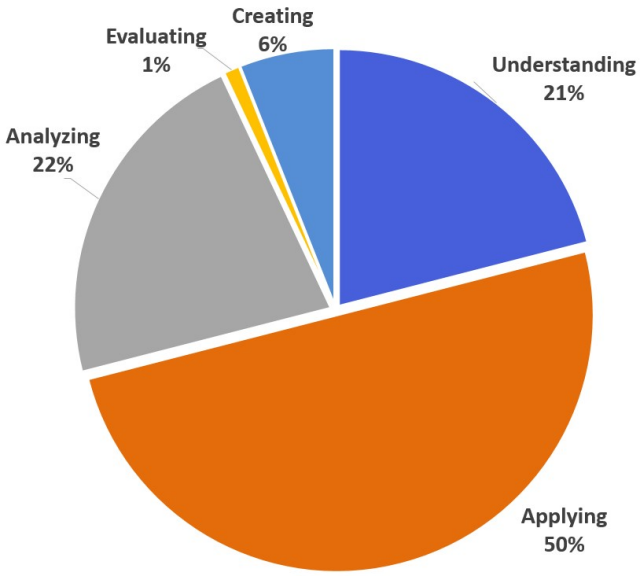

Figure 1: Distribution of learning outcomes in CSTransfer2017 across levels of Bloom's Revised Taxonomy.

Table 2 shows a sampling of the learning outcomes in the guidance, with cyber-infused LOs bolded. Bloom's levels are indicated after each learning outcome. The table of verbs used for the learning outcomes, categorizing the verbs according to Bloom's level, can be found at ccecc.acm.org/assessment/blooms.

Table 2: Selected Learning Outcomes from CSTransfer2017

KA-#. Learning Outcome [*Bloom's level*]
AL-01. Analyze best, average, and worst-case behaviors of an algorithm. [*Analyzing*]
AL-06. Investigate the use of random/pseudo random number generation in cybersecurity applications. [*Applying*]
AR-01. Diagram the digital components of computing architecture. [*Applying*]
CN-01. Illustrate the concepts of modeling and abstraction with respect to problem solving. [*Applying*]
CYB-15. Construct input validation and data sanitization in applications, considering adversarial control of the input channel. [*Creating*]
DS-11. Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument. [*Applying*]
GV-05. Perform information hiding through steganography in images, messages, videos, or other media files. [*Applying*]
HCI-06. Analyze the interaction between a security mechanism and its usability. [*Analyzing*]
IM-05. Describe approaches to scale up information systems. [*Understanding*]
NC-01. Diagram the basic structure of the Internet. [*Applying*]
OS-09. Illustrate the concepts of virtual memory, including paging, thrashing, and partitioning. [*Applying*]
OS-10. Investigate the features and limitations of an operating system used to provide protection and security. [*Applying*]
PD-01. Analyze the goals of parallelism and concurrency. [*Analyzing*]
PL-01. Design a simple class hierarchy using superclasses, subclasses, and abstract classes. [*Creating*]
PL-03. Use access and visibility modifiers to secure class data and methods. [*Applying*]
SDF-04. Create simple programs that use abstract data types (ADTs). [*Creating*]
SDF-19. Carry out a code review on a program component using a provided security checklist. [*Applying*]
SE-03. Diagram the phases of the secure software development lifecycle (SecSDLC). [*Applying*]
SE-10. Describe the cost and tradeoffs associated with designing security into software. [*Understanding*]
SF-01. Illustrate the basic building blocks of computers and their role in the historical development of computer architecture. [*Applying*]
SP-08. Support the ethical responsibility of ensuring software correctness, reliability, and safety. [*Evaluating*]
SP-16. Use effective oral, written, electronic, and visual communication techniques with stakeholders. [*Applying*]

Of the 64 cybersecurity-infused learning outcomes, 25 are in the Cybersecurity (CYB) knowledge area. The rest are distributed throughout the other knowledge areas, as shown in Figure 2. The majority of the cybersecurity-related LOs appearing in other knowledge areas demonstrates the importance of security being integrated into the computer science curricula, rather than (only) a separate topic.

