Enforcement of Prerequisites in Computer Science

Ernst Bekkering
bekkerin@nsuok.edu

Patrick Harrington
harringp@nsuok.edu

Mathematics and Computer Science
Northeastern State University
Tahlequah, OK 74464

Abstract

This paper describes the study of enforcement of prerequisites in the Computer Science program at a regional university in the Southwest. Prerequisites are a significant factor in programs of study in higher education. Allowing students to register in courses may assume that they have existing knowledge and skills. Some programs treat prerequisites as advisory, while others consider them mandatory. In the latter case, procedures usually exist to make exceptions in the form of registration overrides. The state of prerequisite enforcement at our university over the years, and some factors that may have influenced adherence to the prerequisite structure over the years, will be discussed in this paper.

Keywords: enrollment, curriculum, prerequisites, scheduling, graduation.

1. INTRODUCTION

In higher education, courses may have prerequisites and/or co-requisites to ensure that students taking the course are sufficiently prepared. In reality, enforcement of these prerequisites may be difficult. For instance, enrollment systems cannot check prerequisites until after the semester end because course grades have not been awarded and enrollment for the next semester starts well before the current semester ends (Boyer & Bucklew, 2019). A variety of processes exist to deal with this timing problem.

The issue of enforcing prerequisites is especially important when the demand for a course exceeds the course capacity. Students who should not be taking the course (yet) take a seat that should have gone to students who do qualify (Soria & Mumpower, 2012).

This research will examine the compliance with successful completion of prerequisites and enrollment in co-requisites in the Computer Science program where the investigators teach. The paper is organized as follows. In the next section, we discuss the relevant literature. Next, we discuss the methodology of our study, data collection, and data analysis. We end with conclusions and recommendations.

2. LITERATURE REVIEW

Prior knowledge and skills needed to be successful in a course can take multiple forms. Prerequisites can be defined as courses or tests that must be successfully completed prior to registering for the target course. In some cases, special tests are given at the start of a course to identify students who need additional support to get caught up. Co-requisites are courses that must be taken at the same time as another course. In the case of labs partnered with lecture sections, the reason for the split is usually administrative − multiple lab sections with fewer students are needed to give all students sufficient individual attention in the lab. In other cases, the material in one course is supportive of the target course. For instance, Discrete Mathematics may

be co-requisite for Basic Computer Architecture to cover Boolean logic more in-depth in the mathematics course.

Rationale for expecting prior knowledge

Curriculum design involves combining multiple courses into an integrated program of study. Considering limitations of time and other resources, programs should maximize coverage of content while minimizing overlap. As Diamond (1998) states:

> "One of the most prevalent problems in course and curriculum design is the tendency of faculty to make false assumptions about the knowledge and skills that students bring to their courses. These incorrect assumptions lead to failure for the students who are ill prepared, boredom for their classmates who are often more than adequately prepared, and frustration for the faculty."

Performance in advanced courses depends on how the material in prerequisite courses is used and evaluated (Nelson et al., 2020). Lack of correct prerequisites and lack of continuity between courses in a sequence are specifically mentioned as causes of student failure (Babb et al., 2014). "Depth of knowledge" assumes a hierarchical structure of programs (Reynolds et al., 2016). This is not limited to a single program of study. If programs have more than one concentration, two or more sequenced courses within the concentration are required (Downey et al., 2008).

Babb et al. (2014) propose that students must "step-wise and incrementally, engage in the persistent and iterative pursuit of programming" of at least 15 out of 23 topics over the course of the curriculum. Longenecker et al. (2013) suggest a series of three programming courses with database as prerequisite for the last course. Decreasing the number of programming courses in the early 2000s backfired, and industry continues to demand technical skills (George & Marett, 2019).

Prerequisites do not come without drawbacks. Students can suffer delays in graduation, and universities may have to offer more courses with a deeper prerequisite structure (Reynolds et al., 2016). For some students, it can be a reason to select another major (Li et al., 2014). It is therefore important to impose only prerequisites that are necessary and effective.

Effectiveness

Much of the research on prerequisites has focused on their effectiveness in a program's curriculum. The effect of prior knowledge has been tested statistically through correlation of prior coursework and final grades in the target course, and by correlation with special pretests at the start of the target course.

Examples of studies that use final grades in the target course are Blaylock & Lacewell (2008), Krause-Levy et al. (2020), Liao et al. (2019), and Soria & Mumpower (2012). All found positive relationships between prerequisites and target courses.

Passing previous courses may not be effective. The use of proficiency tests instead of prerequisite courses has been described by Rondeau & Li (2009). Instituting the test created a backlog of students who failed the test, and the test did not fully cover the required knowledge. Also, not all prior knowledge can be obtained in a single course. Blaylock & Lacewell (2010) used a prerequisites test covering topics from multiple disciplines to demonstrate that prior knowledge led to better final grade results. Sargent (2013) used proficiency tests for Intermediate Accounting with good result. Abou-Sayf (2009) concluded that using entrance tests in the target course provided more accurate results than using final grades.

Higher education has become increasingly fluid. Whereas decades ago, students tended to complete degrees in one institution, the last decade has seen an increasing number of students who transfer from community colleges to four-year institutions to finish their degree. Transfer agreements between schools and state transfer guides (for instance OSRHE (2022)) help students to combine courses from multiple sources into a single degree. Needless to say, this assumes sufficient similarity of course contents to be successful. The effect of transfer guides on the number of prerequisites taken elsewhere is small, and mature students benefit more (Spencer, 2019). Furthermore, Catanese et al. (2018) found no significant difference in a third computing course between native students and transfer students who took the first two courses elsewhere.

Finally, some problems exist in quantitative measurement of the effect of prerequisites. Students are more mature in the target course, the students in the prerequisite and target course partially overlap, self-selection happens due to students not following up with the target course,

and a cause-and-effect relationship has not been demonstrated (Abou-Sayf, 2009).

Selection process

The second main area for prerequisites research is in the selection process. One of the first questions in delivering courses is the issue of prerequisite knowledge (Johnson et al., 2002). In some cases, this is easy. Database II should have Database I as direct prerequisite (Reynolds et al., 2016). Computer Science II should be preceded by successful completion of Computer Science I, and Object-Oriented Programming (in Java) might need the coverage of objects in C++. Model curricula may suggest this type of sequential prerequisites. In IS2010 and IS2020, Foundations of Information Systems is prerequisite for most other courses, and all other courses must be passed before culminating courses like the capstone course (Leidig et al., 2019; Leidig & Salmela, 2021).

Other prerequisites may be less intuitive. Some courses have non-sequential prerequisites from other disciplines. For instance, some programming courses may need the logical thinking from specific Mathematics courses. White found that prerequisites are needed to develop the proper cognitive style in order to be successful in Visual Basic programming (2012). In a follow-up study, White and Sivitanides suggested that freshman level mathematics courses are good indicators for success in Visual Basic (2003). To fully understand the issues round technology-based entrepreneurship, a course on information systems in business is essential (Jones & Liu, 2017).

Level of enforcement

Registration systems contribute to the problem of skipping prerequisites (Wilkerson et al., 2019). As mentioned before, course registration systems have difficulty enforcing prerequisites if registration for a future semester is allowed before grades in the current semester have been posted. Several solutions to this dilemma exist.

First, universities may use conditional enrollment pending successful completion of the prerequisite. Second, appropriate staff can issue overrides. Academic advisors may have permission to issue an override if transfer courses have not been posted in the transcript system yet, and faculty may override enrollment blocks in courses they teach if they deem the prerequisite unnecessary for specific students. Departments can give overrides for courses in their department. An example of this process can be found at OSU (2021) . Third, enrollment without prerequisites may not be blocked. Course descriptions may merely mention prerequisites, and students can not notice or ignore them. In that case, it would be up to faculty to check transcripts before or at the start of the course. Finally, the university may only open enrollment after grades in the current semester have been posted. This pushes enrollment back compared with early registration and is unattractive from an administrative point of view. Soria & Mumpower (2012) describe their university's switch to an automated, mandatory prerequisite enforcement system and found that it led to better academic outcomes.

The need to check grades does not exist for co-requisites. If the course has been taken before it is available in the transcript system, and if not must be taken in the same semester as the target course. Registration systems could check if the co-requisite is covered and either disable registration until it is met or issue a warning that the co-requisite course must also be registered.

In closing, student compliance with prerequisites and co-requisites is a multi-faceted issue. The literature indicates that they are effective, gives some guidelines for their selection, and that they may not always be followed. We have not been able to find literature indicating how often they are skipped, and that is the focus of this study.

3. METHODOLOGY

This section describes the methodology of measuring compliance with successful completion of prerequisites and enrollment in co-requisites for the Computer Science program.

Online university systems

Students register on Banner and transcripts can be checked on DegreeWorks.

Course registration system

The Banner system (Ellucian Company LP, 2022a) is a comprehensive suite that includes student course registration and instructor functions like generating course enrollment listings and grade entry. Enrollments are synchronized nightly with the course management system BlackBoard. The gradebooks in BlackBoard contain the student identifiers for each course.

Registration overrides may be customizable by the university, allowing different users and groups different permissions. At our university, overrides for undergraduate courses can be issued by faculty, department, and academic advisors (Figure 1). Prerequisites are not listed as

a reason. Academic advisors contact faculty for prerequisite overrides. This typically happens between Software Engineering and its target Capstone course.

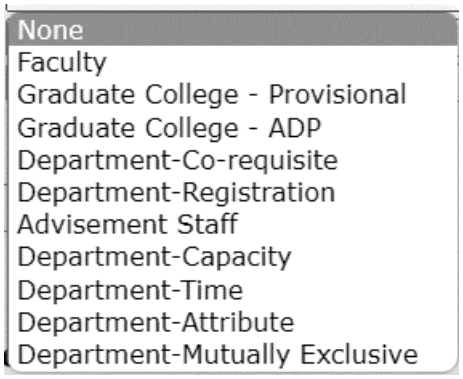


Figure 1 - Types of overrides

Transcript system

DegreeWorks (Ellucian Company LP, 2022b) allows selecting single students based on their N number (a unique number unconnected with other identifiers such as social security numbers) but also has a search function that includes searching on majors and minors. The results of each search are displayed in a popup window with name and N numbers.

The frequency of synchronization between grade entry on the Banner system and the transcripts in Degreeworks is not known but assumed to be frequent since both packages are sold by the same company. The "historic audit" dropdown shows multiple snapshots during the semesters, and the "date refreshed" field can be used to refresh the current transcript on the fly.

Transcripts show all required courses for the program of study, the semester when taken or scheduled, the course grade status, and special notes for course transfers and substitutions. Sample (anonymized) transcripts for are shown in Appendix A.

Prerequisites in the CS program

The Computer Science program has multiple courses with prerequisites or co-requisites. Some are sequential, such as Computer Science I and Computer Science II which even use the same textbook. Other courses are non-sequential, such as Object Oriented Programming and Software Engineering. The course content in the programming course is not used directly in the target course, but the intent is to ensure that students have sufficient programming background to start preparing for the Capstone course. A complete listing of common courses with prerequisites is provided in Table 1. Some courses with prerequisites in the course catalog are seldom or never offered, and we ignored those. We also omit courses restricted to instructor permission only.

