2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #50348

Integration of Best Practices (BP) into Computing Gateway Courses: Implications for Success in the Discipline

Dr. Trevor Cickovski, Florida International University

Dr. Trevor Cickovski is an Associate Teaching Professor and Associate Director of the Knight Foundation School of Computing and Information Sciences (KF-SCIS) at Florida International University (FIU). He received his Ph.D. from the University of Notre Dame in 2008 and is a member of ACM, IEEE and the National Learning Assistant Alliance (LA). As a Center for the Advancement of Teaching (CAT) Fellow at FIU, Trevor participated in a University-wide effort to improve teaching evaluation and developed a website of resources for faculty. He has received funding from NVIDIA to bring GPU computing into the classroom and is lead software developer of PluMA, a plugin-based toolkit that facilitates classroom bioinformatics projects by allowing students to build their ideas in a language of choice and seamlessly integrate them into larger pipelines. He has led the development of two interdisciplinary computing degrees at FIU: one in computer science and biology, the other in computer science and psychology.

Jason Liu, Florida International University

Jason Liu is the Interim Director and a University Eminent Scholar Chaired Professor at the Knight Foundation School of Computing and Information Sciences, Florida International University (FIU) in Miami, Florida, USA.

Dr. Mark Allen Weiss, Florida International University

Mark Allen Weiss is Distinguished University Professor and Interim Vice Dean of the College of Engineering and Computing

Mrs. Monique S. Ross, The Ohio State University

Monique Ross earned a doctoral degree in Engineering Education from Purdue University. She has a Bachelor's degree in Computer Engineering from Elizabethtown College, a Master's degree in Computer Science and Software Engineering from Auburn University

Carla E. Brodley, Northeastern University

Integration of Best Practices (BP) into Computing Gateway Courses: Implications for Success in the Discipline

Computing "gateway" courses, i.e., Programming I (CS1) and Programming II (CS2), in addition to providing the first introduction to the discipline, also present students with foundational knowledge that is critical for success in the discipline. The gateway courses are not only the most critical for retention in the major but also for success in the major. At a R1 institution with more than 4,300 computing majors, we implemented four interventions known for improving student success rates (which we collectively term "Best Practices" or BP) into computer science (CS) gateway courses: synchronized content, common assessment, hands-on learning, and undergraduate teaching assistants. Over a two-year period, we tracked students (more than 6,000 anonymized student records) who took BP gateway course sections and compared their passing rates in multiple upper-division CS courses to the pass rates of students who took non-BP gateway sections. We also perform ANalysis Of VAriance (ANOVA) on the data to determine if there is a statistically significant connection between enrollment in a BP-gateway CS course and passing an upper-division CS course. We performed these analyses over the same time period to remove external confounding factors such as COVID-19 and the ubiquity of large-language models (LLMs), and additionally confirmed teaching evaluations as a non-confounding factor through subsequent ANOVA analysis. In the end, we found that for all CS upper-division core courses, the pass rate for students who had completed a BP gateway course section were higher than the pass rate for students who did not, and with very few exceptions, this difference was both a double-digit increase and statistically significant. Also, for every course where these interventions were implemented, student success rates improved. These results illustrate that these four BP interventions can create a positive feedback loop when implemented across the board, both improving the success rates of their target gateway course and additionally translating to improved performance in upper-division courses (even those that have not implemented BP).

1 Introduction

The term "gateway" course refers to courses that are critical for success in a major [1]. Gateway courses are typically taken early in the curriculum, providing foundational knowledge upon which other major requirements build, and are therefore some of the most important courses for major retention [2]. This also holds in the computing field. Statistically, most students who switch out of computing do so sometime during the gateway sequence, especially the first course in the programming sequence (CS1) [3]. Pioro [4] showed statistically significant correlations between performance on both programming problems and multiple choice questions on final exams from an introductory CS course and GPA two years later. Beck *et al*. identified a specific set of questions from an introductory programming course that could predict future success in the discipline [5].

Gateway courses offer both challenges and opportunities with respect to being high risk and highly enrolled, and computing is no exception [6]. Relative to other courses in the computing discipline, computing gateway courses tend to have higher DFW rates [7]. Additionally, computing gateway class sizes tend to be larger than non-gateway courses [8], and students enter these courses with a wide range of prior programming experience, ranging from none to expert [9]. Collectively, this creates an environment with the most heterogeneous set of student needs, with one of the highest student-to-faculty ratios, in courses that are most critical for discipline retention and success.

1.1 Related Work

These same challenges illustrate the opportunities that arise from and the potential power of interventions in the computing gateway sequence. Previous studies have connected pedagogical interventions in gateway courses with success in those same gateway courses. For example, Benkarroum et al. [10] demonstrated improved student engagement and problem-solving proficiency upon infusing computational thinking into their CS1 course. A team of faculty at Alabama A&M University [1] improved student cognitive and non-cognitive skills, in addition to DFW rates, by integrating problem- and project-based learning within their CS1 course. Davis [11] demonstrated improvements in multiple metrics (final grades, DFW rates, CS

Concept Inventory, tutorial completion rates and test performance) upon integration of various hands-on learning practices within a CS1 course.

Additionally, studies have illustrated the relationship between success in gateway courses and success in upper-level computing courses. Bisgin et al. [14] reported a statistically significant correlation ($p < 1e\text{-}3$) between CS2 and Data Structures grades over more than one hundred students. Sondakh and Pungus [15] demonstrated a statistically significant correlation between Programming I (CS1) and Programming II (CS2) grades with over six years of student data at Universitas Klabat. This study found that students who received a "C" in Programming I and/or Programming II took an average of one semester longer to graduate compared to students who received an "A". In Programming II, students who received a "C-" actually took 1.5 semesters longer to graduate.