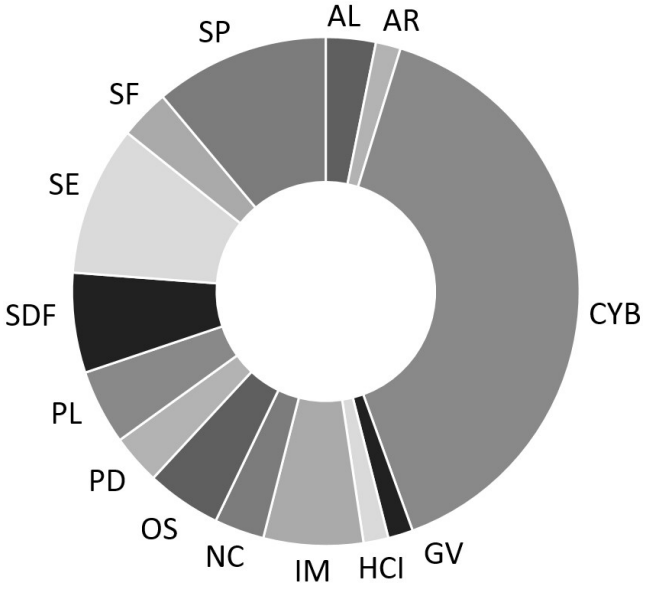

Figure 2: Cybersecurity-infused learning outcomes across the knowledge areas of CSTransfer2017.

Table 3 shows a small number of learning outcomes with their full three-tiered assessment rubric. The lowest level in the rubric, Emerging Standard, represents students who are still emerging in their competency and have not yet achieved the indicated outcome. The Developed Standard represents students who have developed the competency, and the Highly Developed Standard represents those students who have gone above and beyond the given outcome.

Table 3: Selected Learning Outcomes with Assessment Rubric

Learning Outcome	Assessment Rubric
AR-03. Illustrate how fixed-length number representations could affect accuracy and precision, causing vulnerabilities. [*Applying*]	**Emerging Standard:** Explain how fixed-length number representations could affect accuracy and precision, causing vulnerabilities. [*Understanding*] **Developed Standard:** Illustrate how fixed-length number representations could affect accuracy and precision, causing vulnerabilities. [*Applying*] **Highly Developed Standard:** Examine how fixed-length number

	representations could affect accuracy and precision, causing vulnerabilities. [*Analyzing*]
CYB-11. Use the principles of secure design. [*Applying*]	**Emerging Standard:** Demonstrate some of the principles of secure design as related to cybersecurity. [*Understanding*] **Developed Standard:** Use the principles of secure design, such as least privilege, isolation, fail-safe, and deny-by-default. [*Applying*] **Highly Developed Standard:** Choose appropriate secure design principles for a given cybersecurity scenario. [*Evaluating*]
SDF-06. Create programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. [*Creating*]	**Emerging Standard:** Investigate defensive programming techniques. [*Applying*] **Developed Standard:** Create programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. [*Creating*] **Highly Developed Standard:** Create complex programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. [*Creating*]
SDF-11. Compare the efficiency of basic operations across various data structures. [*Analyzing*]	**Emerging Standard:** Investigate the efficiency of basic operations across various data structures. [*Applying*] **Developed Standard:** Compare the efficiency of basic operations, such as insertion and deletion, across various data structures. [*Analyzing*] **Highly Developed Standard:** Critique the efficiency of basic operations across various data structures. [*Evaluating*]

The full list of learning outcomes and assessment rubrics can be found in the published guidance [2] or explored interactively at the ACM CCECC computer science curricular guidance website: ccecc.acm.org/guidance/computer-science-2017.

