SAINT LOUIS UNIVERSITY.

Program-Level Assessment: Annual Report

Program Name (no acronyms): Computer Science	Department: Computer Science
Degree or Certificate Level: MS	College/School: School of Science and Engineering
Date (Month/Year):	Assessment Contact: Erin Chambers

In what year was the data upon which this report is based collected? AY 2021-2022

In what year was the program's assessment plan most recently reviewed/updated? 2018

Is this program accredited by an external program/disciplinary/specialized accrediting organization or subject to state/licensure requirements?

If yes, please share how this affects the program's assessment process (e.g., number of learning outcomes assessed, mandated exams or other assessment methods, schedule or timing of assessment, etc.):

1. Student Learning Outcomes

Which of the program's student learning outcomes were assessed in this annual assessment cycle? (Please provide the complete list of the program's learning outcome statements and **bold** the SLOs assessed in this cycle.)

This year, assessment was targeted at the following outcome:

PLO 1 - Design, implement, evaluate and test a software system that meets a given set of computing requirements.

PLO 2 – Apply computer science theory, knowledge of computer systems and software development fundamentals to produce computing-based solutions.

PLO 6 – Function effectively as a member of a team in developing computing technology and solving technical problems.

2. Assessment Methods: Artifacts of Student Learning

Which artifacts of student learning were used to determine if students achieved the outcome(s)? Please describe the artifacts in detail, identify the course(s) in which they were collected, and if they are from program majors/graduates and/or other students. Clarify if any such courses were offered a) online, b) at the Madrid campus, or c) at any other off-campus location.

CSCI 5150: Computational Geometry was the only theory course offered in this academic year. It was required for any MS student who planned to graduate, and initially had about 30 students enrolled. It was assessed only on the main SLU campus, as Madrid does not offer graduate CS programs.

The planned assessment gathering was for one assignment, covering the design of a theoretical algorithm to solve a problem in planar computational geometry. The question is included below:

3. Computational biologist often compute physical properties of large molecules. One of these properties is something called the accessible surface area of the molecule. We will consider this problem in a 2-dimensional setting.

   In this problem, you are given a protein molecule P that consists of n atoms. Each atom is represented by a circular disk in the plane of radius r (see Figure 3). Let $P = \{p_1, \ldots, p_n\}$ denote the centers of these atoms. The protein molecule lives in a solution of water. A water molecule is represented by a circular disk of radius $r' > 4$ (see Figure 3a).

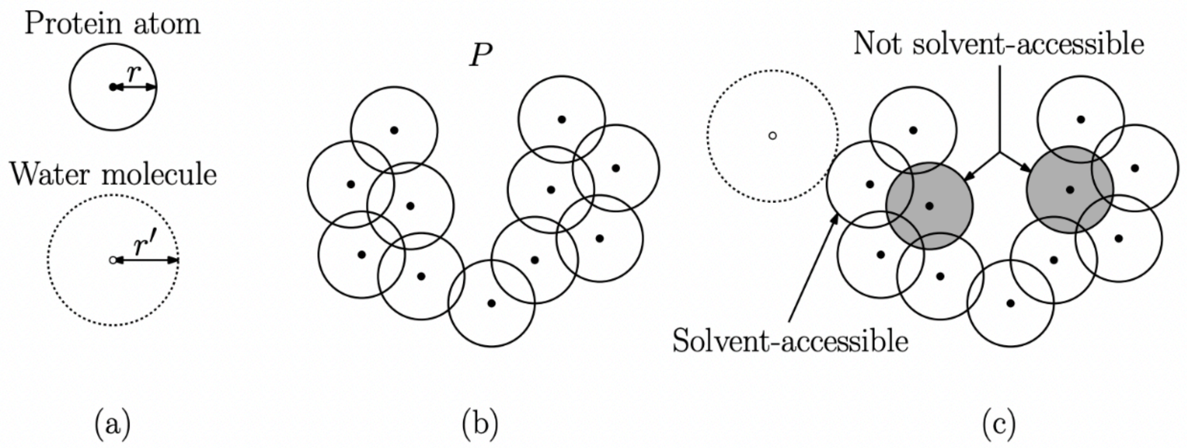

Figure 2. Solvent accessibility illustration.

