Bloom's for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines

IronDog Draft

27 February 2022

Association for Computing Machinery (ACM)
Committee for Computing Education in Community Colleges (CCECC)

Bloom’s for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines

Adeleye Bamkole, Passaic County Community College, NJ
Markus Geissler, Ph.D., Cosumnes River College, CA
Koudjo Koumadi, Ph.D., Prince George’s Community College, MD
Christian Servin, Ph.D., El Paso Community College, TX
Cara Tang, Ph.D., Portland Community College, OR *
Cindy S. Tucker, Bluegrass Community and Technical College, KY

* Task Force Chair

Table of Contents

Introduction

The ACM Committee for Computing Education in Community Colleges (CCECC) has long been using Bloom's Revised Taxonomy in composing learning outcomes and competencies for curricular guidance published for associate degree programs. The six levels in the taxonomy represent the varying levels of cognitive depth a student is expected to demonstrate. The CCECC learning outcomes have been carefully crafted using verbs from standard lists associated with each of the six levels. Through this experience, curriculum project members have often struggled with finding the right verb from the verb list to best express the desired learning outcome. A common case in which this struggle occurs concerns technical tasks for which a technical verb would be appropriate but is not available on the verb list. This repeated experience, across projects and across task group members, provided the initial motivation for taking on the project of suggesting computing-specific verbs to supplement the standard lists. Positive response from the community to the first draft of this report offered feedback that this is something the community is interested in and provided further motivation. The verbs found in this report are not limited to technical computing verbs, but are more generally verbs that could be useful for learning outcomes and competencies in computing programs and curricular guidance.

The *Bloom's for Computing* project began in July of 2020, and a first draft of about 90 potential new verbs was disseminated in March - April of 2021 via a poster at SIGCSE 2021 and sent out on various mailing lists. A survey was used to gather feedback on the verbs, with 46 responses. This feedback has been incorporated into the current report containing 77 proposed verbs. This IronDog report also offers an explanation for each verb along with at least two sample learning outcomes showing how the verb could be used.

The target audience for this report is anyone who drafts competencies or learning outcomes for computing disciplines - whether programs, courses, individual modules, or curriculum guideline reports; whether two-year, four-year, graduate, or K-12 level; whether educators, instructional designers, or program coordinators.

We look forward to further community input on this draft, which will be incorporated into an expected final report around the end of 2022.

Bloom's Revised Taxonomy

Assessment at Educational Institutions

Institutions must ensure an ongoing and effective process for assessing student learning. In particular, computing courses and programs of study must incorporate clearly defined, measurable student outcomes which demonstrate that student achievement at the course level promotes successful attainment of program goals. Competencies and learning outcomes, which

indicate a student's ability to do something successfully or efficiently, may also be used to measure achievement of objectives within a curricular module or unit.

This relationship is commonly demonstrated when:

- For each program of study a collection of program outcomes is identified;
- For each course in the program a collection of student learning outcomes is identified;
- For each course, topics of study and learning activities are selected and designed to support the course student outcomes;
- Each course student outcome supports one or more program outcomes; and
- Each program outcome is supported by one or more course outcomes.

Effective assessment provides valuable feedback to faculty and academic leaders for continuous improvement of pedagogy, course content, and program outcomes, in order to better prepare students for future studies and careers. In addition, effective assessment fosters articulation between institutions and promotes student transfer, and documents employment readiness and facilitates job placement. Accreditation requirements, performance-based funding and public demands for accountability also make effective educational assessment a necessity.

Bloom's Cognitive Domain

The foundational Taxonomy of Educational Objectives: A Classification of Educational Goals was established in 1956 by Dr. Benjamin Bloom, an educational psychologist, and is often referred to as Bloom's Taxonomy [b56]. This classification divided educational objectives into three learning domains: Cognitive (knowledge), Affective (attitude) and Psychomotor (skills). In 2000, Lorin Anderson and David Krathwohl updated Bloom's seminal framework to create Bloom's Revised Taxonomy, focusing on the Cognitive and Affective Domains [a01].

In computing curricula, the Cognitive domain is often used to assess student mastery of learning outcomes. There are six levels in the taxonomy for the Cognitive domain, progressing from the lowest order processes to the highest:

- *Remembering* - Retrieving, recalling, or recognizing information from memory. Students can recall or remember information. Note: This process is the most basic thinking skill.
- *Understanding* - Constructing meaning or explaining material from written, spoken or graphic sources. Students can explain ideas or concepts.
- *Applying* - Using learned materials or implementing materials in new situations. Students can use/apply information in a new way.
- *Analyzing* - Breaking material or concepts into parts, determining how the parts relate or interrelate to one another or to an overall structure or purpose. Students can distinguish between different parts.
- *Evaluating* - Assessing, making judgments and drawing conclusions from ideas, information, or data. Students can justify a stand or decision.

- *Creating* - Putting elements together or reorganizing them into a new way, form or product. Students can create a new product. Note: This process is the most difficult mental function.

In the framework of Bloom's Revised Taxonomy learners need not start at the lowest taxonomic level and work up; rather, the learning process can be initiated at any point, and the lower taxonomic levels will be subsumed within the learning scaffold. To wit:

- Before we can understand a concept we have to remember it;
- Before we can apply the concept we must understand it;
- Before we analyze it we must be able to apply it;
- Before we can evaluate its impact we must have analyzed it; and
- Before we can create, we must have remembered, understood, applied, analyzed and evaluated.

