DOI:10.1145/3538878

Employers seek recruits who can apply the knowledge, skill, and culture they acquire in college to solve problems as soon as they enter the workforce.

BY JUAN CHEN, SHEIKH GHAFOOR, AND JOHN IMPAGLIAZZO

Producing Competent HPC Graduates

COMPUTING COMPETENCY IS becoming an essential quality needed by industry. For decades, the gap between baccalaureate computing graduates and industry needs was a discussion topic. Most graduates seek employment in deference to continuing their full-time graduate (master's or doctoral) programs. While the percent of such choice varies by institution, it is estimated that about 5% of computing graduates choose full-time graduate study upon graduation, meaning that 95% of computing graduates seek jobs in business, government, or industry.[15] While computing graduates may acquire jobs in today's world, they often lack the competencies (skills and dispositions) expected in the workplace.

Most undergraduate computing-degree programs want to produce job-ready graduates who are productive on the first workday. They often seek local advisory boards composed of industry, government, and business representatives to help develop a functional computing curriculum for their students. Information technology and computing disciplines are changing, and new fields appear continuously. Computing curricula and undergraduate programs are challenged to keep up with this rapid change. Employers are looking for competent graduates who can apply the knowledge, skill, and culture they acquire in college to solve problems as soon as they enter the workforce.

High-performance computing (HPC) and parallel and distributed computing (PDC) have become pervasive. People use them on various platforms, from supercomputers and server farms containing multicore central- and graphical-processing units to personal computers, laptops, mobile phones, and embedded computing devices. HPC systems have broad use for solving large-scale scientific computing problems in various domains, such as atmosphere and ocean studies, astrophysics, industrial design and manufacturing,

» key insights

- **High-performance computing (HPC) and parallel and distributed computing (PDC) are becoming pervasive to more efficiently process an increasingly larger volume of data and computation.**
- **ExaFLOPS computing has become the current standard for HPC.**
- **Competency statements should soon replace learning outcomes in HPC/PDC computing education. Computing competency that promotes dispositions, skills, and knowledge should become the basis for producing competent graduates.**
- **HPC professionals prefer competency-based learning, where dispositions and skills receive greater emphasis beyond knowledge.**

and new energy materials. Specifically, HPC has also become a powerful booster for today's popular computing areas, such as artificial intelligence (AI), big data, and the Internet of Things (IoT). Therefore, understanding how parallelism and distributed computing affect problem-solving is essential for every computing and engineering professional. However, due to the interdisciplinary complexity of the HPC field, finding competent graduates who meet employer needs is becoming difficult.

The notion of competency refers to applying or using the set of related knowledge, skills, and behaviors required to perform "critical work functions or tasks in a defined work setting."[24] Most current computing programs and curricula focus on imparting domain knowledge (topics) to students. A small number of computing programs and curricula have also begun to emphasize teaching students how to apply this knowledge to solve problems in the real world. The word "skills" captures such applications. However, other traits besides domain knowledge and application (skills) exist. People also require passion for work, initiative, critical thinking, problem solving, communication, decision making, adaptation, modification, and extension to perform a job competently.

Teachers have always paid attention to knowledge and have recently begun to attach importance to skills. However, they are often not transparent or cannot understand the drivers of human quality and internal driving forces needed to understand and apply knowledge to complete a job. As a result, employers often have to invest in on-the-job training to help recruits develop the competency required for the jobs at hand. This practice is not good for improving the computing competency of HPC graduates.

Parallel computing is ubiquitous: HPC has penetrated various industries and has become deeply integrated with many computing fields. Industry views HPC in a new light of importance rather than as an ongoing battle to build the fastest supercomputer. HPC has become an integral necessity of scientific and engineering survival. It is the new frontier for business, industry, and government for a new generation of computing.

The main goal of HPC is to reduce the execution times of applications, programs, and operations. PDC, on the other hand, represents how concurrency, and the conceptual decomposition of an application, may occur in a parallel or distributed program. It may not be easy to distinguish between HPC and PDC because modern-day HPC relies on PDC to achieve HPC's high-performance goals, which is the purpose of HPC.[22] Therefore, this article assumes that PDC is a subset of HPC for practical purposes.

Teaching a range of HPC knowledge and skills at multiple levels in computing curricula is necessary. For example, since the early 1990s, computing educators have realized that parallel computation should appear in undergraduate computing curricula.[13] Later, many teaching research works appeared covering curricular experience with parallel computational thinking,[18] parallel programming course practices,[19] and HPC course development and practice.[3,4,17] A recent publication highlighted some critical changes in PDC software and hardware platforms and their effect on educators. It proposed valuable strategies and practical advice for choosing these options and PDC pedagogy.[2]

The changing computing landscape is also presenting challenges in undergraduate computing programs. Employers seek competent graduates who can fit into an industry or business setting, who can apply their knowledge and skills to solve problems in rapidly changing environments, and even those who can create something new to meet future challenges. Unfortunately, current computing curricula and programs focus primarily on knowledge, rarely teach or assess skills, and pay little or no attention to the human element, called dispositions. As a result, employers must invest in job training for recruits. Currently, no competency-based HPC curriculum exists. The closest HPC model is the NSF/IEEE Technical Committee on Parallel Processing (TCPP) curriculum, published in November 2020.[20] Therefore, this work aims to present a rationale for competency-based learning in studying all computing and, in particular, high-performance computing.