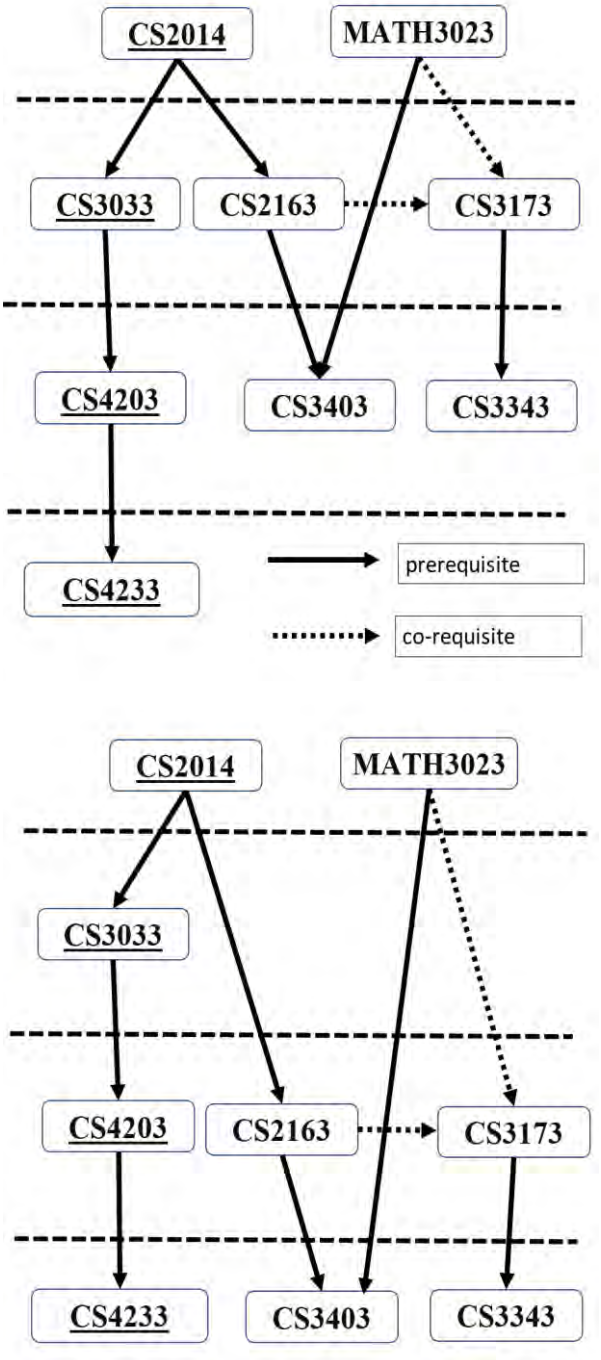
Course	Prerequisite/co-requisite
CS 2014 Computer Science I	Applied Mathematics (co) or College Algebra (co) or ACT >=23 and computer proficiency
CS 2163 Computer Science II	Computer Science I with C minimum
CS 3033 Object Oriented Programming	Computer Science I with C minimum
CS 3173 Basic Computer Architecture	Computer Science II (co) and MATH 3023 Discrete Mathematics (co)
CS 3343 Computer Operating Systems	Basic Computer Architecture
CS 3403 Data Structures	Computer Science II with C minimum and MATH 3023 Discrete Mathematics
CS 4343 Database Management Systems	Computer Science II and MATH 3023 Discrete Mathematics
CS 4203 Software Engineering	Object Oriented Programming with C minimum
CS 4233 Capstone	Software Engineering

Table 1 - Courses and prerequisites

Based on prerequisites in required courses, the program has a critical path to graduate in four semesters after General Education requirements have been met. The critical path is shown in Figure 2. In all three versions, completing the upper division courses in four semesters is only possible with the sequenced prerequisites Computer Science I, Object Oriented Programming, Software Engineering, and Capstone (Professional Development in CS).

This critical path was shortened about six years ago. Before the change, the prerequisite for Object Oriented Programming was Computer Science II, not Computer Science I. The critical path at that time had a length of five courses with Computer Science I, Computer Science II, Object

Oriented Programming, Software Engineering, and finally the capstone course. Even though we felt that Computer Science II was a better prerequisite for Object Oriented Programming than Computer Science I, we decided to shorten the path. This demonstrates that some of the considerations in the use of prerequisites are political as noted by Abou-Sayf (2009).



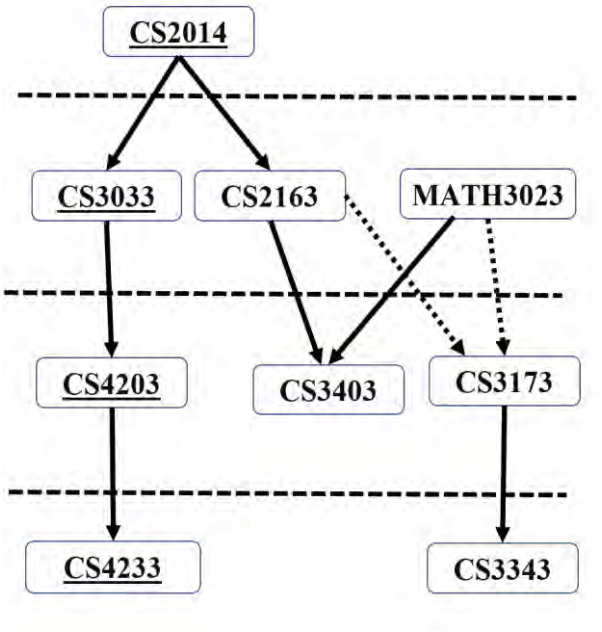


Figure 2 - Critical Path in CS

Course rotation

The CS program is offered on two campuses. In the past, courses were only offered face to face (f2f) on alternating campuses. With the advent of powerful online meeting tools like BlackBoard Collaborate and later Zoom, courses were increasingly split in two sections, with one section f2f and the other online concurrently. This allowed students to take courses each semester independent of their main campus, and the course rotation is shown in the first image in Appendix B.

Starting with the Fall 2021 semester, the course offerings were reduced to once a year, regardless of campus. Faculty could select the campus for the f2f section and stream the course to the other campus. The faculty with doctoral degrees were reduced from five to three by releasing two faculty on the tenure track and not replacing them. The road map with implicit course rotation is shown in the second image in Appendix B.

The road map for CS shows that CS courses are spread over four years and eight semesters. In our program, approximately half of the CS majors complete all four years at our university, but the other half consists of transfer students from community colleges. For those transfer students, and students who do not declare CS as their major until after completion of their General Education courses, students can combine the major and minor courses for the first and second years in the appropriate semesters, as well as

combining the third and fourth years. For all students though, most of the required courses are offered only once a year. This complicates proper course selection for our majors.

Scraping web data

All course registration data and all transcripts are accessible online at the university website. One of the current popular tools for collecting online data is a combination of the Python programming language and the Selenium module to automate user actions in web browsers (Chapagain, 2019). The Scrapy module (Zyte, 2022) is faster due to multithreading but does not render JavaScript and is not very user-friendly. BeautifulSoup (Richardson, 2015) needs additional modules for sending requests and parsing HTML pages. Selenium is easy to use, sends its own requests, and can pull data that is only available when JavaScript is loaded (Grimes, 2022).

4. DATA COLLECTION

We filtered the transcript system on Computer Science majors and copied the list of students to a text file. Student identification numbers are in the format Nxxxxxxxx, where x is a digit. The N numbers were extracted with regular expressions, and we used Python scripts with Selenium to download transcripts and test scores. Transcripts and test scores were saved with corresponding random numbers for file names and identifying information in the files was removed. Finally, we used Python scripts with Selenium and BeautifulSoup to extract the data we needed in csv format.

The csv file was used as the data source for the analysis spreadsheet. Separate tabs for each course reviewed extracted the data for the prerequisites and co-requisites and used lookup tables to convert semesters to sequential numbers. The semester numbers of prerequisites and co-requisites were compared with the number of the target course to check if they fell before or with the target course and stored in true/false format. Separate lookup tables were used to check for passing grades in prerequisites and co-requisites and stored in true/false format. Formulas with AND() and OR() were constructed to check if all rules were met.

We filtered first on target courses being taken, and then on meeting all the rules. Finally, we assigned categories for the reasons. Figure 3 shows an example for Basic Computer Architecture, which has two co-requisites with passing grades of D.

co_cs2163withD	co_math3023withD	ALL_RULES_MET	COMMENTS
FALSE	TRUE	FALSE	co-requisite(s) not taken
TRUE	FALSE	FALSE	co-requisite(s) failing grade
FALSE	TRUE	FALSE	co-requisite(s) not taken
TRUE	FALSE	FALSE	co-requisite(s) failing grade
TRUE	FALSE	FALSE	co-requisite(s) failing grade
TRUE	FALSE	FALSE	co-requisite(s) failing grade
TRUE	FALSE	FALSE	co-requisite(s) failing grade
TRUE	FALSE	FALSE	co-requisite(s) failing grade
FALSE	TRUE	FALSE	co-requisite(s) not taken

Figure 3 - Categorization

Finally, the data were summarized on a separate tab by semester and categories in tabular and graphical format for analysis.

5. ANALYSIS AND DISCUSSION

We will now discuss the CS major courses. We will start with a brief description of the course and its requirements, followed by the results and our interpretation. Appendix C shows graphs for the number of times requirements were not met, the occurrences relative to the total times the course was taken in the semester, and a breakdown of the reasons. The historical rate mentioned is the rate of unmet requirements over the history of the program, based on all available transcripts.

Computer Science I is the introductory programming course. The course presents the basics of programming in C++. Two Math courses can be taken as a co-requisite, or students can qualify with an ACT of 23 or higher. All students must have computer proficiency. The course has a substantial number of students who do not meet the requirements. The historical rate is 17.8%. One of the prerequisites is usually taken before the course, but a substantial number of students fail the MATH course. The main reason for failing the requirements is not taking them, with a strong second not passing when taken before. Very few students fail to get a passing grade when taken concurrently.

Computer Science II presents more programming in C++. In contrast to Computer Science I, it has a low historical rate of 4.1%. The course has Computer Science I with a minimum of C as a prerequisite. The majority of students not meeting it are transfer students from a Community College who either failed to get a C or took the courses concurrently.

Object Oriented Programming is taught in Java. The prerequisite is Computer Science II, which has to be passed with a C. The historical rate is extremely low at 2.5%, with more than half

related to transfer students who fail to get a C or taking the prerequisite late.

Basic Computer Architecture presents the theoretical foundation of computers, including the basic hardware components and an introduction to assembly language. It has two courses that can be taken before or concurrently. The historical rate is 11.5%. About twice as many students fail to pass Discrete Mathematics as Computer Science II.

Computer Operating Systems follows Basic Computer Architecture, its only prerequisite. The course presents topics like concurrency, processes, and threads. The historical rate is lower at 7.9%. Both courses are taught in opposite semesters and taking the prerequisite two semesters later generally delays graduation.

Data Structures introduces both common data structures and algorithms used in Computer Science. The historical rate is 8.6%, of which more than three quarters is caused by not passing Discrete Mathematics and the rest by failing to pass Computer Science II or both.

Database Management Systems presents multiple forms of databases as storage and retrieval tools for data. It has a historical rate of 10.8%, with two thirds due to not meeting the Discrete Mathematics requirement and one third not passing Computer Science II.

Software Engineering presents the software development process including requirements analysis, modeling, and testing. Combining this course with previous programming courses prepares the students for the integrative Capstone course. The historical rate is higher again at 14.7%, with only a very minor part due to failing the prerequisite. The remainder is evenly split between taking the courses concurrently and taking the prerequisite after the software engineering course.

The capstone course has students develop an individual integrative project of their choice. The historical rate for not meeting the Software Engineering prerequisite is 16.0%, with virtually all due to taking the capstone and its prerequisite concurrently.

Table 2 presents a summary of the historical rate of unmet requirements. The rate appears to vary based on three factors. First, courses with Mathematics prerequisites tend to have higher rates. This is true for especially Computer Science I. Second, sequential courses tend to have lower rates. This is true for Computer Science I and II (4.1%), Computer Science I and Object Oriented Programming (2.5%), and Basic Computer Architecture and Computer Operating Systems (7.9%). Finally, courses later in the program tend to have higher rates. We attribute this to the pressure to graduate on time.

Course	Sequential	Percent Unmet
Computer Science I	no	17.8%
Computer Science II	yes	4.1%
Object Oriented Programming	yes	2.5%
Basic Computer Architecture	no	11.5%
Computer Operating Systems	yes	7.9%
Data Structures	no	8.6%
Database Management Systems	no	10.8%
Software Engineering	no	14.7%
Capstone	no	16.0%

Table 2 - Historical Rate of Unmet Requirements

6. CONCLUSIONS AND RECOMMENDATIONS

Our analysis shows a significant level of failing to meet requirements for courses in our Computer Science program. As involved faculty members, the results do not surprise us. We do see this as an opportunity to take actions that alleviate this problem.