1.2 Hypothesis

Given these two observations, namely:

(1) Pedagogical interventions in gateway courses translate to success in those same gateway courses.
(2) There is a relationship between success in gateway courses and success in upper-level computing courses.

Our hypothesis is that a transitive dependence holds, i.e.: pedagogical interventions in gateway courses translate to success in upper-level computing courses.

However, to our knowledge, no explicit connections have been established in the literature. A twelve-year study by Salguero et al. [13] illustrated statistically significant correlations between pedagogical interventions in CS1 and more global metrics, such as major retention and switch (into) rates, time to degree for students with prior programming experience, and upper-division GPAs for students without prior programming experience. Their interventions included media computation (teaching programming in the context of manipulating images and sounds), pair programming (alternating "driver" and "navigator" roles while programming), and peer instruction (with lecture structured as a discussion between instructor and other peers).

In our study, we propose to implement pedagogical interventions in a subset of our CS1 and CS2 course sections, follow students through the discipline, and determine if students who took the pedagogically-modified sections experience greater success in subsequent CS coursework than those who did not.

1.3 Study Objectives

We chose four interventions with documented success for improving success rates in both general and computing-specific coursework, to which we collectively refer as "Best Practices" (BP). These include:

Synchronized Content. This addresses the problem of *course coordination*, as a method for ensuring consistency across different sections of gateway courses [16]. For CS1, this amounted to an average of six sections every semester (420 students total), and for CS2, an average of nine sections (469 students total), Many models for implementing course coordination exist, with various associated levels of content synchronicity. In an attempt to balance synchronized content while upholding academic freedom and recognizing that instructors of a course are the ultimate experts in the associated material, we choose a hybrid uniform/non-uniform model [17]. Every semester we formed two committees (one for CS1, the other for CS2), that each consisted of the faculty teaching those respective courses in that semester. Upon initial formation, these committees were tasked with developing a uniform master syllabus for their respective courses. These committees then met bi-weekly and agreed on the portions of this master syllabus that should be uniform, and those that could be non-uniform. The presence of the master syllabus was used to help create a mentality where uniformity would be the "default", and non-uniformity would be the "exception". Newly integrated faculty every semester would first use the master syllabus as a reference, and any change they proposed would need to be collectively approved by their respective committees.

Common Assessment. Uniform assessment practices across sections can hold many advantages in terms of reducing cross-instructional variability and resource sharing, in addition to flexible support across different sections of a course [18]. Once again, to achieve an adequate balance with upholding academic freedom, we used a hybrid model similar to our strategy for synchronized content. Every semester, the CS1 and CS2 committees would collaboratively build an exam bank for their respective courses. Each examination would have (1) a certain percentage of its questions from the bank and be consistent across all sections, (2) a certain percentage of its questions randomly pulled from the bank, and (3) a certain percentage of its questions developed independently by each instructor. These percentages were determined by the respective committees and had the flexibility to change every semester as the course instructors deemed necessary. Assignments were also shared amongst the faculty, once again with the potential for some variation (assignments unique to a specific section, or slightly different flavors from another sections); but as with examinations, any such variation needed to be mutually agreed upon by all members of the committee.

Hands-On Learning. Hands-on learning, centered around the concept of *learning by doing*, has its underpinnings in the methods by which humans construct knowledge by interacting with their environment [19] and experiential learning theory [20]. The benefits of hands-on learning on knowledge retention are well-established and common practice [21][22][23][24][25][26][27][28][29][30], explicitly recognized for computer programming courses [31], general STEM courses [32], and even in non-academic climates for training workers [33][34]. Experiential learning has also been shown to be vital for computing education accessibility [35], even for upper-level machine learning [36] and theory [37] courses. The primary resource used to administer hands-on learning in our programming gateway courses was laboratory programming assignments, conducted under the guidance of the instructor and undergraduate teaching assistants (described next) during the normal classroom session. While the exact percentages of lecture and activity were not identical across all sections, faculty teaching our gateway courses agreed that the majority of their session time was to be devoted to the activities rather than the lectures.

Undergraduate Teaching Assistants (UTAs). Incorporating undergraduates as teaching assistants in the classroom has been shown to both improve achievement of learning outcomes [38] and promote hands-on learning [39] in undergraduate courses. There are several key advantages to UTAs, compared to graduate teaching assistants (GTAs). First, a UTA will likely have taken the exact course for which they are assisting, at the same institution. Second, a UTA can be more relatable [40] because they are closer in age and have propagated through the same undergraduate program; UTAs are more likely to share similar experiences with the students they are assisting. A "side-benefit" for the instructor is that UTAs can provide an avenue through which an instructor can receive more accurate feedback about a student's well-being in the course [41]. Numerous benefits of UTAs have been documented in computer science [42], including unique perspectives on course material [43], creating better community [44], more detailed tailored feedback [45], and finally improving student participation [46], retention [47], motivation [48], and satisfaction [49]. For our gateway courses (both BP and non-BP) we supplied support at a rate of one UTA (20 hours per week) for every twenty-five students. During our intervention period, gateway courses were the only courses eligible for UTAs and instructors for gateway courses were allowed and encouraged to recruit their own UTAs. Additionally, we set a requirement that a UTA must have passed the course for which they are assisting with a "B" or above. Responsibilities for UTAs included: completing a required training course and orientation, attending weekly meetings the assigned course instructor, leading review and sessions, assisting inside and outside of class (including office hours), facilitating class projects, writing study guides, notes, and practice problems, and proctoring quizzes and exams.