When using Bloom's Revised Taxonomy in crafting competencies and learning outcomes, lists of verbs associated with each level in the taxonomy are often used. These verb lists may vary slightly depending on the source but many of the same verbs are commonly found at particular levels. For example, Define at the Remembering leve, Describe at the Understanding level, etc. The table below presents one such verb list, this one being the standard in use by the ACM CCECC (see ccecc.acm.org/assessment/blooms).

Assessment Verbs by Bloom's Level

Lower Order Thinking Skills — Higher order Thinking Skills

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Define	Classify	Apply	Analyze	Appraise	Assemble
Duplicate	Convert	Calculate	Attribute	Argue	Construct
Find	Demonstrate	Carry out	Categorize	Assess	Create
Identify	Describe	Edit	Compare	Choose	Design
Label	Differentiate	Diagram	Contrast	Critique	Develop
Locate	Exemplify	Illustrate	Deconstruct	Defend	Formulate
Memorize	Explain	Implement	Deduce	Estimate	Hypothesize
Name	Infer	Investigate	Discriminate	Evaluate	Invent

Recall	Interpret	Manipulate	Distinguish	Judge	Make
Recognize	Paraphrase	Modify	Examine	Justify	Plan
Retrieve	Report	Operate	Integrate	Support	
Select	Summarize	Perform	Organize	Test	
State	Translate	Produce	Outline	Value	
		Solve	Structure	Verify	
		Use			
		Write			

Limitations of Bloom's

While it provides a useful framework which applies to many disciplines, Bloom's Revised Taxonomy is not without limitations. Though its linear arrangement into six levels adds simplicity and thus makes the framework more usable, one of its major criticisms is that some verbs could be applied at several cognitive levels, and this can lead to the design of learning outcomes or competencies which may not accurately indicate the cognitive level. As the English language evolves, sometimes even regionally, the meaning and usage of some verbs will change and may eventually require a revision of the Taxonomy.

Those who create learning outcomes are increasingly also looking to include elements from the Affective Domain, which "includes the manner in which we deal with things emotionally, such as feelings, values, appreciation, enthusiasms, motivations, and attitudes" [c15]. For this reason, Bloom's Taxonomy Cognitive Domain may no longer suffice to serve, as it does for many curriculum committees, as the sole source of verbs for defining learning outcomes or competencies which are increasingly expected to allow for the assessment of dispositions as well as skills.

Other work with learning taxonomies for computing disciplines includes [f07,s17].

Bloom's Verbs for Computing

New Verbs (Proposed)

The following are 77 proposed verbs, at the cognitive levels shown, to be offered as an enhancement to the standard verb list presented earlier.

Remembering	**Understanding**	**Applying**	**Analyzing**	**Evaluating**	**Creating**
Enumerate Reference	Annotate Comment Follow	Backup Build Code Compile Compute Configure Connect Decrypt Deploy Document Encourage Encrypt Experiment Graph Hash Install Interview Iterate Map Measure Patch Provision Randomize Recover Remove Restore Schedule Store Train Unit-test Virtualize	Articulate Automate Contextualize Correlate Detect Facilitate Generalize Model Monitor Parallelize Predict Promote Simulate Trace Translate Update	Adapt Administer Balance Debug Decide Defend Delegate Derive Hack Moderate Optimize Prioritize Propose Prove Transform Validate	Collaborate Compose Generate Program Reengineer Refactor Script Secure Visualize

As one may note in reviewing the list above, it includes verbs that can be used for technical computing tasks (such as Code, Compile, Encrypt, Parallelize, Hack, Script), verbs that can be used for dispositional or soft skills (such as Encourage, Articulate, Collaborate), as well as verbs for more general use (such as Measure, Translate, Decide). All of these verbs can be used to write learning outcomes and competencies that students in computing programs may be expected to achieve.

It should be noted that the task force does not expect the final version of this report to contain 77 verbs. We hope to use feedback from the community on this present draft to remove those verbs that will be of little use and to offer a set of verbs that best enhance the standard verb lists without making the combined list too unwieldy. Your feedback is thus very important to this project.

Verb Explanations and Sample Learning Outcomes

When considering how a verb might be used in a competency or learning outcome, it is helpful to see examples. Below we share at least two examples for each verb to demonstrate how they might be used in a curriculum framework for a computing discipline. The verbs' usage should not be seen as being limited to the meanings expressed in these examples. As mentioned earlier, we acknowledge that a particular verb could represent varying levels of cognitive depth, but our usage of each verb is intended to represent the cognitive depth of the Bloom's level it has been placed in. In addition to the sample competencies and learning outcomes we provide a brief explanation / rationale for the inclusion of each verb.

The tables below include, for each level of Bloom's, the list of proposed verbs at that level, with an explanation for why each was included, and numbered sample learning outcomes for each verb.

Remembering

Verb	***Explanation* and Sample Learning Outcomes**
Enumerate	*Similar to List in non-computing uses, and also has computing uses* 1. Enumerate the essential steps of the cybersecurity kill-chain process. 2. Enumerate the steps in the software development lifecycle (SDLC). 3. Enumerate the values of a collection in a data collection set.
Reference	*Adds a skill not included directly in the original Bloom's list* 1. Reference multiple NIST Special Publications in a cyber-risk analysis report. 2. Reference sources of borrowed codes as comments within a computer program.

Understanding

Verb	***Explanation* and Sample Learning Outcomes**
Annotate	*Drawing and clearly annotating diagrams is required in computing tasks such*

	as designing system and network flows 1. Annotate an attack graph in the context of cybersecurity threat modeling. 2. Annotate a network diagram with names of the components.
Comment	*Commenting codes is a recommended practice* 1. Comment a given segment of a program or script. 2. Comment on a network design presented on a diagram.
Follow	*Common when translating designs into code, and can be used more generally* 1. Follow a flowchart to write a program that solves a real-life problem. 2. Follow instructions in a user guide to set up a cybersecurity appliance.