The tools used for this study can be converted to multiple proactive tools. We can generate lists for academic advisors with students needing to take the specific courses in the following semester. These lists can be used in advising sessions, to contact students proactively by email or text message, and to check courses needed against actual enrollments. The same lists can be used by the department to estimate demand for the next semester so an adequate number of course sections can be scheduled. Finally, we can generate faculty lists with checked prerequisites and co-requisites at the start of the semester to maximize the enrollment of eligible students by eliminating non-eligible students.

Finally, our study does not address the effect of unmet prerequisites on student performance in the target course. We plan to make this a separate study.

5. REFERENCES

Abou-Sayf, F. K. (2009). Challenges inherent in the assessment of the impact of prerequisite courses. *Journal of Applied Research in the Community College*, *17*(1), 9−15.

Babb, J. S., Longenecker, B., Baugh, J., & Feinstein, D. (2014). Confronting the Issues of Programming In Information Systems Curricula: The Goal is Success. *Information Systems Education Journal*, *12*(1), 42.

Blaylock, A., & Lacewell, S. K. (2008). Assessing Prerequisites as a Measure of Success in a Principles of Finance Course. *Academy of Educational Leadership Journal*, *12*(1), 51−62.

Blaylock, A., & Lacewell, S. K. (2010). Ex Ante Prerequisite Knowledge and Student Success in Principles of Finance. *Journal of Learning in Higher Education*, *6*(1).

Boyer, N., & Bucklew, K. (2019). Competency-based education and higher education enterprise systems. *The Journal of Competency-Based Education*, *4*(1), e01180. https://doi.org/10.1002/cbe2.1180

Catanese, H., Hauser, C., & Gebremedhin, A. H. (2018). Evaluation of native and transfer students' success in a computer science course. *ACM Inroads*, *9*(2), 53−57. https://doi.org/10.1145/3204471

Chapagain, A. (2019). *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others*. Packt Publishing Ltd.

Diamond, R. M. (1998). *Designing and Assessing Courses and Curricula: A Practical Guide. Second Edition. The Jossey-Bass Higher and Adult Education Series*. Jossey-Bass, Inc.

Downey, J. P., McMurtrey, M. E., & Zeltmann, S. M. (2008). Mapping the MIS Curriculum Based on Critical Skills of New Graduates: An Empirical Examination of IT Professionals. *Journal of Information Systems Education*, *19*(3), 351.

Ellucian Company LP. (2022a). *Banner ERP System for Universities*. Ellucian. https://www.ellucian.com/solutions/ellucian-banner

Ellucian Company LP. (2022b). *Ellucian Degree Works*. Ellucian. https://www.ellucian.com/solutions/ellucian-degree-works

George, J. F., & Marett, K. (2019). The Times they are a Changin': How Non-Technology Factors have Affected IS Curriculum over Time. *Journal of Information Systems Education*, *30*(4), 222.

Grimes, J. (2022, March 27). *Scrapy Vs. Beautifulsoup Vs. Selenium for Web Scraping*. Best Proxy Reviews. https://www.bestproxyreviews.com/scrapy-vs-selenium-vs-beautifulsoup-for-web-scraping/

Johnson, D. W., Wilkes, F., Ormond, P., & Figueroa, R. (2002). Adding Value to the IS'97… Curriculum Models: An Interactive Visualization and Analysis Prototype. *Journal of Information Systems Education*, *13*(2), 135.

Jones, C. G., & Liu, D. (2017). Approaches to Incorporating IT Entrepreneurship into the Information Systems Curriculum. *Journal of Information Systems Education*, *28*(1), 43.

Krause-Levy, S., Valstar, S., Porter, L., & Griswold, W. G. (2020). Exploring the Link Between Prerequisites and Performance in Advanced Data Structures. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 386−392. https://doi.org/10.1145/3328778.3366867

Leidig, P., Ferguson, R., & Reynolds, J. (2019). IS2010: A Retrospective Review and Recommendation. *Journal of Information Systems Education*, *30*(4), 298−302.

Leidig, P., & Salmela, H. (2021). *IS2020 Final Report − IS2020 ACM-AIS Curriculum*. https://is2020.hosting2.acm.org/2021/06/01/is2020-final-draft-released/

Li, L., Zhang, C., & Zheng, G. (2014). Promoting Information Systems Major to Undergraduate Students—A Comprehensive Investigation</title>. *Journal of Information Systems Education*, *25*(3), 211.

Liao, S. N., Zingaro, D., Alvarado, C., Griswold, W. G., & Porter, L. (2019). Exploring the Value of Different Data Sources for Predicting Student Performance in Multiple CS Courses.

Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 112–118. https://doi.org/10.1145/3287324.3287407

Longenecker, H. E., Feinstein, D. L., & Babb, J. S. (2013). Is there a need for a Computer Information Systems model curriculum. *Proceedings of the Information Systems Educators Conference ISSN*, *2167*, 1435–1447.

Nelson, G. L., Strömbäck, F., Korhonen, A., Begum, M., Blamey, B., Jin, K. H., Lonati, V., MacKellar, B., & Monga, M. (2020). Differentiated Assessments for Advanced Courses that Reveal Issues with Prerequisite Skills: A Design Investigation. *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education*, 75–129. https://doi.org/10.1145/3437800.3439204

Oklahoma State University. (2021, October 26). *Banner Registration Permits/Overrides for Faculty and Staff—Oklahoma State University*. https://registrar.okstate.edu/banner-registration-permits-overrides-faculty-staff.html

OSRHE. (2022). *Oklahoma State Regents for Higher Education | Students | Transfer Students | Course Transfer*. https://www.okhighered.org/transfer-students/course-transfer.shtml

Reynolds, J., Ferguson, R., & Leidig, P. (2016). A Tale of Two Curricula: The Case for Prerequisites in the Model Curriculum. *Information Systems Education Journal*, *14*(5), 17.

Richardson, L. (2015). *Beautiful Soup Documentation—Beautiful Soup 4.4.0 documentation*. https://beautiful-soup-4.readthedocs.io/en/latest/

Rondeau, P. J., & Li, X. (2009). The Impact of a Computer Proficiency Exam on Business Students' Admission to and Performance in a Higher-Level IT Course. *Journal of Information Systems Education*, *20*(4), 477.

Sargent, C. S. (2013). Find it, fix it, and thrive: The impact of insisting on proficiency in prerequisite knowledge in intermediate accounting. *Issues in Accounting Education*, *28*(3), 581–597.

Soria, K. M., & Mumpower, L. (2012). Critical Building Blocks: Mandatory Prerequisite Registration Systems and Student Success. *NACADA Journal*, *32*(1), 30–42.

Spencer, G. (2019). Can Transfer Guides Improve the Uptake of Major Prerequisites? Evidence from Ohio's Transfer and Articulation Policy Reform. *Research in Higher Education*, *60*(4), 458–484. https://doi.org/10.1007/s11162-018-9522-2

White, G. (2012). Visual Basic Programming Impact on Cognitive Style of College Students: Need for Prerequisites. *Information Systems Education Journal*, *10*(4), 74.

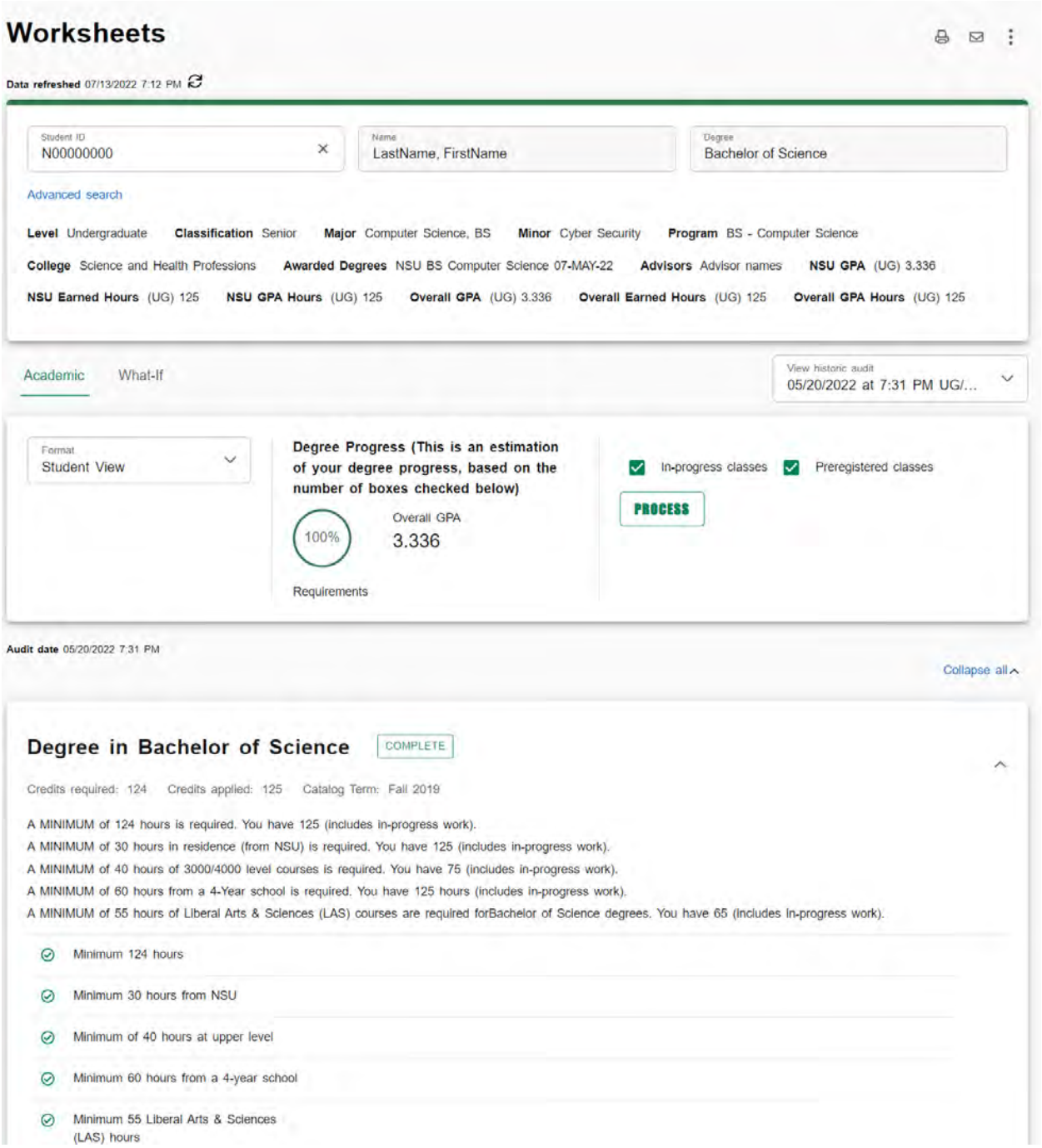
White, G., & Sivitanides, M. (2003). An Empirical Investigation of the Relationship Between Success in Mathematics and Visual Programming Courses. *Journal of Information Systems Education*, *14*(4), 409.

