|  |
| --- |
| **CSM**  **BS Program in Computer Science**  **Department of Computer Science** |
| **Annual Assessment Report for AY 2015-16** |

|  |
| --- |
| 1. **What learning outcome(s) did you assess this year?** |
| During AY 2015-16, we have assessed the following learning outcome:   * Demonstrate solid skill of problem solving by applying appropriate data organization, programming methods, algorithms, and communications; * Gather and analyze software requirements, apply appropriate design, implement software using one or more modern programming languages, and test the correctness of the software; |

|  |
| --- |
| 1. **What instruments did you use to assess them?** |
| For learning outcomes B & C, we used direct method A.2 (Programming Projects).  Two programming intensive courses, CSci 40 (Introduction to Programming and Problem Solving) and CSci 152 (Software Engineering), were selected to assess student programming skills (Rubrics in Appendix I).  We also implemented indirect assessing method B.1 (Exit Interview) and B.3 (Faculty Discussion of Student Strength and Weakness). |

|  |
| --- |
| 1. **What did you discover from these data?** |
| **a. CSci 40 (Introduction to Programming and Problem Solving)**  CSCI 40 (Introduction to Programming and Problem Solving) was assessed in spring 2016. Totally 2 sections, 20 students, were assessed in the area of programming capabilities using the SOAP rubric form (A.2) designed by the Computer Science Department. The evaluation scales from 0-5. Students were evaluated on a programming assignment designed to test object-oriented programming in C++. The assignment was auto-graded using an online textbook environment.  Student’s scores ranged from 0 to 5 points. Since the assignment was auto-graded, only those submissions that compiled were scored. Approximately 30% of the students either did not submit to this assignment, or submitted code that did not compile. Of the students who were able to submit compilable code, 56 percent received 5 points, and 75% received 4 or 5. From the set of compilable code a closer evaluation was done using the rubric from (A.2).  Since students knew assignments were auto-graded, there was no incentive for comments so those elements of the rubric were not included. The analytic results of CSci 40 assessment conducted in spring 2016 are as follows.   * Strength   + Program meets its functional specifications and expectation of advisor: 3.7     - Students were able to apply their knowledge gained from the class to work through the assignment to completion. Many students submitted many different versions of program until they were able to fully solve problem. * Weaknesses   + Program contains error checking code and handles all special cases: 1.85     - Student focus was on meeting auto-grader requirements, leaving some special cases and error checking out of code.   + Documentation: 1.45     - Since students knew assignment would be auto-graded, there was not incentive for auxiliary documentation. This component was not included in summary score.   Given the results of this analysis the following are actionable items:   1. Inclusion of more auto-grading test cases will help insure students add additional error checking code for special cases. 2. Inclusion of some hand graded elements for some assignments will enable grading for documentation. Preferably, grading support will facilitate this additional level of grading. |
| **b. CSCI 152 (Software Engineering)**  CSci 152 is the second course of Software Engineering sequence offered every spring. At the beginning of the semester, each student acts as a program manager and proposes a semester-long team project proposal to recruit those interested in the same topic(s). Instructor groups students into 3-5 people teams based on students’ Computer Science backgrounds and proposal preferences. Each team then exercises software development that comprises requirements, design, implementation, testing, demonstration as well as tools used for increasing productivity.  Due to the nature of team-based setting and learning objectives of Software Engineering course, the rubric to assess programming skills used for CSci 150 and 152 is different from the one provided by the Department, which basically is for assessing individual programming skills. (In short, from the perspective of Software Engineering, software development spans from requirements to maintenance, instead of just implementation/programming). The assessment form is attached at the end of the report. In the form, each team is assessed by 7 items. Additionally, each student’s team-project presentation is also assessed. Since this AY’s assessment only emphasizes programming skills, the data and analysis of student presentation are not included.  