Applying

Verb	***Explanation*** **and Sample Learning Outcomes**
Backup	*Common computing task, especially in system administration and data security* 1. Backup multiple local or remote files and folders to a backup volume. 2. Backup marked data to a remote location using an automation script.
Build	*Commonly used in many computing areas, such as operating systems, IT, systems programming* 1. Build a machine learning model to detect network intrusions. 2. Build a working multi-router network using an automation script.
Code	*Required task in many aspects of computing, including building virtualized infrastructure* 1. Code a Python program from a given flowchart. 2. Code a cloud infrastructure orchestration using a scripting language.
Compile	*Common software development task, especially when coding in compiled languages such as C and C++* 1. Compile a program using the command line interface. 2. Compile a low level code for a specific operating system.
Compute	*Common task in computer science* 1. Compute conditional probabilities. 2. Compute message digest of saved files using various hashing algorithms.
Configure	*Common real-word task in system administration, networking, and cybersecurity* 1. Configure a UNIX-based operating system in order to harden it. 2. Configure a firewall to block or allow traffic based on IP addresses and port numbers.
Connect	*Required in networking, system configuration, and other information exchange*

	environments 1. Connect two remote sites using a tunneling technology. 2. Connect multiple branch routers to a headquarters' router using virtual private networks. 3. Connect a workstation to a router to configure the router.
Decrypt	*Commonly required for information security objectives* 1. Decrypt a ciphertext using various cryptographic techniques. 2. Decrypt a received email using a private key.
Deploy	*Common computing task* 1. Deploy a machine learning pipeline to achieve a successful algorithm. 2. Deploy multiple instances of a template virtual machine.
Document	*Common computing task, includes source code documentation as well as system documentation* 1. Document the processes of an information system at multiple levels of detail. 2. Document a project for version control by keeping track of the software environment.
Encourage	*Verb used in computing tasks and disposition for soft skills context* 1. Encourage cyber hygiene among peers. 2. Encourage computational thinking for a given problem.
Encrypt	*Common computing task with emphasis in cybersecurity* 1. Encrypt files or folders using an appropriate system utility. 2. Encrypt computer users data with RSA algorithm.
Experiment	*Verb commonly used in programming methodology and in computational thinking areas* 1. Experiment with program inputs to find vulnerability to control hijacking. 2. Experiment with what-if scenarios on given data.
Graph	*Verb used in data structures and algorithms as well as data science and analytics* 1. Graph a data set using an appropriate chart type. 2. Graph traverse a set of vertices using depth-first and breadth-first search.
Hash	*Common computing task in cybersecurity area and data structures* 1. Hash a password using an appropriate hash function. 2. Hash given data values into a hash table.
Install	*Common computing task* 1. Install a given software patch. 2. Install a docker engine in a given operating system.
Interview	*Verb used in computing projects, particularly in software engineering; also a disposition for soft skills context* 1. Interview users to elicit feature requirements for an application. 2. Interview a computing professional and recognize industry competencies

	required in the field.
Iterate	*Common computing task in programming and data science areas* 1. Iterate through a list in a given program language. 2. Iterate a machine learning algorithm and indicate the number of times the algorithm's parameters are updated.
Map	*Common computing verb in IT and programming* 1. Map an entity relationship diagram to tables in a relational database. 2. Map a shared folder by assigning a drive letter
Measure	*Common computing verb in IT and computer science* 1. Measure performance of algorithms on dimensions of time and space. 2. Measure the round trip time (RTT) between two servers.
Patch	*Verb used in computing particularly in IT and cybersecurity* 1. Patch the expected or verified vulnerabilities in an object so that the related risks are mitigated in support of business continuity. 2. Patch an existing vulnerability by improving the functionality, usability, or performance of a given program.
Provision	*Common computing task in cloud computing areas* 1. Provision resources in a virtual environment. 2. Provision cloud equipment, software or services to users/clients.
Randomize	*Common verb in algorithms, data sciences, and cybersecurity areas* 1. Randomize numbers in an algorithm to simulate random behavior. 2. Randomize an encryption algorithm by using a degree of randomness in its logic or within a procedure.
Recover	*Common computing task* 1. Recover files from an operating system as a forensics application. 2. Recover data, based upon a given search term, from an imaged system.
Remove	*Common computing task* 1. Remove sensitive data that is no longer needed. 2. Remove common software vulnerabilities for a given piece of code.
Restore	*Common computing task* 1. Restore files and folders from a backup volume to a local or remote drive/folder. 2. Restore code for a program or application from an earlier copy of the software using version control.
Schedule	*Common computing task, especially in IT* 1. Schedule backups to meet data management policies in an organization. 2. Schedule steps in a cybersecurity assessment for a given scenario.
Store	*Common computing task* 1. Store data securely in the cloud. 2. Store data and software securely at an offsite location.

Train	*Used in machine learning and data science* 1. Train a prediction model by using online data to improve its accuracy. 2. Train a machine learning model with stored data.
Unit-test	*Common task especially in computer science* 1. Unit-test a given program and specification for correctness. 2. Unit-test a function for correctness using both a manual and automated approach.
Virtualize	*Action verb in lieu of "use virtualization"* 1. Virtualize components/resources at the server and client levels for a given scenario. 2. Virtualize an application deployment for a given scenario.