Wilkerson, J. M., Palmer, D. W., & Meyer, J. C. (2019). Disintegrating Business Degree Curricula by Skipping Prerequisite Courses. *Journal for Excellence in Business & Education*, *6*(1), Article 1. http://www.jebejournal.org/index.php/jebe/article/view/113

Zyte. (2022). *Scrapy | A Fast and Powerful Scraping and Web Crawling Framework*. https://scrapy.org/

Appendix A – Sample anonymized transcripts

Graduated student



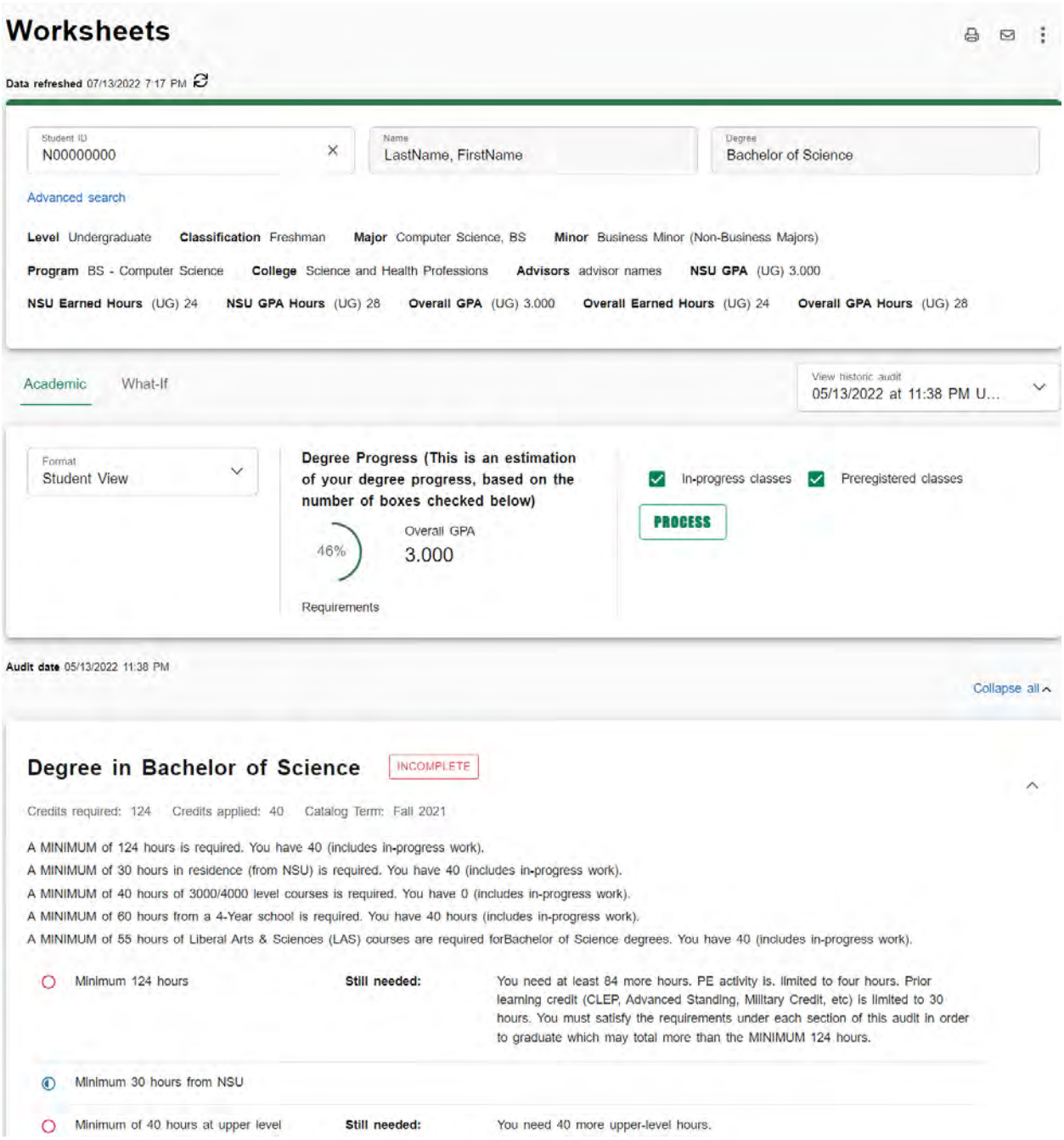
Major in Computer Science, BS COMPLETE

Credits required: 42 Credits applied: 43 Catalog Term: Fall 2019

Your current major GPA is 3.348.

	Course	Title	Grade	Credits	Term	Repeated
Computer Science I	CS 2014	COMPUTER SCIENCE I	B	4	Fall 2018	
Computer Science II	CS 2163	COMPUTER SCIENCE II	A	3	Spring 2019	
Object Oriented Programming	CS 3033	OBJECT ORIENTED PROGRAMMING	A	3	Spring 2020	
Basic Computer Architecture	CS 3173	BASIC COMPUTER ARCHITECTURE	D	3	Fall 2019	
Computer Operating Systems	CS 3343	COMPUTER OPERATING SYSTEMS	A	3	Spring 2020	
Data Structures	CS 3403	DATA STRUCTURES	C	3	Fall 2019	
Software Engineering	CS 4203	SOFTWARE ENGINEERING	B	3	Spring 2021	
Professional Development in Computer Science	CS 4233	PROF DEV IN COMPUTER SCIENCE	C	3	Spring 2022	
Database Management Systems	CS 4343	DATABASE MANAGEMENT SYSTEMS	A	3	Spring 2020	
Professional & Technical Writing	ENGL 3083	PROF & TECHNICAL WRITING	A	3	Fall 2020	
6 Hrs-Group A or B Electives	CS 3023	OBJECT BASED VISUAL PROG	A	3	Fall 2021	
	CS 3203	APPLICATION DEVELOPMENT IN C++	A	3	Fall 2019	
5 Hrs-Group B Electives	CS 4103	LT: ADVANCED JAVA	A	3	Fall 2021	
	CS 4553	PARALLEL PROGRAMMING	A	3	Fall 2020	

Ongoing student



Major in Computer Science, BS INCOMPLETE

Credits required: 45 Credits applied: 3 Catalog Term: Fall 2021

Unmet conditions for this set of requirements:

45 hours are required. You have 3, (includes in-progress work) and you need at least 42 more hours.
A minimum of 21 hours must be taken in residence. You have 3 (includes in-progress work) and you need 18 more hours. Hours in residence (from NSU) EXCLUDE prior learning credit such as CLEP, Advanced Placement, Advanced Standing, Military Credit, etc.
A minimum of 21 upper level hours are required. You have 0 (includes in-progress work) and need 21 more hours.
Minimum GPA unsatisfied

Your current major GPA is 0.000.

	Course	Title	Grade	Credits	Term	Repeated
○ Computer Science I	**Still needed:**	1 Class in **CS** 2014				
◐ Computer Science II	CS 2163	COMPUTER SCIENCE II	IP	(3)	Fall 2022	
○ Object Oriented Programming	**Still needed:**	1 Class in **CS** 3033				
○ Basic Computer Architecture	**Still needed:**	1 Class in **CS** 3173				
○ Computer Operating Systems	**Still needed:**	1 Class in **CS** 3343				
○ Data Structures	**Still needed:**	1 Class in **CS** 3403				
○ Software Engineering	**Still needed:**	1 Class in **CS** 4203				
○ Professional Development in Computer Science	**Still needed:**	1 Class in **CS** 4233				
○ Database Management Systems	**Still needed:**	1 Class in **CS** 4343				
○ Professional & Technical Writing	**Still needed:**	1 Class in **ENGL** 3083				
○ Discrete Mathematics	**Still needed:**	1 Class in **MATH** 3023				
○ 6 Hrs-Group A or B Electives	**Still needed:**	6 Credits in **CS** 1033 or 2023 or 3023 or 3101 or 3102 or 3103 or 3203 or 3223 or 3623 or 3633 or 3643 or 3663 or 4023 or 4101 or 4102 or 4103 or 4113 or 4143 or 4223 or 4233 or 4253 or 4363 or 4463 or 4553 or **IS** 3053 or **MATH** 4223 or 4233				
○ 5 Hrs-Group B Electives	**Still needed:**	5 Credits in **CS** 3223 or 4023 or 4101 or 4102 or 4103 or 4113 or 4143 or 4223 or 4233 or 4253 or 4363 or 4463 or 4553 or **MATH** 4223 or 4233				

Appendix B – Course rotations

Campus 1 - Computer Science

CS 2013 Computer Science I	F, S
CS 2163 Computer Science II	F, S
CS 3023 Object Based Visual Prog.	F, S
CS 3033 Object Oriented Programming	S
CS 3173 Basic Computer Architecture	F
CS 3203 Application Development in C++	Odd fall (ITV to Campus 1)
CS 3343 Computer Operating Systems	S
CS 3403 Data Structures	F
CS 4203 Software Engineering	F
CS 4233 Professional Dev. in CS	F, S
CS 4343 Database Mgmt	S

Even Fall	Odd Spring
CS 3623	CS 3663

Odd Fall	Even Spring
CS 3643	CS 3633

Campus 2 – Computer Science

Even Fall	Odd Spring
CS 3033, evening	CS 3173, evening
Group B Elective, evening	CS 3403, evening
MATH 3023, Evening	CS 3623, evening
CS 3633, evening	
	CS 4343 (ITV to Campus 2)
CS 4203 (ITV to Campus 2)	Group B Elective (ITV to Campus 2)
Group B Elective (ITV to Campus 2)	Group B Elective (ITV to Campus 2)

Odd Fall	Even Spring
CS 3343, evening	CS 4203, evening (ITV to Campus 1)
CS 4343, evening	Group B Elective, evening
CS 3663, evening	CS 3643, evening
CS 3203, evening (ITV to Campus 1)	

CS 3173 (ITV to Campus 2)	CS 3033 (ITV to Campus 2)
CS 3403 (ITV to Campus 2)	CS 3343 (ITV to Campus 2)
MATH 3023 (ITV to Campus 2)	

Note that CS 4233 is an arranged course and can be offered as needed.

September 30, 2016

B.S. Computer Science Road Map

First Year

Fall Semester	Spring Semester
CS 2014 Computer Science I	CS 2163 Computer Science II
H ED 1113 Personal Health or NUTR 1653 Basic Nutrition	MATH 3023 Discrete Mathematics
MATH 1513 College Alg. (if necessary) otherwise free elective	***Physical Science
ENGL 1113 Freshman Composition I	ENGL 1213 Freshman Composition II
UNIV 1003 University Strategies	Free Elective (3 hours)
Total Hours 17	**Total Hours 15**

Second Year

Fall Semester	Spring Semester
CS 3403 Data Structures	CS 3033 Object Oriented Programming
CS Group A or B Elective	Minor Course
Minor Course	***Humanities (First Course)
***Biological Science	***Communications Course
POLS 1113 American Federal Government	HIST 1483 or 1493 American History
Total Hours 15	**Total Hours 15**

Third Year

Fall Semester	Spring Semester
CS 3173 Basic Computer Architecture	CS 3343 Computer Operating Systems
CS Group A or B Elective	CS 4343 Database Management System
Minor Course	ENGL 3083 Technical Writing
***Humanities (Second Course)	Minor Course
***Global Perspectives Course	Free Electives (3 hours)
***Social and Behavioral Sciences Course	
Total Hours 18	**Total Hours 15**

Fourth Year

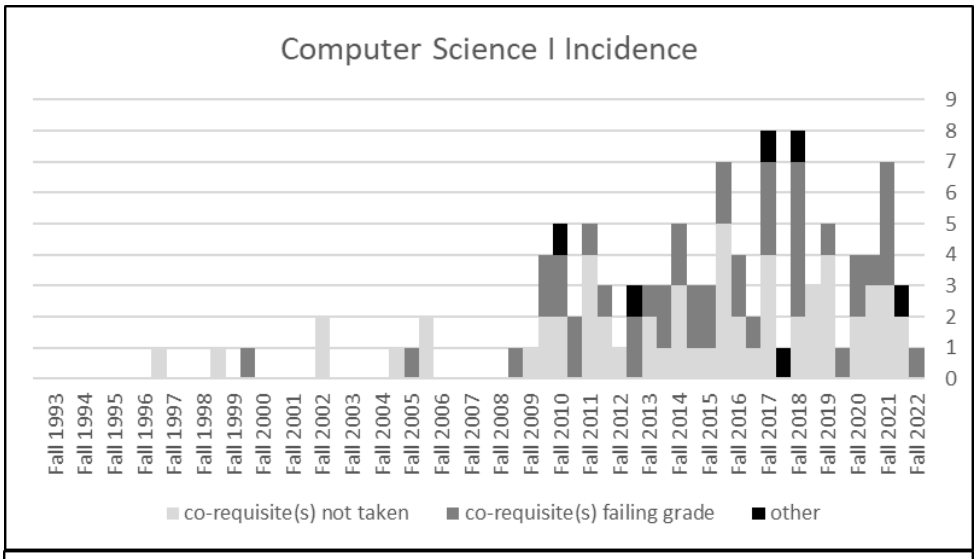
Fall Semester	Spring Semester
CS 4203 Software Engineering	CS 4233 Professional Development in CS
CS Group B Elective	CS Group B Elective
Minor Course	Minor Course
Free Electives (6 hours)	Free Elective (5 hours)
Total Hours 15	**Total Hours 14**