Competency and Computing Curricula

In recent years, computing educators have witnessed a shift from knowledge-based to competency-based learning. Knowledge-based learning revolves around the student's knowledge and understanding by doing work;[14] it is an accepted ontology for a specific domain. Knowledge-based learning usually includes an exhaustive list of topics and an accompanying set of learning outcomes associated with a particular knowledge domain.

An essential link between academic preparation and industry needs is competency. A dictionary definition of competency is "possession of sufficient knowledge or skill," implying a quality that expedites achievement or accomplishment.[8] The Association for Computing Machinery (ACM) already uses competency to describe its respective curricular guidelines for baccalaureate undergraduate education. For example, the Information Technology IT2017 Report[11] shifted the focus from knowledge-based learning (knowledge areas, knowledge units, topics, and learning outcomes) to competency-based education, particularly since most IT graduates enter industry or government workplaces upon graduation. Also, IT2017 created the canonical triad whereby:

Competency =[Knowledge +Skills + Dispositions], in Context

This model used competency as the centerpiece of learning. The IT2017 report indicates that competence refers to the performance standards and the assessing of some workplace performance. The purposes of knowledge, skills, and dispositions are valuable to better understand competency. Knowledge is the "know-what" dimension of competency through comprehending facts. An element of knowledge designates a core concept essential to competency. Skills express the application of knowldge components and are the "know-how" dimension of competency. Dispositions describe the "know-why" dimension of competency and shape the "know-what" and "know-how" of competency. Dispositions encompass socio-emotional attributes, behaviors, and attitudes. The set of dispositions is an essential characteristic of a well-structured competency.

Several years ago, ACM and IEEE launched the CC2020 project, a 50-member task force representing 20 countries from six continents. In addition, a 15-member steering committee formed a subset of the task force, and four members formed its editorial committee. The project sought to summarize the current state of curricular guidelines for undergraduate academic programs in computing, guide a transition from knowledge-based to competency-based learning, and set pathways for the future of computing education. The report[1] contrasts knowledge-based learning with competency-based learning, provides visualizations of these paradigms, offers a global perspective on undergraduate computing education, and fosters worldwide integration of computing programs. The report also suggests an alternative interpretation of competency, whereby competency includes three components within a task's performance, represented as:

Competency = [Knowledge + Skills + Dispositions], in Task

which is equivalent to the IT2017 interpretation. Figure 1 illustrates two equivalent illustrations of competency. Figure 1a shows the canonical triad of competency derived from the IT2017 report, while Figure 1b illustrates competency derived from the CC2020 report with knowledge and skills viewed as a pair. Hence, competency emphasizes the overlapping or intersection of knowledge, skills, and dispositions in both cases.

The CC2020 project has already achieved worldwide recognition. Nine organizations have funded the project, and 21 global computing societies have endorsed the CC2020 report. Countries are already starting to consider and implement some of its attributes, and different computing disciplines have already begun to adopt competency as the model for

Figure 1. Competencies from the IT2017 report (a) and from the CC2020 report (b).