Our study was conducted in three phases:

Phase A. We implemented BP in the CS1 and CS2 courses and compared success rates of students taking BP sections with those taking non-BP sections. Registration was conducted using normal practices at our institution, and students openly registered for CS1 and CS2 sections without *a priori* knowledge of whether the section was BP or non-BP.

Phase B. We tracked students who took BP sections of CS1 and CS2 and compared their subsequent performance in three upper-level computer science (CS) courses further down the prerequisite chain, with the performance of students who took non-BP CS1 and CS2 in the three upper-level classes.

Phase C. After Phase B was complete, we implemented BP in the three upper-level courses, and in the CS1 and CS2 equivalents in our information technology (IT) program – across all modalities: in-person, online (asynchronous and synchronous), and hybrid. We then compared the success rates of students in the BP sections compared to those taking the non-BP sections.[1]

Success particularly in Phase B illustrates the power of pedagogical intervention in computer science gateway courses and the impact it has on student success in upper-level coursework. From the perspective of a computing faculty and/or administrator, success in phases A and C would verify these BP practices as a robust set of strategies that could improve success rates independent of computing "flavor" (CS or IT) or location in the prerequisite chain, with phase B creating a

[1] The CS1 and CS2 course in our IT degree differ somewhat in their outcomes from the CS gateway sequence. The CS1 in IT only covers I/O (particularly file) slightly, much less than what is covered in CS. The CS2 in IT completely omits recursion and introductory data structures, such as linked-lists, stacks and queues, since our IT degree does not require data structures. In contrast, such topics are well covered in the CS2 course in CS. This is consistent with Accreditation Board for Engineering and Technology (ABET) requirements [50], for which our BS degrees in both disciplines have received accreditation.

positive feedback loop for the latter. It should be noted that grades only provide one perspective on academic success [51], and future work should consider other metrics such as sense of belonging [52], engagement [53], career readiness [54], etc., in order to establish a more complete picture.

2 Findings/Results

Our institution has a core prerequisite chain of required computer science courses shown in Figure 1. Our gateway courses (shaded) are CS1 (Programming I), and CS2 (Programming II). BP interventions commenced in CS1 during the Fall 2021 (FA21) semester, and in CS2 during the Spring 2022 (SP22) semester.

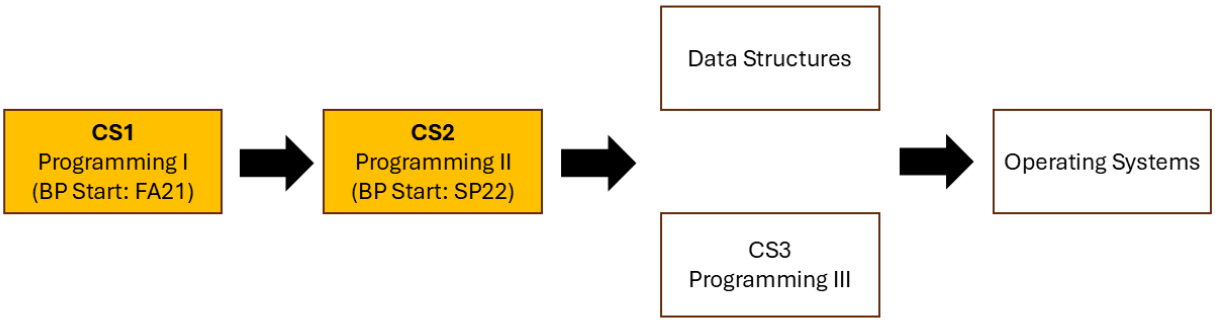

Figure 1. CS required course prerequisite chain.

2.1 Phase A: Impact of BP on Gateway Course Success Rates

For this phase, we track all data for BP and non-BP gateway course sections from the starting period of intervention (CS1 FA21, CS2 SP22) through SP23. Table 1 shows demographics (ethnicity and gender) of this student population. Departmentally during this this period, our undergraduate population consisted of mostly (40.5%) the standard 18-21 age group, followed by 22-24 (26.7%), 25-29 (19.1%), 30-39 (10.7%), 40+ (2.5%) and <18 (0.5%).

	CS1 (1,890 students)	**CS2 (1,757 students)**
Ethnicity		
American Indian/Alaskan Native	1	2
Asian	94	73
Black/African American	227	205
Hispanic	1177	1124
Nonresident Alien	124	129
Unknown	12	13
Pacific Islander	0	1
Two or More Races	41	35
White	214	175

Gender		
Male	1452	1417
Female	438	340

Table 1. CS1 and CS2 demographics from the start of intervention through SP23.

For our purposes throughout this study, we define "success rate" as the passing rate, calculated as the percentage of students who received a grade of "C" or above. Note that in our department, a grade of "C" is the minimum requirement for passing each of these courses and we do not have grades of "C-". Raw passing rates were tracked by section and semester of CS1 and CS2 allowing us to track whether a course section was BP or non-BP. Table 2 shows the improvements.

Gateway Course (Total Students)	**Non-BP Success Rate (%)**	**BP Success Rate (%)**	**Improvement**
CS1 (1,890)	82	89	+7%
CS2 (1,757)	77	88	+11%

Table 2. CS1 and CS2 success rates for students who took non-BP sections, compared to BP sections.

As each of these pedagogical practices incorporated in BP have already been documented as successful for improving success rates in the literature (both for general and computing-specific courses), we would expect the same result here. The results support the conjecture that BP instructors implemented these pedagogical interventions in a correct and effective manner, achieving improvements consistent with expectations.