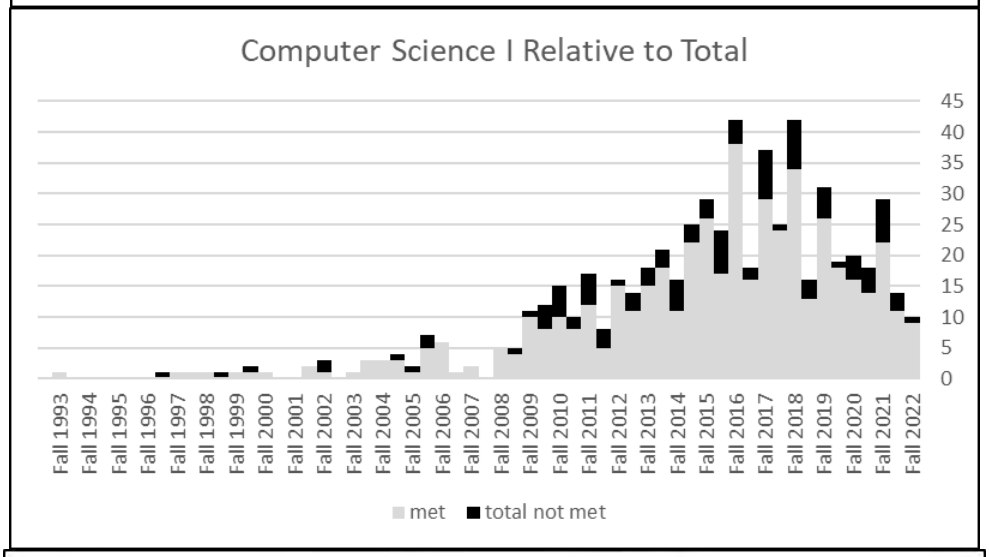
Total Degree Plan Hours 124

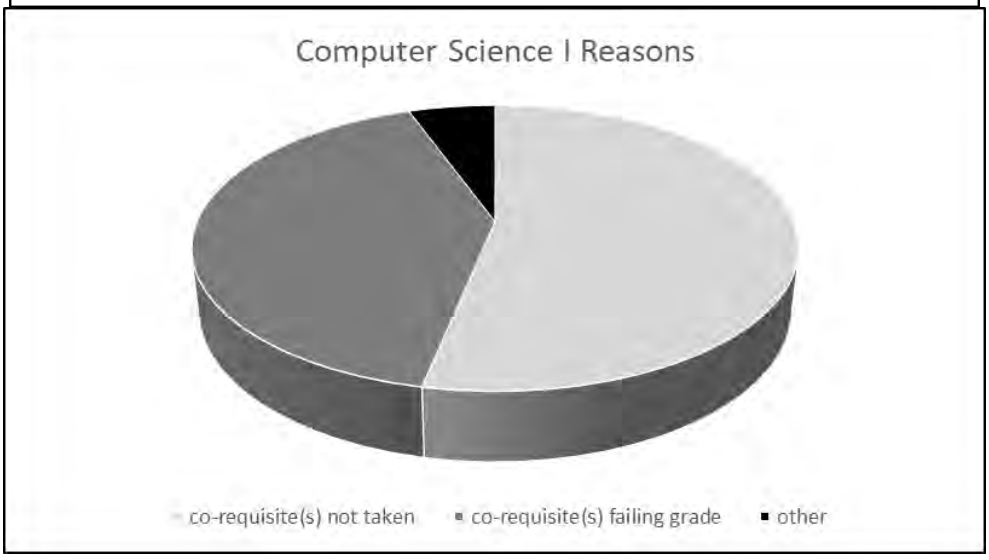
*** See current catalog "General Requirements" for selection.
Note: Courses which *may* be offered during the fall and spring based on need include CS 2014 and CS 2163.