Analyzing

Verb	***Explanation*** **and Sample Learning Outcomes**
Articulate	*Allows for a deeper cognitive level when compared to other Bloom's synonyms such as Explain and Describe* 1. Articulate a testing strategy for a given function. 2. Articulate legal issues, authorities, and processes related to digital evidence.
Automate	*Common task in computing disciplines, requiring prior analysis* 1. Automate a user function, such as generating an electronic invoice or generating an error message/alert. 2. Automate an initial incident response for a security breach for a given scenario.
Contextualize	*Allows a deeper insight into computing concepts by studying the context which surround given scenarios* 1. Contextualize security vulnerabilities for a given piece of software within a given scenario. 2. Contextualize social and professional aspects of computing with respect to ethical codes of conduct.
Correlate	*Analyzes the relationship between multiple computing concepts and the impact they have on one another* 1. Correlate the cost of computing resources with improved security. 2. Correlate privacy, security and trust for a given scenario.
Detect	*Common task, especially in cybersecurity* 1. Detect a potential vulnerability in given source code. 2. Detect an instance of false sharing.
Facilitate	*Improving or making an action easier is a common computing task*

	1. Facilitate a discussion on security requirements. 2. Facilitate dynamic file expansion using hashing.
Generalize	*Allows a deeper cognitive level when compared to other Bloom's synonyms such as Summarize and Explain* 1. Generalize the specific tasks completed by a point-of-sale system. 2. Generalize security vulnerabilities in various data structures.
Model	*Common computing task and uses an action verb in lieu of "use modeling"* 1. Model a business problem mathematically or with an algorithm. 2. Model a real-world problem using graphs and trees, such as representing a network topology or the organization of a hierarchical file system.
Monitor	*Common anomalies deterrence and uncovering error task in computing disciplines* 1. Monitor network traffic for security vulnerabilities. 2. Monitor input controls into an end-point to prevent invalid or erroneous data from entering the system. 3. Monitor the identity of users or groups who request access to sensitive resources.
Parallelize	*Variety of uses, such as in programming and during changeover phases in computing disciplines* 1. Parallelize a given algorithm/program. 2. Parallelize the design features of two operating systems. 3. Parallelize an old system with its replacement system until users are assured that the replacement functions correctly.
Predict	*Common analysis and decision making task applicable to many computing and technology projects* 1. Predict the output of an algorithm after it has been trained on a historical dataset and applied to new data. 2. Predict the cost and benefits trade-off that may be associated with a system recovery plan process. 3. Predict for an organization what management, organization, and technology factors are responsible for the difficulties in building electronic medical records systems.
Promote	*Common task in communication and presentation skills when interacting with upper management and stakeholders* 1. Promote an efficient technical solution to upper management. 2. Promote the impact and/or benefit of open source vs. proprietary software for a given task. 3. Promote consideration of the ethical challenges in a new technology.
Simulate	*Common verb applicable to planning, analysis and testing in computing disciplines* 1. Simulate a disaster recovery response. 2. Simulate the role of personnel in a given business group working environment. 3. Simulate mobile device usage during web application testing and

	prototyping.
Trace	*Common task applicable to many computing, system administration, networking, and cybersecurity tasks* 1. Trace a variable's value as it changes throughout a code module. 2. Trace a list attached or linked to personnel who are not allowed access to any part or function of the system. 3. Trace an activity to detect the condition that creates a deadlock.
Translate	*Common task in programming, information systems, networking and security* 1. Translate a business problem into an artificial intelligence and data science solution. 2. Translate a given system design specification into software program code. 3. Translate written communication to oral communication.
Update	*Common computing task* 1. Update a data-based model for a given scenario. 2. Update the features of a website to match the information requirements of an organization or user needs. 3. Update software to include new features.

Evaluating

Verb	***Explanation* and Sample Learning Outcomes**
Adapt	*Common computing task in computing disciplines* 1. Adapt a given network defense strategy to a new network configuration. 2. Adapt a script feature to suit a specific function or task. 3. Adapt software to changing industry standards.
Administer	*Common task in information systems, networking and security* 1. Administer virtual machines serving a wide user base. 2. Administer an information system for a local business such as a doctors office, grocery store or a fast-food restaurant. 3. Administer a web server. 4. Administer tools and techniques to design and implement a technical solution for a system process.
Balance	*Common analysis and decision making task applicable to many computing and technology projects* 1. Balance security and usability in a given scenario. 2. Balance individual activity with group activity in a working environment. 3. Balance energy savings and performance settings for a hardware power plan. 4. Balance between efficient code and well designed applications.

Debug	*Verb used in troubleshooting and error checking task; common task across computing disciplines* 1. Debug a given segment of code. 2. Debug a common printing problem. 3. Debug an entity relationship diagram of a database for anomalies. 4. Debug software by utilizing appropriate techniques.
Decide	*Common verb used in evaluation of best practices in any computing areas* 1. Decide between several appropriate methods for securing a wireless network. 2. Decide on the appropriate resolution to a help desk request. 3. Decide the best practices for safety when working with networks and end points. 4. Decide on the best data structure for a particular scenario.
Defend	*Common task in information systems, cybersecurity and networking domains* 1. Defend a scenario that exposes a potential security weakness. 2. Defend an information system from security threats. 3. Defend the need for faster backbone for a data center. 4. Defend an information system design proposed to an organization
Delegate	*Common verb used in networking and technology activities* 1. Delegate tasks to team members with the expertise necessary to accomplish them. 2. Delegate management of shared resources to users. 3. Delegate permissions in an Active Directory.
Derive	*Useful skill for analysts in most computing disciplines* 1. Derive potential causes for a system failure from log file entries. 2. Derive system specifications from information gained in end user interviews.
Hack	*Common computing task* 1. Hack a client's local area network to determine its resiliency against brute-force attacks. 2. Hack user account passwords under varying password policies using common attacker tools.
Moderate	*Useful soft skill when interacting with computing colleagues and end users* 1. Moderate an electronic discussion on a computing topic within an educational forum. 2. Moderate corporate communications from management to employees.
Optimize	*Common computing goal* 1. Optimize a segment of code by using fast functions. 2. Optimize network flow by configuring Access Control Lists on firewall ports.
Prioritize	*Multiple possible uses in computing as well as soft skills* 1. Prioritize a search algorithm using a data structure such as binary heaps. 2. Prioritize network traffic by enabling quality of service functionality.

