# 2022-2023 Assessment Report for Department: *Computer Science & Engineering -draft*

**General Education Core Curriculum Area:**

**Undergraduate Major:** *Computer Science*

**Department Mission Statement:**
*The mission of the Computer Science Program is to produce computer science graduates who, trained in the design, implementation, and analysis of computational systems and skilled in technical communication, will contribute towards the advancement of computing science and technology.*

**Program Outcomes (*a.k.a.* *Student* Outcomes)**:
Owing to changes in ABET-CAC requirements, we altered our program (student) outcomes. The new outcomes are the following.

By the time of their graduation, the undergraduate academic program in Computer Science should enable our graduates to

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions;
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline;
3. Communicate effectively in a variety of professional contexts;
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles;
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline; and
6. Apply computer science theory and software development fundamentals to produce computing-based solutions.

*Note*[1]: The above program / student outcomes are ABET outcomes.

[1] This addresses a comment we received from a past review.

## Curricular Map*:*

Each numeric entry in the matrix below (between 1 and 3) represents the relative weight / contribution of a required course (row) towards a program / student outcome (column). *Note*[2]: This is the *final* matrix. It was obtained after trimming entries with smaller contributions ( = 1) in an *initial* matrix in order to keep the assessment effort manageable. This is why there are no 1-entries and the row for CSE 101 is blank.

| Required Courses | Program/Student Outcomes | | | | | |
|---|---|---|---|---|---|---|
| | 1.<br>Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions | 2.<br>Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline | 3.<br>Communicate effectively in a variety of professional contexts | 4.<br>Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles | 5.<br>Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline. | 6.<br>Apply computer science theory and software development fundamentals to produce computing-based solutions |
| CSE 101 | | | | | | |
| CSE 113 | | **2** | | | | |
| CSE 122 | | **2** | | | | |
| CSE 213 | | **2** | | | | |
| CSE 221 | | **2** | | | | |
| CSE 222 | | **2** | | | | |
| CSE 241 | | | | | | **2** |
| CSE 324 | **3** | **3** | | | | |
| CSE 325 | **3** | | | | **3** | |
| CSE 326 | | | **3** | | **3** | **3** |
| CSE 331 | **3** | | **2** | | | |
| CSE 342 | | | | | | **3** |
| CSE 344 | | | | | | **3** |
| CSE 353 | **2** | **3** | | | | |
| CSE 382 | | | **3** | **3** | | |
| CSE 423 | **3** | **3** | | | | **3** |

[2] [2] This addresses a comment we received from a past review.

**Our process**:

- The above curricular map was obtained by first considering all courses and assigning weights between 1 and 3 (inclusive) to reflect the strength of their contribution, with the following interpretation.

| *Contribution* | *Interpretation* |
|---|---|
| 1 | Introductory / preliminary |
| 2 | Reinforcement / extension / application |
| 3 | Major component |

  Next, it was pruned keeping only the required courses, and then pruning it further by eliminating weights of 1 and 2 unless one of three criteria (omitted here) were met.
- For each Program (/Student) Outcome, we obtain a number between 1 and 4 through a weighted sum of scores from contributing courses as per the curricular map shown above. Our acceptance threshold for each student outcome is 3.0.
- The scores from contributing courses come from the respective instructors who are required to submit an assessment report for each offering of such courses. This report outlines the relation between the scores and the course learning outcomes.
- The report also provides comments regarding successful strategies and plans for future modifications. While we have a numeric acceptance threshold, the instructors' comments are always important.
- The numeric score for the $j^{th}$ student outcome is a normalized weighted sum

