DIRECTIONS FOR INSTRUCTOR USE OF THE SOFTWARE ENGINEERING I ASSESSMENT RUBRIC

This rubric is intended for use in evaluating student ability to use current techniques, skills, and tools necessary for computing practice. 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 I 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 appropriate software development process models.	The student applies the same process model to every software system being developed	The student alternates between one predictive model and one agile model for various systems	The student demonstrates the ability to apply several agile and predictive models to different software system development situations.	The student modifies classical process models to accommodate the particular needs of a given software development situation.
The student thoroughly models planned software system to effectively plan the future implementation	The student begins software implementation without significant modeling of software processes and interactions	The student lays out preliminary model of software processes and interactions prior to the start of coding	The student applies modern modeling tool (e.g., UML) to model the primary classes and functionality of a planned software system	The student applies a modern modeling tool to develop a detailed model of a planned software system's classes and interactions, including the planning of actual client usage.
The student develops accurate time and size estimations for a planned software system	The student merely guesses about the time and size associated with a planned project	The student extrapolates time estimates on projects from experience on previous projects	The student applies linear regression to estimate time and size of a new project based on data from previous projects	The student applies linear regression to estimate the time and size of a new project, with an alternative estimation technique used if data correlation is insufficient.