	3. Prioritize daily and weekly tasks to accomplish critical work and meet deadlines.
Propose	*Useful soft skill when interacting with computing colleagues and end users; similar to design, but added communication aspect of proposing to an audience* 1. Propose a network cable plan capable of transmitting large video files at high speed. 2. Propose system requirements revisions to stakeholders.
Prove	*Useful in mathematical and forensic environments* 1. Prove elementary theorems on probability. 2. Prove the presence of a security vulnerability in endpoint devices.
Transform	*Similar to Translate but at a deeper cognitive level* 1. Transform a recursive method into an iterative method. 2. Transform a legacy report into a turnaround document by adding QR codes.
Validate	*Useful skill in several computing disciplines* 1. Validate software security within a given test plan. 2. Validate a server cluster using an appropriate system utility.

Creating

Verb	***Explanation*** **and Sample Learning Outcomes**
Collaborate	*Useful soft skill when interacting with computing colleagues and end users* 1. Collaborate with team members to produce a given computer artifact. 2. Collaborate with fellow teammates to conduct a needs analysis for a point-of-sale system.
Compose	*Useful integrative skill; more flexible than Program* 1. Compose a new class using principles of object-orientation to support a new feature in a software system. 2. Compose shell scripts that run without direct interaction.
Generate	*Useful skill for developing content from existing data or tools* 1. Generate meaningful reports from various data sets. 2. Generate revealing statistics from spreadsheet data by using appropriate Microsoft Excel functions.
Program	*Common computing task* 1. Program a routine that starts code segments on multiple devices when a certain condition is met. 2. Program an automated routine that will pause the smart sprinkler controller when local rainfall exceeds a given threshold.

Reengineer	*Useful skill in some computing disciplines* 1. Reengineer an iterative-based algorithm using recursion. 2. Reengineer a business process based on new hardware capabilities.
Refactor	*Useful skill across several computing disciplines* 1. Refactor a computer program using well-known design patterns. 2. Refactor legacy code to reduce its vulnerability against modern cyberattacks.
Script	*Common computing task; more specific than Program* 1. Script an algorithm to perform form verification using pattern matching and regular expressions. 2. Script commands to instantiate virtual resources.
Secure	*Common computing task across all computing disciplines* 1. Secure a transaction between a browser and a web server. 2. Secure a local area network against denial-of-service attacks.
Visualize	*Useful integrative skill with certain computing tools* 1. Visualize a data model using well-known libraries for data analysis. 2. Visualize a project schedule via a Gantt chart.

Reformulating Learning Outcomes in Existing ACM Curriculum Guidelines Using Proposed Verbs

To further demonstrate how the proposed verbs could be used, we have taken some competencies and learning outcomes (LOs) from the below listed ACM curriculum guidelines and reformulated them using a new verb from the proposed verbs list.

- CCDS2021 - Computing Competencies for Undergraduate Data Science Curricula [a21]
- CSEC2017 - Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity [j17]
- IT2017 - Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology [t17]
- CS2013 - Curriculum Guidelines for Undergraduate Programs in Computer Science [j13]

These examples in the table below show how a formulation can be improved, made more direct, or targeted at a more appropriate cognitive level.

ACM Guideline	**Existing Competency / LO**	**Possible Competency / LO with New Verb**
CCDS2021	Demonstrate contexts in which Bayesian networks can be useful	**Contextualize** effective Bayesian networks (e.g., diagnostic

	(e.g., diagnostic problems).	problems). *Analyzing*
CCDS2021	Justify the need for probabilistic reasoning.	**Simulate** a data model using probabilistic reasoning. *Analyzing*
CCDS2021	State what a Markov Decision Process is, giving a small or medium sized example.	**Trace** a Markov Decision Process with a small or medium sized example. *Analyzing*
CSEC2017	Demonstrate the ability to implement approaches for detection and mitigation of social engineering attacks.	**Detect** social engineering attacks. *Analyzing* NOTE: Mitigation would need to be incorporated separately.
CSEC2017	Explain the various authentication techniques and their strengths and weaknesses.	**Monitor** various authentication techniques. *Analyzing*
CSEC2017	Describe a buffer overflow and why it is a potential security problem.	**Patch** a buffer overflow security problem in a given program. *Applying*
IT2017	Implement virtualization for applications, desktops, servers, and network platforms.	**Virtualize** applications, desktops, servers, and network platforms for a given scenario. *Applying*
IT2017	Illustrate the goals of secure coding, and show how to use these goals as guideposts in dealing with preventing buffer overflow, wrapper code, and securing method access.	**Secure** a given piece of software with respect to buffer overflow, wrapper code, and method access. *Creating*
IT2017	Perform simulations and describe security and performance issues related to wireless networks.	**Simulate** various security and performance issues related to wireless networks. *Analyzing*
CS2013	Use refactoring in the process of modifying a software component.	**Refactor** a software component. *Creating*
CS2013	Explain what is meant by “best”, “expected”, and “worst” case behavior of an algorithm.	**Measure** “best”, “expected”, and “worst” case behavior of a given algorithm. *Applying*
CS2013	Give examples that illustrate time-space trade-offs of algorithms.	**Compute** run-time analysis and Big-O notation including time and space for an algorithm. *Applying*