   Note that water molecule and protein molecules cannot overlap. We say that an atom of P is *solvent-accessible* if there exists a placement of a water molecule that touches this atom, but does not intersect any of the other atoms of P. (In Figure 3c, only the two shaded atoms are not solvent-accessible.)

   Give an algorithm to determine all the atoms that are solvent accessible in $O(n \log n)$ time.

   (Hint: consider Voronoi diagrams here! If a molecule is solvent-accessible, can you say anything about its Voronoi cell? Can you use this structural property to quickly check if something is solvent-accessible once you know its Voronoi cell?)

The rest of the assessment was to complete a final project, implementing an algorithm or providing a demo. The assignment options specified were:

- Theoretical work: Attempt to solve an interesting, non-trivial, and preferably open theoretical problem related to computational geometry.
- Experimental: Implement and experimentally evaluate some algorithm or data structure in computational geometry. Projects of this type can include new data structures or algorithms that are hard to analyze, or that are practically efficient (in some interesting context) despite theoretical inefficiency.
- Scholarly: Write a comprehensive survey of a topic related to computational geometry. Surveys should be about 15-20 pages long, and should include a history of the topic, a description of motivating applications, a summary of known results, sketches of the most important data structures and proofs, suggestions for further research, and a thorough bibliography. Pictures are also highly encouraged to illustrate the workings!
- Creative: Make a youtube video, or interactive demo of a computational geometry topic, preferably one that doesn't have a lot of information available on the web. Or you can substantially improve the Wikipedia

articles; note this might be a challenge, for reasons I'm happy to discuss. Or come see me if you have other creative ideas! The idea is to show significant knowledge on some advanced data structures topic, but I'm prepared to be flexible for this category.

CSCI 5300: Students were asked to participate in an in-class assessment and were rewarded with a small participation credit. Students were instructed to not use any internet resources to answer assessment questions. Since the grade was based on completeness and not correctness, most of the answers likely represent students' learning outcomes.

Assessment details:

1. Pick one data structure you are using in your team project. Briefly describe the data structure and the purpose it serves in your project.
2. Analyze the choice of this data structure for the purpose it serves in terms of program efficiency, coupling, and/or cohesion.
3. What alternative data structure could you have used? Analyze if this alternative would be a better choice for your project.
4. Explain what the term "security" means in the context of software.
5. Describe what measures you would take to ensure that the software you produce is "secure".
6. State and explain what you believe is the ideal team size for a: Small project (about the size of our class project) and a medium project
7. Given your ideal team size and project requirements, explain how you would organize your team and approach the development process to deliver the required software.
8. Describe the git workflow we have utilized in this class for the team project.
9. Explain the difference between the workflow we used in this class and the workflows you have used in other situations.
10. What considerations do you need to take into account when deciding on what workflow to use?
11. Explain how we applied various development tools to assure code quality.

Two other courses were supposed to provide assessments, but were taught by adjuncts or visitors who did not complete the data as required.

3. Assessment Methods: Evaluation Process

What process was used to evaluate the artifacts of student learning, and by whom? Please identify the tools(s) (e.g., a rubric) used in the process and **include them in/with this report document** (please do not just refer to the assessment plan).

The rubric used to assess is attached.

4. Data/Results

What were the results of the assessment of the learning outcome(s)? Please be specific. Does achievement differ by teaching modality (e.g., online vs. face-to-face) or on-ground location (e.g., STL campus, Madrid campus, other off-campus site)?

About one third of the students enrolled either failed or withdrew from the course, after failing a significant amount of the work, which means data is available for only 19 students. (This represents a significant failure already, as discussed in the next section.)