3 PROGRAM EXAMPLES

Examples of degree and certificate programs that align with CSTransfer2017 are part of a growing repository hosted on the CCECC website. These program examples demonstrate the adaptability of this competency-based curriculum approach to a variety of computing programs. Three program examples are described here, with a summary of how each program correlates to the guidance depicted in Table 4. The table shows what percentage of CSTransfer2017 learning outcomes from each knowledge area are included in the given school's program.

3.1 El Paso Community College (EPCC)

The two-year Computer Science Field of Study at El Paso Community College prepares students to transfer directly into a Bachelor's Degree Program in Computer Science at a four-year institution. The Computer Science Field of Study provides a balanced program which gives students a strong concentration in computer programming. In this program, it is highly recommended that students complete the Math and Physics sequence at the same institution. The Computer Science Field of Study consists of the following four courses:

- **COSC 1436: Programming Fundamentals I.** Introduces the fundamental concepts of structured and object-oriented programming, and provides a comprehensive introduction to programming for computer science and technology majors. Includes topics on software development methodology, data types, control structures, functions, arrays, and the mechanics of running, testing, and debugging. This course assumes computer literacy.
- **COSC 1437: Programming Fundamentals II.** Focuses on the object-oriented programming paradigm, emphasizing the definition and use of classes along with fundamentals of object-oriented design. Includes basic analysis of algorithms, searching and sorting techniques, and an introduction to software engineering processes. Students will apply techniques for testing and debugging software.
- **COSC 2336: Programming Fundamentals III.** Provides further applications of programming techniques, introducing the fundamental concepts of data structures and algorithms. Includes topics on recursion and fundamental data structures, including stacks, queues, linked lists, hash tables, trees, and graphs.
- **COSC 2425 Computer Organization and Machine Language.** Introduces the organization of computer systems using assembly language. Includes topics on basic concepts of computer architecture and organization, memory hierarchy, data types, computer arithmetic, control structures, interrupt handling, instruction sets, performance metrics, and the mechanics of testing and debugging computer systems. Introduces systems and device interfacing

3.2 Bluegrass Community and Technical College (BCTC)

The A.S. Transfer degree in Informatics at Bluegrass Community and Technical College focuses on software development and databases. In addition to general education requirements, the degree includes the following five courses:

- **INF 120: Elementary Programming.** An elementary introduction to programming for those with no previous programming experience. Emphasis on understanding of algorithm development, procedural and object-oriented software development, secure coding, testing, and execution of program. This is a CS1-level course, which includes secure software development techniques.
- **INF 260: Object-Oriented Programming.** Elementary object-oriented programming concepts and practice: types, decisions, loops, methods, arrays, classes; design and

problem solving. An intensive introduction intended for students with programming experience. This is a CS2-level course, which includes secure software development techniques.

- **INF 282: Introduction to Databases**. Core concepts for the design, creation, and manipulation of relational databases. Analysis of data requirements, conceptual modeling, definition of the relational model, relational database design and normalization, and database implementation; manipulation of relational databases using relational algebra with SQL.
- **CIT 120: Computational Thinking.** Promotes understanding of computer programming and logic by teaching students to "think like a computer". Covers skills needed to develop and design language-independent solutions to solve computer-related problems. Covers development and design basics including use of variables, control and data structures, and principles of command-line and object-oriented languages.
- **CIT 111: Computer Hardware and Software.** Presents a practical view of computer hardware and client operating systems. Covers computer hardware components; troubleshooting, repair, and maintenance; operating system interfaces and management tools; networking components; computer security; and operational procedures.

3.3 Folsom Lake College (FLC)

Folsom Lake College's Associate Degree in Computer Science provides a comprehensive exposure to computer science in preparation for upper-division computer science courses. The program also prepares students for entry level employment in the computer and related industries. Students must complete either CISP 400 or CISP 401, and either CISP 310 or ENGR 303: Introduction to Logic Design (not analyzed here). The Computer Science Associate of Science (AS) Degree may be obtained by completion of the required program, plus general education requirements, and sufficient electives to meet a 60-unit total.