$$Score\_Outcome_j = \frac{\sum_i (n_{ij} * s_{ij})}{\sum_i n_{ij}}$$

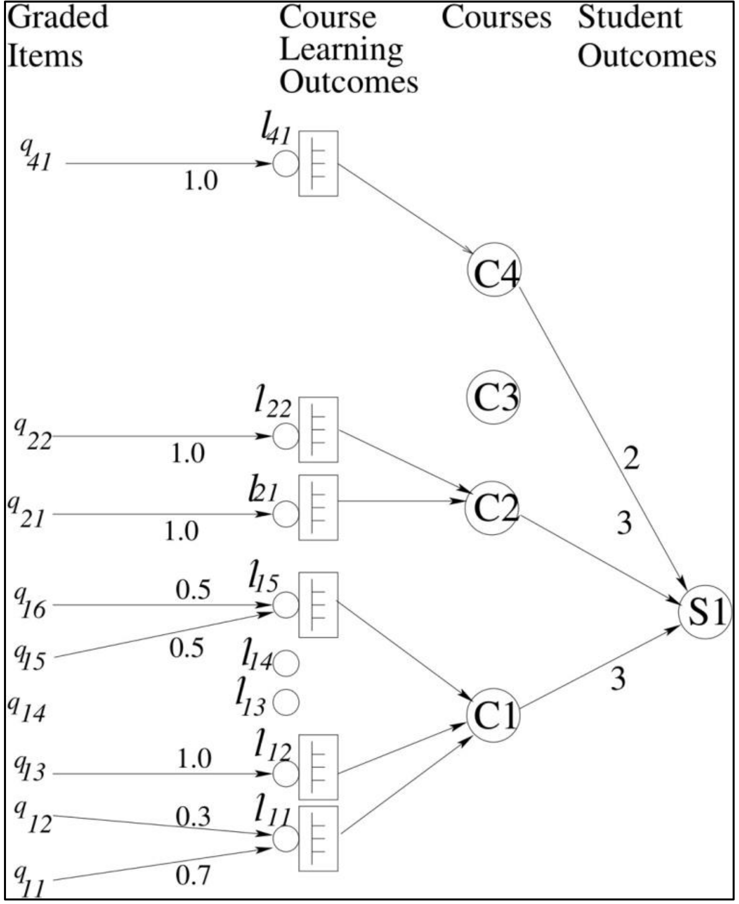
where the weights $n_{ij}$ are the non-zero entries in the column for student outcome *j* in the curricular map, and each value $s_{ij}$ is a score that comes from the assessment of the $i^{th}$ course specifically for the $j^{th}$ Student outcome. For example, the curricular map shows that Student outcome 3 (technical communication) will be measured using three courses CSE326 Software Engineering, CSE423 Compiler Writing, and CSE331 Computer Architecture, with impact factors of 3, 3, and 2 respectively. If the numeric scores assessed by those three courses are 3, 2, and 4 respectively, then the score computed for Student outcome 5 is given by (3*3 + 2*3 + 4*2)/(3+3+2), i.e., 2.88.

We limit the score $s_{ij}$ (reported by a course *i* for a Student outcome *j*) to a number between 1 and 4 with the following interpretation.

| *Student Outcome score* | *Interpretation* | |
|---|---|---|
| 1 | Unsatisfactory | 🟥 |
| 2 | Marginal | 🟦 |
| 3 | Satisfactory | 🟨 |
| 4 | Excellent | 🟩 |

- The instructor of the $i^{th}$ course computes a score $s_{ij}$ for the $j^{th}$ Student outcome as follows.

*Example*: computing the score for Program/Student Outcome S1 from relevant courses C1, C2, and C4.



The first step is to identify the largest disjoint set *L* of course learning outcomes corresponding to the Student outcome at hand at hand. For that set *L*,

1. The instructor decides on a performance metric to interpret an average score for a course outcome as unsatisfactory, marginal, satisfactory, or excellent, resulting in the basis for a four-point scale; this takes care of variations among courses in grading, e.g., relative versus absolute, partial credit versus all-or-none grading.
2. Each course outcome $l$ in *L* is tied to a set of gradable items in the course, e.g., a project, specific questions in the final exam, a presentation, etc. The sets of items should be disjoint among learning outcomes. In the above figure, course outcome $l_{11}$ would be tied to questions $q_{11}$ and $q_{12}$.
3. Weights are assigned to these questions or items (in Figure 1, 0.7 and 0.3 for questions $q_{11}$ and $q_{12}$ respectively); using them, a formula is written to compute a normalized weighted sum from the scores for those questions or items;
4. From a table of scores of the students on those gradable items, one numeric score is computed for each student per course outcome $l$.
5. Those numeric scores are then averaged over the whole class to get one numeric score $p_l$ for each course outcome $l$.
6. Using the performance metric, a number $q_l$ is obtained by quantizing $p_l$ to a four-point scale.
7. The above is repeated for each $l$ in *L.*
8. The scores $q_l$ (in the four-point scale) are averaged over all $l$ in *L.*

The result is $s_{ij,}$ the numeric score (between 1 and 4) from course $i$ to student outcome $j$.

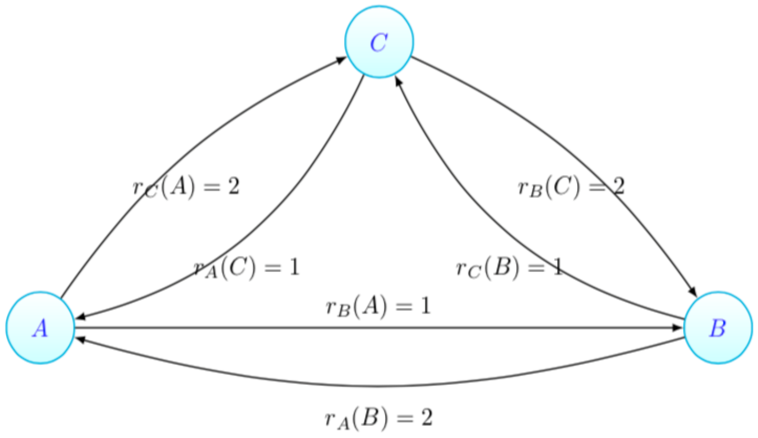
- *New assessment method for team (/group) work.*
  We found it quite difficult to measure the effectiveness of team work using direct measures. Invariably, the instrument became corrupted with the quality of the projects or deliverables. The issues differed with the content of the courses; a workable solution for all concerned courses proved elusive. Beginning in Spring 2019, we decided to use an indirect measure using surveys following a suggestion of Dr. Gloria Rogers, the facilitator of an assessment workshop organized at NMT by AVPAA Peter Mozley. We adopted three yardsticks from her examples, and constructed the following computational scheme to obtain a numeric measure consistent with our scheme.

