6-2021

Rapid Entry into Masters in Computing Program for Non-Majors

Gary S. Krenz
Marquette University, gary.krenz@marquette.edu

Thomas Kaczmarek
Marquette University

John C. Moyer
Marquette University, john.moyer@marquette.edu

ITiCSE '21 : Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education, Vol. 1 (June 2021): 505-511. DOI. This article is © Association for Computing Machinery and permission has been granted for this version to appear in e-Publications@Marquette. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

Rapid Entry into Masters in Computing Program for Non-Majors

Gary Krenz
Marquette University, Milwaukee, WI
Thomas Kaczmarek
Marquette University, Milwaukee, WI
John F. Moyer
Marquette University, Milwaukee, WI

Abstract

The (program name removed for dual-anonymous review process) graduate curriculum initiative strives to provide a rapid entry pathway to a professional Master of Science (MS) degree for individuals who do not have an undergraduate degree in computing, but who wish to cross over to a career in the computing field. The goal of our curriculum is to minimize the time students spend preparing for graduate study and maximize experiences relevant for work after graduation. The (acronym removed) curriculum initiative is similar in concept to other post-baccalaureate conversion programs. However, customization of the (acronym removed) bridge course and curriculum pathway makes it possible for conversion students to complete the bridge course in summer then move directly to standard graduate program courses in fall. The highly focused bridge course includes learning to program in two popular high-level languages, abstract data structures, professional practices and various computing concepts that prepare students for the rigors of the MS program. To recruit low-income students from populations under-represented in the computing field, federal funding and institutional support was obtained. The funding provided low-income students with financial support, enabling them to complete the degree in two years. With projected computing talent shortfall and the advantages of workforce diversity, it is imperative that educational institutions create conversion programs that can be completed relatively quickly, with the needs of non-traditional students in mind.

CCS Concepts

Social and professional topics → *Model curricula*.

Keywords

bridge course, computing education, conversion program, curriculum initiative, master's degree, non-traditional students, underrepresented populations

1 Motivating Context

Since its inception in 1999, our Master of Science (MS) in Computing has offered a career-change opportunity for individuals with undergraduate degrees with little or no formal computing education. However, like United States (US) graduate computing programs at the time, MS admission required various Computer Science (CS) prerequisites; our minimum prerequisites a semester-long course each in programming and data structures. However, for career-changing individuals with little formal computing education, these prerequisites were typically undertaken in traditional undergraduate courses over two or more semesters. We believed we could shorten the study time frame for career-change individuals by a rapid entry pathway; a pathway providing prerequisites relevant for success in a professional MS program while also introducing and practicing skills used by successful computing professionals.

Also in the late 1990's, universities within the United Kingdom (UK) began post-baccalaureate programs to train graduates for careers in new subject areas, *e.g.*, non-CS majors trained in software engineering [3]. In the UK and several other countries (*e.g.*, New Zealand), these career-change academic programs are referred to as *conversion degrees* or *conversion courses*. Similar US-based programs, such as the multi-campus Align program [1], typically provide several semesters of coursework as part of their bridge to an MS program [16]. While there is a growing handful of US graduate programs that provide an *accelerated* entry pathway for individuals whose baccalaureate study contained little or no formal computing education, *e.g.*, [2, 4, 14, 17], the number of US-based graduate conversion courses/programs and delivery options lag behind those of international colleagues. This article discusses our experience providing non-CS majors rapid entry into a professional MS in Computing program. For overviews of UK conversion programs in artificial intelligence, data science, software engineering and other computing offerings, see [11, 12, 15].

While computing conversion programs typically target a career-change focused adult population under-served by higher education institutions, there are other under-served populations. Taulbee Sur-vey data have revealed ongoing gender and racial disparities in North America computing program enrollments and degree production [18]. These disparities have generated ongoing calls for the computing profession to fulfill its moral and economic obligations by seeking out and encouraging individuals from underrepresented populations to become a significant part of the profession [9]. Mellors-Bourne and Williams note similar issues within the UK, highlighting concerns regarding gender diversity and access for individuals from less advantaged socio-economic backgrounds [12]. A post-baccalaureate rapid bridge to MS provides a means to increase the total flow of individuals into computing-related sectors. For individuals who originally did not complete, or did not see themselves as candidates for, an undergraduate computing degree [9], graduate conversion programs could provide these individuals with a pathway into the computing profession [1, 7, 10].