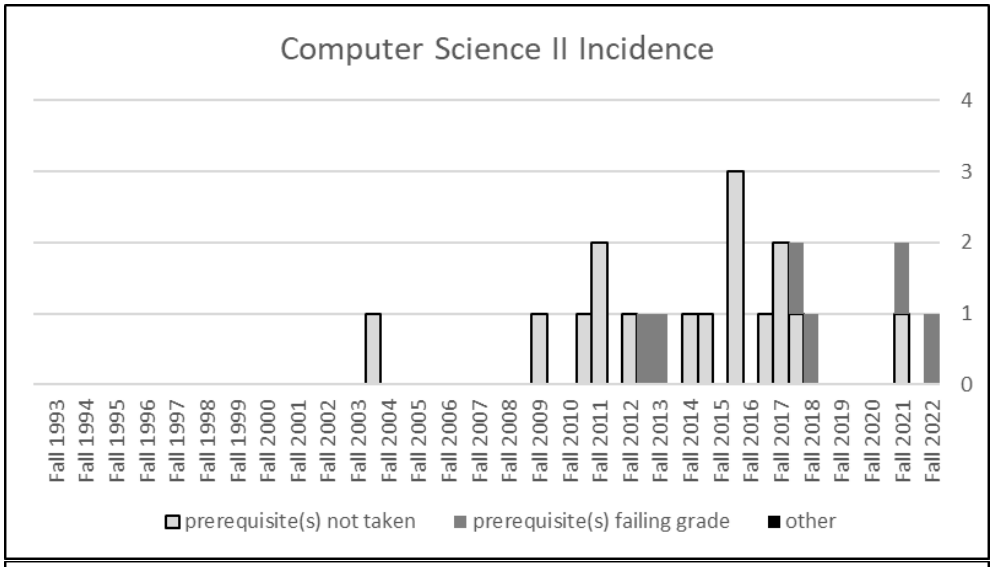
June 2021

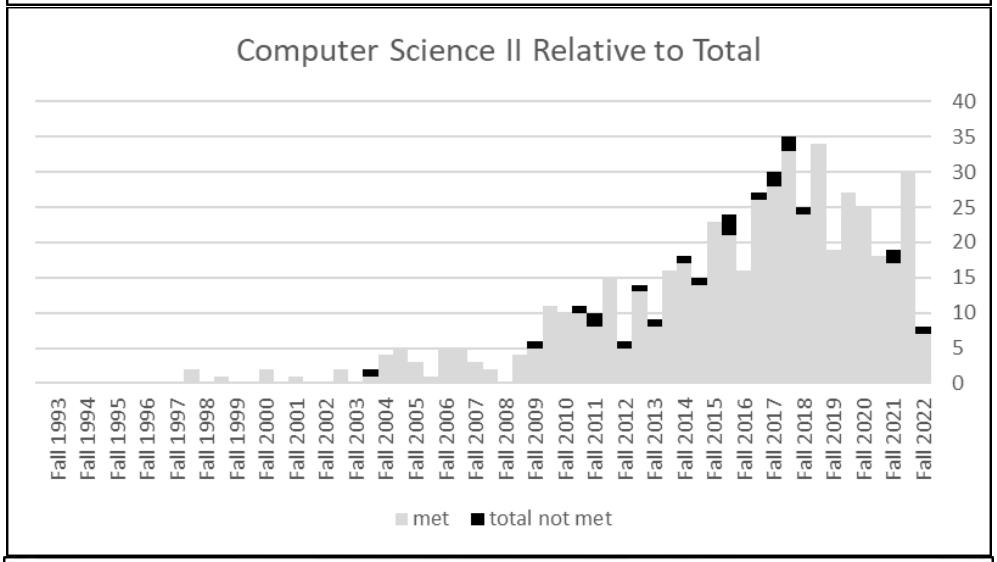
Appendix C – Results by Course

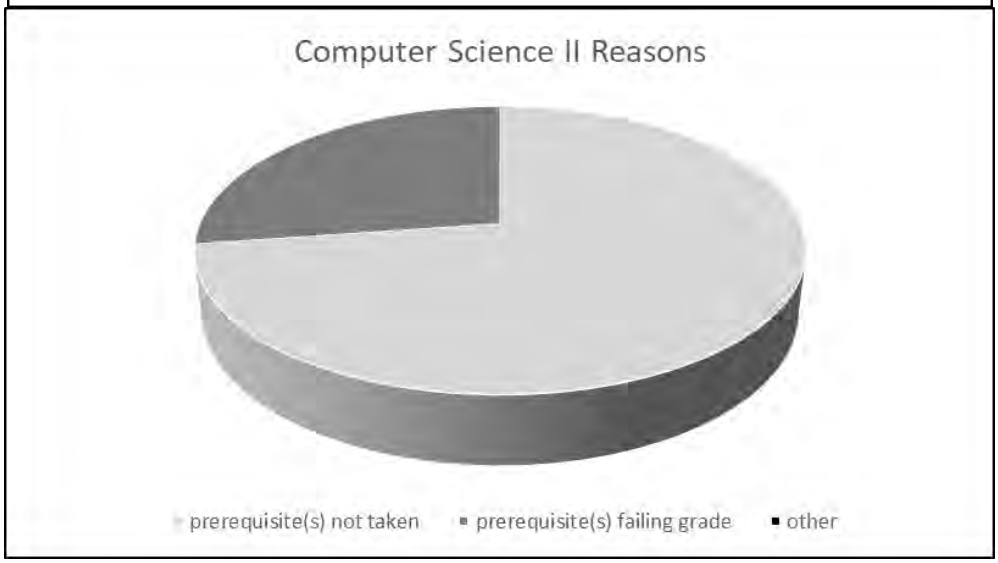


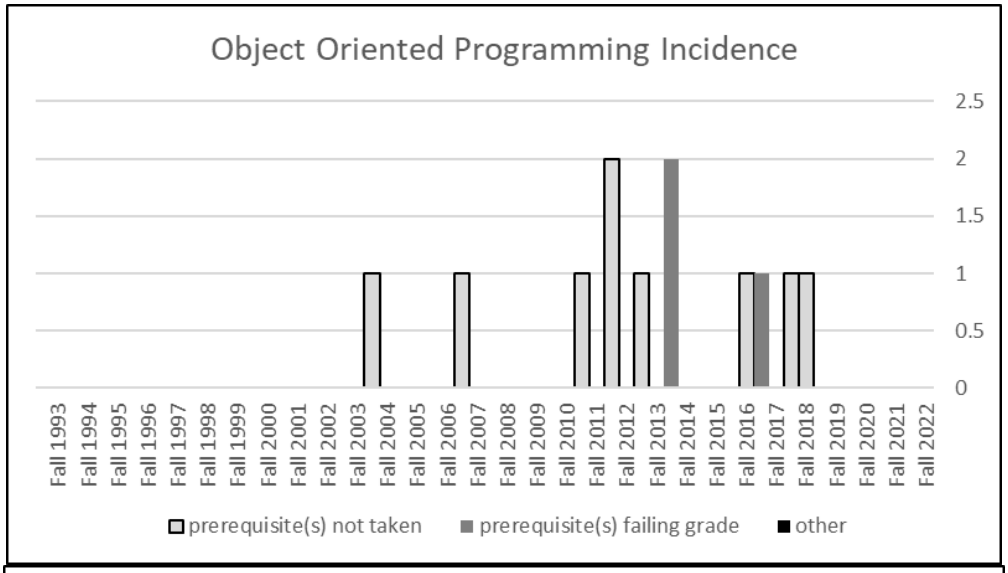


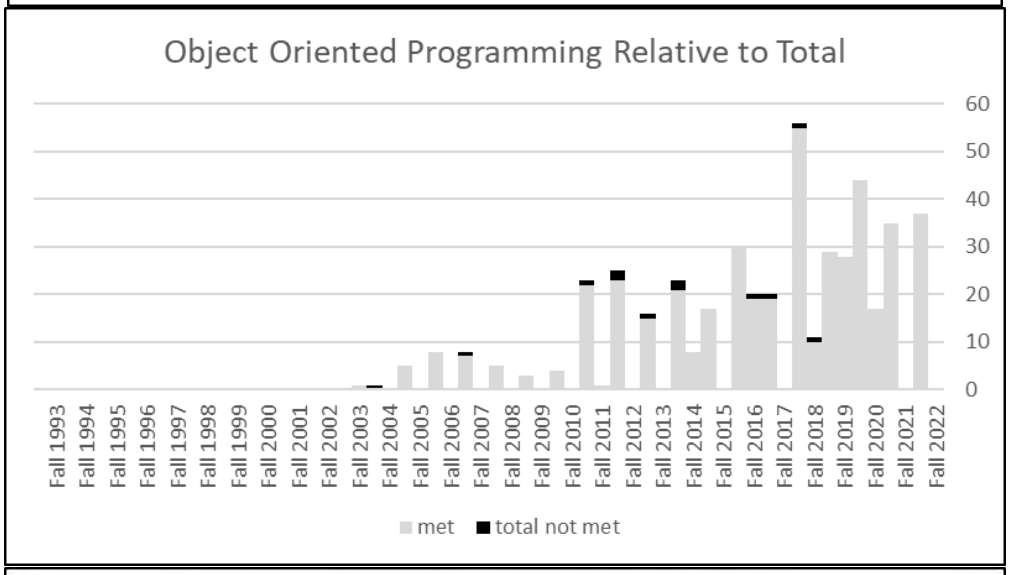


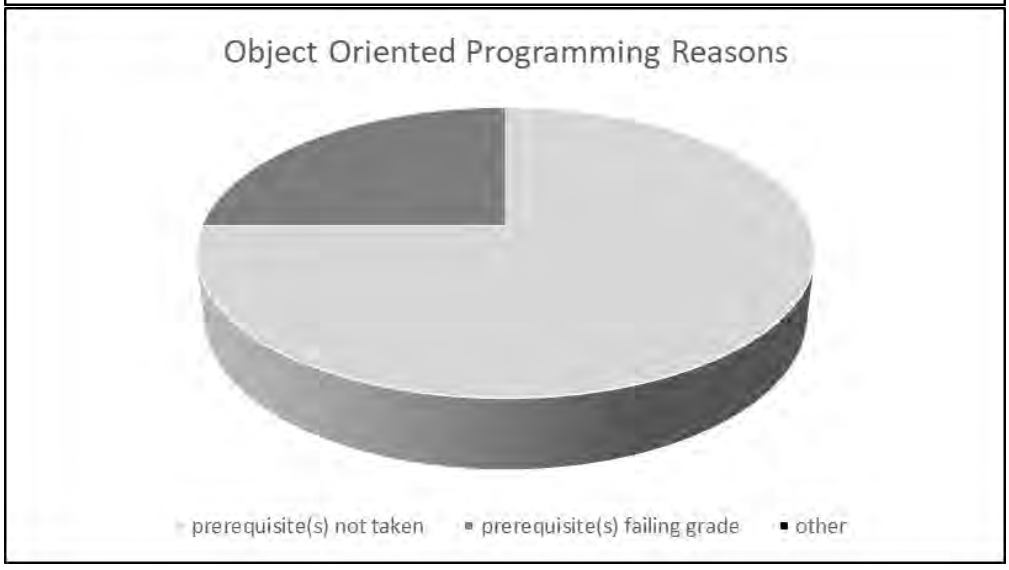

Computer Science II Incidence
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other

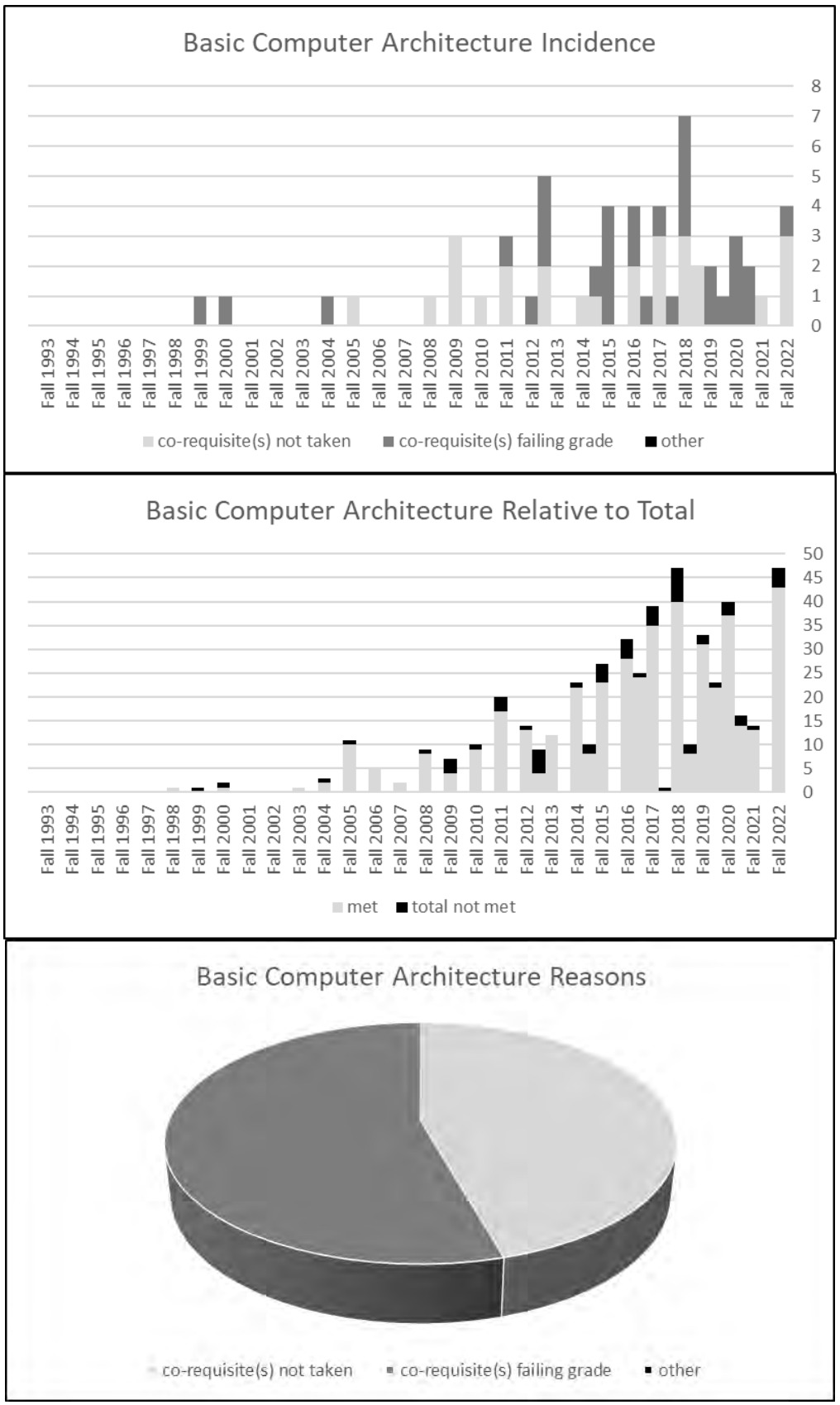

Computer Science II Relative to Total
40
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met

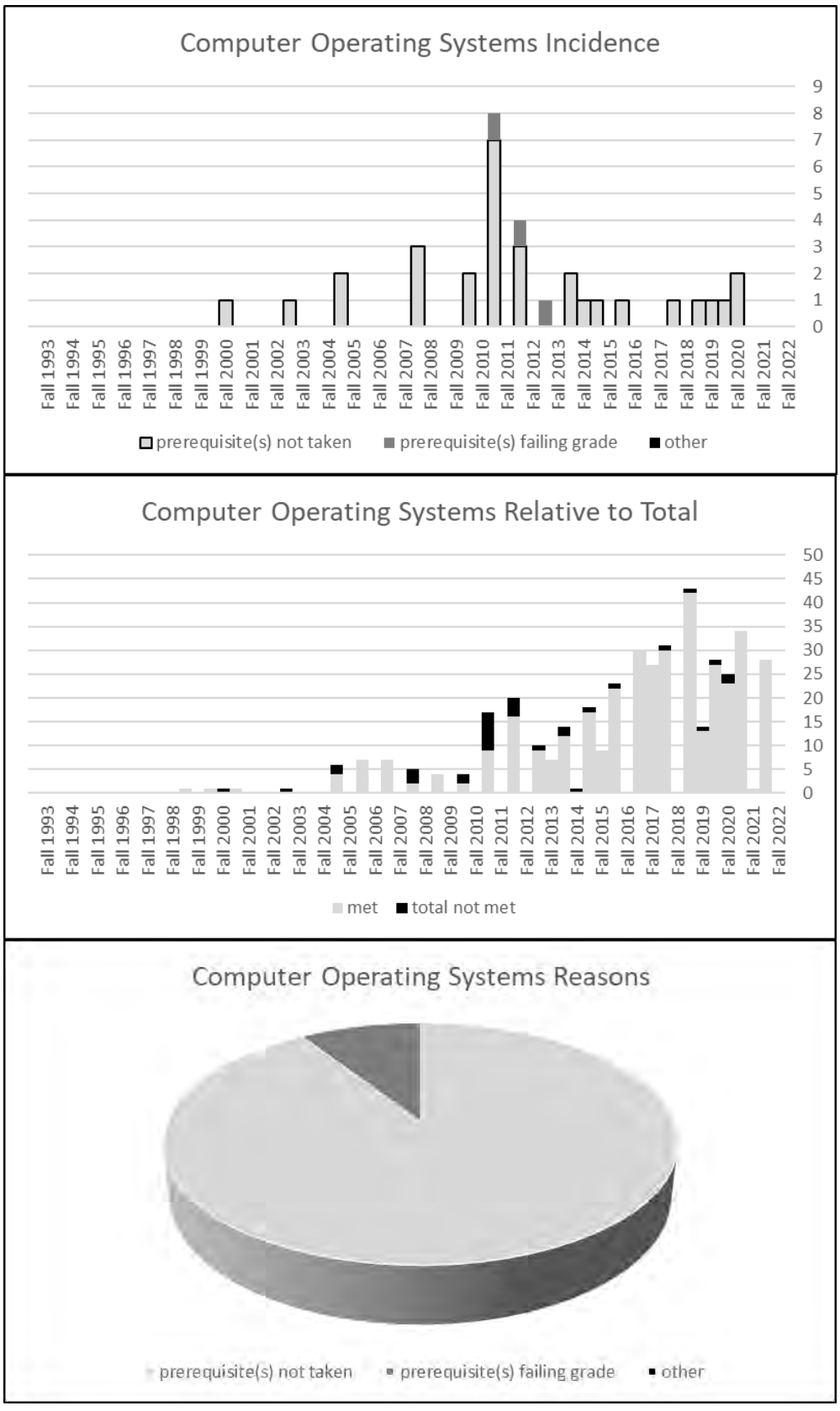

Computer Science II Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other

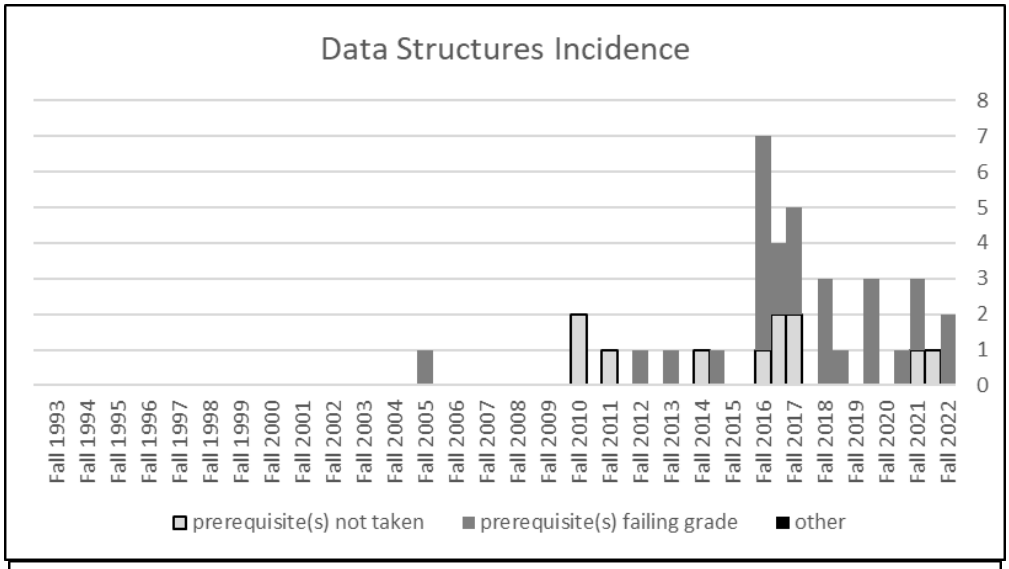

Object Oriented Programming Incidence
2.5
2
1.5
1
0.5
0
Fall 1993 Fall 1994 Fall 1995 Fall 1996 Fall 1997 Fall 1998 Fall 1999 Fall 2000 Fall 2001 Fall 2002 Fall 2003 Fall 2004 Fall 2005 Fall 2006 Fall 2007 Fall 2008 Fall 2009 Fall 2010 Fall 2011 Fall 2012 Fall 2013 Fall 2014 Fall 2015 Fall 2016 Fall 2017 Fall 2018 Fall 2019 Fall 2020 Fall 2021 Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other