- **CISC 310: Introduction to Computer Information Science.** This course is an examination of information systems and their role in business. The focus is on information systems, database management systems, networking, e-commerce, ethics and security, computer systems hardware and software components. Students will develop experience applying these concepts and methods through hands-on projects creating computer-based solutions to business problems.
- **CISP 300: Algorithm Design/Problem Solving.** This course introduces the Computer Science major to methods for solving classical computer problems through algorithm design. Topics covered include introduction to structured design, control structures, arrays, object oriented programming, and file processing. Students will learn how to assess and analyze computer problems in a top-down, divide-and-conquer approach that leads to a programming solution. It also includes creating programming plans and detailed design documents from which source code versions of programs will be created.
- **CISP 310: Assembly Language Programming for Microcomputers.** This course is an introduction to computer architecture using assembly language programs. Topics include binary representation of data and instructions, memory addressing modes, subroutines and macros, operating system interrupts, processor architecture, and interfacing with high level languages.
- **CISP 360: Introduction to Structured Programming.** This course is an introduction to structured programming. The topics covered include: top-down design, input/output considerations, control structures and flow control, variables, constants, the use of libraries, simple to intermediate data structures, functions, and arguments. An introduction into objects will be included.
- **CISP 400: Object Oriented Programming with C++.** This course is an introduction to object-oriented programming using the C++ programming language. This course is designed to enhance students' abilities to implement object-oriented programs and to further develop programing proficiency. Detailed topics include classes, storage class and scope, encapsulation, polymorphism, inheritance, function overloading and overriding, virtual functions, operator overloading, templates, exception handling, stream I/O, file processing, and the Standard Template Library. Also covered are introductions to Graphical User Interface (GUI) development using class libraries, and object-oriented design methodology.
- **CISP 401: Object Oriented Programming with Java.** This course is an introduction to Object Oriented Programming using the Java language. Topics include: objects, classes, UML, function overloading, inheritance, static and dynamic class relationships, polymorphism, components, graphical user interfaces, event driven programming, class associations, interfaces, error handling, threads, file I/O, testing and debugging. This provides the student with a well-rounded background in Java and is good preparation for advanced topics.
- **CISP 430: Data Structures.** This is a course in data structures for Computer Science. Topics include time complexity analysis and big-O notation, recursion, searching and sorting, linked lists, stacks, queues, priority queues, binary trees, B-trees, graphs, hashing, and basic encryption algorithms.
- **CISP 440: Discrete Structures for Computer Science.** This course is an introduction to the essential discrete structures used in Computer Science, with emphasis on their applications. Topics to be covered include: binary number representation and arithmetic, sets, relations, functions, formal propositional logic and proofs, digital logic and combinational circuits, finite state machines, regular expressions and formal grammars. Students will implement programs to illustrate principles of discrete structures.

Table 4: Program Examples

Knowledge Area	EPCC Computer Science Transfer	BCTC Informatics Transfer	FLC Computer Science Transfer
AL	88.2%	76.5%	94.1%
AR	100%	63.6%	81.8%
CN	33.3%	0%	66.7%
CYB	72%	72%	88%
DS	48.4%	0%	85.3%
GV	60%	60%	60%
HCI	100%	50%	83.3%
IM	46.1%	84.6%	92.3%
NC	33.3%	75%	75%
OS	83.3%	61.5%	84.6%
PD	20%	20%	20%
PL	100%	80%	100%
SDF	94.7%	100%	89.5%
SE	57.1%	64.3%	14.2%
SF	66.6%	33.3%	33.3%
SP	86.3%	45.5%	40.1%