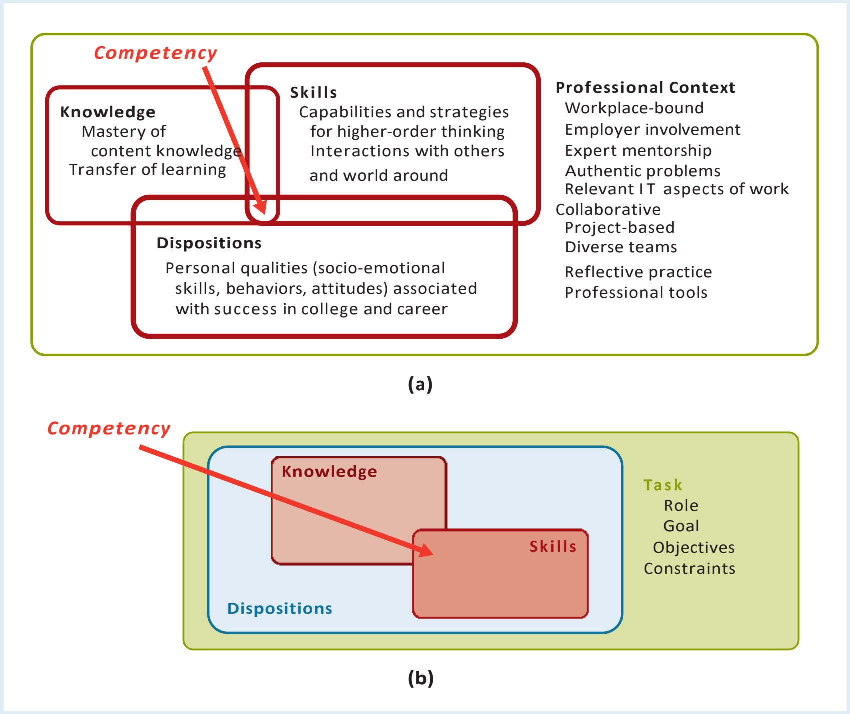

Table 1. Baseline (professional knowledge) elements from CC2020.

Baseline Elements	
Analytical/critical thinking	Collaboration and teamwork
Ethical/intercultural perspectives	Mathematics and statistics
Multi-task prioritization and management	Oral communication and presentation
Problem-solving and troubleshooting	Project and task organization and planning
Quality assurance and control	Relationship management
Research and self-starter learner	Time management
Written communication	

curricula development. In addition to the IT2017 report, an information systems committee has developed a competency-based curricular model in its IS2020 report.[12] Additionally, ACM's data science DS2021[7] report reflects a competency-based recommendation. High-performance computing is beginning to build a competency-based model, and the literature has already proposed this concept.[22] Other areas of computing will likely transform into a competency-based approach to learning.

Table 1 shows 13 baseline elements (or professional knowledge) presented in Table 4.2 of the CC2020 report. These elements represent fundamental components that all competent professionals should possess. For example, all computing professionals should know how to communicate, collaborate, and time-manage their activities. However, the CC2020 report and this work do not enumerate computing or HPC skills, for two reasons. Firstly, a skill is an application of knowledge at a particular level. Secondly, the CC2020 report specified skills as skill levels applicable to knowledge in the context of a task. This article does not try to list any HPC skills except through examples.

Table 2 describes six skill levels, presented in Table 4.3 of the CC2020

Table 2. Levels of skills from CC2020.

Skill Levels		
Remembering	Understanding	Applying
Analyzing	Evaluating	Creating

report, which are commensurate with modern taxonomies to define levels of performance. Finally, Table 3 describes 11 dispositions presented in Table 4.4 of the CC2020 report. Dispositions are essential characteristics of well-structured competency. Therefore, the vital elements for competency for all computing professionals are baseline or professional knowledge (Table 1), cognitive skill levels (Table 2), and dispositions (Table 3).

Table 3. Elements of disposition from CC2020.

Disposition Elements			
Adaptable	Collaborative	Inventive	Meticulous
Passionate	Proactive	Professional	Purpose-driven
Responsible	Responsive	Self-directed	

Table 4. Sample of computing machines and their performance measured in FLOPS.

Phrase	FLOPS	Machine	Year	Speed
FLOPS	1	ENIAC	1946	385 FLOPS
kiloFLOPS = KFLOPS	1 thousand	IBM 704	1954	12 KFLOPS
megaFLOPS = MFLOPS	1 million	IBM 360	1964	1 MFLOPS
		CDC Cray-1	1976	160 MFLOPS
gigaFLOPS = GFLOPS	1 billion	CDC Cray-2	1985	1.9 GFLOPS
		Fujitsu Numerical Wind Tunnel	1993	124 GFLOPS
teraFLOPS = TFLOPS	1 trillion	ASCI Red	1997	1.1 TFLOPS
		IBM Blue Gene/L	2006	136.8 TFLOPS
petaFLOPS = PFLOPS	1 quadrillion	Roadrunner	2008	1 PFLOPS
		Tianhe-2	2013	33.9 PFLOPS
		IBM Summit	2018	122.3 PFLOPS
		Fujitsu Fugaku	2020	442 PFLOPS
exaFLOPS = EFLOPS	1 quintillion	ORNL Frontier	2022	1.102 EFLOPS

Table 5. HPC computing elements of knowledge from the TCPP Initiative.

TCPP Elements	Knowledge Scope	
Pervasive Concepts	Concurrency	Asynchrony
	Locality	Performance
Parallel Architecture	Data parallelism	Control parallelism
	Shared memory	Distributed memory
	False sharing	Energy efficiency
	Floating-point representation	Instructions per cycle
	Memory and network bandwidth	
Parallel Algorithms	Concurrency	Non-determinism
	Costs of computation	Load balancing
	Communication and synchronization	Sorting
	Searching	
Parallel Programming	Shared-memory programming	Distributed-memory programming
	Hybrid heterogeneous programming	Parallel data
	Synchronization	Critical regions
	Load balancing	Scheduling
	Mapping	
Emerging Topics	Internet of Things	Edge computing
	Mobile computing	Distributed security
	Artificial intelligence	Machine learning

High-Performance Computing Curricula

How do we differentiate one computing domain (for example, high-performance computing) from another (for example, computer engineering)? For HPC, computing knowledge and its elements derive from the Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER) project and the TCPP initiative.[20] Furthermore, cognitive skill levels are self-evident and do not require justification. Therefore, it is possible to address how professional knowledge areas (Table 1) and dispositions (Table 3) apply to HPC.