  First, we chose three team performance yardsticks:
  - fulfilling team role duties,
  - sharing of team work, and
  - listening to teammates.

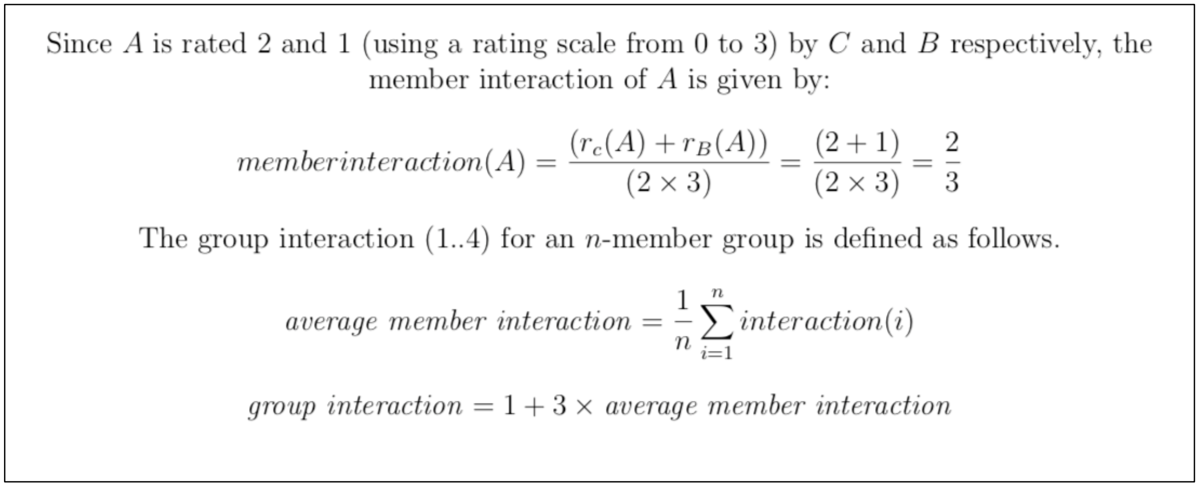
  For each of those yardsticks, for each group, the following information was obtained. Each individual in the group was rated on that yardstick by each of his/her team-mates on a *0..M* scale.
  The information (for a given yardstick and given group) is modeled using a weighted directed graph. Each member of the group is represented by a vertex; each rating of a member *u* by member *v* is represented by an out-edge from *u* to *v*, the weight on that edge being the rating (a number in *0..M*.
  - A *member interaction* metric for each team member is then defined as the sum of the weights on the out-edges from the vertex representing that member divided by the maximum possible sum, which is *(N-1)*M.*

- The *group interaction* for a group is essentially an average over all its members. But since that average is in the closed interval [0,1], it is translated to 1..4 by multiplying by 3 and adding 1 (a linear transformation).
- The *interaction* on a given yardstick is the average over all the groups in the class.
- Finally, the averages of the three yardsticks is the *class team work* measure for the entire class.

The following example outlines the initial steps of our method. Suppose the following graph captures the result of a survey of a group with members *A, B,* and *C* for a given yardstick using a rating scale of 0..3. The sum of the two out-edges from *A* is 3 (A is rated 2 and 1 by C and B respectively) while the maximum such sum is 3*(3-1) = 6 (the scenario in which both would have rated *A* at 3); thus the interaction of *A* is 3/6 = 0.5. Similarly, the interaction of *B* and *C* are also 0.5; thus the interaction of the group (on this yardstick) is 0.5; that is transformed into 2.5 on our 1..4 scale.



Example of a team with $N = 3$ members $A, B$, and $C$. $A$ rates $B$ and $C$ at 2 and 1 respectively.
$r_C(A) = C$'s rating of $A$'s interaction
$r_B(A) = B$'s rating of $A$'s interaction



Since $A$ is rated 2 and 1 (using a rating scale from 0 to 3) by $C$ and $B$ respectively, the member interaction of $A$ is given by:

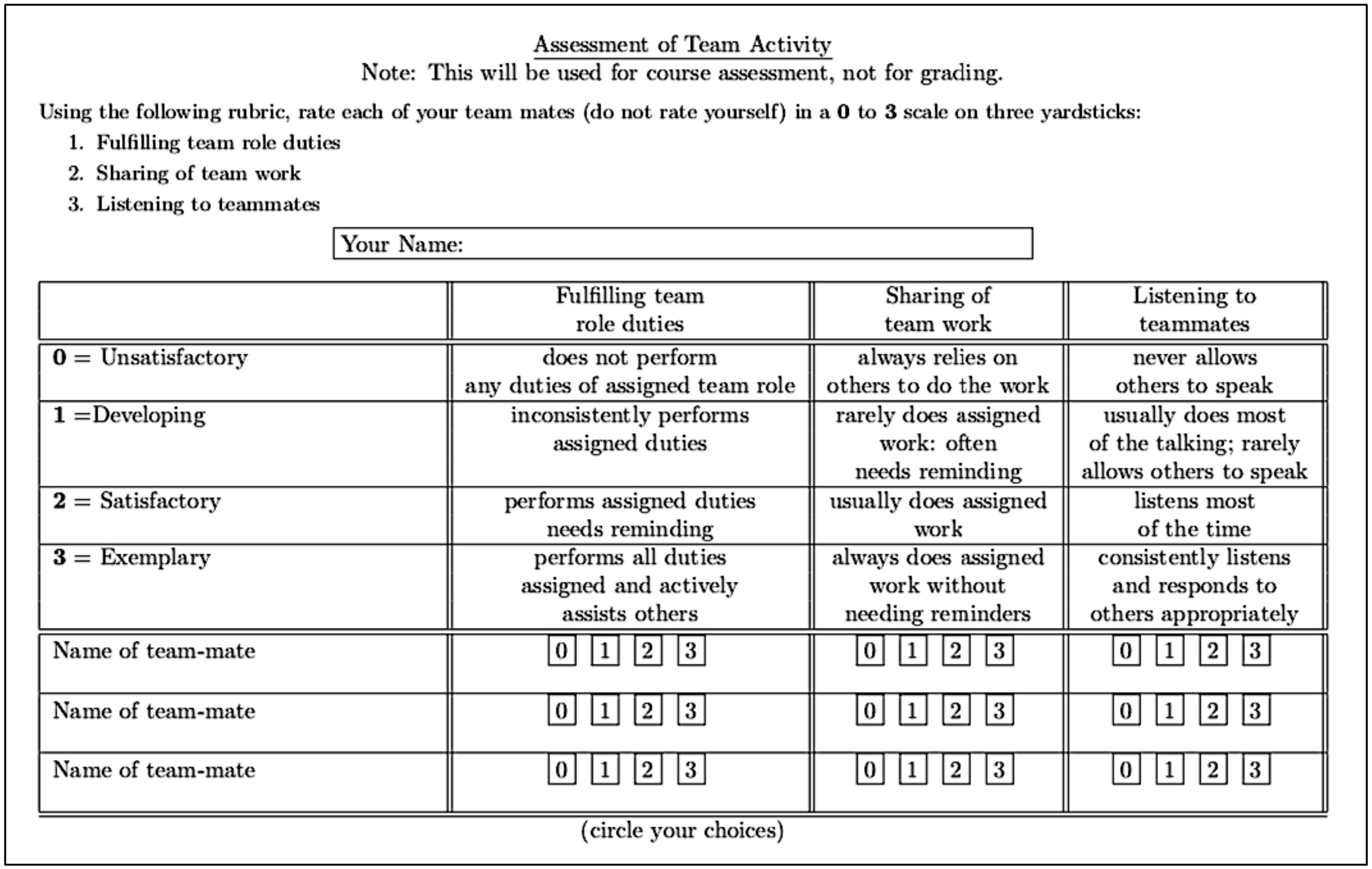
$$memberinteraction(A) = \frac{(r_c(A) + r_B(A))}{(2 \times 3)} = \frac{(2+1)}{(2 \times 3)} = \frac{2}{3}$$

The group interaction (1..4) for an $n$-member group is defined as follows.

$$average\ member\ interaction = \frac{1}{n}\sum_{i=1}^{n} interaction(i)$$

$$group\ interaction = 1 + 3 \times average\ member\ interaction$$

*Assessment Instrument for team (/group) work*: We ask each participant to rate the others in the group on a 0 to 3 scale, enter the data in a spreadsheet, and compute the interaction score for each group using the above. The students are informed that the survey results will not impact their grades.



Assessment of Team Activity

Note: This will be used for course assessment, not for grading.

Using the following rubric, rate each of your team mates (do not rate yourself) in a **0** to **3** scale on three yardsticks:

1. Fulfilling team role duties
2. Sharing of team work
3. Listening to teammates

Your Name:

| | Fulfilling team role duties | Sharing of team work | Listening to teammates |
|---|---|---|---|
| **0** = Unsatisfactory | does not perform any duties of assigned team role | always relies on others to do the work | never allows others to speak |
| **1** =Developing | inconsistently performs assigned duties | rarely does assigned work: often needs reminding | usually does most of the talking; rarely allows others to speak |
| **2** = Satisfactory | performs assigned duties needs reminding | usually does assigned work | listens most of the time |
| **3** = Exemplary | performs all duties assigned and actively assists others | always does assigned work without needing reminders | consistently listens and responds to others appropriately |
| Name of team-mate | 0 1 2 3 | 0 1 2 3 | 0 1 2 3 |
| Name of team-mate | 0 1 2 3 | 0 1 2 3 | 0 1 2 3 |
| Name of team-mate | 0 1 2 3 | 0 1 2 3 | 0 1 2 3 |

(circle your choices)

Rubric for assessment of teamwork.