2. Rapid Entry Structure

2.1 Bridge Course

For the first three years of offering (program name suppressed during review), our rapid entry pathway began with an 11 week/5 days a week/8 hours a day, in-person summer bridge course. Typically, the first four hours of each class was instruction, while the last 4 hours students worked on exercises or a team project.

Former and prospective students indicated attending an all-day, in-person class while also working to support themselves and/or a family was a significant challenge. To address this feedback, we planned synchronous online delivery of the summer 2020 course, Monday through Thursday, 5-9 PM, over 14 weeks; which would be recorded in case a student was unable to attend a session. The revised delivery would also allow the course to be offered in a similar manner during a standard university fall or spring semester. We also planned three two-day [Friday/Saturday] in-person meetings at weeks 1, 7 and 14 in order to facilitate building a sense of community. Due to covid-19 assembly restrictions, the Friday/Saturday sessions were pivoted to online meetings.

Bridge course design incorporated regional employer input and our past experience with career-change individuals. In both in-person and online format, course delivery employed traditional lecture, active-learning, student presentations, independent study, and a team project. Foundational topics, such as abstract data types, were intermixed with the study of software development and common computing architectures. The approach is similar to aspects found in several of the University of Pennsylvania's MCIT Online core courses [5]. Bridge course content, exercises, team work and student presentations strive to develop prerequisite CS knowledge, employer-desired soft skills, as well as begin to move students towards the MS degree program's learning outcomes.

2.1.1 Programming.

Assuming students have no previous programming experience, the course begins with an introduction to Python 3 and the IDLE IDE via an application-focused approach. Students are introduced to Python expressions, variables, assignments, primitive data types, objects and classes, data abstraction, data encapsulation, control structures, user defined functions, errors and exceptions, programming patterns, containers, namespaces, overloaded operators, recursion, modular design, *etc*. The application-focused approach has students completing various exercises, *e.g.*, integrating a program with the underlying OS, using a simple GUI toolkit, accessing content on the web, and accessing databases. Access to OS resources (files and other services) introduces APIs, buffering and other concepts. Exercises creating application GUI interfaces introduces cooperating processes and interprocess communication. The section on the web introduces scraping, concepts of formal language description as used in computer languages, as well as a discussion of standard web architecture and the exchange of information between browsers and applications. The database (DB) excursion introduces data organization (schema), modeling, and the predominant long-term storage mechanisms used by enterprises. As part of exploring web-based user interface design via the Flask tutorial, Flaskr [6], the summer 2020 students were introduced to PyDev in Eclipse before the study of abstract data types using Java. Although software version control is critically important, with concepts, content and systems already in the course, we felt there was insufficient time to introduce students to a version control system. Students were strongly encouraged to investigate version control systems, *e.g.*, Git, in the term following the course.

2.1.2 Professional Practice.

After several weeks of introductory programming, students conduct a self-study of professional programming practices; including debugging, testing, and project coordination. Other self-study topics include HTML/CSS and SQL. During the week 7 two-day session, students give presentations on their self-study. The self-study, and presentation guidance and feedback, aid students in beginning to develop the skills necessary for life-long learning and expressing technical concepts to others.

2.1.3 SQL.

One of the course's last Python programming applications includes rudimentary exercises using SQL to create and query DBs. Students undertake additional SQL self-study for individually assigned SQL presentations, as well as to prepare them for understanding SQL aspects of the team project.

2.1.4 OS Commands and Scripting.

The SANS Institute provides free *SANS Cyber Aces Online Tutorials* [8]. Students are introduced to virtual machines, installing Windows and Linux within VMs, file systems, user access control, the Open System Interconnect model, and scripting with Bash and PowerShell. To foster the skills that enable professionals to learn on their own, we do not provide explicit direction regarding Cyber Aces study, but offer assistance as students engage the Cyber Aces content. Students quickly encounter unfamiliar technology, *e.g.*, VMs and command line interfaces, they must comprehend in order to complete the tutorials.