There are also cases in past ACM curriculum guidelines where a competency or learning outcome uses one of the new proposed verbs, demonstrating the usefulness of that verb already. Examples include the following:

- In IT2017: **Collaborate** in the creation of an interesting and relevant app (mobile or web) based on user experience design, functionality, and security analysis and build the app's program using standard libraries, unit testing tools, and collaborative version control.
- In CS2013: **Refactor** a program by identifying opportunities to apply procedural abstraction.
- In CS2013: **Parallelize** an algorithm by applying task-based decomposition.
- In CS2013: **Parallelize** an algorithm by applying data-parallel decomposition.

Original Plus Enhanced Verbs

The tables below present the original list of verbs plus the new proposed verbs. We welcome input on which presentation is the most useful.

Original verbs with **proposed verbs in bold**:

Remembering	Understanding	Applying		Analyzing	Evaluating	Creating
Define	**Annotate**	Apply	**Interview**	Analyze	**Adapt**	Assemble
Duplicate	Classify	**Backup**	Investigate	**Articulate**	**Administer**	**Collaborate**
Enumerate	**Comment**	**Build**	**Iterate**	Attribute	Appraise	**Compose**
Find	Convert	Calculate	Manipulate	**Automate**	Argue	Construct
Identify	Demonstrate	Carry out	**Map**	Categorize	Assess	Create
Label	Describe	**Code**	**Measure**	Compare	**Balance**	Design
List	Differentiate	**Compile**	Modify	**Contextualize**	Choose	Develop
Locate	Discuss	**Compute**	Operate	Contrast	Critique	Devise
Memorize	Exemplify	**Configure**	**Patch**	**Correlate**	Debate	Formulate
Name	Explain	**Connect**	Perform	Decompose	**Debug**	**Generate**
Recall	**Follow**	**Decrypt**	Produce	Deconstruct	**Decide**	Hypothesize
Recognize	Infer	**Deploy**	**Provision**	Deduce	Defend	Invent
Reference	Interpret	Diagram	**Randomize**	**Detect**	**Defend**	Make
Retrieve	Paraphrase	**Document**	**Recover**	Discriminate	**Delegate**	Plan
Select	Report	Edit	**Remove**	Distinguish	**Derive**	**Program**
State	Summarize	**Encourage**	**Restore**	Examine	Estimate	**Reengineer**
	Translate	**Encrypt**	**Schedule**	**Facilitate**	Evaluate	**Refactor**
		Execute	Solve	**Generalize**	**Hack**	**Script**
		Experiment	**Store**	Integrate	Judge	**Secure**
		Graph	**Train**	**Model**	Justify	**Visualize**
		Hash	**Unit-test**	**Monitor**	**Moderate**	
		Illustrate	Use	Organize	**Optimize**	
		Implement	**Virtualize**	Outline	**Prioritize**	
		Install	Write	**Parallelize**	**Propose**	
				Predict	**Prove**	
				Promote	Support	
				Simulate	Test	
				Structure	**Transform**	
				Trace	**Validate**	
				Translate	Value	
				Update	Verify	

Original verbs plus **proposed verbs in bold** at the end of each section:

Remembering	Understanding	Applying		Analyzing	Evaluating	Creating
Define	Classify	Apply	**Decrypt**	Analyze	Appraise	Assemble
Duplicate	Convert	Calculate	**Deploy**	Attribute	Argue	Construct
Find	Demonstrate	Carry out	**Document**	Categorize	Assess	Create
Identify	Describe	Edit	**Encourage**	Compare	Choose	Design
Label	Differentiate	Diagram	**Encrypt**	Contrast	Critique	Develop
List	Discuss	Execute	**Experiment**	Decompose	Debate	Devise
Locate	Exemplify	Illustrate	**Graph**	Deconstruct	Defend	Formulate
Memorize	Explain	Implement	**Hash**	Deduce	Estimate	Hypothesize
Name	Infer	Investigate	**Install**	Discriminate	Evaluate	Invent
Recall	Interpret	Manipulate	**Interview**	Distinguish	Judge	Make
Recognize	Paraphrase	Modify	**Iterate**	Examine	Justify	Plan
Retrieve	Report	Operate	**Map**	Integrate	Support	**Collaborate**
Select	Summarize	Perform	**Measure**	Organize	Test	**Compose**
State	Translate	Produce	**Patch**	Outline	Value	**Generate**
Enumerate	**Annotate**	Solve	**Provision**	Structure	Verify	**Program**
Reference	**Comment**	Use	**Randomize**	**Articulate**	**Adapt**	**Reengineer**
	Follow	Write	**Recover**	**Automate**	**Administer**	**Refactor**
		Backup	**Remove**	**Contextualize**	**Balance**	**Script**
		Build	**Restore**	**Correlate**	**Debug**	**Secure**
		Code	**Schedule**	**Detect**	**Decide**	**Visualize**
		Compile	**Store**	**Facilitate**	**Defend**	
		Compute	**Train**	**Generalize**	**Delegate**	
		Configure	**Unit-test**	**Model**	**Derive**	
		Connect	**Virtualize**	**Monitor**	**Hack**	
				Parallelize	**Moderate**	
				Predict	**Optimize**	
				Promote	**Prioritize**	
				Simulate	**Propose**	
				Trace	**Prove**	
				Translate	**Transform**	
				Update	**Validate**	

