ILLINOIS INSTITUTE
OF TECHNOLOGY
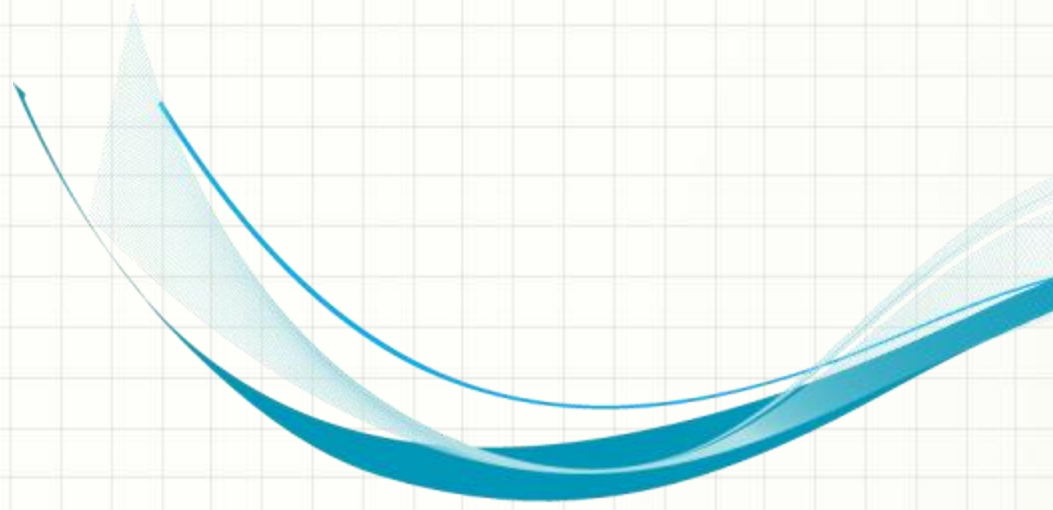
*Transforming Lives. Inventing the Future.*
*www.iit.edu*

