DIRECTIONS FOR INSTRUCTOR USE OF THE SOFTWARE ENGINEERING II ASSESSMENT RUBRIC

This rubric is intended for use in evaluating student ability to apply design and development principles in the construction of software systems of varying complexity. Instructors should share copies of the assessment rubric with students in advance of the students' participation in assignments so that they will understand what is expected of them on the assignment and how they will be evaluated.

To use the rubric, the evaluator should place check marks in the boxes corresponding to their evaluation of the various dimensions of the student's performance.

The rubric is set up with four levels of performance (i.e., unacceptable, developing, competent, exemplary) that can be achieved by the student during the assignment.

- unacceptable: :
  The student does not demonstrate sufficient knowledge, skills or abilities with respect to this dimension and therefore, does not meet the instructor's expectations.
- developing:
  The student demonstrates only the initial knowledge, skills or abilities with respect to this dimension and therefore, does not meet the instructor's expectations.
- competent:
  The student demonstrates sufficient knowledge, skills or abilities with respect to this dimension, and thereby basically meets the instructor's expectations.
- exemplary:
  The student demonstrates greater knowledge, skills, or abilities than expected by the instructor, and thereby exceeds the instructor's expectations with respect to this dimension.

MTSU Computer Science Software Engineering II Rubric version 1.0 Last Change 8/23/2011	
Name of Individual being evaluated:	
Name of Evaluator:	

Performance Criteria	**Unacceptable**	**Developing**	**Competent**	**Exemplary**
The student applies modern design techniques to ensure code meets design specifications	The student develops code without following the design spec and/or without using structured and OO programming techniques. The code must usually be rewritten by others	The student develops code that follows the design spec, is designed based on structured and OO programming techniques, but often must be revised somewhat with the help of others before it is acceptable	The student develops code that follows the design spec, is designed based on structured and OO programming techniques, utilizes design patterns where appropriate, and is delivered with little or no help from others	The student performs competently and in addition notices flaws in or improvements that can be made to the design spec and consistently delivers as well as helps others to deliver code that is of exceptional quality
The student follows established processes and utilizes design and code reviews in software system development	In team assignments, the student neglects reviews of teammates' work and fails to deliver design and code for review by his/her teammates	In team assignments, the student conducts superficial reviews of teammates' work and delivers design and code too late for his/her teammates to review at length	In team assignments, the student conducts basic reviews of teammates' work and delivers design and code in time for a basic review by his/her teammates.	In team assignments, the student conducts rigorous reviews of teammates' work and delivers design and code in time for a thorough review by his/her teammates
The student performs thorough testing of developed software system	The student performs minimal unit testing of own code, concentrating exclusively on the simplest, most obvious cases	The student performs black-box unit testing of own code, using reasonable sample of average and extreme test cases	The student plans and executes thorough list of test cases for black-box testing of his/her own code as well as the team's integrated code, with expected results specified	The student plans and executes thorough list of test cases for black-box unit and system testing, as well as white-box testing of modules produced by his/her teammates