Original verbs plus **proposed verbs in bold** in a separate section:

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Define	Classify	Apply	Analyze	Appraise	Assemble
Duplicate	Convert	Calculate	Attribute	Argue	Construct
Find	Demonstrate	Carry out	Categorize	Assess	Create
Identify	Describe	Edit	Compare	Choose	Design
Label	Differentiate	Diagram	Contrast	Critique	Develop
List	Discuss	Execute	Decompose	Debate	Devise
Locate	Exemplify	Illustrate	Deconstruct	Defend	Formulate
Memorize	Explain	Implement	Deduce	Estimate	Hypothesize
Name	Infer	Investigate	Discriminate	Evaluate	Invent
Recall	Interpret	Manipulate	Distinguish	Judge	Make
Recognize	Paraphrase	Modify	Examine	Justify	Plan
Retrieve	Report	Operate	Integrate	Support	
Select	Summarize	Perform	Organize	Test	
State	Translate	Produce	Outline	Value	
		Solve	Structure	Verify	
		Use			
		Write			

Remembering	Understanding	Applying		Analyzing	Evaluating	Creating
Enumerate	**Annotate**	**Backup**	**Interview**	**Articulate**	**Adapt**	**Collaborate**
Reference	**Comment**	**Build**	**Iterate**	**Automate**	**Administer**	**Compose**
	Follow	**Code**	**Map**	**Contextualize**	**Balance**	**Generate**
		Compile	**Measure**	**Correlate**	**Debug**	**Program**
		Compute	**Patch**	**Detect**	**Decide**	**Reengineer**
		Configure	**Provision**	**Facilitate**	**Defend**	**Refactor**
		Connect	**Randomize**	**Generalize**	**Delegate**	**Script**
		Decrypt	**Recover**	**Model**	**Derive**	**Secure**
		Deploy	**Remove**	**Monitor**	**Hack**	**Visualize**
		Document	**Restore**	**Parallelize**	**Moderate**	
		Encourage	**Schedule**	**Predict**	**Optimize**	
		Encrypt	**Store**	**Promote**	**Prioritize**	
		Experiment	**Train**	**Simulate**	**Propose**	
		Graph	**Unit-test**	**Trace**	**Prove**	
		Hash	**Virtualize**	**Translate**	**Transform**	
		Install		**Update**	**Validate**	

Guidance for Writing Learning Outcomes and Competencies

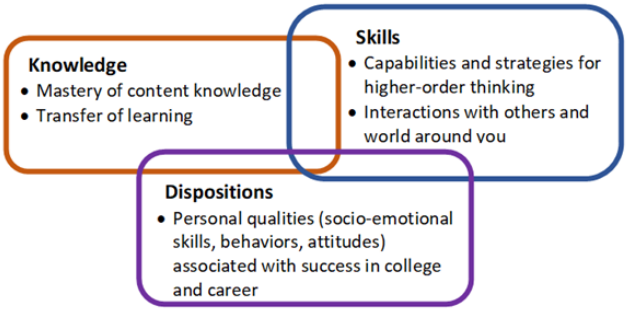
Bloom's verbs are commonly used to write program, course, and lesson competencies and student learning outcomes. For our purposes, competencies follow the definition presented in Modelling Competencies for Computing Education beyond 2020: A Research Based Approach to Defining Competencies in the Computing Disciplines [f18]: "Competency integrates knowledge, skills, and dispositions and is context-situated." Knowledge ("know-that") refers to "mastery of core concepts and content knowledge." Skills ("know-how") are "qualities that

people develop and learn over time with practice and through interactions with others." Dispositions ("know-why" and "know-yourself") include "attitudinal, behavioral, and socio-emotional qualities of how disposed people are to apply knowledge and skills to solve problems." Context is the setting in which competencies manifest, the "authentic situations related to problems/issues and aspects of work." [f18]

Learning outcomes focus on achievement, what students can do in some measurable way, rather than what students know. They are used to develop lesson and course outcomes. Student learning outcomes are more detailed than competencies and are often accompanied with outcome metrics, such as rubrics. It is not unusual to define multiple learning outcomes for a given competency. Learning outcomes contribute to achievement of course competencies which in turn contribute to program competencies.

In summary, competencies are written as general statements that describe desired knowledge, skills, and dispositions of students within an academic program or course while learning outcomes express, in a measurable detailed manner, what a student can do at the end of the lesson, course or program.

Competencies = Knowledge + Skills + Dispositions



Learning Outcomes = Measurable Achievement

Using the six (6) cognitive levels in Bloom's Revised Taxonomy, each of the proposed enhanced computing verbs has an assigned cognitive level. Below are a few examples.

Bloom's Level	***Sample Computing Verbs***
Remembering	Enumerate, Reference
Understanding	Annotate, Comment

Applying	Code, Encrypt, Decrypt
Analyzing	Detect, Trace, Translate
Evaluating	Debug, Defend, Hack
Creating	Script, Refactor, Visualize

A variety of computing concepts are taught and learned at a lower cognitive level initially and progress to a higher cognitive level over time. The enhanced computing verbs provide select "families" of verbs to facilitate writing competencies and learning outcomes which illustrate student achievement and growth from lower to higher cognitive levels. For example,

1. **Code**, **Script**, and **Program** are similar concepts. However, **Code** is at the Applying level whereas **Script** and **Program** are at the Creating level. Students may begin their learning at an Applying level but ultimately demonstrate what they know and can do at the Creating level. Learning outcomes can be written to demonstrate this progress.
2. **Design**, **Develop** and **Build** are also similar. **Design** and **Develop** are verbs from Bloom's Revised Taxonomy at the Creating level. **Build** is a proposed new computing verb at the Applying Level. Again, this provides a way for students to provide a computing solution to a problem but at different cognitive levels.
3. **Unit-test, Test,** and **Validate** can be used to illustrate a growth in how students test and validate software. **Unit-test** is at the Applying level, **Test** and **Validate** are at the Evaluating level.
4. **Convert, Translate,** and **Transform** can be used to illustrate changes to computing artifacts at three different cognitive levels (Understanding, Analyzing, and Evaluating respectively).