2.2 Phase B: Impact of BP in Gateway Courses on Success in Upper-Division Courses

Next, we track students who took BP and non-BP sections of our gateway courses and compare their performance in upper-division courses in our core prerequisite chain shown in Figure 1. These upper-division courses include Data Structures, Programming III, and Operating Systems. For CS1, we also track student performance in CS2. Collectively this amounts to seven course pairs consisting of one gateway and upper-division course in our prerequisite chain: CS1+{CS2, Data Structures, Programming III, Operating Systems} and CS2+{Data Structures, Programming III, Operating Systems}. Fully anonymized data was provided by our Office of Accountability and Information Management (AIM), with each record consisting of: a unique hash value for every student (we were not provided the mapping), section and semester of gateway course, section and semester of upper-division course, and grade in the upper-division course (the section and semester of a gateway course is sufficient to know if the section is BP or non-BP).

For each pair, we first note the semester of the intervention in the BP course, and for its corresponding upper-division course, the earliest point of possible impact of these interventions. We then collect data from this earliest point, until the end of SP23. This data is summarized in Table 3. For example, with CS1 – the BP intervention took place in FA21. After CS1, CS2 is next in the prerequisite chain, so impacts in that course could be felt as early as the next semester

(SP22), and we begin data collection then. Data Structures and Programming III are further down the prerequisite chain, so impact there would commence as early as two semesters later (SU22), and for Operating Systems three semesters later (FA22).

Gateway Course (Intervention Time)	Upper-Division Course	Earliest Impact	Data Collected	BP Students	Non-BP Students
CS1 (FA21)	CS2	SP22	SP22-SP23	559	913
	Data Structures	SU22	SU22-SP23	329	442
	Programming III	SU22	SU22-SP23	250	563
	Operating Systems	FA22	FA22-SP23	68	232
CS2 (SP22)	Data Structures	SU22	SU22-SP23	302	773
	Programming III	SU22	SU22-SP23	238	911
	Operating Systems	FA22	FA22-SP23	46	427

Table 3. Summary of each gateway course and intervention semester, the earliest potential impact on each upper-division course, and the number of students enrolled in the upper-division course who took BP and non-BP sections of the gateway course.

Table 4 shows the success rate in each upper-division course in our prerequisite chain in Figure 1, collected from the starting point of potential impact through the SP23 semester.

Gateway Course	Upper-Division Course	Non-BP Pass Rate (%)	BP Pass Rate (%)	Improvement
CS1	CS2	71	83	+12%
	Data Structures	80	91	+11%
	Programming III	66	77	+11%
	Operating Systems	88	96	+8%
CS2	Data Structures	82	92	+10%
	Programming III	63	74	+11%
	Operating Systems	89	96	+7%

Table 4. Gateway courses, and success rates in upper-division CS courses for students who took BP sections of the gateway course, compared to those who did not (non-BP).

Improvements in upper-division pass rates range from 7 to 12% for students who took BP gateway course sections over those who took non-BP sections. For both gateway courses, Operating Systems had the smallest improvements in pass rates at 8% and 7%, although worth noting the non-BP pass rate of Operating Systems was already high (at 88 % and 89%), resulting in less room for improvement.

In addition to simply comparing success rates, we also want to determine if there is a *statistically significant relationship* between upper-division course success based on enrollment in a BP or non-BP section of the gateway courses. For each student record, we assign a binary value for the gateway course (1=BP section, 0=non-BP section). Then for the corresponding upper-division course, we assigned a binary value (1=PASS, 0=FAIL) and perform binary ANOVA analysis to check impact of enrollment in BP Sections on PASS, computing both F-values and p-values, taking

a *p*-value of less than 0.05 as statistically significant. We show these results in Table 5, with statistically significant results denoted with an asterisk (*).

Gateway Course	Upper-Division Course	F-Value	P-Value (*=Significant)
CS1	CS2	26.753	0.0000002*
	Data Structures	19.711	0.00001*
	Programming III	4.264	0.039*
	Operating Systems	1.381	0.241
CS2	Data Structures	16.023	0.00006*
	Programming III	9.846	0.002*
	Operating Systems	1.778	0.183

Table 5. Gateway courses, and ANOVA results indicating the impact of enrollment in a BP section of that course and passing an upper-division CS course (*=statistically significant).

The only non-significant impact (for both gateway courses) was in Operating Systems. The most plausible reason for this is the large difference in sample size between BP and non-BP groups (students had not matriculated far enough in the pipeline), which will tend to increase uncertainty.

To ensure student evaluations are a non-confounding factor and that we are adequately distributing instructors with respect to their student-evaluation data, we also perform ANOVA analysis to check for any dependence between BP section and our "Student Perceptions of Teaching Survey (SPOTS)" scores.[2] We show these results in Table 6. The F- and P-values both indicate no significant connection and affirm that to be the case.

Gateway Course	F-Value	P-Value
CS1	0.182	0.692
CS2	1.112	0.332

Table 6. ANOVA results that assess the impact of gateway course faculty teaching evaluations on assignment to a BP or non-BP section of that same gateway course.

As we collected this data, we realized this also provided an interesting opportunity to assess the impact of gateway BP interventions on success in the upper division, compared to the impact of gateway-course faculty SPOTS score. To this end, we took every record and assigned a value corresponding to the pre-BP SPOTS score of the gateway course instructor, and the same binary value (1=PASS, 0=FAIL) for the upper-division course. We then perform general ANOVA analysis, once again computing F-scores and p-values and adopting the same definition of statistical significance ($p < 0.05$). Table 7 shows these results.

Gateway Course	Upper-Division Course	F-Value	P-Value (*=Significant)
CS1	CS2	0.179	0.672
	Data Structures	0.002	0.962

[2] We are using SPOTS scores to serve as a proxy for quality of teaching, which is likely flawed, as studies have shown that students have biases in how they evaluate student that are related to faculty race, gender and age [55].