For CSCI 5150:

For the HW question:

Score	Data structures	Algorithms
4	0	6
3	6	6

2	6	4
1	9	5

For the final project:

Score	Data Structures	Algorithms
4	4	6
3	3	6
2	7	4
1	5	3

For CSCI 5300:

Score	Data Structures	Computer systems: Security	Software development: Team and organization	Software development: Team and workflow
4	9	1	10	5
3	3	4	3	11
2	4	7	1	0
1	0	4	2	0

5. Findings: Interpretations & Conclusions

What have you learned from these results? What does the data tell you? Address both a) learning gaps and possible curricular or pedagogical remedies, and b) strengths of curriculum and pedagogy.

The result of this is that theory graduate courses need a major redesign. For students coming from other countries, the variability of theory background means that our assumptions about what a CS undergraduate degree will include are fundamentally flawed. We will not teach this course at a broad level to MS students, but rather plan in future years to offer general theory courses which cover more introductory material. It is also worth noting that scores were lower in the data structures category, although in some cases that is because the class was much less focused on data structures in content, and only some of the final projects incorporated them in any major way.

On software engineering, we saw reasonable scores in terms of the application of data structures, as well as software development processes. Security and systems was a weak point in this assessment, but we wish to evaluate the systems classes to see how students perform in core classes from this area.

As a result of the two classes which failed to gather assessment data, we had quite a bit of discussion about how to handle assessment when faculty are not full time at the university, which has led to further discussions of quality control in general on such courses.

6. Closing the Loop: Dissemination and Use of Current Assessment Findings

A. When and how did your program faculty share and discuss the results and findings from this cycle of assessment?

This discussion occurred in fall faculty meetings, as well as in administrator meetings with the associate dean for the new college.

B. How specifically have you decided to use these findings to improve teaching and learning in your program? For example, perhaps you've initiated one or more of the following:

Changes to the Curriculum or Pedagogies	• Course content • Teaching techniques • Improvements in technology • Prerequisites	• Course sequence • New courses • Deletion of courses • Changes in frequency or scheduling of course offerings
Changes to the Assessment Plan	• Student learning outcomes • Artifacts of student learning • Evaluation process	• Evaluation tools (e.g., rubrics) • Data collection methods • Frequency of data collection

Please describe the actions you are taking as a result of these findings.

As a result of both this year's discussion and the growth we are experiencing, we have resolved as a department to transition our assessment plan and discussion to a new model. Assessment will be conducted in area clusters, focused around the following broad areas: introductory classes, ethics, software engineering, AI/ML, theory, and systems/networking/security.

Each faculty area group is charged to meet over the course of the year, to reflect on current content, agree on an assessment plan, and report back to the faculty in department meetings in the next 1-2 years with a developed plan. Given the increasing size of the program, we expect the next 1-2 years to involve significant overall revision of course sequencing and content, as well as pedagogy, since many of these classes will transition to larger sizes in the next few years. It is likely this will necessitate that we include smaller statical samples of direct student assessment, supplemented by overall grades in the larger classes.

In addition, one of the more helpful pieces of assessment in discussion was in faculty reflections. The chair will collect individual reflections in the fall for every class taught, so that faculty can share these within areas and use them to develop improvements to the content both individually and in groups.

We will also avoid the use of adjuncts as much as possible for key assessment courses, as it is clear gaps can enter if the faculty choose not to report the data and/or do not come back to teach and share it.

Finally, in the coming year we will also try to incorporate exit interviews for all students, to gain a more big-picture and holistic view of the student experience.

If no changes are being made, please explain why.

7. Closing the Loop: Review of Previous Assessment Findings and Changes

A. What is at least one change your program has implemented in recent years **as a result of previous assessment data**?

This is no prior round of successful assessment, since the MS program began just as the COVID pandemic hit. As a result, this represents our first round of solid assessment data.

