3CS 2017

acm CCECC
Committee for Computing
Education in Community Colleges

ACM Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity

Panel: Elizabeth Hawthorne, Cara Tang, Cindy Tucker,
Christian Servin, Markus Geissler

http://ccecc.acm.org

Agenda

Introduction & Milieu

Overview of Guidance

Cybersecurity Infused Learning Outcomes

CAE2Y & AP CS-A Mappings

Program Examples and Call for More

Staying in Contact

Question and Answers

ACM CCECC

Committee for Computing Education in Community Colleges

- Celebrating **40++ years** of service to computing education
- Standing committee of the ACM Education board for 25++ years

Global Mission: Serve and support community and technical college educators in all aspects of computing education

Engage in curriculum and assessment development, community building, and advocacy in service to this sector of higher education

CS-Transfer Guidelines Background

ACM Curricular Guidance – www.acm.org/education/

2009: Guidelines for Associate-Degree Transfer Curriculum in Computer Science

2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science – CS 2013

New knowledge area: Information Assurance and Security (IAS)

KA ≠ course

2015: BoF @ SIGCSE: Perspectives on How CS 2013 Influences Two-Year College Programs – Standing room only!

2015: Joint task force on Cybersecurity Education formed – ACM, IEEE-CS, AIS-SEC, CEP

2015: NSF C5 Project – Catalyzing Computing and Cybersecurity in Community Colleges

2016: Computer Science for All – U.S. Government initiative (www.whitehouse.gov/blog/2016/01/30/computer-science-all)

CS-Transfer Guidelines Background

Nov 2015: CS-Cyber Task Group formed

Divide CS 2013 knowledges areas (KAs) into 3 clusters, form 3 teams

- Team leads: Teresa Moore, Lambros Piskopos, Christian Servin

For each CS2013 knowledge unit (KU): Appropriate for associate-degree level?

Draft learning outcomes for each KU

- Sources: CS 2013, NSA CAE2Y, NICE Framework, IT 2017 v0.51, Bloom's Taxonomy

Mar 2016: SIGCSE workshop developing learning outcomes & assessment rubric

June 2016: StrawDog released; 2 surveys for input

- Over 50 feedback responses from 8 different countries

CS-Transfer Guidelines Background

July 2016: Poster at ITiCSE, Arequipa, Peru

Oct 2016: IronDog released

Jan – Feb 2017: Final input incorporated

KAs, KUs, learning outcomes, assessment rubric, and Bloom's levels reviewed, tweaked, and finalized

Mar 2017: Pre-release of Final version @ SIGCSE 2017, Seattle
ACM Education Board Endorsed

June 2017: **Final version release @ 3CS 2017**
Available ACM Digital Library

Acknowledgements

Team Leaders and Task Force Members

- Prof. Lambros Piskopos, Wilbur Wright College, IL, Team Leader
- Dr. Markus, Geissler, Cosumnes River College, CA, Team Leader
- Prof. Kimberly Bertschy, Northwest Arkansas Community College, AR
- Prof. Colleen Case, Schoolcraft College, MI
- Prof. Rafael Escalante, El Paso Community College, TX
- Dr. Becky Grasser, Lakeland Community College, OH
- Prof. Charles Hardnett, Gwinnett Technical College, GA
- Prof. Amardeep Kahlon, Austin Community College District, TX
- Prof. James Kolasa, Bluegrass Community and Technical College, KY
- Dr. Shamsi Moussavi, MassBay Community College, MA
- Prof. Pam Schmelz, Ivy Tech Community College, IN
- Prof. Melissa Stange, Lord Fairfax Community College, VA
- Prof. Khallai Taylor, Miami-Dade College, FL
- Prof. Carole Tharnish, Northeast Community College, NB