	Programming III	0.074	0.786
	Operating Systems	5.484	0.022*
CS2	Data Structures	0.092	0.762
	Programming III	0.041	0.840
	Operating Systems	0.039	0.843

Table 7. Gateway courses, and ANOVA results assessing the impact of instructor evaluations (based on teaching evaluations) of that same gateway course and passing the corresponding upper division course (*=statistically significant).

We note that in almost all cases, gateway course teaching evaluations had no significant impact on success in upper-division CS courses, supporting a theory that these BP gateway course pedagogical interventions are more reliable for building a solid foundation and have a far more significant impact on longitudinal student success than the teaching evaluations of faculty assigned to gateway courses.

Interestingly, the one anomaly again involves Operating Systems – where CS1 instructor evaluations were revealed to have a significant impact on student success in Operating Systems. Being furthest in the chain, the Operating Systems student population had the lowest percentage of BP students (17%, fewer than one in five). We hypothesize that with the majority of the CS1 instructor evaluations being from non-BP sections this increased the significance of these evaluations, "all else being equal". This same significance was not found between CS2 instructor evaluations and Operating Systems success, however.

It should also be noted that during this time period (FA21-SP23), BP was only implemented in face-to-face sections of CS1 and CS2 (due to their higher conductivity with hands-on learning and UTAs), while non-BP sections included both in-person and online sections. While BP in-person sections all implemented hands-on learning and UTAs, non-BP in-person sections may or may not have (nothing was enforced), and did not use UTAs. In reviewing data over the past five years (2020-2025, thousands of students, including time outside the intervention period), students who took either online or in-person sections of CS2 experienced identical passing rates in successor courses. Students who took online CS1 had only a slightly lower (3%) pass rate that those that took in-person sections of CS1. This supports our conjecture that modality is at best a small confounding factor, if at all.

2.3 Phase C: Impact of BP on Success Rates in CS Upper-Division Courses, and IT Gateway Courses

As Phase B data collection terminated in SP23, beginning in FA23, we implemented the BP interventions in the three upper-division courses (Data Structures, Programming III, Operating Systems), in addition to gateway courses in our information technology (IT) program: CS1 (IT), and CS2 (IT), and in all remaining sections of CS1 and CS2. We collected data in the same manner as Phase A above – accounting for all data starting from FA19-FA24, with course section and semester data provided by Accountability dashboards with restricted access. As with Phase A, section and semester were enough to distinguish BP and non-BP sections of these courses.

Table 8 shows these results. Improvements range from 7-25% for the upper-division courses and 6-19% for the IT gateway courses.

Course	Non-BP Success Rate (%)	BP Success Rate (%)	Improvement
Data Structures	86	94	+8%
Programming III	68	93	+25%
Operating Systems	90	97	+7%
CS1 (IT)	73	79	+6%
CS2 (IT)	72	91	+19%

Table 8. BP and non-BP success rates in upper-level CS courses, and IT CS1/CS2.

3 Discussion

Implications. We have learned much from our experience implementing and maintaining these BP interventions. We pursued a low student-to-UTA ratio because, as mentioned in the introduction, computing gateway courses have the most heterogeneous array of student needs to satisfy, thus increased support increases this capacity. Use of UTAs will generally be far less expensive compared to graduate TAs (about one-third the cost) – but particularly for large class sizes (which gateway courses often are), a reliable pipeline of student assistants is ideal to avoid instructor overhead in recruiting multiple UTAs every semester which then must be screened and background checked for meeting departmental and institutional requirements. We have recently begun incorporating UTA evaluations, so that every semester when we assign UTAs to gateway courses, we have a knowledge base containing which UTAs worked well for which gateway course sections and faculty members. On our immediate horizon is construction of a Student Assistant Pipeline (SAP), with experienced UTAs for specific course sections and faculty training new hires every semester, further reducing the burden on faculty by saving them the challenge of recruiting their own UTAs.

For those looking to implement BP interventions in their computing curricula, we recommend a similar "phase-in" process, where these practices are integrated slowly, one course at a time, with one set of faculty at a time. This approach also afforded us the opportunity to study the intervention and its impact. Reconciling concepts such as "synchronized content" and "common assessment" with "academic freedom" [56] can be challenging (we experienced this in some cases as well), particularly when the benefits for the students are not immediately clear - as will be the case by definition when the practices are first integrated and results are not yet available. In our case, our interventions in FA21 began with just two CS1 faculty who were willing to attempt and buy into the intervention. It may also be necessary to supply such faculty with additional compensation, as course refactoring and integration of practices like BP will require additional time and likely periodic meetings to coordinate content across sections. It will also be necessary to implement procedures for integrating new faculty into pedagogically modified sections. We accomplished this through clear delineation of faculty teams teaching each BP course every semester, eventually culminating in the formation of a Programming Gateway Committee (PGC).

Limitations. Incorporating BP into online course sections had its share of challenges. Although the first two BP interventions (synchronized content and common assessment) remained

unaffected, the latter two (hands-on learning, facilitated by UTAs) were far more challenging. Hands-on learning in online course sections is of course a well-known challenge [57]. During the BP intervention period, we upheld these interventions as much as possible through faculty and UTA office hours (both physical and virtual) and limited synchronous meeting times (our institution limits this to 20% for online asynchronous courses), but one inescapable reality became the optionality from the student's perspective of taking advantage of these interventions, compared to in-person sections. As more faculty become certified to teach in a hybrid (50% in-person, 50% remote) environment, we continue to migrate more of our online gateway courses to that modality. However, we continue to pursue solutions to optimize the impact of these interventions in online asynchronous gateway courses, because like many institutions, we have three online degrees, which must by definition accommodate students independent of location and time zone.