2.1.5 Data Structures and Algorithms.

Algorithm analysis and computational complexity are considered when examining data structures, abstract data types, object-oriented design, algorithms and alternative implementations. This portion of the course also introduces the Java programming language. Java was chosen for its support of abstraction and to introduce students to another commonly used programming language. We review programming concepts, introduce data structures, and compare choices made by Python and Java designers. We cover common abstractions, such as queues, stacks, and maps. Programming exercises continue development of algorithmic thinking while emphasizing abstraction, encapsulation, implementation, application and the impact of design decisions.

2.1.6 Team Project.

We use development of a web-based blackjack game as a final course project. Students build on an object-oriented representation of a blackjack game in Python. They are asked to implement players and win/loss records DBs and to use a modern user interface toolkit (Flask) to support interaction with the application. Students are encouraged to divide the work and interact with one another to build an integrated solution. A distributed architecture is introduced involving the typical web-based application consisting of browser-based user interface, web server, application server, and DB. In the DB portion of the project, students move from using SQLite locally to MySQL on a remote server. The user interface and application server is proposed as a modification of the Flaskr application, which includes ideas of application testing and packaging applications for broader use. The student team eventually hosts the application on one VM interacting with a remote MySQL on a second VM. The project emphasizes using APIs, simple web architectures, abstractions, and project management/teamwork. The course ends with a final blackjack game live demonstration and presentations regarding design choices. Students gain experience preparing demonstrations, and learn it is common to encounter problems during a live client presentation. Technical presentation guidance, and other communication skills, are explicitly addressed during the course, and is part of the final project critique.

2.1.7 Computing Culture.

Since the course meets interactively over 11 weeks (in-person delivery) or 14 weeks (online delivery), there are many opportunities to question and explicitly discuss ideas and approaches that are part of the culture of computing as seen from the inside of the business. While students have end-user experience with computers, cellphones, laptops and other computing services and systems before they arrive, the course exposes them to some of the practices of building these services. They experience how to sort out the rough edges of computing technology that professionals are asked to master and the ecosystem that supports modern computing.

2.2 Follow-up Term

The single 7 credit bridge course provides conversion students with initial coding skills and a knowledge base they continue to develop during their MS program. In the term following the bridge course, conversion students mix with other students in the MS program. Conversion students are advised to take three regular MS program courses to continue their introduction to the computing field, as well as provide them with skills that could be discussed at potential internship interviews: 1) Principles of Database Systems – examines standard database concepts and architecture, data modeling, formal query languages such as relational algebra, commercial query language SQL, database access from application programs and a brief examination of advanced concepts including transactions, distributed databases, security and XML. 2) Elements of Software Development – explores software design and development processes through a term project in which students are exposed to requirements gathering and analysis, mapping requirements to a design, sound coding and documentation practices, configuration management, testing and quality assurance, system deployment and maintenance. 3) Professional Seminar in Computing – a topic chosen each term from among issues important to professionals in computing. Bridge course exercises and blackjack project highlight professional practices, programming in two common languages, patterns, object oriented design, APIs, data structures, DB/persistent storage, web technologies, tiered architecture, client/server concepts, SQL, VM, documenting design decisions, and other concepts. The bridge course prepares conversion students for the DB course, in particular, for implementing schema and project(s). Likewise, the bridge course prepares students for the software development course, where students perform requirements gathering and execute agile deployment of an instructor approved software project.

2.3 The Professional MS Program

The MS in Computing program requires students to have least 12 credit hours related to a primary career focus, at least six credit hours related to a secondary career focus, an overall credit total and maintain a minimum GPA. Currently, common student foci are cybersecurity, data analytics, software engineering, and application architectures. Two career focus areas have been formalized as MS program specializations, whose required coursework constitutes about one half of the required credits for graduation. The MS program specializations are in cybersecurity and big data/data analytics. Our conversion program is a third specialization that requires students to take the bridge course plus numerous electives, depending on student career goals. The MS in Computing program offers a coursework-only option (36 semester hour credits), a thesis option (30 semester hour credits) and the conversion program option (42 semester hour credits). Students not pursuing the thesis option may do a practicum or professional project as part of their academic plan. Anecdotally, we have observed employer hesitation when considering individuals with non-traditional educational pathways. Conversion students are encouraged to explore internships to add computing work experience to their resume.