#1 Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions;

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Direct measures. Graded items are weighted and linked to courses; courses are weighted, aggregated, and linked to student outcomes. Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions</td><td>Direct Measure:<br>Quantitative Assessment Procedure on CSE 324, 325, 331, 353, and 423.</td><td><table>
<tr><th><i>Course</i></th><th><i>Score</i></th><th><i>Weight</i></th><th><i>Overall</i></th></tr>
<tr><td>CSE 324 S23</td><td>4</td><td>3</td><td>3.4</td></tr>
<tr><td>CSE 325 F22</td><td>3</td><td>3</td><td></td></tr>
<tr><td>CSE 331 S23</td><td>3.1</td><td>3</td><td></td></tr>
<tr><td>CSE 353 F22</td><td>4</td><td>2</td><td></td></tr>
<tr><td>CSE 423 S23</td><td>2.9</td><td>3</td><td></td></tr>
</table></td><td>The overall scores for Student outcome 1 are higher than <b>3.0</b>, our acceptance threshold.</td></tr>
</table>

**Adjustment/Improvement**

CSE 324: The score is lower by 5% compared to last year. The orthogonality of computing OC-4 and the 4 special case students (almost 13% of the class body) issues still affecting the score. Yet, the students still show great interest in semester projects is the reason for such increase. I continued to ask the students to report on some prominent modern programming languages (not covered in class) from the literature. The resulting semester projects reports continued to be very impressive. In addition, the second semester project is to use C++ to implement a LISP interpreter. Not all students are with strong background in C++, hence I also allowed students to use other high-level languages of their interest/experience; most students are still really involved and liked final project for its practicality, reflected at the Lisp-Interpreter scoring of 85%.

CSE 325: The course’s lab software (xv6 on x86) is a bit dated and unsupported at this point, and does not run easily on all Linux distributions. A more portable lab platform is desirable.

CSE 331: I took over this course once more following the departure of the instructor who taught it in S21 and S22. I managed to cover the majority of the topics, except for loop unrolling, and I assigned all the designed assignments to the students. Some students complained about the clarity of the homework questions and lab descriptions, issues that I intend to address in the future offerings of the course.

CSE 353: More in class examples in addition to the homework exercises and quizzes in the basic electrical/light domain signaling and data encoding topics. The class TA might offer some tutoring in the low-level signals’ encodings and their analysis. More assignments on the LAN/MAN/WAN protocol design, including programming assignments, paper homework, and reports that relate to modern networking technology (e.g., WiFi, UDWDM).

CSE 423: The results on the midterm exam question regarding applied context free grammars indicate that students need additional practice with grammars. This may be a side effect of allowing them to work in groups, since only one team member might have born the brunt of making their team's parser work. A possible remediation would be to require all students to do certain homeworks, switching into a group mode only for assignments where division of labor would be especially beneficial.

#2 Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Direct measures. Graded items are weighted and linked to courses; courses are weighted, aggregated, and linked to student outcomes. Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline</td><td>Direct Measure: Quantitative Assessment Procedure (see below) of CSE 113, 122, 221, 222, 324, 353, and 423.</td><td>
<table>
<tr><th>Course</th><th>Score</th><th>Weight</th><th>Overall</th></tr>
<tr><td>CSE 113 F22</td><td>3.8</td><td>1</td><td rowspan="11">3.4</td></tr>
<tr><td>CSE 113 F22</td><td>3.0</td><td>1</td></tr>
<tr><td>CSE 113 S23</td><td>3.0</td><td>1</td></tr>
<tr><td>CSE 122 F22</td><td>3</td><td>1</td></tr>
<tr><td>CSE 122 S23</td><td></td><td>1</td></tr>
<tr><td>CSE 213 S23</td><td>3.8</td><td>2</td></tr>
<tr><td>CSE 221 F22</td><td></td><td>2</td></tr>
<tr><td>CSE 222 S23</td><td></td><td>2</td></tr>
<tr><td>CSE 324 S23</td><td>3.67</td><td>3</td></tr>
<tr><td>CSE 353 F22</td><td>3.7</td><td>3</td></tr>
<tr><td>CSE 423 S23</td><td>3</td><td>3</td></tr>
</table>
</td><td>The overall score is higher than **3.0,** our acceptance threshold.</td></tr>
</table>

**Adjustment/Improvement**

CSE 113:

For inexperienced programmers. The GUI used in the Game of Life project has reached end of life and is no longer being updated. The Game of Life project will be updated for the next semester to utilize the terminal so that students can focus more on the memory management and matrix elements of the assignment. We have seen a higher rate of students purchasing M1 Apple laptops, and in order to better serve these students with learning debugging tools, lecture material, in-class activities, and labs will reflect the addition of lldb for general debugging and leaks to show dynamic memory allocation. In Spring 23, exam performance fell from Fall 2022, despite providing similar review materials outside of class time. For Fall 2023, an additional in-class review will be held alongside the longer review held outside class sessions.