Additionally, recommended best practices regarding teaching assistants will require some level of investment in TA training [58]. Particularly in the case of UTAs in large, active learning environments - an elaborate, multi-session training program is ideal [59], involving well-organized sessions where students are reviewing content, practicing asking questions, discussing processing skills, writing journal reflections, leading study sessions, etc. In our case, we encapsulated the above into an online course customized to the needs of our curricula, but this still required both hosting fees, and a dedicated faculty member to manage hundreds of UTAs across all of the BP sections every year, with applicable compensation and course releases. Such an investment will either need to be consistently sustained throughout implementation of BP, or a viable alternative will need to be discovered.

Finally as mentioned earlier, metrics other than success rates based on grade should be considered for a more comprehensive analysis. One in particular we would like to pursue is career readiness – i.e., the degree to which students are prepared for success in their post-graduate work (i.e. industry or academia). Establishment of a solid infrastructure for alumni tracking, ideally including a database and web interface along with some networking incentives for participation would essentially enable us to continue tracking the impact of interventions such as BP beyond upper-level coursework and into the professional realm, completing the feedback loop.

Conclusion. Returning to our original objectives, our results support the opportunity for pedagogical interventions in computer science gateway courses to improve success rates in upper-division courses, even more reliably than gateway course instructor evaluations (as measured by our student evaluation data). Phases A and C showed across-the-board improvements in success rates the CS gateway and upper-division levels and IT courses as well, suggesting these BP practices as powerful interventions that not only improve success rates in a wide range of gateway and upper-division computing courses, but also create a positive feedback loop between intervention in the gateway and success in the upper-division courses (Phase B).

References

[1] X. Zhao, S. Chowdhury, and T. Chowdhury, "Integrating Evidence-based Learning in Engineering and Computer Science Gateway Courses," in *2020 ASEE Virtual Annual Conference Content Access Proceedings*, Virtual On line: ASEE Conferences, Jun. 2020, p. 34842. doi: 10.18260/1-2--34842.

[2] A. K. Koch, *Improving teaching, learning, equity, and success in gateway courses*, vol. 180. in New Directions for Higher Education, vol. 180. San Francisco: Jossey-Bass.
[3] P. Kinnunen and L. Malmi, "Why students drop out CS1 course?," in *Proceedings of the second international workshop on Computing education research*, Canterbury United Kingdom: ACM, Sep. 2006, pp. 97–108. doi: 10.1145/1151588.1151604.
[4] B. Pioro, "Performance in an Introductory Computer Programming Course as a Predictor of Future Success for Engineering and Computer Science Majors," presented at the International Conference on Engineering Education, Gainesville, FL, 2004.
[5] L. Beck, P. Kraft, and A. W. Chizhik, "Predicting Student Success in CS2: A Study of CS1 Exam Questions," in *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education*, Providence RI USA: ACM, Feb. 2022, pp. 140–146. doi: 10.1145/3478431.3499276.
[6] M. O. Hegazi and M. Alhawarat, "The Challenges and the Opportunities of Teaching the Introductory Computer Programming Course: Case Study," in *2015 Fifth International Conference on e-Learning (econf)*, Manama, Bahrain: IEEE, Oct. 2015, pp. 324–330. doi: 10.1109/ECONF.2015.61.
[7] J. Bennedsen and M. E. Caspersen, "Failure rates in introductory programming," *SIGCSE Bull.*, vol. 39, no. 2, pp. 32–36, Jun. 2007, doi: 10.1145/1272848.1272879.
[8] R. Mason, Simon, B. A. Becker, T. Crick, and J. H. Davenport, "A Global Survey of Introductory Programming Courses," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, Portland OR USA: ACM, Mar. 2024, pp. 799–805. doi: 10.1145/3626252.3630761.
[9] I. Groher, M. Vierhauser, B. Sabitzer, L. Kuka, A. Hofer, and D. Muster, "Exploring diversity in introductory programming classes: an experience report," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Software Engineering Education and Training*, Pittsburgh Pennsylvania: ACM, May 2022, pp. 102–112. doi: 10.1145/3510456.3514155.
[10] Y. Benkarroum and M. Q. Azhar, "Infusing Computational Thinking into a Computer Science Gateway Course," *Journ Comp Sci Colleges*, vol. 39, no. 8, pp. 143–157, 2024.
[11] D. Davis, "Combining Active Learning Approaches for Improving Computing Course Outcomes at Minority-Majority Institutions," presented at the 2017 ASEE Annual Conference & Exposition, ASEE, 2017.
[12] U. Omer, M. S. Farooq, and A. Abid, "Introductory programming course: review and future implications," *PeerJ Computer Science*, vol. 7, p. e647, Jul. 2021, doi: 10.7717/peerj-cs.647.
[13] A. Salguero, J. McAuley, B. Simon, and L. Porter, "A Longitudinal Evaluation of a Best Practices CS1," in *Proceedings of the 2020 ACM Conference on International Computing Education Research*, Virtual Event New Zealand: ACM, Aug. 2020, pp. 182–193. doi: 10.1145/3372782.3406274.
[14] H. Bisgin, M. Mani, and S. Uludag, "Delineating Factors that Influence Student Performance in a Data Structures Course," in *2018 IEEE Frontiers in Education Conference (FIE)*, San Jose, CA, USA: IEEE, Oct. 2018, pp. 1–9. doi: 10.1109/FIE.2018.8659300.
[15] D. E. Sondakh, S.Kom, M.T, Ph.D and S. R. Pungus, "Can Early Programming Performance Predict Computer Science Students' Success?," *CogITo Smart Journal*, vol. 8, no. 2, pp. 574–584, Dec. 2022, doi: 10.31154/cogito.v8i2.452.574-584.