With the rapid development of science and technology, the pursuit of high-performance computer speed is endless, and the quest for computing power is unlimited. Computing power has increased tremendously since one of the earliest computers, the ENIAC, appeared in 1946. According to the June 2022 issue of the TOP500 List, the Frontier is the first machine to break the Exaflop ceiling.[23] Jack J. Dongarra, the 2021 ACM A.M. Turing Award recipient, started compiling the TOP500 List in 1993 with Erich Strohmaier and Horst Simon. The ENIAC could perform 385 floating-point (fractional addition/multiplication) operations per second (FLOPS), while the Frontier performs 1,102 quadrillion (million-billion) FLOPS. Table 4 lists some supercomputers, in their time, to show the increase in computing power over the last seven decades. Performance is measured in FLOPS; each principal row is a thousand times greater than the previous row from the first row.

It is interesting to note two representative supercomputers in the world today. The Tianhe-2 (Figure 2a) supercomputer, developed by China's National University of Defense Technology (NUDT), was rated the world's fastest supercomputer six times between June 2013 and November 2015. The name was changed to Tianhe-2A in 2018. Figure 2b shows today's fastest supercomputer, the Frontier, developed by Oak Ridge National Laboratory (ORNL).

Exascale computing (exaFLOPS)[9] is 1,000 petaFLOPS. The Frontier is the first exascale machine in the

TOP500 List with 1.102 exaFLOPS, equalling 1,102 petaFLOPS. In about 10 years, the world expects to see zettaFLOP machines that are 1,000 times faster than exaFLOP machines. These achievements should motivate students to become competent HPC graduates in the future.

While computer performance accelerates exponentially, its academic relationship seems to suffer from a lack of direction. Little or no HPC educational activity has taken place worldwide. Isolated pockets of such activity exist at different universities, where departments have formulated some HPC courses or an internal HPC curriculum. However, these activities are local and do not necessarily expand globally. For example, Table B.1 of Appendix B in Raj et al.[22] lists representative HPC courses globally. Unlike other computing disciplines with globally published curricular guidelines (for example, ACM/IEEE CE2016 and CC2020 reports), little has occurred with HPC.

In the U.S., some curricular activity has occurred in the HPC domain. The U.S. National Science Foundation (NSF) has supported grants to scientists and engineers to integrate computational and modeling skills into a curriculum. HPC centers funded by national and international agencies, national laboratories, and universities have also provided training and education programs. As modeling and simulation have become an integral part of science and engineering research, there has been a growing need to educate students about the concepts and tools required to create, implement, and validate models.

Several projects have focused on creating educational modules useful in classrooms and workshops.[7,16] Another project developed a set of capabilities for undergraduate students in science and engineering,[10] partitioned into seven major categories. These include modeling and simulation, programming and algorithms, differential equations and discrete dynamical systems, numerical methods, optimization, parallel computing, and scientific visualization. This mix of mathematics, computing principles, and domain science will guide undergraduate curricular development. It is also crucial to introduce parallel computational thinking for non-computing majors in science, technology, engineering, and mathematics (STEM). If students grasp these basic concepts, they might better understand HPC system operating environments and underlying principles of scientific computing.

Figure 2. The Tianhe-2 supercomputer (left) and the Frontier supercomputer (right).

天河
FRONTIER
OAK RIDGE National Laboratory
U.S. DEPARTMENT OF ENERGY
Hewlett Packard Enterprise
AMD

The ACM/IEEE documents have been updated in recent years. They are beginning to recognize that computer professionals must have skills in designing and using large-scale parallel and distributed computing systems, multicore and many-core systems-related programming, and algorithmic requirements. For example, the HPC-related abilities embedded in the CS2013 computer science curriculum document[6] include seven related categories, including an introduction to modeling and simulation and multi-processing. Likewise, the CE2016 computer engineering report[5] provided 11 explicit HPC-related activities, such as identifying concurrency, parallel algorithm structure, parallelism on distributed systems, and decomposition. These activities for computer engineering appeared in three broad segments: algorithmic analysis of parallel algorithms, parallelism from an architectural perspective, and parallelism from a systems perspective.

