ECE 4960: Computational and Software Engineering

Spring 2017

Week 1: Class Introduction

Reading Assignments:

1. Chap. 1, D. Bindel and J. Goodman, *Principles of Scientific Computing*, 2009.
2. Chaps. 2 and 8, B. Einarsson, Ed., *Accuracy and Reliability in Scientific Computing*, SIAM 2005.
3. (Optional) Chaps. 6 – 8, S. Oliveira and D. Stewart, *Writing Scientific Software: A Guide to Good Style*, Cambridge 2006.

1. Class logistics

1.1 ECE 4960 is:

- A senior/first-year graduate level of programming class that teaches you how to write **robust** and **verifiable** software to interface with the physical world described by mathematical laws. The implementation cycle is shown in Fig. 1.

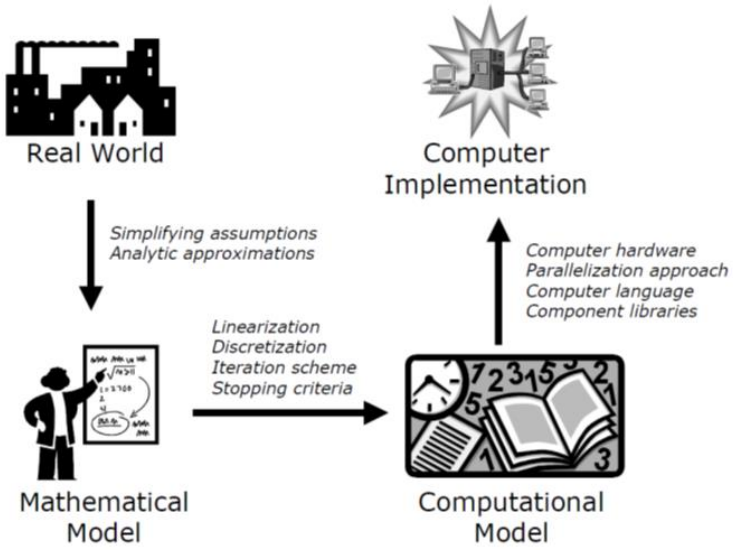

Fig. 1. Programming to interface or to emulate the real world from the mathematical modeling. Both computational and software perspectives are essential.

- The main focus is on problem solving for scientific and engineering problems, as well as problems derived from virtual reality and augmented reality. Most of these problems are still too large to ignore the hardware implementation, such as circuit simulation of a chip, weather simulation, 3D image/geometry processing, etc.
- We will differentiate between **hardware-aware** programming and **fast-prototype** programming, where execution efficiency tradeoffs with *programmer productivity* and *platform tolerance*, as shown in Fig. 2. For example, for most assignments, you will need to be able to estimate the computational efforts and memory usage.
- Role reversal on assessment: In commercial software, developers, instead of users, provide specification, validation, and test cases. In this class, likewise you will assume similar responsibilities in defining how to assess your results. Overspecification or oversupervision

during the training phase results in immature learners as well as programmers, as shown in Fig. 3 for a Chicago mathematical education assessment.

- Developers should also be in the position of novel users and attackers! Robustness mostly comes from comprehensive testing in all levels from the user perspective. War strategy: "There is no guaranteed victory, but there are careful ways to avoid failure!"

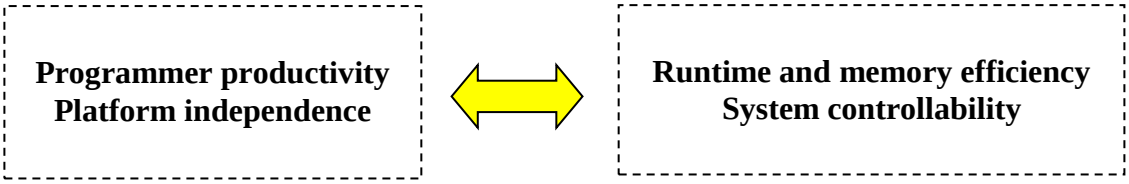

Fig. 2. Programming abstraction level vs. System performance and controllability.