Lab 9, doubly-linked lists, was removed permanently from course content and shifted into the following course CSE 122 - Algorithms and Data Structures.

For experienced programmers.

• The course's instructional design was designed to engage students and build over their programming experience. As a result, the attendance was high. Students performed weekly group work classroom activities and labs, individual assignments, and two midterms and one final exams. The coursework and feedback helped students to develop the expected learning outcomes.

- The lowest score was for learning outcome number 2, "Recognize how to use basic data structures, including C struct, arrays, strings, and linked lists." Students found working with linked lists and strings hard due to their natural complexity in C language. Also, the linked lists were introduced at the last quarter of the course, and students took time to develop a mature understanding of using them in intermediate-level complexity programs. The recommendation is to start introducing them earlier and cooperate with the following course (CSE 122) to reinforce their understanding of them.
- The challenges were that some previous misunderstandings of programming concepts took time to address. Also, some students who have been programming for more than three years in Python found the pointers, dynamic memory allocation, and linked lists more complex than those with less or no programming experience. Also, it took a lot of work to design engaging and challenging problems for students to practice, with the difference in programming experience with different programming languages.
- The recommendations are 1) design a placement test for the students' programming skills before registering for the course, 2) divide students according to their programming level into different lab sections so the labs' materials can be customized, 3) design activities for the gifted students and connect them with a program that follows their progress during multiple courses, and this can also be supported by the students' success programs at the university.

CSE 122: There were only two exams given in the class which covered a substantial amount of course material. This had three effects, I found the exams although they covered the material, did not go into the level granularity I would like for the exam; the clusters of knowledge incorporated in the exams covered disparate topics; and students found studying for the exams difficult and. In the future, I plan to give three exams over the course of the semester each covering a general knowledge area for the course. Most lab assignments asked the students to implement a data structure, while a substantial portion of the class covered analysis of algorithms. In the future I plan to have the students perform and analysis of the algorithms they implement to strengthen their grounding in algorithm analysis.

CSE 213: The lowest score was for learning outcome number 1, " Recognize object-oriented concepts that include objects, classes, inheritance, instances, methods, interfaces, packages, encapsulation, and polymorphism." The students found that developing programs was easier than the theoretical concepts of object-oriented models. The recommendation is to add reading tasks with quizzes to the coursework.

CSE 221: There were only two exams given in the class which covered a substantial amount of course material. This had three effects, I found the exams although they covered the material, did not go into the level granularity I would like for the exam; the clusters of knowledge incorporated in the exams covered disparate topics; and students found studying for the exams difficult. In the future, I plan to give three exams over the course of the semester each covering a general knowledge area for the course. The homework assignments were designed to address the topics of each chapter of the textbook. Some of the chapters cover the topics in detail only a portion of the material was incorporated in the homework assignments. In the future I plan to have more homework dealing with computer architecture
to give the students a better grounding in this area.

CSE 222: There should be three, instead of two, exams for this course, covering the topics included in Course Learning Objectives. The grading of exams and homework should be more strict. Homework and exams should be more challenging for the students.

CSE 324: (from three course outcomes mapped to this program outcome)  I started going over introductory material from earlier classes (CSE113&122&213) myself in my class (though it affected the class timeline of covering topics), yet showed the showed increase in the students' comprehension of the subjects related to outcome-1. In addition, I also went over a much-needed concepts of MATH proofs & abstractions concepts from other MATH/CS classes. I will continue to provide more examples (as I did last year) from well known languages such as C, C++, Java, FORTRAN, etc, about *tradeoffs* of different language design factors. Class discussion involving most of the students helped very much. In addition, offering extra points for students who answer critical questions about language design factors helped to increase the students awakness to the lectures; students showed better understanding of the covered lecture subjects. I plan to keep the same approach for more improvement. The same difficulty issue of such evaluation score is that the related outcome is not very "orthogonal" in the assigned class works (exams/quizzes' questions, HWs, Projects) to be easily labeled as outcome-3. Hence, due to its implicit involvement and mixing with other class-outcomes, it requires a special way of capturing/extracting its score (I did my best!). Moreover, the aforementioned (in outcome-1) factors affected the students' performance, especially COC 3 which covers the entire language domain paradigms.

CSE 353: More on the most advanced internetworking protocols, especially over the wireless networking (wireless TCP for WSNs) and fiber optics domains (e.g., examples are packet over SONET, and UDWDM). I plan to continue coverage of network security and assign homeworks in the subject. More coverage of security fundamentals to break down the subject complexity and possible inclusion of security coding in the class final project.

