Middle States Committee, 2015
Course Project Report on Graduate Course CMSC/AMSC 663, Spring 2015

Background
CMSC/AMSC 663 is a project-based course where students pick a large-scale software engineering task and implement it over the course of a semester. The project assignment contains numerous components. In addition to developing the software tool, students are responsible for making several presentations, as well as for producing documents for the project proposal and for the final project summary. Students are expected to find a faculty advisor to oversee their project. However, the project is not intended to be research-oriented, but rather to be an implementation of established methods.

Difficulty of project/assignment
The project is very flexible, and allows students to target a level of difficulty that is appropriate for them. However, the instructors that run the course have a lot of control over the requirements that must be completed during the course of the project, and this prevents students from having too little structure.

Appropriateness of project design and expectations
While the project is not necessarily difficult per se, the amount of work required of students is perhaps a bit too heavy for a 600-level class, given that most students at that level have TA- and course- responsibilities outside this class. For example, the students must prepare two 15-20 page documents, one for the proposal and one for the summary. The students also do three presentations, one about themselves at the project start, one for the project proposal, and one for the project summary.

It would perhaps be more appropriate to consolidate some of these tasks. This would make the project a bit more focused, and allow students to spend more time on software development. That being said, it is good to include some writing and presentations components to the course because these are important skills for academic success. However, these skills are perhaps a bit over-stressed in the project.

Project oversight by instructors
This is one thing the instructors do exceptionally well. As part of the project, the students are required to present a timeline for project completion, with a number of intermediate milestones. There is also periodic review by instructors to make sure things get completed on time.
Student performance reviews

Below, we include reviews of student projects. We include the following criteria:

- **Code Structure**: how well is the code organized and planned out? Is it clear that the code is extensible and maintainable? Are modules loosely coupled and self-contained?
- **Code Readability**: is the purpose of individual lines of code made clear? Is the code self-documenting?
- **Code Formatting**: Does the code obey standard organization and formatting conventions that make the code easily parsed by a human reader?
- **Code Evaluation/testing**: Does the student have a plan for testing the performance of the software, and evaluating its performance relative to other options?
- **Document Organization**: is the project summary appropriately organized as a technical document? Is it clearly divided into meaningful sections that make it easy for the reader to access information?
- **Document Readability**: is the student’s writing clear and concise?

The criteria are evaluated on a 5-point scale corresponding to (1) excellent, (2) very good, (3) good, (4) fair, and (5) poor. The score for one implementation will be the average of the scores on the individual categories.

We also rate the difficulty of the project, with -5 indicating that the project was too easy, and 5 indicating the project was too hard. A score of 0 is ideal. We have anonymized some of the identifying details to protect students’ privacy.

**Student project 1**: On computational fluid dynamics.

- **Code Structure**: 2
- **Code Readability**: 1
- **Code Formatting**: 1
- **Code Evaluation/testing**: 3
- **Document Organization**: 1
- **Document Readability**: 1

Overall score: 1.8 difficulty rating: 2

Comments: This student implemented a solver for basic problems in fluid dynamics, incorporating parallel computing as well. This is a very difficult task for a second-year student but the student did very well. Evaluation was only done by comparing to another solver for similar problems. However, conventional verification methods, such as measuring convergence rates, were not explored.

**Student project 2**: Also on computational fluid dynamics.

Structure:

- **Code Structure**: 1
- **Code Readability**: 3
- **Code Formatting**: 3
- **Code Evaluation/testing**: 1
- **Document Organization**: 3
Comments: This student implemented standard solvers for fluid-dynamics problems. This method is not very difficult to implement, but is a great learning task for a student trying to get familiar with this complex field. The verification portion of the project was very strong (student measured performance and accuracy for several standard test problems), but the presentations were a bit lacking in background material.

Student project 3: On large-scale data analysis.

- **Document Readability:** 1
- **Overall score:** 2     **difficulty rating:** 0

Comments: The student used standard methods to analyze data from a certain domain. The algorithms used seemed a bit generic and simple. The code was well written, but the sheer number of availability of implementations of these methods appeared to obviate the need for this project.

Student project 4: On computational medicine.

- **Document Readability:** 1
- **Overall score:** 1.5   **difficulty rating:** 2

Comments: This project combined two particular complex topics, making this a difficult project: the student seemed to handle it well though. The implementation was benchmarked and verified on small test problems against a variety of standard solvers. This verification was a strong point of the project. Presentations were a bit weak because they included too much text on slides. However, the project overall was well done.