Among the 7 assessed items, “Requirements” assesses whether a team is capable of articulating functional and non-functional features/requirements of the software in IEEE suggested format. Final version of requirements are posted on github.com (a source code repository) wiki page. “Design” assesses a team’s software design, represented in UML diagrams. Not only correctness use of UML symbols but also how good software design is will be assessed. “Implementation” assesses a team’s programming capabilities. In this course context, it means how many features that a team can deliver under time constraints. On the other hand, “testing” assesses how well implementation is. Namely, how to introduce smallest set of test cases in order to reveal the largest set of defects. “Demonstration” assesses whether software can be normally run as promised and could possibly handle unexpected errors. It is also a way to assess programming capabilities. “Tools” assesses whether a team is familiar with modern software development tools (e.g., Github, Trello, Slack, Eclipse Testing plugin) in order to increase productivity. Each item is scaled with very poor (score 1) poor (score 2), average (score 3), good (score 4) and excellent (score 5).  Among 6 teams, the average score on all 7 assessed items is 4.44 and standard deviation is 0.0878. The highest score is 4.52 and lowest is 4.26. Three teams performed above the average. From the data, it can be seen that only one team’s score relatively lower than others (0.18 lower).  For each of the 7 items assessed on the rubric, the average score is also 4.44 (lowest is 4.27, highest is 4.53 and standard deviation is 0.1396). Among 7 assessed items, “Demonstration” obtained highest score. “Requirements”, “Design” and “Tools” scores are between 4.46~4.48. “Implementation” score is 4.41 and “Testing” obtained the lowest score 4.27.  From the analysis of the assessment data, it can be observed that students either do not have time to conduct sufficient testing. Or they are not interested in verifying whether software performs correctly. Instead, students are interested in introducing/implementing new features, since this is more visible and rewarding to classmates. This analysis result encores the finding from Software Engineering community (ironically even though students have finished two undergraduate software engineering courses) and this is why more education and promotion of good software engineering principles and practices are needed so that such good principles and practices can be naturally embedded into students’ mindset when developing software.  In addition to numerical analysis, lab instructor also conducted exit interview to help improve the course. Based on her findings, students felt software testing topic is introduced too late (Week 9, after midterm). It is more preferable testing topic can be introduced earlier so that there could be more time to digest testing concepts and explore and utilize suitable tools appropriately.  Based on the numerical analysis results and exit interview, instructor will   * Shift software testing topic to Week 4. * Instructor will specifically emphasize how important software testing is and how it may influence software quality in final stage. * In addition to different testing techniques that have been covered, with more time to cover testing topic, instructor may also cover more advanced testing documentation and management topics to alleviate future maintenance headache. * A side note from instructor’s own observation, students’ requirements and design skills, even though earning 4.47and 4.48 respectively from rubric assessments (the score also includes evaluation from peers), still have room for improvement. Experience is usually a key factor for such a finding. Instructor will encourage students continue to take or self-study advanced software engineering courses.   Last but not least, even though the rubric is not the same as the one offered by the Department, the Software Engineering specific rubric embodies the majority of programming assessment items requested by the Department. For example, “Demonstrates synthesis of solutions and creates alternatives by combining knowledge and information” and “Demonstrates a clear understanding of how various pieces of the problem relate to each other and the whole” can be assessed by our “Design”. Likewise, “Uses software tools and computing resources correctly and effectively” can be assessed by our “Tools”; “Program contains error checking code and handles all special cases” can be assessed by our “Testing” and/or “Demonstration”; and finally “Program meets its functional specifications and expectation of advisor” can be assessed by our “Requirements” and “Demonstration.” Only “Documentation” and “Presentation” categories are not covered in the assessment.  **c. Exit Interview**  We conducted an Exit Interview through Google Form in an attempt to improve the number responses. Unfortunately, same as last year, no responses were received.  **c. Discussion of Student Strength and Weakness**  On July 15, we discussed the strength and weakness of undergraduate students during the department annual retreat. The results are as follows.  **Strength**   * Students are better prepared for upper division courses * More students have been active and motivated on programming and community projects. * Students continued to pursue research opportunities and graduate school admissions.   **Weakness**   * + Only a small percentage of students worked on senior projects or capstone projects, due to limited resources of faculty supervision. |