CSE 423: After a past concern that only ~60% of students finished and submitted a working compiler at semester's end, in Spring 2023 the course was revised to do group team projects. The students were asked to write a compiler for a small subset of Python, a popular language with some strange lexical and syntax rules such as indentation-based syntax for which the class had to figure out appropriate Flex regular expressions, Bison context free grammars, and C code as-needed.

The use of group projects appears to have helped some more weaker students to achieve satisfactory homework grades. Unfortunately, this does not tell us anything about whether they learned more or just enjoyed the free ride. Judging from final exam grades, it seems that they may not have learned as much. The instructor remains torn about whether to stick with group projects, or provide more assistance, or to request more programming skills be taught in prerequisite courses. At a previous university, some improvements in compiler writing were achieved by introducing compiler tools (flex and bison) in the principles of programming language course that is a prerequisite. At NMT this would be CSE 324, and the compiler tools in question could be taught as part of a course module on the declarative paradigm.

#3 Communicate effectively in a variety of professional contexts

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Direct measures. Graded items are weighted and linked to courses; courses are weighted, aggregated, and linked to student outcomes. Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>3. Communicate effectively in a variety of professional contexts</td><td>Direct Measure: Quantitative Assessment Procedure (see below) on CSE 326, and 331.</td><td><table>
<tr><th><i>Course</i></th><th><i>Score</i></th><th><i>Weight</i></th><th><i>Overall</i></th></tr>
<tr><td>CSE 326 S23</td><td>3.5</td><td>3</td><td rowspan="3">3.6</td></tr>
<tr><td>CSE 331 S23</td><td>4</td><td>2</td></tr>
<tr><td>CSE 382</td><td>3.5</td><td>3</td></tr>
</table></td><td>The overall score is higher than <b>3.0,</b> our acceptance threshold.</td></tr>
</table>

| **Adjustment/Improvement** |
|---|
| |
| CSE 326: The overall assessment results for this course in 2023 were generally consistent with previous years, although there was a slight decline in the class average scores for CLO3, CLO5, and CLO6 compared to previous years. Despite the ongoing improvement in student performances in object design and testing, achieved through measures such as increased hands-on homework assignments and constructive feedback, it is evident that additional actions may be necessary to elevate the CLOs to an excellent level. To address this, several action items are being considered, including the utilization of the latest tools and development frameworks for object design and testing, as well as inviting guest speakers from the industry to share their experiences and insights.<br>CSE 331: None.<br>CSE 382: To improve the scores on 4b (Understand and communicate digitally), I plan to make new recordings to share with the students on organization, design, and creativity in making persuasive multimedia content. These resources will provide additional instruction without taking more class time. |

#4 Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Direct measures. Graded items are weighted and linked to courses; courses are weighted, aggregated, and linked to student outcomes. Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>4. [<b>Applications:]</b> exposure to one or more computer science application areas;</td><td>Direct Measure: Quantitative Assessment Procedure (see below) on CSE 382.</td><td><table><tr><th><i>Course</i></th><th><i>Score</i></th><th><i>Weight</i></th><th><i>Overall</i></th></tr><tr><td>CSE 382 S23</td><td>3.7</td><td>4</td><td>3.7</td></tr></table></td><td>The overall score is higher than <b>3.0,</b> our acceptance threshold.</td></tr>
</table>

| **Adjustment/Improvement** |
|---|
| |
| CSE 382: None provided |

#5 Function effectively as a member or leader of a team engaged in activities appropriate to the program’s discipline

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Indirect measure. Team members were surveyed on three yardsticks. A score for interaction effectiveness for the entire class was computed.<br>Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>5. <b>[Tech Comm:]</b> technical communication skills in written and oral form;</td><td>Direct Measure: Quantitative Assessment Procedure (see below) on CSE 325, and 326.</td><td><table>
<tr><th><i>Course</i></th><th><i>Score</i></th><th><i>Weight</i></th><th><i>Overall</i></th></tr>
<tr><td>CSE 325 F22</td><td>3</td><td>3</td><td rowspan="2">3.5</td></tr>
<tr><td>CSE 326 S23</td><td>4</td><td>3</td></tr>
</table></td><td>The overall score is higher than <b>3.0,</b> our acceptance threshold.</td></tr>
</table>

| Adjustment/Improvement |
|---|
| |
| CSE 325: The previous instructor (Tong) had not been assessing for Program Outcome 5. The course learning outcomes need to be modified to more explicitly address the team project program outcome.<br>CSE 326: See earlier comments about CSE 326 in SLO #3. |

#6 Apply computer science theory and software development fundamentals to produce computing-based solutions.