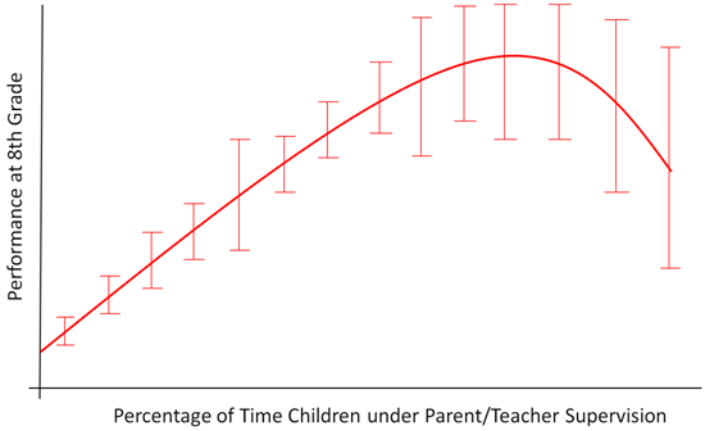

Fig. 3. The philosophy behind self assessment of program assignment. The parenting curve is drawn from: Stephen J. Dubner and Steven Levitt, *Freakonomics: A Rogue Economist Explores the Hidden Side of Everything, Chap. 5*, William Morrow, 2005.

Group discussion: How to make it fair to prescribe your own testing methods for class assignments?

1.2 ECE 4960 is not:

- This is not a class to teach fundamental program skills: You are expected to have substantial programming experience in a high-level, object-oriented language. Although you are allowed to use any language and developer platform of your choice, Gnu C++ will be most often used as programming practice examples. Programming techniques such as sorting and tree trasversing are taught in basic data structure classes, and are implicitly assumed.
- This is not a mathematical class to prove convergence or error bounds, although concepts for testing and verification will be reviewed when needed.
- This is not a class to focus on Internet or Cloud computing environment where programmer productivity and machine compatibility from virtual wrappers are emphasized.
- This is not an operating system or compiler class to deal with system setup and resource allocation, although you will gain some practical knowledge of the system. Detailed knowledge of compilers and operating systems belongs to other courses.

1.3 Syllabus:

- Sign up for Blackboard access: ECE 4960 Special Topics in ECE Kan, E (18129_2016SP)
- Lectures MWF 11:15am – 12:05pm in Hollister 362. I will not do role call, and attendance is explicitly considered in grading. However, there will be "real-time hacking assignments" to enhance your programming ability. An announcement will be posted on Blackboard for you to finish the assignment within a given time, but you will miss the "speed bonus" if there is one.
- Recitation/Lab: Tuesday: 7:30pm – 9pm in Phillips 213. The are two main functions: introduction of good large-scale programming practice and project practice. This will also be the time reserved for make-up lectures if necessary.

Group discussion: What is the most important ONE thing you hope to learn in this class?

2. Software development overview

2.1 The lifecycle of programming

- Three things for quality assurance: Testing, testing, and testing.
- Three things for longetivity and re-use of codes:
    1. Clear specification and object models;
    2. Evolution of source code together with regressional tests;
    3. In-code and external documentation

2.2 Source of errors in software

Murphy's Law: "Anything that can go wrong, will go wrong."

Yhprum's Law: "Anything that can go wrong, will go right." (Or Chinese saying: "The path will become straight after you make the turn." "The boat can go straight after passing the bridge."

Murphy's Law for things with complexity: "If one thing goes wrong, everything else will, and at the same time"

Bugs, bugs, bugs:

- Logic errors
- Coding errors
- Approximation and convergence errors
- Conversion errors
- Memory errors
- Memory and resource leaks

- Multi-threaded or race errors
- Timing errors
- Distributed application errors
- Storage errors
- Procedure integration errors
- Version errors and backward compatibility

Discussion: What is your most vivid experience in programming errors?