Competency for HPC

The TCPP Curriculum Initiative[17,20] developed parallel and distributed computing curriculum recommendations for undergraduate computing curricula in 2012 and revised them in 2020. The TCPP initiative recommends topic-level elements for PDC knowledge. The TCPP curriculum divides these knowledge elements into five categories: pervasive concepts, architecture, algorithms, programming, and emerging topics. The curriculum suggests topics covered appropriately in relevant undergraduate classes under these categories. The TCPP curriculum also recommends a set of fundamental pervasive concepts, meaning that all undergraduate computing curricula should cover these concepts in some depth. Table 5 provides a high-level summary of the recommended TCPP curriculum. Details of the knowledge elements of the TCPP curriculum appear in Prasad et al.[20,21]

HPC competency would have the same baseline (professional knowledge) and dispositions as other computing areas. The difference lies in the specific technical elements and related skills of HPC curricular recommendations. Specifically, CS2013, CE2016, and TCPP curriculum recommendations provide elements that describe an HPC technical knowledge/skill foundation.

In addition to the five TCPP curricular categories, HPC users can involve three activities: system design, application development, or system administration. Furthermore, people may have additional HPC knowledge, skills, and dispositions across all categories. For example, an HPC system designer would require in-depth knowledge and skill in computer system design.[4] At the same time, an HPC application developer would need specific domain knowledge about the application domain[3] but would also require some HPC system knowledge. Therefore, it is essential to strengthen HPC competencies in students to cultivate them into the above three specialists.

HPC skills are skill levels based on technical knowledge or application in the HPC field. Examples of the six skill levels related to Table 5 follow:

1. *Remembering* refers to recalling pervasive HPC concepts, the primary/critical concepts in parallel architecture and algorithms.

2. *Understanding* refers to comprehending the differences and characteristics of specific HPC concepts—for example, several parallel architectures, the consideration of data partitioning, load balancing, and overhead in parallel algorithm design.

3. *Applying* refers to using HPC principles such as designing parallel algorithms, solving specific problems, and writing and debugging parallel programs.

4. *Analyzing* refers to decomposing HPC situations and problems, such as exploring performance bottlenecks of existing parallel programs, exploiting performance or energy-efficiency improvement opportunities on a specific parallel platform, or studying the system requirements of a domain-oriented program on the parallel computing system.

5. *Evaluating* refers to integrating HPC principles, such as determining an application's parallel efficiency and scalability or the energy-efficiency ratio of a parallel system.

6. *Creating* refers to HPC inventiveness, such as designing a new architecture to adapt to a specific emerging application, inventing a new architecture suitable for AI applications, or developing various system-level optimization algorithms.

Disposition, in addition to knowledge and skills, is essential for all students. Since problem-solving in the HPC field is challenging and innovative, all HPC graduates must display dispositional attributes, such as being self-directed, passionate, proactive, responsible, collaborative, and adaptable. Such traits should be the most critical abilities they need to advance in follow-up and arduous research. Disposition is critical as students gain more knowledge and skills while progressing through a curriculum and achieving more substantial knowledge and skills.

While computer performance accelerates exponentially, its academic relationship seems to suffer from a lack of direction. Little or no HPC educational activity has taken place worldwide.

For example, suppose students partition into three achievement levels, from L1 to L3, where L1 is the lowest and L3 is the highest regarding student knowledge and skills. Disposition could influence three types of students differently. L1 students must mainly improve their knowledge and skills in this primary stage. Therefore, the demand for disposition may not be meaningful. Here, for L1 students, HPC competency mainly reflects their capacity in knowledge and skills. When students' knowledge and skills improve to the moderate level, L2 students might feel that the disposition affects their knowledge and skills. Student performance may increase because they are more collaborative, proactive, and responsible for performing better. When the students rise to level L3, elements such as being passionate or adaptable may directly affect how they can face setbacks, continue to improve knowledge and skills, and even continue to learn. Hence, disposition ultimately could lie in a dominant position in HPC competency. This anecdotal observation could be the subject of further research on how disposition plays a role in HPC education.

Like other computing domains, being 'collaborative' is vital for HPC. A collaborative disposition means that learners can convey information to their peers or others while receiving information from their audience. The following three examples illustrate this understanding:

1. System designers and developers must communicate with domain experts who have different needs, concerns, and knowledge backgrounds.

2. Message-passing between processes in an application is very common in HPC applications. However, in designing a message-passing library, what a system designer cares about may differ from an application developer who uses the message-passing system.

3. System designers would focus more on efficiently sending a message from the sending buffer to the receiving buffer of the destination process. How-

ever, an application developer may not care about system design and implementation details but would be more concerned about application-level issues, such as how sending/receiving message size affects the application's communication latency and scalability. Using various knowledge domains—combined with purpose-driven, collaborative professional skills and other dispositions to optimize message passing in distributed-memory systems—is an example of competency.