<table>
<tr><th>Student Learning Outcomes</th><th>Assessment Procedures</th><th>Assessment Results</th><th>Assurance</th></tr>
<tr><td>Learning Outcomes of the Program—Students will be able to:</td><td>Process/Instrument used:<br>Direct measures. Graded items are weighted and linked to courses; courses are weighted, aggregated, and linked to student outcomes.<br>Covered Fall 2022, and Spring 2023.</td><td>What were your findings?<br>Score range: 1 (unsatisfactory),<br>2 (marginal),<br>3 (satisfactory), and<br>4 (excellent).</td><td>Our department believes we fulfill this Learning Outcome because:<br>(state evidence in 30 words or less)</td></tr>
<tr><td>6. <b>[Theory+Dev:]</b> theory and development based solutions</td><td>Direct Measure: Quantitative Assessment Procedure (see below) on CSE 241, 326, 342, 344, and 423.</td><td>
<table>
<tr><th><i>Course</i></th><th><i>Score</i></th><th><i>Weight</i></th><th><i>Overall</i></th></tr>
<tr><td>CSE 241 F22</td><td>3</td><td>2</td><td rowspan="5">3.1</td></tr>
<tr><td>CSE 326 S23</td><td>3.67</td><td>3</td></tr>
<tr><td>CSE 342 S23</td><td>3</td><td>3</td></tr>
<tr><td>CSE 344 F22</td><td>3.8</td><td>3</td></tr>
<tr><td>CSE 423 S23</td><td>1.9</td><td>3</td></tr>
</table>
</td><td>The overall score is greater than <b>3.0,</b> our acceptance threshold.</td></tr>
</table>

**Adjustment/Improvement**

CSE 241: This year, I gave a pretest for recursion at the beginning. A substantial portion of the class needed review (a few needed introduction). I reviewed recursion over several labs. I plan to this next year as well. The case was similar for induction. I want to try out next year and consider adjusting the syllabus since reviewing basic recursion and covering everything else up to recursion on formulae and programs and induction on these structures will not be feasible. I would like for 241 prerequisites (some biology students barely remembered programming) and syllabus to be discussed in full with the theory subcommittee and the chair, to unsure I can cover enough to prepare students for the following courses. Since I was unsure of what would be retained of concepts learned in the covid year. The test performance showed that the students had not retained much of recursion. Since I only did this towards the end of the semester, I did not have time to remedy this and cover all the materials planned. Specifically, I did not have time to cover recursion and induction proofs over trees, formulae etc. We did cover recursion and induction over numbers and arrays of numbers. I plan to give a pretest earlier in the coming semesters in order to be able to address this.

CSE 326: See earlier comments about CSE 326 in SLO #3.

CSE 342: The class time was moved from 9:30 am to 8:00 am. This caused some problems to students who had arranged their work schedules at this time, and to those who do not live on campus. The main issue this year’s offering had was attendance. A substantial portion of the students who had to readjust would come to class late.

A more general issue: I noticed that when a class is systemically disrupted as above, I tend to readjust the syllabus to focus more on what would be core topics or topics that would be needed in later courses. I do cover others, but do not test students on it. I would like to discuss with the chair whether this practice may be continued.

CSE 344: Good undergraduate TAs are needed in the labs to provide personal attention to students.
- As in the year before, students were given problem sets to work on but not all of them worked on them. I remain unsure about incentive options.
- Last year, I wrote “The instructor has tried his utmost to wean students from looking up solutions to problems and encourage them to attempt to think and solve problems. The Associate VPAA has drawn our attention to websites such as chegg.com, which, for a small fee, allow access to the solutions of every problem in the text. Perhaps reducing the weight of homeworks will help.” The weights have been reduced.

- A crucial issue is that students lack preparation in proofs and logic and lack appreciation for the theoretical aspects of computer science.

- Another unresolved problem is the large number of students in the class and the lab.

CSE 423: It is conceivable that the test questions were too hard or graded too harshly, or that students were not prepared properly for their final exam. The instructor believes that the exam problems asked for appropriate knowledge but were in a different format than that to which students were accustomed. Additional in-class demonstration and practice problems in the format of the exam questions should leave students better prepared to solve these types of problems. The exact questions asked are shown below. Students were given one whole page to work each problem. They often drew somewhat of a syntax tree, with structural errors and missing details.

> 9. (20 points) Analyze the Python code fragment below. Draw syntax trees for the executable statement(s). Report what a PunY compiler’s type checker would do in order to determine whether the types were correct. Then report what the outcome of PunY type checking would be.
>
> 10. (20 points) Draw a syntax tree for the following Python code. Generate intermediate three address code (in the form of a linked list diagram) for them.

**Concluding Comments**:

Based on ABET requirements, we continue to use last year’s set of program / student outcomes and curricular map.

Overall, the 2022-2023 year was affected by covid but not as significantly as in previous years.

This report has been read and accepted by the department faculty on Jan 25, 2024.

All course assessment reports can be supplied on request.

Submitted by: Clinton Jeffery Department Chair: Clinton Jeffery Date: 1/2x/2024

Reviewed by Assessment Director/Director Signature: Date: (by 10/01)
Comments:

Reviewed by Faculty Senate Assessment Committee/Committee Chair Signature: Date: (by 11/01)
Comments:

Reviewed by Associate VP of Academic Affairs/AVPAA Signature: Date: (by 11/15)
Comments:

Submitted to Vice President of Academic Affairs/Date: (no later than 9/15)