Other Contributors

- Dr. Anne Applin, Southern Maine Community College, ME
- Prof. Bryce Barrie, Saskatchewan Polytechnic, Canada
- Prof. Michael Bauer, Leeward Community College, HI
- Prof. Paul Dadosky, Ivy Tech Community College, IN
- Prof. Andrea DeMott, Ohio University, OH
- Dean Jamie Edwards, Wytheville Community College, VA
- Dr. Larry Forman, San Diego City College, CA
- Prof. Guy Garrett, Gulf Coast State College, FL
- Prof. Dianne Hill, Jackson College, MI
- Dr. Nancy Jones, Coastline Community College, CA
- Prof. Marc Nester, Wytheville Community College, VA
- Dr. Dean Nevins, Santa Barbara City College, CA
- Dr. Michael Posner, Villanova University, PA
- Prof. Kristopher Roberts, Ivy Tech Community College, IN
- Prof. Barry Sullens, Ivy Tech Community College, IN
- Prof. Robert Surton, Columbia Gorge Community College, OR

Differences from CS2013

- 17 of 18 KAs included
  - 1 KA not included: Intelligent Systems
- 1 KA name change:
  - IAS Information Assurance and Security → CYB Cybersecurity
- Various KUs included for each KA
- Learning Outcomes updated
  - Utilize Bloom's Revised Taxonomy
- Assessment rubric added for every learning outcome
- No topics

Why Learning Outcomes Approach?

Focus is on **student achievement**

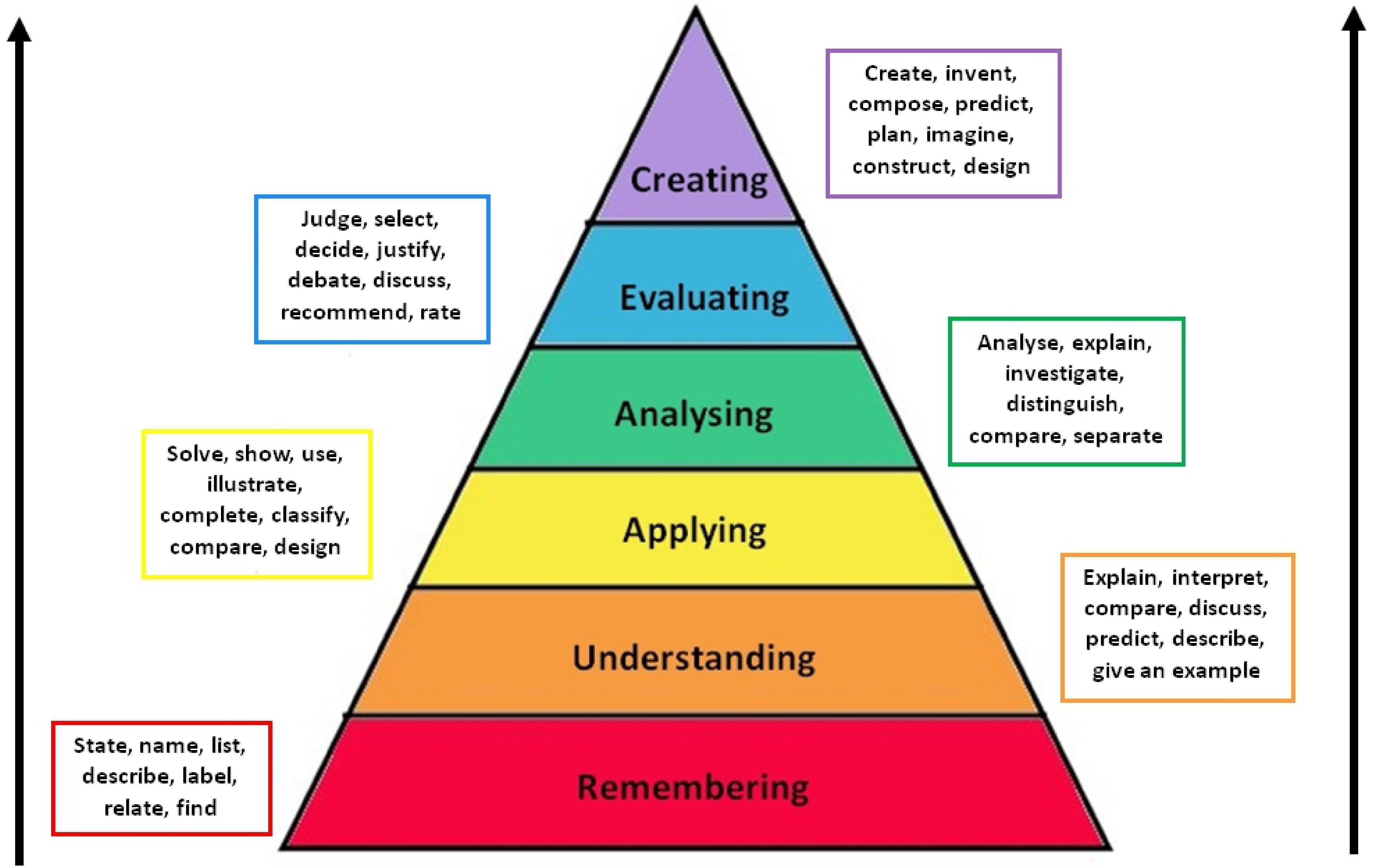
Supports modification of existing courses (easier to add outcomes than entire courses)