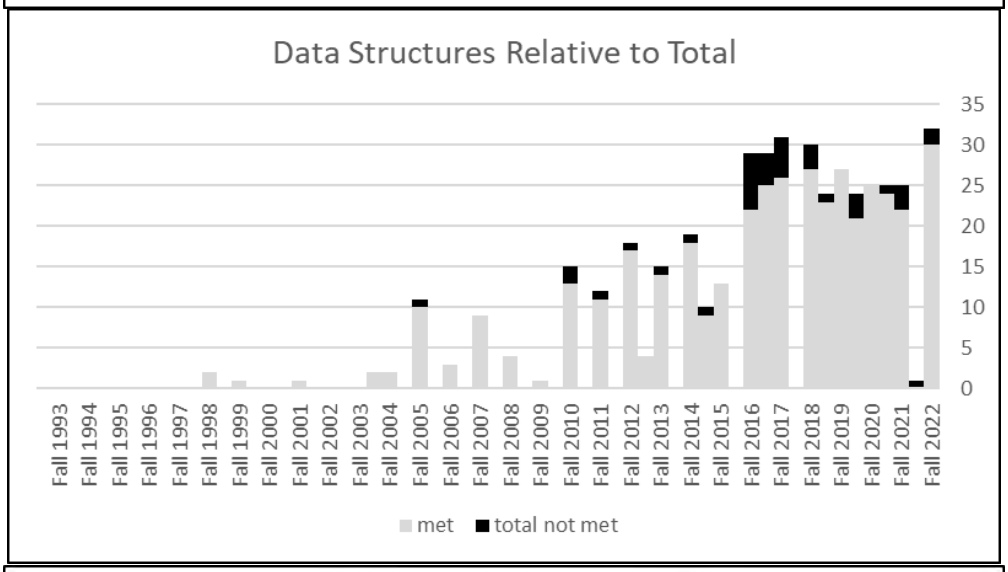

Object Oriented Programming Relative to Total
60
50
40
30
20
10
0
Fall 1993 Fall 1994 Fall 1995 Fall 1996 Fall 1997 Fall 1998 Fall 1999 Fall 2000 Fall 2001 Fall 2002 Fall 2003 Fall 2004 Fall 2005 Fall 2006 Fall 2007 Fall 2008 Fall 2009 Fall 2010 Fall 2011 Fall 2012 Fall 2013 Fall 2014 Fall 2015 Fall 2016 Fall 2017 Fall 2018 Fall 2019 Fall 2020 Fall 2021 Fall 2022
met
total not met

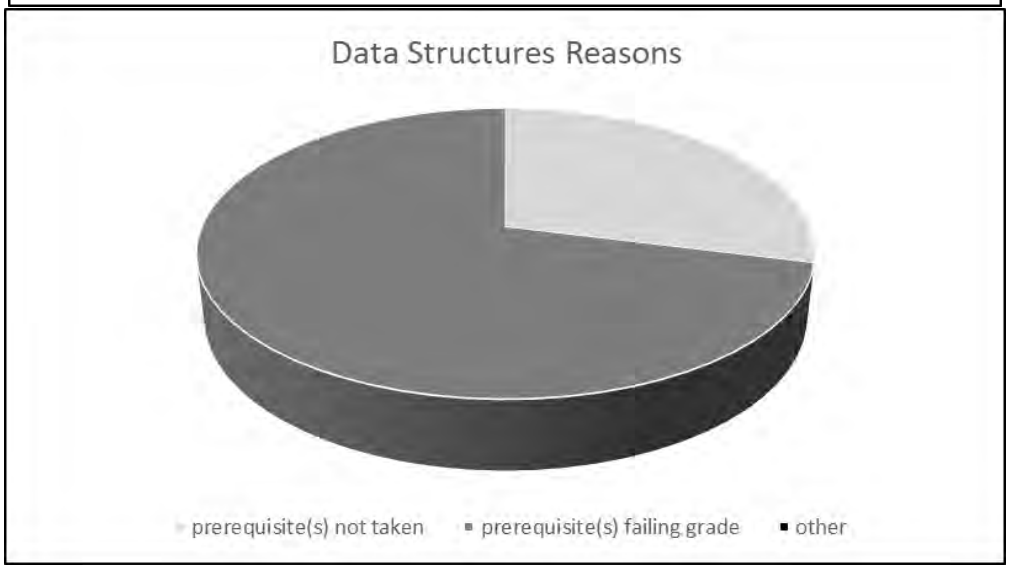

Object Oriented Programming Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other

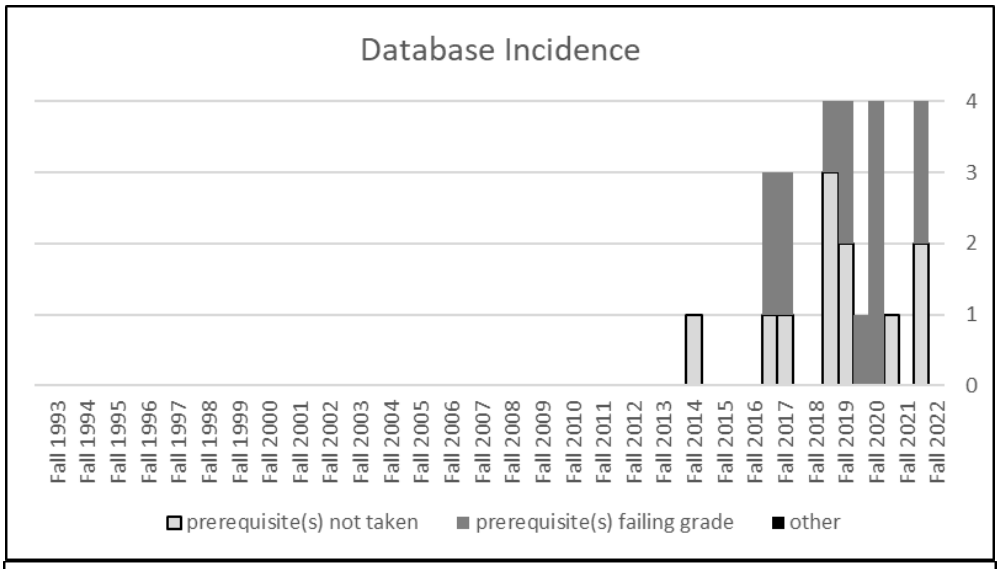

Basic Computer Architecture Incidence
8
7
6
5
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
co-requisite(s) not taken
co-requisite(s) failing grade
other
Basic Computer Architecture Relative to Total
50
45
40
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met
Basic Computer Architecture Reasons
co-requisite(s) not taken
co-requisite(s) failing grade
other

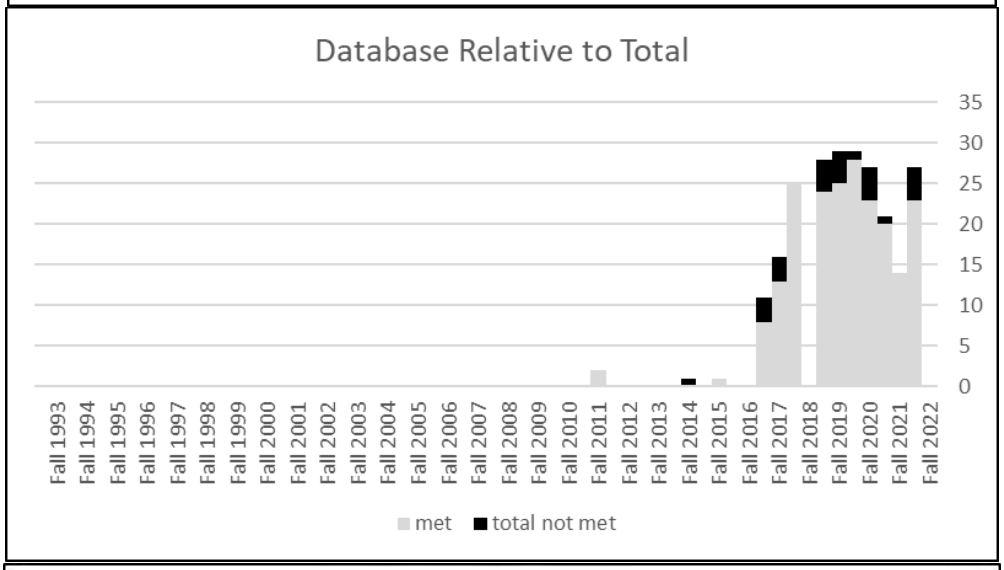

Computer Operating Systems Incidence
9
8
7
6
5
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other
Computer Operating Systems Relative to Total
50
45
40
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met
Computer Operating Systems Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other

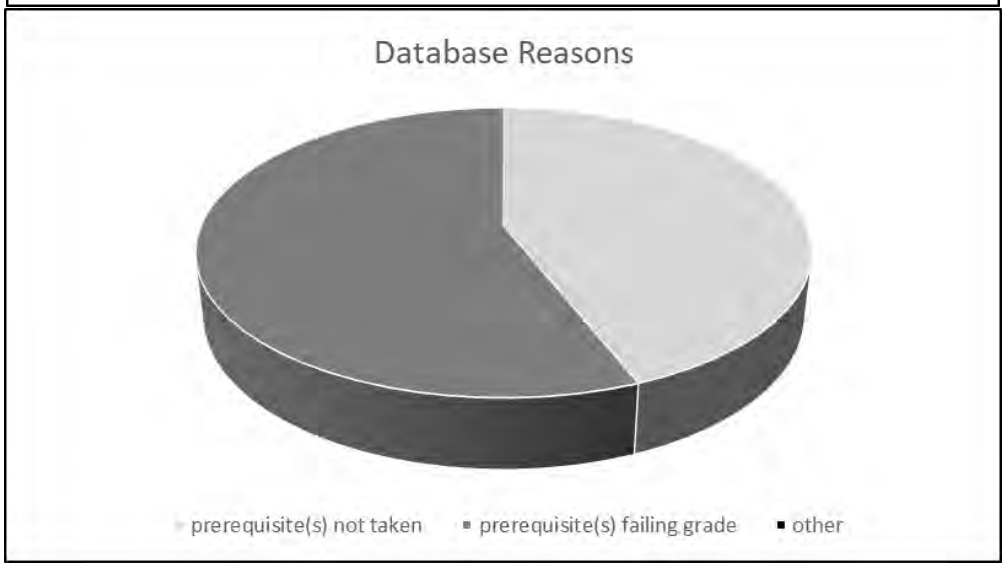

Data Structures Incidence
8
7
6
5
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other