3. Experience Report

3.1 Student Data

While our data set is small, it provides useful insights. Our conversion program began in summer 2017 with an 8 student cohort. Five members of the first cohort graduated with an MS in Computing, two members left the degree program, and although we supplied federal and institutional financial aid covering much of the tuition, one member has temporarily suspended study in order to earn a means to pay for the remainder of their program. The five completions are employed as systems administrator, risk engineer, two are data analysts and one decided they loved their original career and returned to teaching. The two who left the program are employed in computing, one as a web designer, the other as a desktop field service technician. All 3 individuals in our 2018 summer cohort completed their MS programs in two years. Unfortunately, students in 2018 cohort encountered a covid-19 decimated job market when they began their job search in mid-2020; however, one has accepted a data analytics position with a Fortune 500 company, another will begin IT rotations with a leading Fortune 100 automaker. The 2019 and 2020 summer cohorts were each comprised of 6 individuals.

Of the 23 students students who attempted the bridge course, 22 completed the course. The self-reported student demographics are: Undergraduate degrees: accounting (2), administration of justice/law enforcement (3), biochemistry (1), business administration (2), criminology (2), communication (1), English (1), geography (1), information science and technology (1), liberal arts/philosophy (2), history and Spanish (1), math (1), math and Spanish education (1), psychology (2), sociology (1), urban planning (1). Throughout, parentheses indicate the number of reporting students, e.g., two students had accounting degrees. Gender: male (15), female (8). Ethnicity: Hispanic or Latino (4), not Hispanic or Latino (18), one did not provide a response. Race: American Indian or Alaska Native (2), Asian (4), Black or African American (7), White (12); one did not wish to provide. Total more than 23; three individuals reported biracial. Previous computing study/experience: None (3); some self-study (18); related coursework (9); certification (2); worked in field (5; 4 at Help Desks, 1 non-technical IT associate).

Twenty-one of the 23 students taking the bridge course were eventually deemed eligible for US federal scholarships by our university's Office of Financial Aid; thus over 90% of the conversion students taking the bridge course were deemed low-income.

At the start of the bridge course, students reported the hours per week they intended to work a job while pursuing full-time study: Full-time (7), 20 or more, but not F/T (6), 10 (3), 0 (7).

3.2 Instructor Survey

In May 2019, after course grades were posted, we surveyed 7 consenting instructors who had taught one or more conversion students during the spring 2019 term. There were 21 enrollments by 8 conversion students in 9 classes. At that time, the first conversion student cohort was at or near graduation, having completed the bridge course and 5 terms in the MS program. The second cohort was approximately half way to graduation, having completed the bridge course and 2 additional terms. The purpose of the survey was to help determine which aspects of the conversion students' knowledge, achievements, and interactions were sufficient for coursework in the MS program and which aspects were deficient. The survey was to be repeated in May 2020, however informed consent and data collection were hampered by challenges arising from the pandemic.

The survey's first question asked whether or not the instructors agreed that their conversion students were able to demonstrate sufficient background knowledge in each of 25 CS topics needed for their course. The instructors were asked to choose "Not necessary" only if knowledge of the topic was not necessary for their course.

The second question asked instructors how strongly they agree that their conversion students are achieving the MS program learning outcomes.

The third question asked instructors to estimate the frequency with which their conversion students interacted with the instructor and with other students both in and out of class.

The fourth question asked for suggestions to improve the conversion program.

3.2.1 Instructor Survey Results.

Table 1 lists 25 topics that, in 2019, were indicated as components of New York University Tandon School of Engineering's CS Bridge Program. The NYU Tandon CS Bridge Program is wholly online and we had begun to consider moving at least part of our bridge course to an online format back in early 2019. One way to facilitate communication and sharing with NYU Tandon was to establish a common framework for making program comparisons and collaborations. The NYU Tandon's CS online bridge has since expanded their bridge course topics list [13].