Also supports development of new courses

Avoids traditional body of knowledge focus on topics and contact hours that can grow unbounded as new technologies emerge

What topics are eliminated to make room for the new? (food fight)

Bloom's Taxonomy



3-Tiered Assessment Rubric

Every learning outcome has an assessment rubric

Learning Outcome	Emerging Standard	Developed Standard	Highly Developed Standard
CYB-15. Construct input validation and data sanitization in applications, considering adversarial control of the input channel. [*Creating*]	Implement simple input validation and data sanitization in applications. [*Applying*]	Construct input validation and data sanitization in applications, considering adversarial control of the input channel. [*Creating*]	Develop complex input validation and data sanitization in applications, considering adversarial control of the input channel. [*Creating*]

Body of Knowledge

17 knowledge areas

214 learning outcomes with assessment metrics

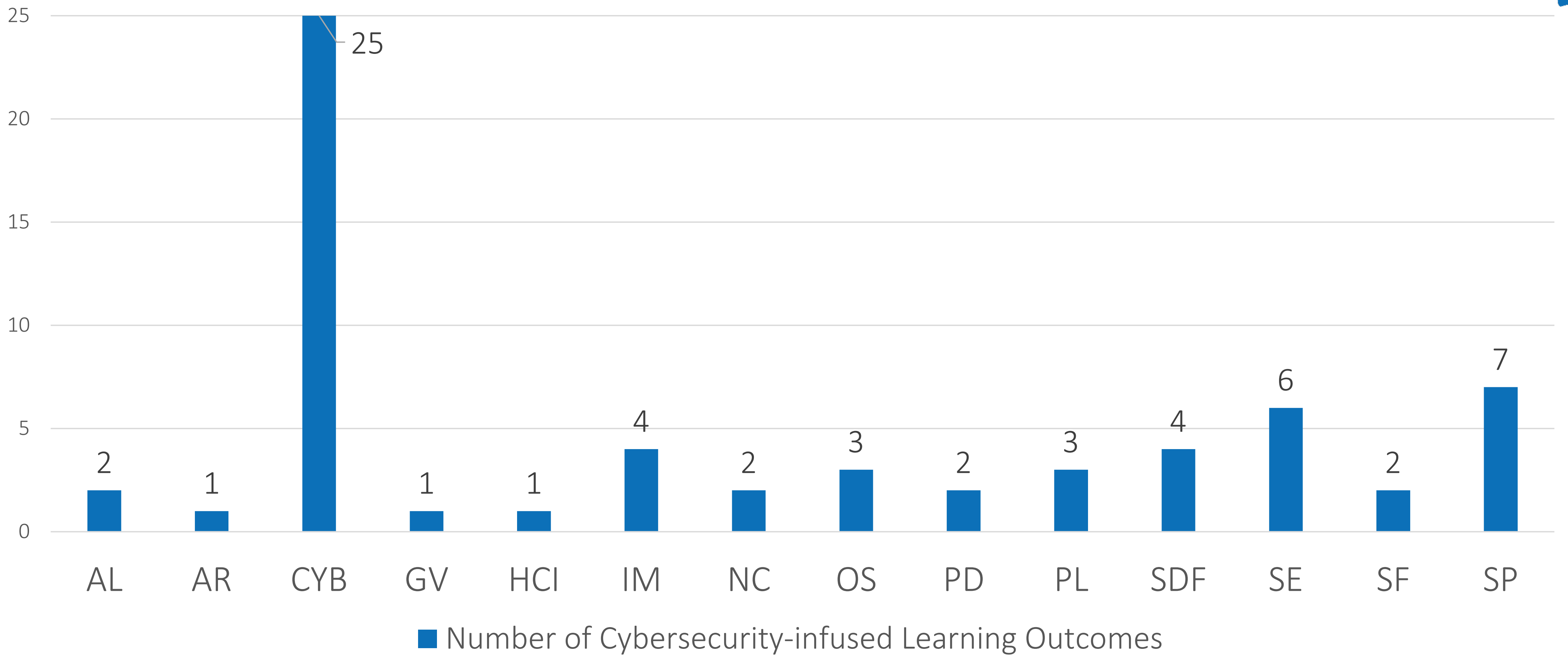
63 learning outcomes specific to cybersecurity