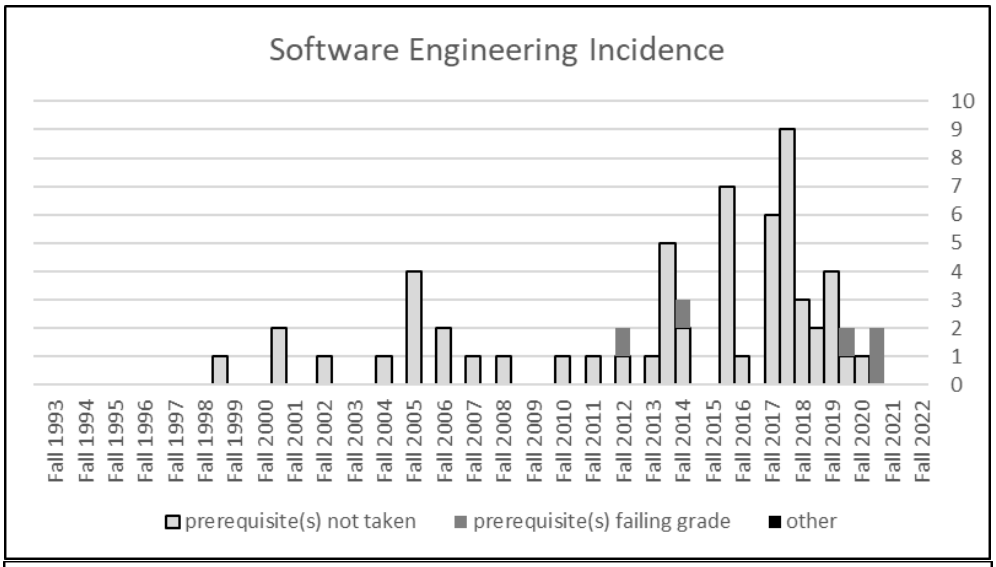

Data Structures Relative to Total
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met

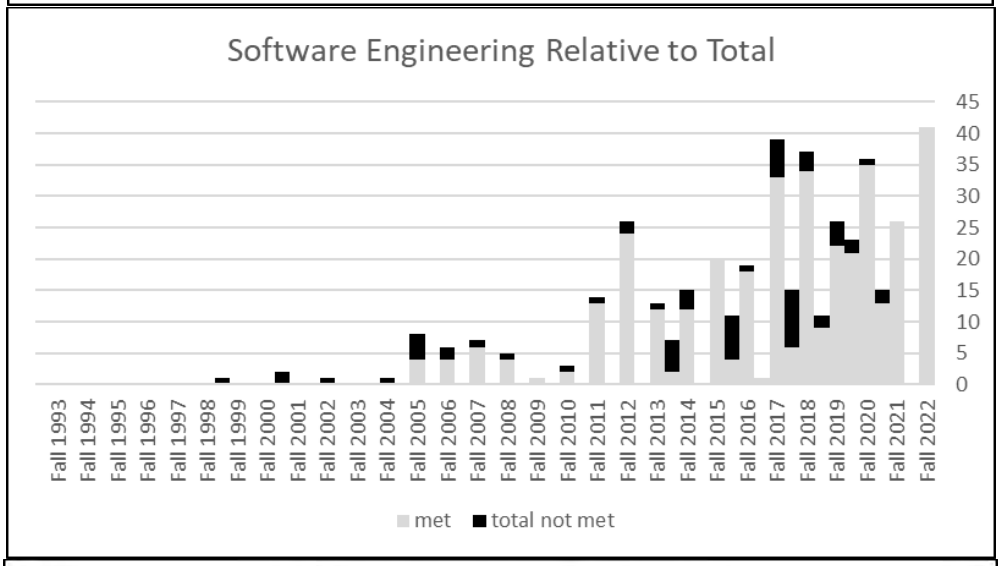

Data Structures Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other

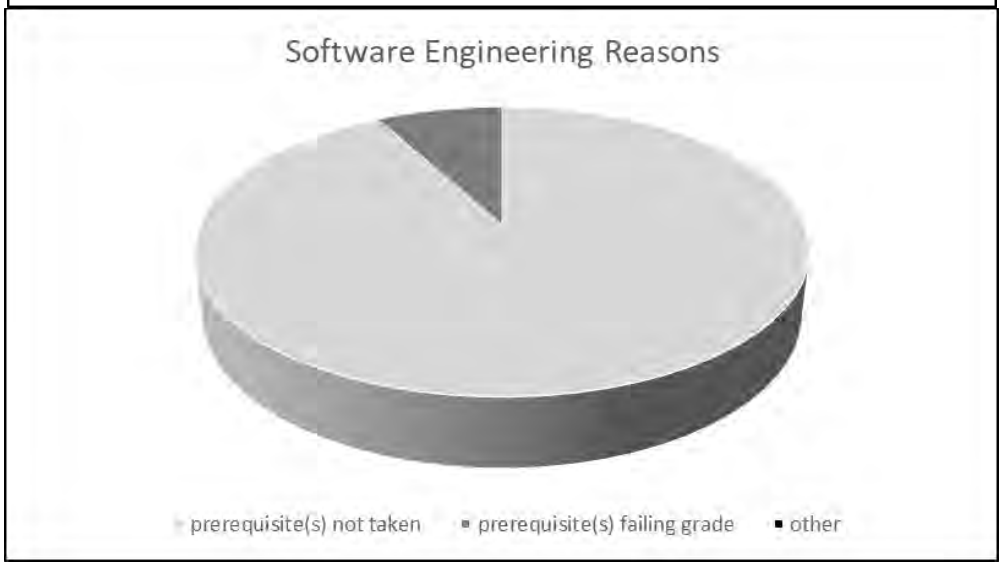

Database Incidence
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other

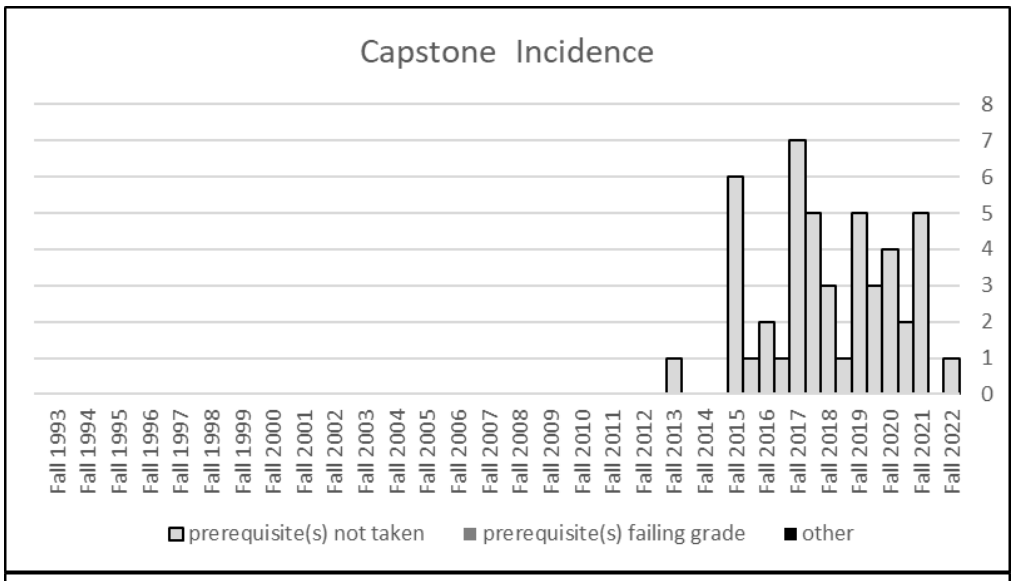

Database Relative to Total
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met

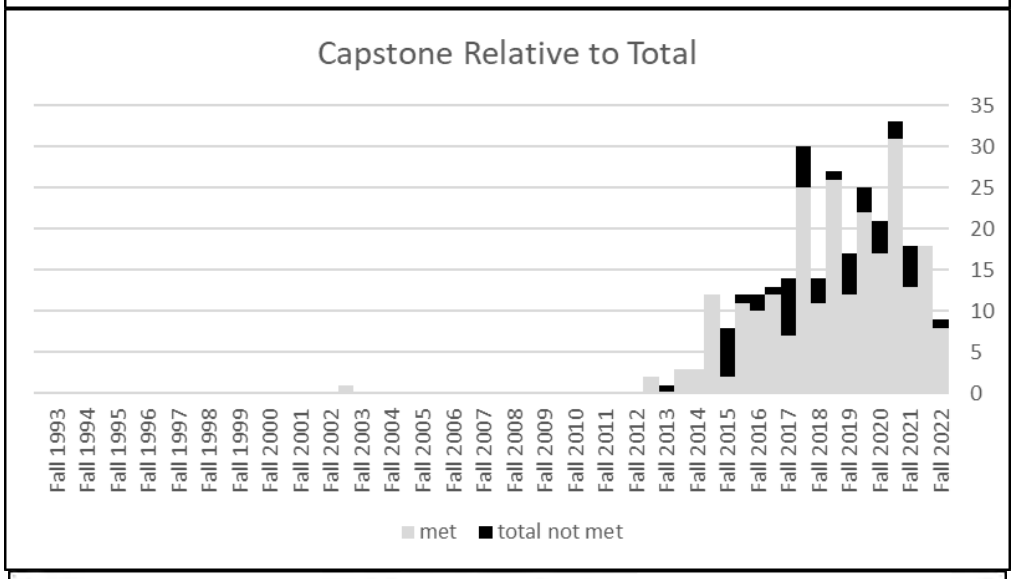

Database Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other

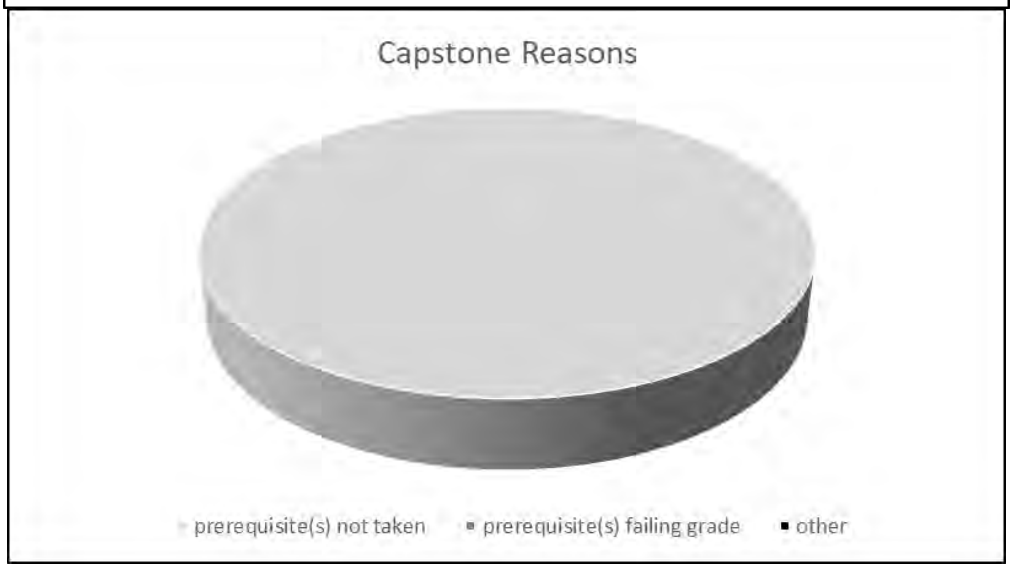

Software Engineering Incidence
10
9
8
7
6
5
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other


Software Engineering Relative to Total
45
40
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met


Software Engineering Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other


Capstone Incidence
8
7
6
5
4
3
2
1
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
prerequisite(s) not taken
prerequisite(s) failing grade
other


Capstone Relative to Total
35
30
25
20
15
10
5
0
Fall 1993
Fall 1994
Fall 1995
Fall 1996
Fall 1997
Fall 1998
Fall 1999
Fall 2000
Fall 2001
Fall 2002
Fall 2003
Fall 2004
Fall 2005
Fall 2006
Fall 2007
Fall 2008
Fall 2009
Fall 2010
Fall 2011
Fall 2012
Fall 2013
Fall 2014
Fall 2015
Fall 2016
Fall 2017
Fall 2018
Fall 2019
Fall 2020
Fall 2021
Fall 2022
met
total not met


Capstone Reasons
prerequisite(s) not taken
prerequisite(s) failing grade
other