ECE 4960

Spring 2017

Lecture 1

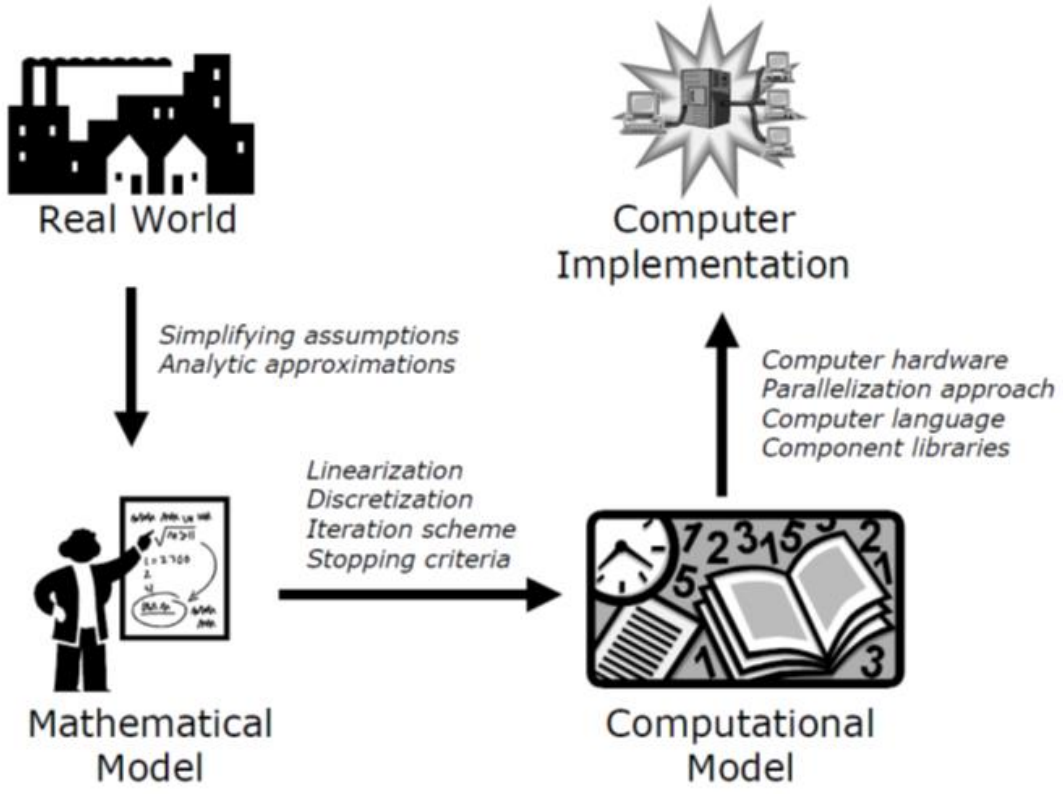
Class Introduction

Edwin C. Kan

School of Electrical and Computer Engineering

Cornell University

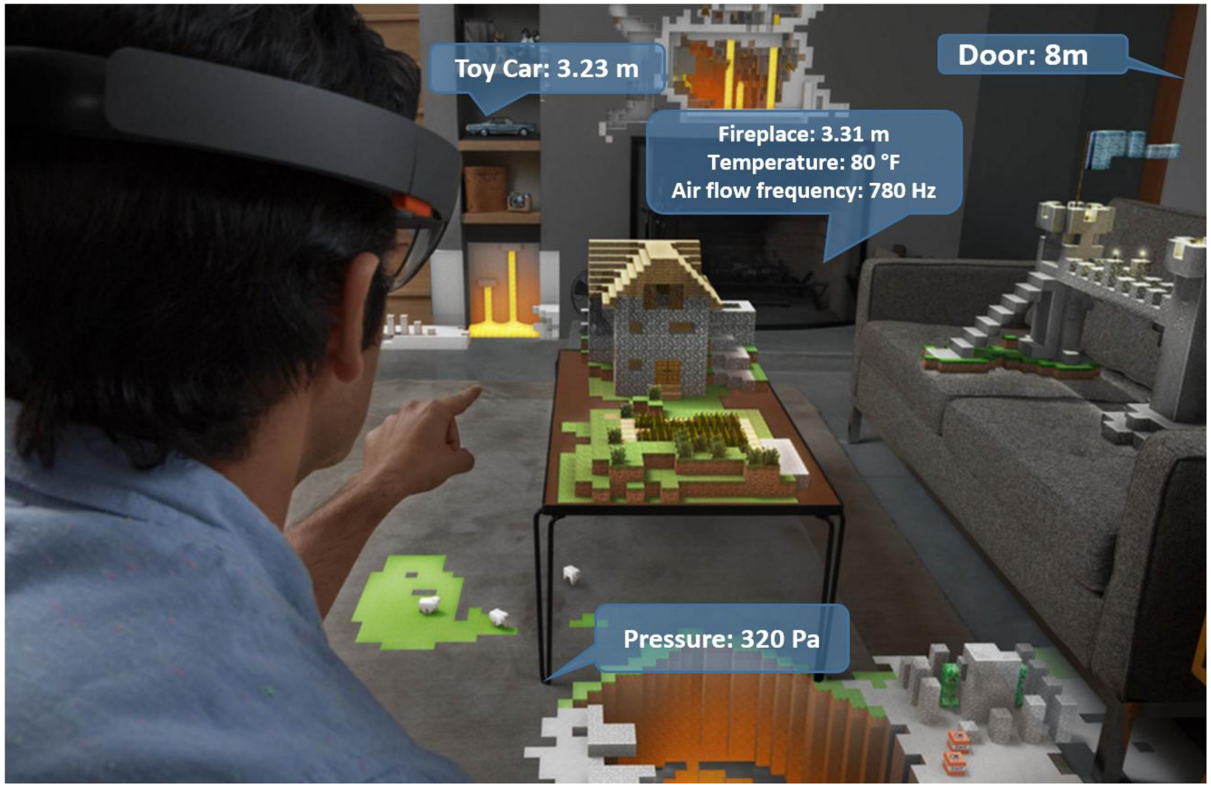
Computational and Software Engineering



- **Robust** and **verifiable** software to interface with the physical world described by mathematical laws.

Main Applications

- Scientific and engineering problems: simulation; design; testing
- 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.



Programming Languages

- **Fast-prototype** programming vs. **Hardware-aware** programming

Programmer productivity

Platform independence

↔

Runtime and memory efficiency

System controllability

Python

Matlab

Basic

Java

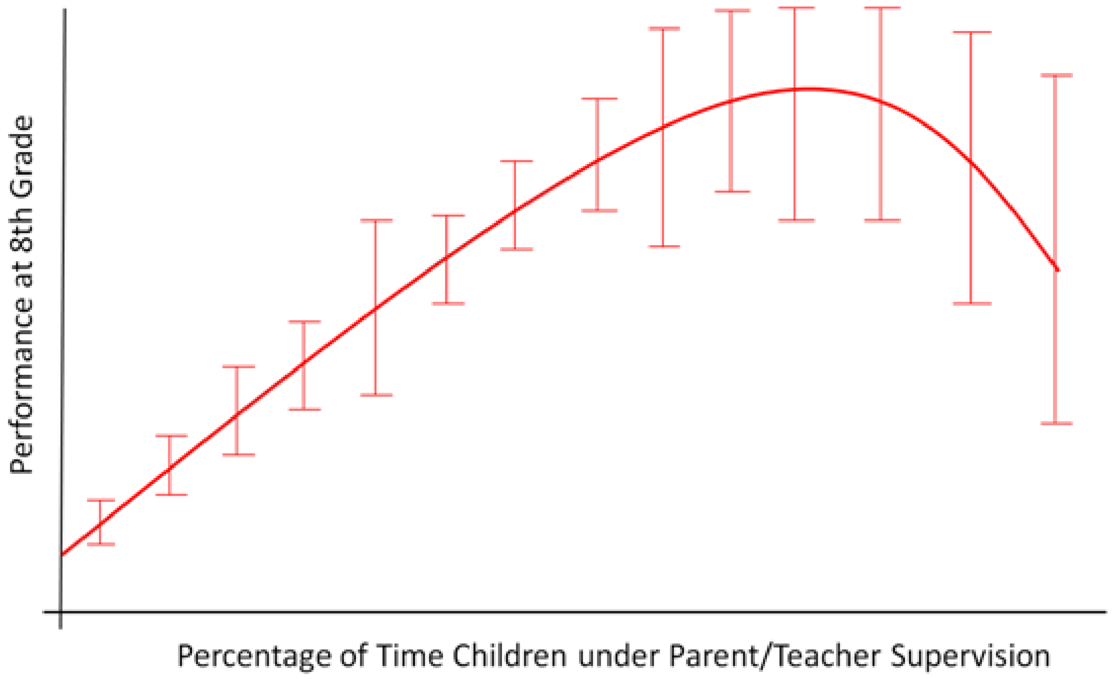
C++

C

Fortran

Software Assessment

- In commercial software, developers, instead of users, provide specification, validation, and test cases.
- Assume 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 a Chicago mathematical education assessment.



Group Discussion

- How to make it fair to prescribe your own testing methods for class assignments?

This class is NOT… (1)

- 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 class is NOT… (2)

- Not a mathematical class to prove convergence or error bounds, although concepts for testing and verification will be reviewed when needed.

- Not a class to focus on Internet or Cloud computing environment where programmer productivity and machine compatibility from virtual wrappers are emphasized.

- 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..

Syllabus

- See handout.

Group Discussion

- What is the most important ONE thing you hope to learn in this class?

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

Will cover in this class

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

Will NOT cover in this class

- ❑ 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?