- 25 in CYB knowledge area
- 38 in other KAs

Body of Knowledge

Algorithms and Complexity (AL) – 17 LOs	Architecture and Organization (AR) – 11 LOs
Computational Science (CN) – 3 LOs	Cybersecurity (CYB) – 25 LOs
Discrete Structures (DS) – 34 LOs	Graphics and Visualization (GV) – 5 LOs
Human-Computer Interaction (HCI) – 6 LOs	Information Management (IM) – 13 LOs
Networking and Communications (NC) – 8 LOs	Operating Systems (OS) – 13 LOs
Parallel and Distributed Computing (PD) – 5 LOs	Platform-based Development (PBD) – No LOs
Programming Languages (PL) – 10 LOs	Software Development Fundamentals (SDF) – 19 LOs
Software Engineering (SE) – 14 LOs	System Fundamentals (SF) – 9 LOs
Social Issues and Professional Practice (SP) – 22 LOs	

Cybersecurity Infused Learning Outcomes



Mappings CS Transfer to Other Curriculum, Frameworks, and Classifications

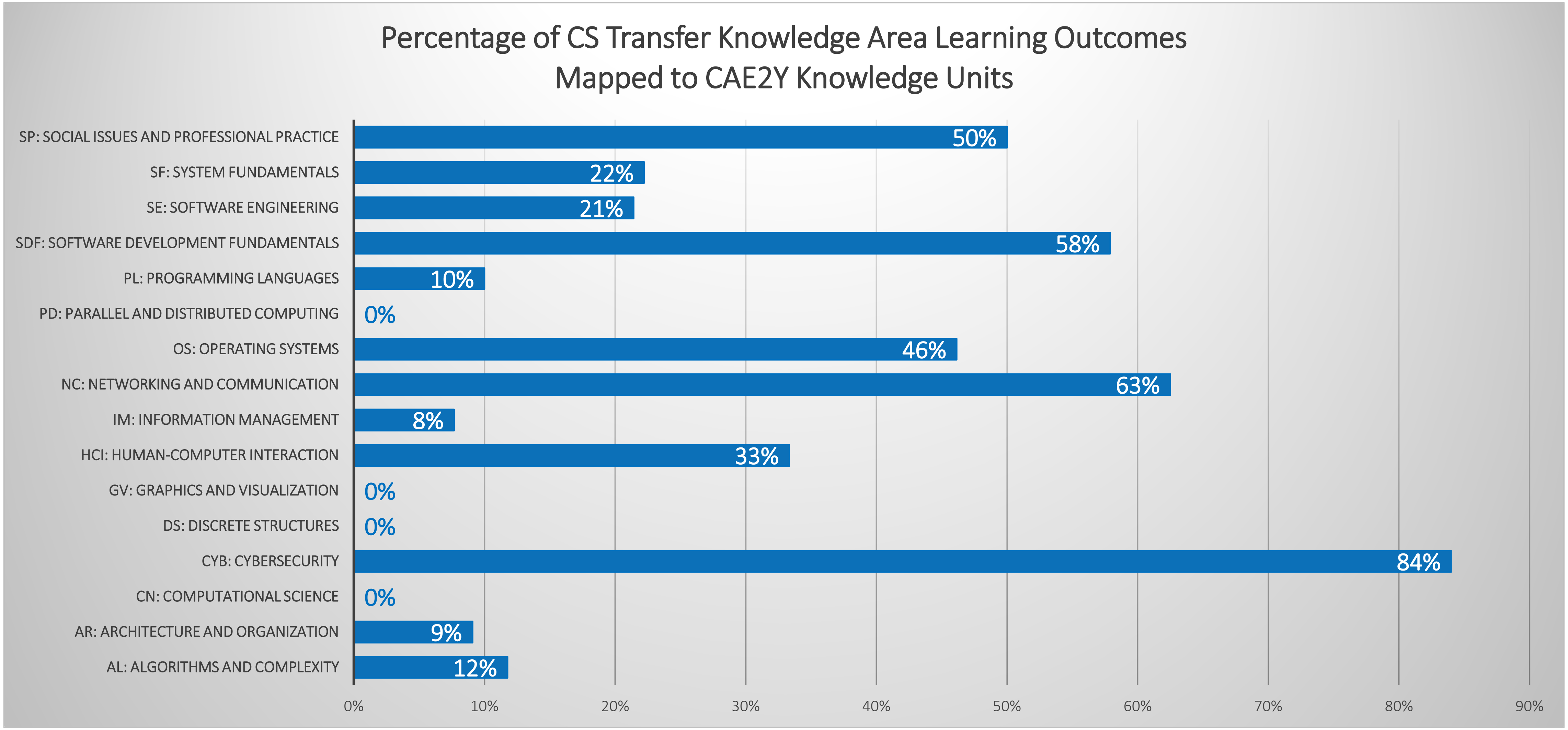
NSA/DHS CAE2Y Knowledge Units

College Board AP Computer Science A

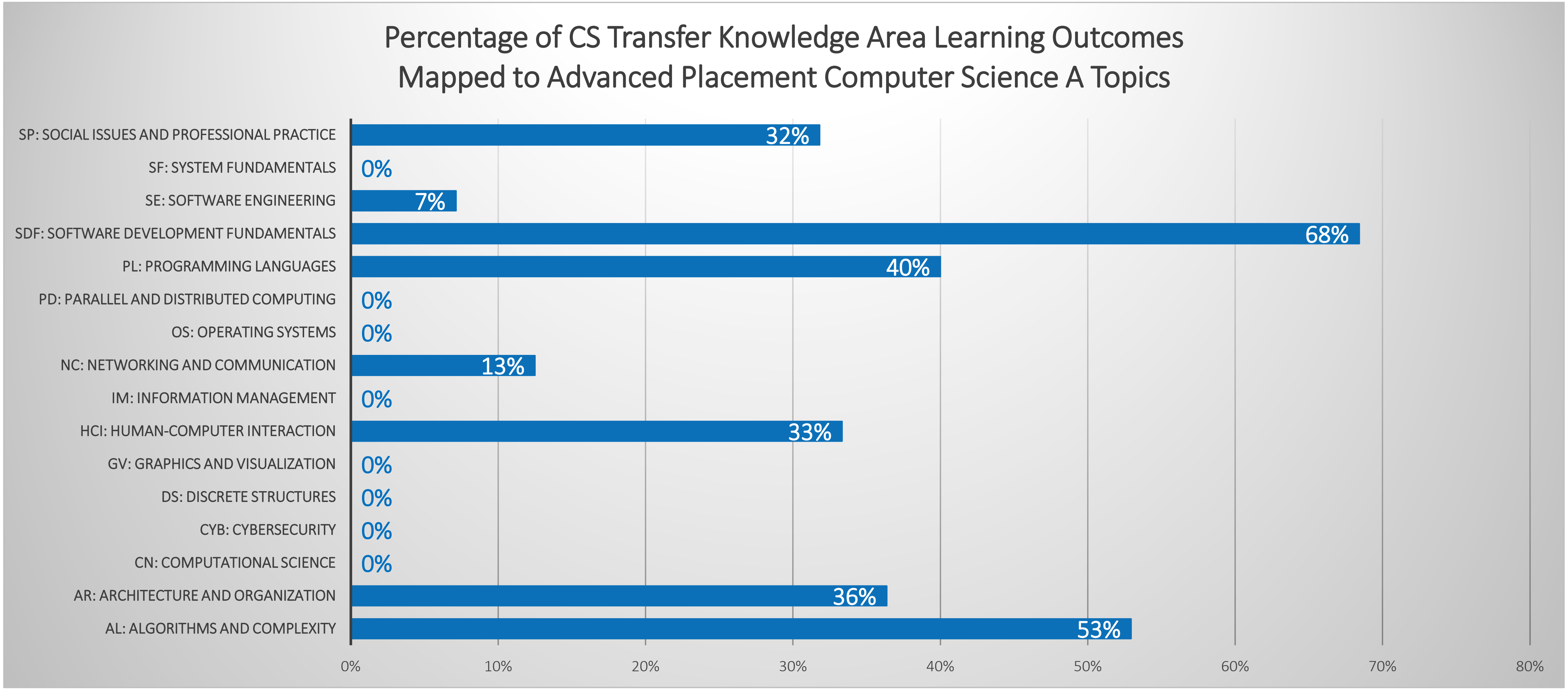
ACM Computer Science 2013 Guidance

Others

NSA/DHS CAE2Y Knowledge Units Mapping



AP Computer Science A Topics Mapping



Program Examples

El Paso Community College

- COSC 1436
- COSC 1437
- COSC 2336
- COSC 2426

Computer Science 2017 Program Example from El Paso Community College

Submitted: 2017-06-19

Description