Teamwork is equally essential for HPC learners because HPC problem-solving involves knowledge in multiple fields. Moreover, the difficulty and scale of HPC problems are usually multidisciplinary and complex. Therefore, people from different domains participate: the team works together to complete the problem definition, analysis, solution, verification, and other elements associated with the problem. It is crucial and necessary to engage in teamwork to solve problems together.

Generating HPC competencies follows the same structure as mentioned in the CC2020 report. In addition to baseline elements, skill levels, and dispositions, it is essential to have a 'computing knowledge' category that is technical for a particular field (for example, HPC) and distinguished from baseline elements as described in Figure 5.6 of the IT2017 report. Table 5 shows the TCPP Initiative's HPC computing elements of knowledge. For example, consider an implied HPC competency statement, such as:

Collaborate with colleagues and researchers to develop a scalable parallel program that solves a problem that utilizes a heterogeneous HPC architecture.

This competency statement requires several technical knowledge elements from different categories in Table 5. The user then selects meaningful professional knowledge elements from Table 1. Next, using the skills level from Table 2, the user forms a coupled pair between each knowledge element and a cognitive skill level. Finally, based on the competency-statement context, the user selects at least one significant disposition from Table 3 that aligns with the statement. Table 6 shows the completed cluster for the sample competency statement.

Table 6. HPC competency cluster from a sample competency statement.

Competency Title (Implied HPC Competency)

Competency Statement
Collaborate with colleagues and researchers to develop a scalable parallel program that solves a problem that utilizes a heterogeneous HPC architecture.

Knowledge	**Skills Level**		
Technical Elements [Table 5]	[Table 2]		
Concurrency	Understanding		
Hybrid heterogeneous programming	Creating/Evaluating		
Performance	Analyzing/Evaluating		
Communication and synchronization	Analyzing/Applying		
Load balancing	Analyzing/Applying		
Memory and network bandwidth	Evaluating		
Baseline Elements [Table 1]	[Table 2]		
Collaboration and teamwork	Applying		
Problem-solving and troubleshooting	Analyzing		
Analytical/critical thinking	Analyzing		
Dispositions [Table 3]			
Collaborative	Purpose-driven	Professional	Responsible

Table 7. Results (important or very important) for professional knowledge (U.S. and Non-U.S.).

Baseline (Professional Knowledge) Elements	**U.S. Results (%)**	**Non-U.S. Results (%)**	**Weighted Average (%)**
Analytical and critical thinking	100.00	95.65	96.41
Collaboration and teamwork	75.00	93.48	90.24
Ethical/intercultural perspectives	70.00	39.78	45.08
Mathematics and statistics	75.00	87.91	85.65
Multi-task prioritization, Management	70.00	91.30	87.56
Oral communication, Presentation	75.00	75.00	75.00
Problem solving, Troubleshooting	100.00	95.65	96.41
Project/task organization, Planning	70.00	85.72	82.96
Quality assurance and control	60.00	85.87	81.33
Relationship management	25.00	51.09	46.51
Research and self-starter/learner	80.00	90.10	88.33
Time management	70.00	92.39	88.46
Written communication	80.00	73.62	74.74
Average	**73.08**	**81.35**	**79.90**

Table 8. Results (important or very important) for dispositions (U.S. and Non-U.S.).

Dispositions	**U.S. Results (%)**	**Non-U.S. Results (%)**	**Weighted Average (%)**
Adaptable	85.00	93.48	91.99
Collaborative	75.00	89.01	86.55
Inventive	75.00	91.31	88.45
Meticulous	70.00	91.31	87.57
Passionate	70.00	85.87	83.09
Proactive	78.95	96.73	93.61
Professional	75.00	95.65	92.03
Purpose-driven	70.00	96.74	92.05
Responsible	80.00	100.00	96.49
Responsive	75.00	64.13	66.04
Self-directed	95.00	95.66	95.54
Average	**77.18**	**90.90**	**88.49**

Need for Competency within HPC

Though educators might feel computing and HPC students should have baseline skills and proper attitudes, the authors believed it was essential to ask practicing professionals and academics their opinions on this assumption. In addition, more detail and concrete evidence were necessary to support these expectations reflecting skills and dispositions and to understand possible differences in perceptions of competencies across countries.

To test the level of interest in competency-based HPC education, the authors surveyed a broad community of approximately 600 HPC people, including industry practitioners, researchers, and academics, gathering their views on the importance of baseline/professional knowledge and dispositions in HPC education. For the 114 responses, Table 7 summarizes the importance of baseline/professional knowledge (important or very important), while Table 8 summarizes the importance of dispositions. The non-U.S. responses were about 4.72 times the U.S. responses; the authors used this ratio to calculate the weighted average for each table.

From Table 7, HPC respondents showed an affinity for the elements of professional knowledge with a weighted average of 79.9% appeal. Likewise, in Table 8, respondents preferred disposition elements with a weighted average of 88.49% appeal. These results suggest professional knowledge and dispositions are valued components of HPC competency.