4 DISCUSSION

The three program examples included here show three diverse ways that a program can align with the CSTransfer2017 guidelines. As seen in Table 4, none of the programs aligns 100% with every knowledge area, but nearly all knowledge areas are included in each program to some extent. The knowledge areas do not represent courses; a typical course is composed of content from multiple knowledge areas. For example, a CS1 course may include content from the KAs Algorithms and Complexity, Cybersecurity, Human-Computer Interaction, Programming Languages, Software Development Fundamentals, Software Engineering, Social Issues and Professional Practice, and others. The learning outcomes in the guidance can be mixed and matched into courses in a variety of ways to meet the needs of a given program and its local situation and transfer partner(s).

The ACM CCECC invites institutions to highlight their computer science degree program by submitting a program example at ccecc.acm.org/correlations. A spreadsheet can be downloaded for easy offline preparation of the program example, which can then be entered in to the online form at the website.

ACKNOWLEDGMENTS

The CCECC thanks the many reviewers and those who provided input and feedback on CSTransfer2017, especially the task group leaders Teresa Moore (Volunteer State Community College, TN), Lambros Piskopos (Wilbur Wright College, IL), and Christian Servin (El Paso Community College, TX). Thanks also to Caleb Fowler (Folsom Lake College, CA) for providing the FLC program example.

REFERENCES

[1] ACM and IEEE Computer Society. 2013. *Computer Science Curricula 2013*. ACM, New York, NY. DOI: http://dx.doi.org/10.1145/2534860.

[2] ACM Committee for Computing Education in Community Colleges. 2017. *Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity*. ACM, New York, NY. DOI: http://dx.doi.org/10.1145/3108241.

[3] ACM Two-Year College Education Committee. 2009. *Computing Curricula 2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science*. Available at http://ccecc.acm.org/files/publications/2009ComputerScienceTransferGuidelines.pdf. Accessed 30 August 2017.

[4] Anderson, L.W. and Kratwohl, D.R. eds. 2001. *A Taxonomy for Learning, Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York, Longman.

[5] Elizabeth Hawthorne, Cara Tang, Cindy Tucker, and Christian Servin. 2017. Computer Science Curricular Guidelines for Associate-Degree Transfer Programs (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, USA, 725-725. DOI: https://doi.org/10.1145/3017680.3022348.

[6] Joint Task Force on Cybersecurity Education. Cybersecurity Curricula 2017 (CSEC 2017) Website. http://www.csec2017.org/. Accessed 30 August 2017.

[7] National Initiative for Cybersecurity Education (NICE). 2017. *NICE Cybersecurity Workforce Framework*. NIST Special Publication 800-181. DOI: https://doi.org/10.6028/NIST.SP.800-181.

[8] Cara Tang, Cindy Tucker, Elizabeth Hawthorne, Christian Servin, and Teresa Moore. 2017. Curricular Guidance for Associate-Degree Transfer Programs in Computer Science with Contemporary Cybersecurity Concepts (Abstract Only). In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17). ACM, New York, NY, 717-717. DOI: https://doi.org/10.1145/3017680.3022398.

[9] Cara Tang, Elizabeth Hawthorne, Cindy Tucker. 2016. ACM Undergraduate Curricular Guidance in Computer Science: The First Two Years. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (ITiCSE '16, Arequipa, Peru). ACM, New York, NY. 357-357. DOI: https://doi.org/10.1145/2899415.2925480.

[10] Cara Tang, Cindy Tucker, Elizabeth Hawthorne. 2016. Updating Curricular Guidelines for Associate-Degree Computer Science Programs. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16). ACM, New York, NY, 490-491. DOI: http://doi.org/10.1145/2839509.2844671.

[11] Cindy Tucker, Cara Tang, Elizabeth Hawthorne. 2015. Perspectives on How Computer Science Curricula 2013 Influences Two-Year College Programs (Abstract Only). In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, 698-698. DOI: 10.1145/2676723.2691833.