This two-year Computer Science Field of Study prepares students to transfer directly into a Bachelor's Degree Program in Computer Science at a four-year institution. Students planning to enter the computer science field or the Associates Degree in Computer Programming or Telecommunications and Networking are advised to select electives which will apply to the academic course working that degree plan. The Computer Science Space Field of Study will provide a balanced program which will give the students a strong concentration in the computer programming area. It is highly recommended that students complete the Math and Physics sequence at the same institution. The student who plans to major in Computer Science may take some of the required courses as both Core Curriculum requirements and Field of Study requirements.

Program Website

https://www.epcc.edu/InstructionalPrograms/Pages/ComputerScience.aspx

Contact

Program Examples (cont-d)

Area	KU-Description	Learning Outcome	COSC 1436	COSC 1437	COSC 2336
AL: Algorithms and Complexity	Basic Analysis	AL-01. Analyze best, average, and worst case behaviors of an algorithm. [Analyzing]		X	X
AL: Algorithms and Complexity	Basic Analysis	AL-02. Estimate time and space complexities for a given algorithm using Big-O notation. [Evaluating]		X	X
AL: Algorithms and Complexity	Basic Analysis	AL-03. Contrast standard complexity classes. [Analyzing]		X	X
AL: Algorithms and Complexity	Basic Analysis	AL-04. Analyze the performances of an algorithm with various input sizes. [Analyzing]		X	X
AL: Algorithms and Complexity	Algorithmic Strategies	AL-05. Apply an appropriate algorithmic approach to a given problem. [Applying]			X
AL: Algorithms and Complexity	Algorithmic Strategies	AL-06. Investigate the use of random/pseudo random number generation in cybersecurity applications. [Applying]			X
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-07. Implement basic numerical algorithms. [Applying]			
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-08. Implement common search algorithms, including linear and binary searches. [Applying]		X	X
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-09. Implement common sorting algorithms, including iterative, quadratic, and recursive. [Applying]		X	X
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-10. Implement hash tables, including collision avoidance and resolution. [Applying]			X
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-11. Explain the runtime and memory efficiency of principal sorting, searching, and hashing functions. [Understanding]		X	X
AL: Algorithms and Complexity	Fundamental Data Structures and Algorithms	AL-12. Investigate factors other than computational efficiency that influence the choice of algorithms. [Applying]			X

Program Examples (cont-d)