# SOFTWARE ENGINEERING
# CS 487

Prof. Dennis Hood

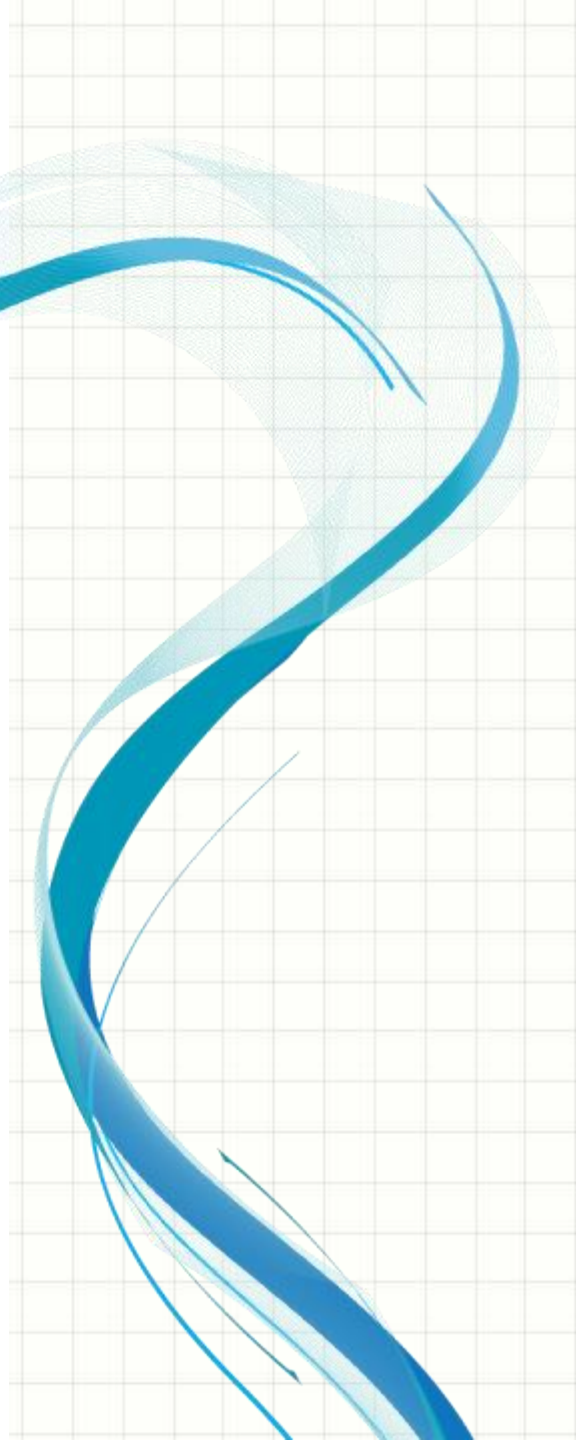Computer Science

# Week 1 Introduction and Motivation

# Instructor

# Dennis Hood

- Background
  - Education
  - Teaching
  - Industry
- Contact
  - [dhood@iit.edu](mailto:dhood@iit.edu)
  - Office Hours
    - TBD - Tue/Thu 1:00pm – 1:30pm
    - SB209-B / virtual - by appointment

# Objectives

# Course Objectives

- To explore the discipline of software engineering and associated activities and processes

- To understand its importance relative to computer science

- To understand its role in business and society as a whole

- To establish a level of proficiency in engineering software systems

- To explore related ethical issues

# Reading

# Reading Materials

- Required textbook
  - *Software Engineering (10$^{th}$ edition)*, Sommerville

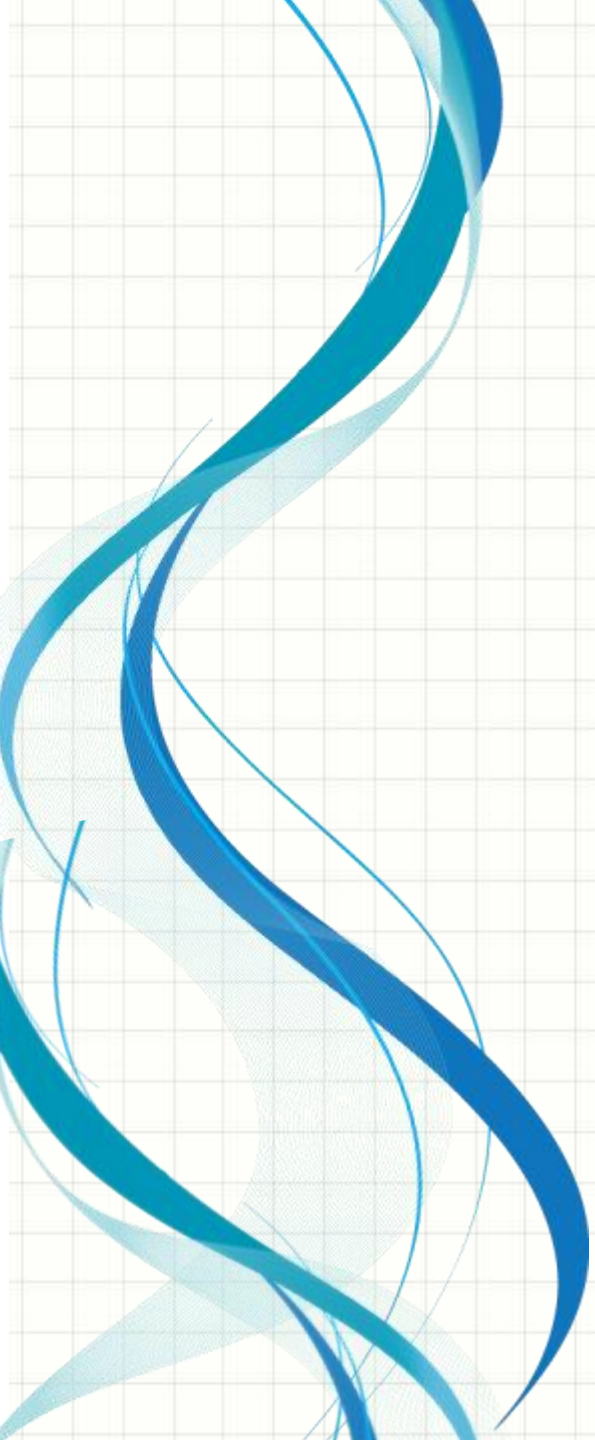- Articles, papers, etc. may be assigned to supplement the textbook