B. How has the change/have these changes identified in 7A been assessed?

N/A

C. What were the findings of the assessment?

N/A

D. How do you plan to (continue to) use this information moving forward?

N/A

IMPORTANT: Please submit any assessment tools (e.g., artifact prompts, rubrics) with this report as separate attachments or copied and pasted/appended into this Word document. Please do not just refer to the assessment plan; the report should serve as a stand-alone document. Thank you.

PLO 3 - Application of Theory, Systems, and Software Development Fundamentals

Outcomes

Graduates of the program will have an ability to...

BA-CS, BS-CS, MS-CS Apply computer science theory, knowledge of computer systems and software development fundamentals to produce computing-based solutions.

Application of Theory Fundamentals

Criterion	4: Exemplary	3: Accomplished	2: Developing	1: Beginning
Data Structures	Student can **critically evaluate** the use of data structures in real contexts and **adapt or create** data structures to accomplish or optimize problem solutions.	Given a problem statement and a data structure, the student can **implement or describe a concrete implementation** using the data structure to solve the problem.	Given a problem statement and a data structure, the student can **reason about tradeoffs** and **articulate** how the data structure solves the problem.	Given a problem statement and a data structure, the student can **describe** the data structure and **generalize** how the data structure might assist in solving the problem.
Algorithms	Student can **critically evaluate** the use of algorithms in real contexts and **adapt or create** algorithms to accomplish or optimize problem solutions.	Given a problem statement and an algorithm, the student can **implement or describe a concrete implementation** using the algorithm to solve the problem.	Given a problem statement and an algorithm, the student can **reason about tradeoffs** and **articulate** how the algorithm solves the problem.	Given a problem statement and an algorithm, the student can **describe** the data structure and **generalize** how the algorithm might assist in solving the problem.

Note: A score of zero should be given for students that do not meet the basic standard.

Notes on the above rubric

- This learning outcome evaluates the students' process of applying learned knowledge and skills to a specific problem, not necessarily the specific skills and learned knowledge itself.
- PLO3 is a broad learning outcome that applies to many courses. This rubric attempts to be general enough so that elements may be applicable to any course covered under PLO3. It is not intended to be specific to the Theory courses. For example, an Operating Systems course can include discussion of specific algorithms and data structures used in the OS, or Computer Architecture can include discussion of how the assumption of sequential execution changes the design of software algorithms from inherently parallel hardware circuit design.

Application of Computer Systems Fundamentals

Criterion	4: Exemplary	3: Accomplished	2: Developing	1: Beginning
Program Execution	Student can **critically evaluate** execution management strategies in real contexts and **adapt or create** new strategies to accomplish or optimize system goals.	Student can **implement or describe a concrete implementation** of different code execution strategies to achieve desired system-level outcomes.	Student can **reason about** how and when a system executes code to accomplish its goals. Students can **compare and contrast** different systems and explain why they manage code execution differently.	Student can **describe** how programs, processes, threads, tasklets, or other runnable code is executed on hardware in an abstract, idealized manner. Student can **describe** mechanisms and algorithms that manage computing time as a resource.
Memory and Data Mangement	Student can **critically evaluate** data management strategies in real contexts and **adapt or create** new strategies to accomplish or optimize system goals.	Student can **implement or describe a concrete implementation** of different data management strategies to achieve desired system-level outcomes.	Student can **reason about** how a system manages data storage and movement to accomplish its goals. Students can **compare and contrast** different systems and explain why they manage data differently.	Student can **describe** how data management systems (memory, cache, databases, etc.) function in an abstract, idealized manner. Student can **describe** how computer data is managed as a resource.
Networking	Student can **critically evaluate** networking strategies in real contexts and **adapt or create** new strategies to accomplish or optimize system goals.	Student can **implement or describe a concrete implementation** of different networked communication strategies to achieve desired system-level outcomes.	Student can **reason about** how distributed systems use communication to accomplish their goals. Student can **compare and contrast** different systems and explain why they manage communication differently.	Student can **describe** how network hardware and software operates in an abstract, idealized manner. Student can **describe** protocols and algorithms that manage the transfer of information between systems.
Security	Student can **critically evaluate** security strategies in real contexts and **adapt or create** new strategies to accomplish or optimize system goals.	Student can **implement or describe a concrete implementation** of different computer security strategies to achieve desired system-level outcomes.	Student can **reason about** how secure systems accomplish their goals. Student can **compare and contrast** different systems and explain why they manage security differently.	Student can **describe** how digital systems are secured in an abstract, idealized manner. Student can **describe** protocols, procedures, and algorithms that achieve security objectives and allow trust in computer systems.