Area	KU-Description	Learning Outcome	COSC 1436	COSC 1437	COSC 2336
SDF: Software Development Fundamentals	Algorithms and Design	SDF-01. Design an algorithm in a programming language to solve a simple problem. [Creating]	X		
SDF: Software Development Fundamentals	Algorithms and Design	SDF-02. Use the techniques of decomposition to modularize a program. [Applying]	X	X	
SDF: Software Development Fundamentals	Algorithms and Design	SDF-03. Compare multiple algorithms for a given problem. [Analyzing]		X	
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-04. Create simple programs that use abstract data types (ADTs). [Creating]		X	X
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-05. Investigate potential vulnerabilities in provided programming code. [Applying]	X		
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-06. Create programs which use defensive programming techniques, including input validation, type checking, and protection against buffer overflow. [Creating]	X	X	X
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-07. Create code in a programming language that includes primitive data types, references, variables, expressions, assignments, I/O, control structures, and functions. [Creating]	X	X	X
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-08. Create a simple program that uses persistence to save data across multiple executions. [Creating]	X	X	
SDF: Software Development Fundamentals	Fundamental Programming Concepts	SDF-09. Create a simple program that uses recursion. [Creating]		X	
SDF: Software Development Fundamentals	Fundamental Data Structures	SDF-10. Create simple programs that include each of the following data structures: lists, stacks, queues, hash tables, graphs and trees. [Creating]		X	X
SDF: Software Development Fundamentals	Fundamental Data Structures	SDF-11. Compare the efficiency of basic operations across various data structures. [Analyzing]			X
SDF: Software Development Fundamentals	Development Methods	SDF-12. Investigate common coding errors that introduce security vulnerabilities, such as buffer overflows, integer errors, and memory leaks. [Applying]	X	X	

Program Examples (cont-d)

Area	KU-Description	Learning Outcome	COSC 1436	COSC 1437	COSC 2336
CYB: Cybersecurity	Foundational Concepts in Security	CYB-01. Describe security as a continuous process of tradeoffs, balancing between protection mechanisms and availability. [Understanding]	X	X	X
CYB: Cybersecurity	Foundational Concepts in Security	CYB-02. Illustrate through examples the concepts of risk, threats, vulnerabilities, attack vectors, and exploits, noting there is no such thing as perfect security. [Applying]		X	
CYB: Cybersecurity	Foundational Concepts in Security	CYB-03. Investigate various countermeasures and security controls to minimize risk and exposure. [Applying]			X
CYB: Cybersecurity	Foundational Concepts in Security	CYB-04. Analyze the tradeoffs of balancing key security properties, including Confidentiality, Integrity, and Availability (CIA). [Analyzing]		X	X
CYB: Cybersecurity	Foundational Concepts in Security	CYB-05. Explain the concepts of trust and trustworthiness related to cybersecurity. [Understanding]		X	
CYB: Cybersecurity	Foundational Concepts in Security	CYB-06. Examine ethical issues related to cybersecurity. [Analyzing]	X		
CYB: Cybersecurity	Foundational Concepts in Security	CYB-07. Illustrate various ways to minimize privacy risks and maximize anonymity. [Applying]	X	X	X
CYB: Cybersecurity	Foundational Concepts in Security	CYB-10. Illustrate with examples the goals of end-to-end data security. [Applying]		X	X
CYB: Cybersecurity	Principles of Secure Design	CYB-11. Use the principles of secure design. [Applying]	X	X	X
CYB: Cybersecurity	Principles of Secure Design	CYB-12. Illustrate the security implications of relying on open design vs the secrecy of design. [Applying]			X
CYB: Cybersecurity	Principles of Secure Design	CYB-14. Analyze the tradeoffs associated with designing security into a product. [Analyzing]		X	
CYB: Cybersecurity	Defensive Programming	CYB-15. Construct input validation and data sanitization in applications, considering adversarial control of the input channel. [Creating]	X	X	X
CYB: Cybersecurity	Defensive Programming	CYB-17. Implement programs that properly handle exceptions and error conditions. [Applying]		X	X
CYB: Cybersecurity	Defensive Programming	CYB-18. Examine the need to update software to fix security vulnerabilities. [Analyzing]	X	X	X
CYB: Cybersecurity	Cryptography	CYB-21. Describe key terms in cryptology, including cryptography, cryptanalysis, cipher, cryptographic algorithm, and public key infrastructure. [Understanding]	X	X	
CYB: Cybersecurity	Cryptography	CYB-22. Use a variety of ciphers to encrypt plaintext into ciphertext. [Applying]	X	X	


Q
&
A


Questions

Stay Involved with ACM CCECC

Follow us on Twitter **@ACMccecc** or Facebook **ACMccecc**

Google group: cc-ed-computing

- Open to anyone interested in computing education at community & technical colleges
- Post any questions / discussion items to have a conversation with the community

Bookmark our Website

CCECC.ACM.org