|  |
| --- |
| 1. **What changes did you make as a result of the findings?** |
| * + From the analysis of the assessment data, we plan to make the following changes.   + **a. Programming Projects** * Add testing requirement specifically in project description * Manually review error checking in student code and test potential scenarios that trigger certain abnormal conditions * Manually review and grade comments and documentation in student projects.   + **b. Exit Interview** * Conduct face to face interview as well as paper based to increase response rate   **d. Student Strength and Weakness**   * + Encourage students to take senior projects CSci 198   + Possibly add one more section for CSci 152 if needed |

|  |
| --- |
| 1. **What assessment activities will you be conducting in the 2015-16 academic year?** |
| During the next academic year, we will work on three assessment methods: |
| Method A.1 Capstone Project Report  Method B.2 Client Survey |
| Method B.3 Discussion of Student Strength and Weakness |
|  |

|  |
| --- |
| 1. **What progress have you made on items from your last program review action plan?** |
| The department has made the following progress on BS action plan. We have submitted a self-study report to program review officers. Review team site visit will be on November 3 and 4, 2016.  **a) Rebuild the faculty**  **Progress**: we recruited two tenure track faculty: Dr. Jin Park and Dr. Cui Lin in Fall 2011. Dr. Cui Lin resigned in December 2013. Dr. Shigeko Seki has retired starting Fall 2014. The department currently has only five tenured and tenure track faculty (Brent Auernheimer, Ming Li, Shih-Hsi Liu, Jin Park, and Todd Wilson), one 3-year temporary faculty (Prudence Lowe), and three part-time temporary faculty (David Ruby, Santanu Banerjee, and Dhanyu Amarasinghe). The department has been approved to conduct a tenure track search in AY 2016-17 in the area of operating systems, cloud systems, distributed systems, and/or security.  **b) Develop new assessment plan (SOAP) for B.S., along with supporting instruments**  A new BS SOAP has been made with the following improvement:   * Five “measurable” learning outcomes have been devised to replace the previous “topic” based “non-measurable” learning outcomes. * “Research Activities” was removed from direct method * Elective courses were removed from curriculum map to ensure consistency assessment * Major Field Test (MFT) as a comprehensive evaluation will be required for all graduating students. So far, 19 students volunteered to take the exam. * Student capstone project was moved to direct assessing methods * Student course project was added to direct methods * Clear criteria were set for all direct methods * Employer survey was added to the indirect methods * A four-year timeline for implementation has been included and will be updated annually to make it current. * We plan to make further update based on the suggestion from University Assessment Director Dr. Melissa Jordine.   **c) Develop CSci 100 for GE Area IB (course title: Introduction to Computational Science)**  **Progress**: the course was successfully offered in Fall 2013. It will be offered every Fall semester.  **d) Establish an Advisory Board**  **Progress**: The department faculty has selected a list of eight potential candidates for an Industry Advisory Board. It is currently pending approval by the college.  **e) Develop Internship and funded-Project procedures**  **Progress**: Dr. Todd Wilson and Dr. Ming Li have been involved in the university Internship Faculty Learning Community (FLC) to understand issues, policies, procedures and required documents for setting up an internship program. Dr. Ming Li has met with Dr. Jaime Arvizu, the Director of the newly established CSM Advising and Resource Center (ARC) and look forward to work with ARC on this effort within the next academic year. In addition, Internship (CSci 196) will be one of the elective courses in the newly proposed CSCI BS curriculum.  **f) Curriculum reviewed with respect to ACM and IEEE Computer Society curriculum recommendations, and ABET accreditation criteria.**  **Progress:** The department faculty understand that it is critical to improve student success. We have established a formal department curriculum committee with members being all full time tenured and tenure track faculty.  The department curriculum committee has studied the ACM Curriculum Recommendation 2013 as a group and proposed a new CSCI curriculum in December 2014. We have received feedback from related programs on the new curriculum and will submit to college curriculum committee for review in Fall 2016.  We plan to reach the “ABET ready” status by May 2018.  **Progress in other items include:**  • Allocation of four faculty research labs.   * Grading support for faculty teaching classes with large enrollment (over 48 students).   • Lab refreshment for computer lab McF 201 in July 2014.   * McF 205 is on schedule to be converted to a hybrid computer lab with capacity of 28. * Science II 258 is on schedule to be converted to a department conference room. |
|  |