The list in Table 1 is separated into two parts, the first 12 topics are covered in our bridge course; the remaining 13 are not. Each part is sorted by descending level of instructor agreement. Of the topics covered in the bridge course, the highest level of instructor agreement that bridge course students had sufficient background knowledge was "File processing" at 100%, while the lowest level was "Algorithm analysis introduction" at 29%. After observing these results, we placed additional emphasis within the bridge course on the seven topics that received agreement scores below 60%. The bridge course students examine the majority of these seven topics while they are coming to grips with implementation details, project management issues, and software integration associated with development of their blackjack application. In bridge course exit surveys, several students remarked they wished they would have spent more time on these topics.

Table 1: CS Topics: Instructor's agreement that conversion students demonstrated sufficient background knowledge. Not: number of instructors who felt topic was not necessary, Nec: number who felt topic was necessary, A: number who agreed topic was necessary and that conversion students demonstrated sufficient background knowledge, N: number who agreed topic was necessary and neutral on sufficient background knowledge, D: number who agreed topic was necessary and disagreed on sufficient background knowledge, %A: 100 × A/Nec.

Topics in Bridge Course	Not	Nec	A	N	D	%A
File processing	2	7	7	0	0	100
Searching & sorting	1	8	7	1	0	88
Recursion	4	5	4	1	0	80
Data structures	1	8	6	1	1	75
Iterative statements	2	7	5	1	1	71
Object oriented concepts	2	7	4	3	0	57
Data types & expressions	2	7	4	2	1	57
Fcns, abstraction & stack	3	6	3	2	1	50
Linked lists	4	5	2	3	0	40
Pointers & memory alloc	4	5	2	1	2	40
Trees and search trees	3	6	2	2	2	33
Algorithm analysis intro	2	7	2	3	2	29
Topics not in Bridge Course						
System hardware	4	5	5	0	0	100
Intro to OS concepts	4	5	5	0	0	100
Compile & execute	3	6	5	1	0	83
Number systems	3	6	4	2	0	67
Assembly language basics	6	3	2	1	0	67
Processes & threads	3	6	3	2	1	50
Sets basics	4	5	2	3	0	40
Predicate logic	3	6	2	4	0	33
Threads & deadlocks	6	3	1	1	1	33

Memory management	5	4	1	2	1	25
Computer organization	4	5	1	3	1	20
Propositional logic	3	6	1	5	0	17
Proving techniques	6	3	0	1	2	0

Colored rows in Table 1 indicate four CS topics where a *majority* of the instructors felt the topics were *not necessary* for their courses: Assembly language basics (6 courses out of 9); Thread concurrency and deadlocks (6 courses out of 9); Proving techniques: rules of inference, proof by contradiction, induction (6 courses out of 9); Memory management (5 courses out of 9). That these particular 4 topics were not viewed as necessary by a majority of spring course instructors was not striking. The MS in Computing program is a professional degree program, and the more theoretical aspects of CS are generally not integral to the curriculum taken by its students. Thompson and Edwards suggest their conversion students sought career paths other than those requiring “high technology skills” [15]. They cite student interest in management progression, combining disciplines, or broadening their overall IT knowledge as motivation for beginning a conversion course. Upon observing this, the University of Sunderland curriculum design team increased study options; they sought to maintain serving students with analyst/programmer aspirations, but also accommodate students primarily interested in the management and application of computing systems. Anecdotally, our observations of conversion students and our Computing program align with observations in the UK report [15], and we continue to encourage and offer a flexible curriculum that serves the breadth of the computing profession. We promote the study of all aspects of computing, namely CS, computer engineering, data science, software engineering, information technology, and information systems by encouraging students to reach out across the campus for coursework related to career goals and computing. We encourage conversion students to consider careers that combine computing and their non-computing knowledge and experience.

Table 2: Program Learning Outcomes (O): Whether instructors agreed their conversion students are achieving MS program learning outcomes O1–O4.

Outcome	Agree	Neither agree nor disagree	Disagree	Not necessary/observable
O1	6	0	1	2
O2	8	1	0	0
O3	4	2	2	1
O4	8	1	0	0

Table 2 represents the results of the second question, whether or not instructors agreed that their conversion students are achieving the MS in Computing program learning outcomes:

On completion of the program, students will be able to

O1 Appraise relationships among a variety of computing practices and technologies to create integrated solutions to computing problems.

O2 Communicate computing problems and suggested solutions to other professionals and with business clients.