# Grading

# Grading

- Homework Assignments (4 / 20%)

- Individual Research Paper (15%)

- Team Design Project (20%)

- Midterm Exam (10%)

- Final Exam (15%)

- Participation (20%)

# Motivation for Studying Software Engineering

# The Purpose of Software – Automation?

- Machines doing the work of humans
  - $$ Faster, cheaper, better $$
  - Repetitive tasks can be documented
  - Communication requires common language
- Example – the Industrial Revolution
- Why not have machines do everything?
  - Exceptions happen
  - Circumstances change
  - Humans are better at "nuance"

# Awareness and Intelligence

- How does an entity become aware?
  - Sensing
  - Existing knowledge (from past learning)
- How does awareness provide the opportunity for decision making (intelligence)?
  - Understanding of situation
  - Anticipation of future state
  - Learning
- How does this form an "architecture" for artificial intelligence?

# Risk and Exception Management

- A risk is a threat to success
  - It may or may not happen (likelihood of occurrence)
  - If it does happen, it may have small, medium, or large consequences (impact)
  - The exposure is Likelihood X Impact (expected value)
- Exceptions are deviations from normal
  - Some might be somewhat expected deviations (e.g., holiday hours)
  - Some could be completely unforeseen (e.g., mother nature)
  - Often requires that we execute an appropriately "abnormal" set of processing steps
  - Which cannot happen unless the exception was "detected"

# Software Development- Success

- Scope – it does what it's supposed to
- Quality – it does it well
- Usable – users have a good experience
- Efficient – minimal impact on resources
- Dependable, reliable, secure, etc.
- Maintainable, portable, scalable, etc.

# Challenges to Success

- Layers of inter-dependence
  - Lack of standardization
  - Lack of accountability
- Rapid evolution of technology
- Understanding user needs
  - Different languages and contexts
  - Users often don't completely know themselves
  - Rapid evolution of user needs (competition)
- "Build from scratch" mentality
- Difficult to measure size, complexity, etc.

# Software Process and Models

- A series of steps for designing and developing software systems
  - Analysis
  - Design
  - Implementation
  - Verification
  - Maintenance
- Models
  - Waterfall
  - Iterative

# Ethics and Professional Responsibility

- Critical systems
- Data privacy
- Resource utilization
- Useful life
- Snooping and confidentiality
- Intellectual property
- Strive for perfection (e.g., usability)
- Deliver what you promised

# Software Engineering vs.

- Computer Science
  - Theory vs. practice
  - Similar to physics:electrical engineering
- Systems Engineering
  - Software is an element of the system
  - Hardware, deployment, process, etc.

# Why Software Engineering?

- History
  - [Report on the 1968 NATO conference](#)
  - Section 7.1
- Addressing the "software crisis"
  - Systems becoming larger and more complex
  - Projects taking too long, costing too much, and failing to deliver effective, reliable systems
- The world is becoming increasingly dependent on software
- Discipline is required to create systems that are
  - Reliable
  - Effective, etc.
- Cost-effectiveness
  - The software often costs more than the hardware
  - Maintenance can easily cost more than development

# The Software Crisis

## 7.1.2. PROBLEM AREAS

*There was a considerable amount of debate on what some members chose to call the 'software crisis' or the 'software gap'. As will be seen from the quotations below, the conference members had widely differing views on the seriousness, or otherwise, of the situation, and on the extent of the problem areas.*

*Dijkstra:* The general admission of the existence of the software failure in this group of responsible people is the most refreshing experience I have had in a number of years, because the admission of shortcomings is the primary condition for improvement.

*Opler:* Either of the following two courses of action would be preferable to the present method of announcing a system:

1.  Do all development without revealing it, and do not announce the product until it is working, and working well.

2.  Announce what you are trying to do at the start of the development, specify which areas are particularly uncertain, and promise first delivery for four or five years hence.