At present, ACM recognizes seven (7) computing disciplines [a20]:

1. Computer Engineering
2. Computer Science
3. Cybersecurity
4. Information Systems
5. Information Technology
6. Software Engineering
7. Data Science

Each of these disciplines has its own unique field of study and computing vocabulary. As such, the new computing verbs to enhance Bloom's Revised Taxonomy are inclusive of all computing disciplines. Some verbs may be applicable for all disciplines while others may only apply to one or two disciplines. For example,

- **Encrypt** and **Decrypt** work well for Cybersecurity, Information Technology, and Computer Science.
- **Patch**, **Install**, and **Schedule** are common tasks in Information Technology.
- **Experiment**, **Update**, and **Collaborate** are used across all computing disciplines.

When using these new computing verbs to develop or revise a computing curriculum or course, it is important to remember each verb implies a specific cognitive level of learning. When writing competencies and learning outcomes, authors should carefully select a verb at the appropriate cognitive level for the intended outcome. Use one verb per competency or learning outcome and compose the statement in a measurable, clear, and concise manner. It is recommended that this enhanced list of computing verbs and the associated list of national and regional endorsements be shared with local curriculum committees and other groups who are reviewing curriculum proposals.

Endorsements

We will be seeking endorsements of the final report from various groups. If you represent a group that may be interested in endorsing the final version of this report, please contact the task force chair at cara.tang@pcc.edu.

Acknowledgements

Melissa Stange, Ph.D. of Lord Fairfax Community College, VA provided invaluable input as a team member during the early stages of the project.

The following individuals offered feedback on the first draft of verbs:

Aaron Willcock	Heidi Ellis	Michael S. Kirkpatrick
Alan Hayes	Jakob Barnard	Mihaela Sabin
Anderson	James Davenport	Nathalie Guebels
Ashish Aggarwal	Jeff Merhout	Nicholas Pierce
Bill Kerney	Jeffrey Paone	R. Venkatesh
Brian Ragué	Jennifer Wong-Ma	Raina Mason
Chris Bourke	Jessen Havill	Randy Britto
David G. Kay	Jim Kiper	Richard Bramante
Deborah Boisvert	Joe Paris	Rukiye Altın
Diana Merkel	John A. Trono	Sally Schaffner
Dr. Yousif Mustafa	Josh Archer	Shana Ponelis
Frank Appunn	José Carlos Metrôlho	Susanna Brown
Greg Gagne	Klaas Stoker	Svetlana Peltsverger
Gunnar Wolf	Kristin Stephens-Martinez	Theresa Schmitt
Heather Miles	Michael Bauer	Tim Preuss

References

[a20] ACM and IEEE Computer Society. 2020. *Computing Curricula 2020 (CC2020): Paradigms for Global Computing Education*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3467967

[a21] ACM Data Science Task Force. 2021. *Computing Competencies for Undergraduate Data Science Curricula*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3453538

[a01] Lorin Anderson, David Krathwohl, et al. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Addison Wesley Longman, Inc.

[b56] Benjamin Bloom. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Longmans, Green.

[c15] Donald Clark. Bloom's Taxonomy: The Affective Domain. 2015 https://knowledgejump.com/hrd/Bloom/affective_domain.html. Accessed February 2022.

[f18] Stephen Frezza, Mats Daniels, Arnold Pears, Åsa Cajander, Viggo Kann, Amanpreet Kapoor, Roger McDermott, Anne-Kathrin Peters, Mihaela Sabin, and Charles Wallace. 2018. Modelling Competencies for Computing Education beyond 2020: A Research Based Approach to Defining Competencies in the Computing Disciplines. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '18 Companion), July 2-4, 2018, Larnaca, Cyprus*. ACM, New York, NY, USA, 27 pages. https://doi.org/10.1145/3293881.3295782

[f07] Ursula Fuller, Colin G. Johnson, Tuukka Ahoniemi, Diana Cukierman, Isidoro Hernán-Losada, Jana Jackova, Essi Lahtinen, Tracy L. Lewis, Donna McGee Thompson, Charles Riedesel, and Errol Thompson. 2007. Developing a computer science-specific learning taxonomy. In *Working group reports on ITiCSE on Innovation and technology in computer science education (ITiCSE-WGR '07)*. ACM, New York, NY, USA, 152–170. DOI:https://doi.org/10.1145/1345443.1345438

[j13] Joint Task Force on Computing Curricula ACM and IEEE-CS. 2013. *Computer Science 2013: Curriculum Guidelines for Undergraduate Programs in Computer Science*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/2534860

[j17] Joint Task Force on Cybersecurity Education. 2017. *Cybersecurity Curricula 2017: Curriculum Guidelines for Post-Secondary Degree Programs in Cybersecurity*. ACM, New York, NY, USA. DOI: https://dx.doi.org/10.1145/3184594.

[s17] Klaas Stoker. 2017. A New Cognitive Hierarchy Model for Applied Computer Science. *Journal of Education Research* Vol 10 Issue 4.

[t17] Task Group on Information Technology Curricula. 2017. *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3173161.