Note: A score of zero should be given for students that do not meet the basic standard.

Notes on the above rubric

- This learning outcome evaluates the students' process of applying learned knowledge and skills to a specific problem, not necessarily the specific skills and learned knowledge itself.
- PLO3 is a broad learning outcome that applies to many courses. This rubric attempts to be general enough so that elements may be applicable to any course covered under PLO3. It is not intended to be specific to the Computer Systems courses. For example, the Algorithms course could incorporate elements of "Program Execution" by analyzing an algorithm's Big-O running time under two models: one where a single instruction occurs per time step (sequential execution) versus another where all possible instructions occur per time step (infinitely parallel execution). Or, the Algorithms course could incorporate elements of "Memory and Data Management" by discussing working-set-size and in-cache versus out-of-cache algorithms or in-core and out-of-core algorithms.
- This rubric attempts to hit Computer Systems concerns at a high and low level. For "Memory and Data Management" a programming course may talk about how the Java garbage collector manages memory, an architecture course may talk about how the CPU cache interacts with memory, an OS course may talk about virtual memory and paging, a database course may talk about database organization, and a security course may talk about where data is encrypted and decrypted.
- In many courses these four dimensions of computer systems will interrelate to one another, even if there are apparently one or two primary dimensions. For example, a networking or distributed systems course might talk about efficiently distributing computation and data storage across client and server, subject to the security concerns of who is trusted to do what kinds of operations.

Application of Software Development Fundamentals

Criterion	4: Exemplary	3: Accomplished	2: Developing	1: Beginning
Team and Work Organization	Student can **critically evaluate** software development strategies in real contexts and **adapt or create** new strategies to accomplish or optimize development goals.	Student can **describe a concrete implementation** of different software development strategies to achieve desired development outcomes.	Student can **reason about** how software developers accomplish their goals. Student can **compare and contrast** different organizations or models and explain why they approach development differently.	Student can **describe** how software is developed in an abstract, idealized manner. Student can **describe** policies, procedures, models, and methodologies that support software development.
Development Tools and Workflows	Student can **critically evaluate** the use of development tools and workflows in real contexts and **select or create** different tools or workflows to accomplish or optimize development goals.	Student can **use** different development tools and workflows to facilitate software development outcomes.	Student can **reason about** how different development tools and workflows accomplish their goals. Student can **compare and contrast** different tools or workflows.	Student can **describe** how development tools support work in an idealized, abstract manner. Student can **describe** how development tools and their workflows are used to support good software development.

Note: A score of zero should be given for students that do not meet the basic standard.

Notes on the above rubric

- This learning outcome evaluates the students' process of applying learned knowledge and skills to a specific problem, not necessarily the specific skills and learned knowledge itself.
- PLO3 is a broad learning outcome that applies to many courses. This rubric attempts to be general enough so that elements may be applicable to any course covered under PLO3. It is not intended to be specific to the Software Engineering courses. For example, all classes that incorporate group work can ask students to reflect on their group work organization and process, and ask what could be done differently next time. Similarly, all classes that involve programming projects can evaluate the use of Git as a software development tool as well as identifying specific processes or practices that makes that kind of work easier.