[16] A. Adeyemi, C. Grant, and K. Sebastian, “Course Coordination: A Necessary Requirement for Consistency,” *PRIMUS*, vol. 33, no. 4, pp. 378–401, Apr. 2023, doi: 10.1080/10511970.2022.2073414.
[17] W. M. Smith, M. Voigt, A. Ström, D. C. Webb, and W. G. Martin, Eds., *Transformational change efforts: student engagement in mathematics through an institutional network for active learning*. in AMS ebook collection. Providence, Rhode Island: American Mathematical Society, 2021. doi: 10.1090/mbk/138.
[18] C. E. Brodley and C. Gill, “The BPC Relevance of Common Assessment in the Introductory Sequence,” *Commun. ACM*, vol. 67, no. 7, pp. 24–26, Jul. 2024, doi: 10.1145/3637891.
[19] J. Piaget, *The origins of intelligence in children.* New York: W W Norton & Co, 1952. doi: 10.1037/11494-000.
[20] D. A. Kolb, *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, N.J: Prentice-Hall, 1984.
[21] M. Schwichow, C. Zimmerman, S. Croker, and H. Härtig, “What students learn from hands-on activities,” *J Res Sci Teach*, vol. 53, no. 7, pp. 980–1002, Sep. 2016, doi: 10.1002/tea.21320.
[22] N. Holstermann, D. Grube, and S. Bögeholz, “Hands-on Activities and Their Influence on Students’ Interest,” *Res Sci Educ*, vol. 40, no. 5, pp. 743–757, Nov. 2010, doi: 10.1007/s11165-009-9142-0.
[23] N. Yannier *et al.*, “Active learning: ‘Hands-on’ meets ‘minds-on,’” *Science*, vol. 374, no. 6563, pp. 26–30, Oct. 2021, doi: 10.1126/science.abj9957.
[24] L. D. Cridlin, “The importance of hands-on learning,” in *International Laser Safety Conference*, San Francisco, California, USA: Laser Institute of America, 2007, pp. 151–156. doi: 10.2351/1.5056625.
[25] B. Van Poppel and S. Reed, “Achieving Course Objectives: The Benefits Of A Hands On Design Competition,” in *2003 Annual Conference Proceedings*, Nashville, Tennessee: ASEE Conferences, Jun. 2003, p. 8.157.1-8.157.14. doi: 10.18260/1-2--12138.
[26] G. I. Holwell, A. Mech, H. Parzer, and A. F. Probert, “Teaching entomology online: Challenges, benefits and examples of effective hands-on activities,” *Austral Entomology*, vol. 64, no. 2, p. e70007, May 2025, doi: 10.1111/aen.70007.
[27] L. Carlson and J. Sullivan, “Hands-on Engineering: Learning by Doing in the Integrated Teaching and Learning Program,” *J Engng Ed*, vol. 15, no. 1, pp. 20–31, 1999.
[28] L. B. Flick, “The meanings of hands-on science,” *Journal of Science Teacher Education*, vol. 4, no. 1, pp. 1–8, Mar. 1993, doi: 10.1007/BF02628851.
[29] M. Suchowski, F. Severance, and D. Miller, “Benefits Of A Hands On Introduction To Electrical And Computer Engineering,” in *2003 Annual Conference Proceedings*, Nashville, Tennessee: ASEE Conferences, Jun. 2003, p. 8.264.1-8.264.10. doi: 10.18260/1-2--12647.
[30] A. Konak, T. K. Clark, and M. Nasereddin, “Using Kolb’s Experiential Learning Cycle to improve student learning in virtual computer laboratories,” *Computers & Education*, vol. 72, pp. 11–22, Mar. 2014, doi: 10.1016/j.compedu.2013.10.013.
[31] V. Handur *et al.*, “Integrating Class and Laboratory with Hands-On Programming: Its Benefits and Challenges,” in *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education (MITE)*, Madurai, India: IEEE, Dec. 2016, pp. 163–168. doi: 10.1109/MITE.2016.041.