O3 Formulate and defend realistic and detailed designs for solutions of problems of enterprise scope.
O4 Evaluate and apply common standards for technology and technology management.

The lowest ranking was "O3 Formulate and defend realistic and detailed designs for solutions of problems of enterprise scope". The result is not surprising for two reasons. First, while our MS curriculum has several courses that help build the knowledge and skills needed to define enterprise systems, these are MS program electives, so not all students meet this outcome. Second, as reported by Thompson and Edwards [15], conversion students have a variety of career goals and do not frequently choose the path of large system development; thus the electives are not chosen.

The third question had three subquestions:

I1 On average, how often did the instructor give additional attention to the conversion students in their class?
I2 Based on instructor observations, how often did the conversion students in their class meet with other students for support or collaboration?
I3 Based on instructor observations, how often did the conversion students in their class engage in out-of-class activities with other students/faculty?

Table 3: Interactions (I): Instructor estimates of the frequency with which conversion students interacted with the instructor and with other students both in and out of class.

Subquestion	Every class	Every other class	Every now and then	Don't know
I1	1	1	6	1
I2	1	2	3	3
I3	0	2	3	4

For each subquestion, the instructor could respond: "Every class," "Every other class," "Every now and then," or "Don't know." Table 3 summarizes the responses, presenting estimates of the frequency with which conversion students interacted with the instructor and with other students both in and out of class.

In question I1, we were interested in determining whether the conversion program was a burden. That is, were we, in effect, asking faculty to provide additional one-on-one tutoring for students not sufficiently prepared for graduate studies? Questions I2 and I3 were secondary probes of academic support structures.

We also asked instructors to make suggestions for bridge course improvement. Although we received five responses to this question, only one of them made an actual suggestion for improvement. This instructor opined that conversion students with backgrounds in humanities and social science were unprepared to keep up. The suggestion for improvement was to ". . . increase the class workload for students who express desire to go into analytical work and do not have the proper background." Two of the other four responses made observations about the areas in which conversion students struggle: (1) enterprise level design and solutions, (2) setting up computing environments and making systems work, (3) operating systems, (4) coding projects of significant size. Two responders complained that conversion students did not work as hard as non-conversion students. Since our IRB was approved for anonymous instructor surveys [to minimize the perceived risk that survey answers may lead to employment termination], we were unable to retrospectively examine whether these instructor(s) were commenting about the work ethic of a single student, or speaking more generally.

In the final question, we asked instructors for comments, suggestions, or insights about the conversion program. We received four responses. One response suggested conversion students' understanding of statistics and computational computing skills should be discussed/reviewed during advising sessions before students enroll in quantitative-based courses. Two responses were effusive in their discussion of the growth they have seen in conversion program participants ". . . semester after semester, and class after class." The fourth response stated the conversion program is very much needed and the investigators are employing good ideas.

3.3 Qualitative Observations

Overall, 4 of the 23 cohort members have permanently left the conversion program. Two individuals were employed full-time while attempting full-time graduate study. These students encountered the challenges low-income student populations face daily, including transportation issues, family responsibilities, financial matters, employment responsibilities and scheduling issues. As expected, full-time employment and graduate study is challenging. In a pilot effort, the power of the bridge course cohort organically developed into ongoing peer support. Unlike our pilot, ongoing peer support did not flow from the bridge course into the next term for our first cohort. We have explored ways to encourage students to form peer study groups because we have found that those groups are a positive factor in their overall success in the program.

While instructor survey data indicate there are opportunities for program improvement, conversion students have also had out-standing accomplishments. Three conversion students who worked together on an Elements of Software Development course project team have been extremely successful. Under the guidance of a faculty mentor, this team had a conference paper accepted for a poster presentation in a student scientific paper competition and the paper published as part of the proceedings. The research publication was a continuation of the team's course project. All three completed the program. One of the team members received an internship directly related to their career interest, and the internship led to full-time employment. Another member of the team: received a scholarship to attend the American Indian Science and Engineering Society (AISES) 2018 Leadership Summit; received a $10,000 Intel scholarship; accepted an Intel internship for summer 2018 where their work involved chip software development, testing and debugging as well as creation of an associated tutorial for the product; and received a GHC Student Scholarship to attend the 2018 Grace Hop-per Celebration. What makes the member's accomplishments so remarkable is on their entry survey, the student selected the "no previous computing experience" check box. Another member served as a graduate student research mentor for a summer Research Experience for Undergraduates (REU). Each of the individuals in our second cohort's bridge course found summer internship placements in computing. Despite covid-19 limitations, two of the 2019 bridge course cohort participated in summer 2020 internships; a third had an internship offer rescinded.