Based on these survey results and author experience, it seems clear that HPC graduates should reflect competency in their education. This understanding has four fundamental reasons: interdisciplinary, competitive, innovative, and ultimate experience. First, the HPC field is interdisciplinary, where graduates learn knowledge from domain experts in other areas, which makes HPC competitive. Second, HPC is competitive because failure in the field is an expectation, so students should have the spirit of optimism and learn to never give up on their challenges. Third, innovation is key to HPC, where students learn to complete tasks independently and cooperatively to achieve true innovation. Lastly, HPC students can have the ultimate experience because HPC and supercomputing are among the world's highest-performance computer achievements. People challenge these unknown areas even when faced with multiple failures.

HPC is competitive because failure in the field is an expectation, so students should have the spirit of optimism and learn to never give up on their challenges.

Examples of supercomputer applications include astrophysics, meteorology, chemical analysis, and biomedical research. One recent example was vaccine development to fight the relatively unknown, constantly mutating COVID-19 virus during the pandemic. The world could not advance the development process without the help of supercomputers. Therefore, competency in HPC is foundational in this important computing field.

What's Next?

As stated in the introduction, this article aims to present a rationale for promoting competency-based learning in the study of computing. From the HPC perspective, industry and academia need to produce more-competent graduates with the basic HPC knowledge and skills required for modern-day computing environments. Of concern is that current computing programs address knowledge-based learning. Even then, most programs do not cover HPC topics, and hardly any program addresses competency in practice. Furthermore, enhancing competency learning would be a multiyear effort involving communities, universities, and industry. Curriculum change requires the broad participation of the community. This article does not focus on how to do it. Instead, this work aims to attract the attention of educational computing communities to consider competency-based learning as an augmented asset to their graduates so they can attain more productive and successful futures.

It would be good if faculty members taught their students as whole people rather than technocrats through competency. It is relatively easy for computing teachers to transfer technical knowledge to students because knowledge is usually specific content. However, learning skills, boosting skill levels, and advancing dispositions is more challenging. Skills are closely related to knowledge and refer to multiple levels of accomplishment.

Industry and government jobs very often require competency (knowledge, skills, and dispositions) to evaluate employees. Yet, universities often ignore this facet of education. For example, computing teachers rarely promote dispositions in computing contexts, even though they are critical aspects of being a whole person. In addition, computing and HPC computing faculty members rarely assess student skills or dispositions as a component of the course grade. Educational competencies should soon influence and modify computing teaching paradigms.

For example, teachers would likely develop more excellent student skills with corresponding skill levels and interpretations for specific computing fields. Teachers would also probably pay more attention to shaping students' dispositions and studying how they affect learning and skills development. Teachers would also begin to incorporate skill and disposition elements into the HPC and computing curricula to form a unified teaching design of the knowledge-skills-dispositions triad. Finally, transitioning from knowledge-based learning to comprehensive competency-based strategies that incorporate knowledge, skills, and dispositions should positively influence HPC, and all computing, by developing graduates with the competence to engage in further research or enter the workplace.

The time has come to transform knowledge-based HPC education into a competency-based approach. Students are humans, not robots. Promoting the triad of knowledge, skills, and dispositions—competency—should be part of every HPC student's experience. It's time to integrate competency with HPC curricula. Even recent curricular reports by ACM and IEEE have taken a competency-based approach to their recommendations. Universities should not produce HPC graduates solely on a knowledge-based system. Instead, departments should seek to develop HPC specialists as competent and whole beings with the skills and dispositional fortitude to become capable contributors or creators to the field. Essential technical knowledge elements are not enough for a student to become a competent professional. Instead, they should hone their skills and develop dynamic, positive, collaborative, and constructive behaviors and dispositions toward solving complex problems and creating more powerful computing systems. More importantly, the TCPP curricular guidelines should augment its knowledge-based, topic-oriented recommendations to include the dimensions of skills and dispositions.

Knowledge with skills and dispositions has arrived, not only for HPC curricula but also for all computing guidelines. HPC education can benefit significantly from competency-based learning. It should provide students with powerful and natural experimental environments to immerse themselves and perform within practical settings. The future is bright for HPC competency-based education. Let us embrace it today and not wait until tomorrow.