[32] E. J. Theobald *et al.*, “Active learning narrows achievement gaps for underrepresented students in undergraduate science, technology, engineering, and math,” *Proc. Natl. Acad. Sci. U.S.A.*, vol. 117, no. 12, pp. 6476–6483, Mar. 2020, doi: 10.1073/pnas.1916903117.
[33] M. Ingaldi and S. T. Dziuba, “CHARAKTERYSTYKA GOSPODARSTW DOMOWYCH Z EKOLOGICZNEGO PUNKTU WIDZENIA,” *QPI*, vol. 08, pp. 52–65, Jul. 2018, doi: 10.30657/qpi.2018.08.05.
[34] T. M. Maus, “Simulation: The Importance of ‘Hands-On’ Learning,” *Journal of Cardiothoracic and Vascular Anesthesia*, vol. 25, no. 2, pp. 209–211, Apr. 2011, doi: 10.1053/j.jvca.2011.01.007.
[35] Y. El-Glaly, W. Shi, S. Malachowsky, Q. Yu, and D. E. Krutz, “Presenting and evaluating the impact of experiential learning in computing accessibility education,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training*, Seoul South Korea: ACM, Jun. 2020, pp. 49–60. doi: 10.1145/3377814.3381710.
[36] L. Wunderlich, A. Higgins, and Y. Lichtenstein, “Machine Learning for Business Students: An Experiential Learning Approach,” in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, Virtual Event Germany: ACM, Jun. 2021, pp. 512–518. doi: 10.1145/3430665.3456326.
[37] Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, M. K. K. Devi, M. S. Thendral, and Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, “Using Kolb’s Experiential Learning Theory to Improve Student Learning in Theory Course,” *JEET*, vol. 37, no. 1, pp. 70–81, Apr. 2023, doi: 10.16920/jeet/2023/v37i1/23133.
[38] D. A. B. Weikle and J. Meinke, ed., “More insights on a peer tutoring model for small schools with limited funding and resources,” *Journal of Computing Sciences in Colleges*, vol. 31, no. 3, pp. 101–109.
[39] L. Fingerson and A. B. Culley, “Collaborators in Teaching and Learning: Undergraduate Teaching Assistants in the Classroom,” *Teaching Sociology*, vol. 29, no. 3, p. 299, Jul. 2001, doi: 10.2307/1319189.
[40] A. Van Dam, “Reflections on an introductory CS course, CS15, at Brown University,” *ACM Inroads*, vol. 9, no. 4, pp. 58–62, Nov. 2018, doi: 10.1145/3284639.
[41] A. Decker, P. Ventura, and C. Egert, “Through the looking glass: reflections on using undergraduate teaching assistants in CS1,” in *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, Houston Texas USA: ACM, Mar. 2006, pp. 46–50. doi: 10.1145/1121341.1121358.
[42] D. Mirza, P. T. Conrad, C. Lloyd, Z. Matni, and A. Gatin, “Undergraduate Teaching Assistants in Computer Science: A Systematic Literature Review,” in *Proceedings of the 2019 ACM Conference on International Computing Education Research*, Toronto ON Canada: ACM, Jul. 2019, pp. 31–40. doi: 10.1145/3291279.3339422.
[43] S. A. Blanco, “Active Learning in a Discrete Mathematics Class,” in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Baltimore Maryland USA: ACM, Feb. 2018, pp. 828–833. doi: 10.1145/3159450.3159604.
[44] M. Minnes, C. Alvarado, and L. Porter, “Lightweight Techniques to Support Students in Large Classes,” in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, Baltimore Maryland USA: ACM, Feb. 2018, pp. 122–127. doi: 10.1145/3159450.3159601.

[45] P. E. Dickson, T. Dragon, and A. Lee, “Using Undergraduate Teaching Assistants in Small Classes,” in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle Washington USA: ACM, Mar. 2017, pp. 165–170. doi: 10.1145/3017680.3017725.
[46] M. J. Canup and R. L. Shackelford, “Using software to solve problems in large computing courses,” in *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*, Atlanta Georgia USA: ACM, Mar. 1998, pp. 135–139. doi: 10.1145/273133.273178.
[47] I. Pivkina, “Peer learning assistants in undergraduate computer science courses,” in *2016 IEEE Frontiers in Education Conference (FIE)*, Erie, PA, USA: IEEE, Oct. 2016, pp. 1–4. doi: 10.1109/FIE.2016.7757658.
[48] P. E. Dickson, “Using undergraduate teaching assistants in a small college environment,” in *Proceedings of the 42nd ACM technical symposium on Computer science education*, Dallas TX USA: ACM, Mar. 2011, pp. 75–80. doi: 10.1145/1953163.1953187.
[49] R. Brent, J. Maners, D. Raubenheimer, and A. Craig, “Preparing undergraduates to teach computer applications to engineering freshmen,” in *2007 37th annual frontiers in education conference - global engineering: knowledge without borders, opportunities without passports*, Milwaukee, WI, USA: IEEE, Oct. 2007, pp. F1J-19-F1J-22. doi: 10.1109/FIE.2007.4418053.
[50] “Criteria for Accrediting Computing Programs: Effective for Reviews During the 2025-2026 Accreditation Cycle.” ABET, 2025.
[51] M. Cachia, S. Lynam, and R. Stock, “Academic success: Is it just about the grades?,” *Higher Education Pedagogies*, vol. 3, no. 1, pp. 434–439, Jan. 2018, doi: 10.1080/23752696.2018.1462096.
[52] T. L. Strayhorn, *College Students’ Sense of Belonging*, 2nd ed. Routledge, 2018. doi: 10.4324/9781315297293.
[53] E. R. Kahu and K. Nelson, “Student engagement in the educational interface: understanding the mechanisms of student success,” *Higher Education Research & Development*, vol. 37, no. 1, pp. 58–71, Jan. 2018, doi: 10.1080/07294360.2017.1344197.
[54] W. Camara, “Defining and Measuring College and Career Readiness: A Validation Framework,” *Educational Measurement*, vol. 32, no. 4, pp. 16–27, Dec. 2013, doi: 10.1111/emip.12016.
[55] Y. Fan *et al.*, “Gender and cultural bias in student evaluations: Why representation matters,” *PLoS ONE*, vol. 14, no. 2, p. e0209749, Feb. 2019, doi: 10.1371/journal.pone.0209749.
[56] C. Russell, *Academic Freedom*, 0 ed. Routledge, 2002. doi: 10.4324/9780203003633.
[57] O. Alowaimer, “Accounting Distance Learning: An Analysis of the Benefits and Challenges,” *Acces la Success*, vol. 26, no. 104, pp. 113–123, 2025.
[58] M. Broeckelman-Post and K. Ruiz-Mesa, “Best Practices for Training New Communication Graduate Teaching Assistants,” *JCP*, vol. 1, no. 1, pp. 93–100, Jun. 2018, doi: 10.31446/JCP.2018.16.
[59] S. M. Ruder and C. Stanford, “Strategies for Training Undergraduate Teaching Assistants To Facilitate Large Active-Learning Classrooms,” *J. Chem. Educ.*, vol. 95, no. 12, pp. 2126–2133, Dec. 2018, doi: 10.1021/acs.jchemed.8b00167.