4 Discussion and Conclusion

The instructor survey helped confirm the pathway is successfully helping conversion students. Instructors are in excellent positions to judge how well the bridge course prepares students to achieve in the courses they teach. Instructors are able to compare conversion students with students in the same courses who have come to graduate work with strong backgrounds in computing. Examining the metric agree/necessary 50% (rather than a far more favorable (agree + neutral)/necessary metric), the survey indicates instructors consider conversion students to be sufficiently prepared in 14 of 25 CS requisites for success in their courses, and consider them to be completely deficient in only one. Furthermore, instructors believe conversion students are achieving all primary learning outcomes of the MS program, although there are weaknesses in O3 achievement. However, these results are remarkable considering conversion students' formal preparation is far less than the formal preparation of traditional students, who constitute the overwhelming majority of students enrolled in the MS program.

Despite these positive results, the survey also revealed aspects of the conversion students' knowledge that could be improved. There were 10 topics for which instructors neither agreed nor disagreed that conversion students were able to demonstrate required knowledge. Although the survey did not probe the instructors' thinking, we can speculate about their reasons for choosing the "Neither agree nor disagree" response. Some of the instructors of the six courses that had two or more conversion students in them may have chosen "Neither" because they felt some conversion students possessed the necessary knowledge but some did not. On the other hand, the instructors of any of the nine courses may have chosen "Neither" because they felt that the students' knowledge was mediocre, bordering on insufficient. A third possibility is that some instructors may have chosen "Neither" because they felt the extremely hard work exhibited by underachieving conversion students, who were aspiring to make career changes under difficult circumstances, made up for their inadequacies.

We are pleased with our conversion students' academic progress and current career placements. The three conversion students who encountered academic challenges in the term following the bridge course were working full-time to provide day-to-day living expenses (data not presented). The conversion student who did not complete the bridge course cited stress from full study, a full time job, and lengthy daily commute. A conversion student from the same summer cohort ended study in the middle of the term following the bridge course, citing various financial concerns. While our conversion students are mature and motivated, outside pressures are often much greater as they typically have families, debt loads, and other responsibilities. Knowing we were also going to target low-income students from underrepresented populations via federal scholarships, our curriculum design was impacted not only by our past experience with career change individuals, but also by a dose of pragmatism. Even when federal scholarships were coupled with generous institutional support, the total financial support falls far short of what was necessary for risk-averse, low-income students. Thus, conversion program curriculum was informed by regional employers input on skills that would help conversion students be competitive for summer internships a year after the bridge course. That each member of our second bridge course cohort and half of our third cohort received computing-related summer internships offers is promising. While there are still opportunities for change in the program, early successes increase our confidence in our program's ability to mentor people into realizing their potential. We employed two standard O'Reilly books for instruction of application focused Python and data structures using Java. Co-author (name suppressed during review) has taught each bridge course; while our focus has been on pathway development using in-person and synchronous delivery, one could envision asynchronous lessons and employing advanced undergraduates or graduate students to answer questions, provide mentoring and computing culture in-sights. This could make the pathway more easily transferable to a large academic program. We encourage others to join the effort to recruit and provide conversion pathways for post-baccalaureate individuals.

5 Acknowledgments

Have been removed for dual-anonymous review process.

