SWEN-220
Mathematical Models of Software

Course Introduction
It’s complicated...

- Engineers build complex systems, whether software, hardware, cars, planes, or bridges.
- **Complexity** is a major source of engineering cost and errors.
- How can we best manage complexity?
What is a Model?

https://www.youtube.com/watch?v=eszhVxE_9-8
Modeling

- What is a model?
  A simplified version of some real or conceptual object or entity.
  - A wind tunnel model of an airplane.
  - A 2D model of a house floor plan

- Why model?
  To explore some aspect of the entity of interest.
  - The flow of air over the airplane's surfaces.
  - The arrangements of doors, walls, appliances, furniture.
Key characteristics of a good model

• Simplicity:
  – Exclude detail irrelevant to model's purpose.
  – “Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away.” Antoine de Saint-Exupéry

• Accuracy
  – All necessary information is included.

• Analyzability
  – Can use model to assess its important properties.
  – Analysis may be formal (e.g., mathematical) or informal.
  – We, of course, will emphasize formal, mathematical models.
"Everything should be made as simple as possible, but not simpler."

Albert Einstein
Nobel Prize winner in Physics
(1879-1955)
What is an Abstraction?

• **Abstraction** is the removal of unnecessary detail in order to reduce complexity.

• **Abstract Models** can be used throughout the engineering life-cycle of a project.
If you can’t explain it **simply**, you don’t understand it well enough.

— Albert Einstein
What are Math Models of Software?

• Tools and techniques based on **mathematics** and **formal logic** used to describe the design and behavior of a software system...

• Typically use discrete mathematics (e.g. sets, relations)

• Used both during design and verification (testing) and debugging.
Math Models vs Traditional Testing

• Testing can only explore *some* behavior of the software system
• Testing can show if an error exists, not the absence of errors
• Using math models can “guarantee” that a software design is bug free*
• Testing can be very costly, especially if high coverage is required

* Depends on resources available
Why Do We Study Math Models?

• Provide you with background in math models for SW systems
• Improve your analytical ability and your ability to be precise.
• Train you on the tools needed to deal with critical design issues
Why NOT Math Models?

• Can be difficult to model complex software systems
• You are forced to think precisely
• Large models can take a long time to check
• Much easier to test software than formally verify it
Success Stories Using Models

• Praxis used formal methods to verify Operational Flight Programs for Lockheed Martin
  – Code quality improved by 10X over industry standards
  – Productivity improved by 4X
  – Development costs cut in half

• Intel now uses formal methods techniques when testing their chips

• Amazon Web Services use of formal methods

• See more at http://alloy.mit.edu/alloy/citations/case-studies.html
The Perils of Concurrency

- **1985-1987 -- Therac-25 medical accelerator.** A radiation therapy device malfunctions and delivers lethal radiation doses at several medical facilities. Based upon a previous design, the Therac-25 was an "improved" therapy system that could deliver two different kinds of radiation: either a low-power electron beam (beta particles) or X-rays. The Therac-25's X-rays were generated by smashing high-power electrons into a metal target positioned between the electron gun and the patient. A second "improvement" was the replacement of the older Therac-20's electromechanical safety interlocks with software control, a decision made because software was perceived to be more reliable.

- What engineers didn't know was that both the 20 and the 25 were built upon an operating system that had been kludged together by a programmer with no formal training. Because of a subtle bug called a "race condition," a quick-fingered typist could accidentally configure the Therac-25 so the electron beam would fire in high-power mode but with the metal X-ray target out of position. At least five patients die; others are seriously injured.

*(More on the Therac-25 disaster later in the term when we cover concurrency)*

(Source: “History's Worst Software Bugs”, Wired)
Boeing 737 Lion Air Flight 610 plunged into the Java Sea less than 15 minutes after takeoff on Oct. 29, 2018 killing all 189 people aboard.

Suspected that sensors indicating the position of the plane’s nose failed & returned faulty readings to the plane’s automated flight control system.

Control system thought the plane was stalling, so pushed the nose down to dive and gain speed.

Pilots did not sense a stall condition and fought the control system by pulling the nose up.
“It’s really hard to find some faults in testing, so it has happened that problems are only discovered after the plane is put into service,” said Gerry Soejatman, an Indonesian aviation expert. “Sometimes weird things happen, and you just can’t anticipate it.”
Course Scope

• Expose you to the challenges in modeling problems related to software systems

• Learn various concepts related to math models of software
  – Discrete mathematics for modeling
  – Information/Data modeling
  – Structural modeling
  – Concurrency modeling