Acknowledgment

The authors acknowledge the support provided by the 2022 Hunan Province Higher Education Teaching Reform Research Foundation of China, grant number HNJG-2022-0007 (Innovative Practice of Training "Informatization + Intelligence" Integrated Talents based on the First-class High-Performance Computing Platform), and by the U.S. National Science Foundation grant numbers OAC 1730417 and OAC 2017309. C

References

1. *A Computing Curricula Series Report Computing Curricula 2020: Paradigms for Global Computing Education.* ACM/IEEE Computer Society (IEEE-CS); https://bit.ly/3TDu0kO.
2. Adams, J.C. Evolving PDC curriculum and tools: A study in responding to technological change. *J. of Parallel and Distributed Computing 157* (2021), 201-219; https://doi.org/10.1016/j.jpdc.2021.07.003.
3. Chen, J., Impagliazzo, J., and Shen, L. High-performance computing and engineering educational development and practice. In *Proceedings of 2020 IEEE Frontiers in Education Conf.* (Oct. 2020), 1-8; https://doi.org/10.1109/FIE44824.2020.9274100.
4. Chen, J. et al. Design of practical experiences to improve student understanding of efficiency and scalability issues in high-performance computing. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (February 2018); DOI: https://doi.org/10.1145/3159450.3162239.
5. *Computer Engineering Curricula 2016: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering.* ACM/IEEE-CS Joint Task Force on Computer Engineering Curricula; https://bit.ly/3TQyrZg.
6. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM/IEEE-CS Joint Task Force on Computing Curricula (January 2013); https://doi.org/10.1145/2534860.
7. *Computing Competencies for Undergraduate Data Science Curricula.* ACM Data Science Task Force (January 2021); https://bit.ly/3VVg0EN.
8. Definition of "Competency." Merriam-Webster; https://www.merriam-webster.com/dictionary/competency.
9. Exascale Computing Project. Los Alamos National Laboratory; http://bit.ly/3FfWRHz.
10. Gordon, S.I., Carey, K., and Vakalis, I. A shared, interinstitutional undergraduate minor program in computational science. *Computing in Science and Engineering 10*, 5 (August 2008), 12-16; DOI: https://doi.org/10.1109/MCSE.2008.127.
11. *Information Technology Curricula 2017: Curriculum Guidelines for Baccalaureate Degree Programs in Information Technology.* ACM /IEEE Computer Society (IEEE-CS); https://doi.org/10.1145/3173161.
12. *IS2020: A Competency Model for Undergraduate Programs in Information Systems.* ACM/AIS IS2020 Task Force; https://bit.ly/3zaZuGQ.
13. John, D.J. Integration of parallel computation into introductory computer science. In *Proceedings of the 23rd SIGCSE Technical Symp. on Computer Science Education* (1992), 281–285; https://doi.org/10.1145/134510.134567.
14. KBL: Knowledge-based learning. TES; http://bit.ly/3DyTMBq.
15. Murphy, T.J. How many people go to graduate school and a few other questions. GradSchoolMatch.com (March 12, 2017); https://blog.gradschoolmatch.com/people-going-graduate-school/.
16. National Computational Science Institute (NCSI); http://www.computationalscience.org/.
17. NSF/IEEE-TCPP curriculum initiative. Center for Parallel and Distributed Computing Curriculum Development and Educational Resources (CDER); https://tcpp.cs.gsu.edu/curriculum/?q=node/21183.
18. Orozco, E., Arce-Nazario, R., Ortiz-Ubarri, J., and Ortiz-Zuazaga, H. A curricular experience with parallel computational thinking: A four years journey. In *Proceedings of Workshop on Parallel, Distributed, and High-Performance Computing in Undergraduate Curricula* (2013).
19. Pollock, L. and Jochen, M. Making parallel programming accessible to inexperienced programmers through cooperative learning. In *Proceedings of the 32nd SIGCSE Technical Symp. on Computer Science Education* (2001), 224–228; https://doi.org/10.1145/364447.364589.
20. Prasad, S.K. et al. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing—Core topics for undergraduates. *NSF/IEEE-TCPP Curriculum Working Group*; https://bit.ly/3f2HkAc.
21. Prasad, S.K. et al. NSF/IEEE-TCPP curriculum initiative on parallel and distributed computing: Status report. In *Proceedings of the 49th ACM Technical Symp. on Computer Science Education* (Feb. 2018), 134-135. https://doi.org/10.1145/3159450.3159632.
22. Raj, R.K., Romanowski, C.J., and Impagliazzo, J. et al. High-performance computing education: Current challenges and future directions. In *Proceedings of the Working Group Reports on Innovation and Tech. in Computer Science Education* (June 2020), 51-74; https://doi.org/10.1145/3437800.3439203.
23. TOP500; https://www.top500.org/lists/top500/2022/06/.
24. What is competency and how is it assessed? Department of Mines, Industry Regulation and Safety, Government of Western Australia; http://bit.ly/3srOO2X.

Juan Chen is a professor at the College of Computer, National University of Defense Technology, Changsha, Hunan, China.

Sheikh Ghafoor is a professor in the Computer Science department at Tennessee Tech University, Cookeville, Tennessee, USA.

John Impagliazzo (john.impagliazzo@hofstra.edu) is a Professor Emeritus at the School of Engineering and Applied Science, Hofstra University, Hempstead, New York, USA.