References

1. Carla Brodley, Mega ine Gill, Ian Gorton, Benjamin Hescott, Bryan Lackaye, Cynthia LuBien, Leena Razzaq, Amit Shesh, Tiffani Williams, and Andrea Danyluk. 2020. An MS in CS for Non-CS Majors: Moving to Increase Diversity of Thought and Demographics in CS. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1248–1254. https://doi.org/10.1145/3328778.3366802
2. Jarrett Carter. 2017. *EducationDive Brief: NYU School of Engineering launches computer science bridge program for liberal arts graduates*. Retrieved August 29, 2019 from https://www.educationdive.com/news/nyu-school-of-engineering- launches-computer-science-bridge-program-for-libe/436046/
3. Helen M. Edwards and J. Barrie Thompson. 2003. Reflections on a UK Masters Level Software Engineering Programme Intended for the Home and International Market. In *Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET '03)*. IEEE Computer Society, Washington, DC, USA, 166–. http://dl.acm.org/citation.cfm?id=794194.794951
4. Penn Engineering. 2018. *Online Master of Computer and Information Technology*. Retrieved August 29, 2019 from https://onlinelearning.seas.upenn.edu/mcit/
5. Penn Engineering. 2020. *MCIT Online Course List*. Retrieved August 24, 2020 from https://onlinelearning.seas.upenn.edu/mcit-online-course-list/
6. Python Software Foundation. 2020. Tutorial — Flask Documentation (1.1.x). Retrieved August 25, 2020 from https://flask.palletsprojects.com/en/1.1.x/tutorial/
7. Katherine G. Herbert-Berger, Nina Goodey, Stephen Ruczszyk, Scott Kight, and Thomas J. Marlowe. 2019. Infusing CS Graduate Transition Curriculum with Professional, Technical and Data Science Competencies. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 1266. https://doi.org/10.1145/3287324.3293828
8. SANS Institute. 2020. *SANS Cyber Aces Online Courses*. SANS Institute. Retrieved August 24, 2020 from https://tutorials.cyberaces.org/tutorials.html
9. Deborah G. Johnson and Keith W. Miller. 2002. Is Diversity in Computing a Moral Matter? *SIGCSE Bull.* 34, 2 (June 2002), 9–10. https://doi.org/10.1145/543812.543814

10. M. Klawe, I. Cavers, F. Popowich, and G. Chen. 2000. *ARC*. Springer US, Boston, MA, 94–101. https://doi.org/10.1007/978-0-387-35509-2_12
11. Karsten Lundqvist, Craig Anslow, Michael Homer, Kris Bubendorfer, and Dale Carnegie. 2018. An Agile Conversion Masters Degree Programme in Software Development. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (Baltimore, Maryland, USA) *(SIGCSE '18)*. ACM, New York, NY, USA, 846–851. https://doi.org/10.1145/3159450.3159540
12. Robin Mellors-Bourne and Keith Williams. 2019. *Evaluation of a scheme to develop pilot engineering and computing conversion Masters courses. March 2019*. Office for Students, Bristol, England, UK.
13. New York University Tandon School of Engineering. 2020. NYU Tandon Bridge Computer Science | NYU Tandon School of Engineering. Retrieved August 26, 2020 from https://www.educationdive.com/news/nyu-school-of-engineering- launches-computer-science-bridge-program-for-libe/436046/
14. Stevens Institute of Technology. 2015. *Master of Science in Software Engineer-ing*. Retrieved August 29, 2019 from https://www.stevens.edu/school-systems-enterprises/masters-degree-programs/software-engineeringk
15. J. Barrie Thompson and Helen M. Edwards. 1999. Providing New Graduate Opportunities: Experiences with a UK Master's Level Computing Conversion Course. *J. Syst. Softw.* 49, 2-3 (Dec. 1999), 135–143. https://doi.org/10.1016/S0164-1212(99)00086-2
16. Northeastern University. 2020. How Align Works – Align MS in Computer Science Program. Retrieved August 20, 2020 from https://align.khoury.northeastern.edu/how-align-works/#:~:text=Full%2Dtime% 20students%20typically%20complete,part%2Dtime%20jobs%20to%20participate.
17. Business Wire. 2018. *University of Pennsylvania's School of Engineering Launches Online Master's in Computer Science*. Retrieved August 29, 2019 from https://www.businesswire.com/news/home/20180725005181/en/University-Pennsylvania's-School-Engineering-Launches-Online-Master's
18. Stuart Zweben and Betsy Bizot. 2020. 2019 CRA Taulbee Survey: Total Under-grad CS Enrollment Rises Again, but with Fewer New Majors; Doctoral Degree duction Recovers From Last Year's Dip. *Computing Research News* 32, 5 (2020), 3–63.