ECE 417/617 Elements of Software Engineering
Spring 2008

Instructor: Stan Birchfield
Office: 207-A Riggs Hall, 656-5912, stb at Clemson
Office Hours: 3:30-5:00 M, or by appointment
Course website: **http://www.ces.clemson.edu/~stb/ece417**
Text (recommended):

- Bjarne Stroustrup, *The C++ Programming Language*, Addison-Wesley, 2000.
- Roger S. Pressman, *Software Engineering*, 6th ed., McGraw-Hill, 2005.

Prerequisites: ECE 329, ECE 352, MTHSC 419; C programming skills

Overview: In this course students will learn to build high-quality, reliable, and extensible software systems that are too large for a single programmer. Emphasis is placed upon good programming practices (object-oriented design and clean, well-documented code), teamwork skills (planning, communicating, and interacting), and bringing the project to completion (testing and verification). A semester-long project complements the theory with practice.

Objectives: By the end of the course, students should be able to do the following:

- *Fundamental concepts.* Describe the basic concepts and terms of software engineering, such as requirements elicitation, tasks and activities, project roles, analysis and system modeling, architecture, object design, user interface design, testing, management, life cycles, agile programming, formal methods, mythical man month, and open source. Draw and analyze UML diagrams. Describe object oriented programming concepts and calculate the functionality of code containing constructors, destructors, copy constructors, and assignment operators.
- *Tools.* Check in and update code using a revision control system (CVS). Create a new workspace with an IDE (Visual Studio), navigate between files, and use the debugger. Create and modify a graphical user interface using an existing library (Win32 and MFC).
- *Programming and teamwork skills.* Write clean, well-documented C++ code to achieve desired functionality. Write code that can be read and modified by others. Read and modify code written by others. Work with a team of programmers to build a large software system. Communicate effectively with team members and take initiative to contribute to the overall goal. Test code thoroughly, check in only code that works, and refactor code as necessary.

Topics:	**Lectures**
• introduction	1
• C/C++ programming language	3
• software tools (CVS, VC++)	2
• software process	3
• modeling	3
• design	3
• user interface design	3

• testing	3
• project management	3
• software practice	2
• formal methods	2
• group meetings	15

TOTAL	43

Grading:

1. *Project Milestones.* Each week, groups will submit a progress report for the previous week, along with goals for the upcoming week. In addition, they will produce work products at various intervals, including models, diagrams, schedules, and checked-in source code. The first milestone will be an individual programming assignment.
2. *Attendance.* Because of the group-based nature of the course, attendance in class and at group meetings is important. Therefore, attendance will be taken at the beginning of each class.
3. *Learning Reports.* Due to the non-linear nature of the material in this course, students are encouraged to learn information on their own, as their interests and needs dictate. Bi-weekly reports will document this learning process in a tangible form. Reports should be the equivalent of a 1-2 page paper but may take a variety of forms (written document, Powerpoint slides, HTML page, etc.); be creative. Near the end of the semester, this material will be used as the basis for a brief oral presentation on some topic within software engineering.
4. *Final exam.* There will be a closed-book, closed-notes final examination covering the lecture material for the entire semester. This will be a pass/fail test similar to a certification exam. The instructor should be notified of any conflict that would prevent a student from taking the exam at least one week in advance, so that alternative arrangements can be made. Without prior approval, a missed exam cannot be made up except in cases of extreme urgency and importance (e.g., sudden illness, death in the immediate family).

Grades will not be determined in the traditional "point" fashion but instead using the following C program:

```
char compute_grade
(
    int m, // number of weekly milestones not successfully completed
    int a, // number of unexcused absences
    int r, // number of learning reports submitted by the due dates
    int f  // final exam grade
)
{
    if (m <= 2 && a <= 2 && r >= 6 && f >= 70)  return 'A';
    else if (m <= 4 && a <= 4 && r >= 3 && f >= 70)  return 'B';
    else if (m <= 6 && a <= 6 && r >= 1 && f >= 70)  return 'C';
    else if (m <= 8 && a <= 8 && r >= 1 && f >= 70)  return 'D';
    else return 'F';
}
```

Students taking the graduate-level version of the course (ECE 617) must, in addition to the above, write a 2-3 page summary of an *IEEE Transactions of S/W Engineering* paper.