Class Objectives

Introduction to processes, techniques, and methods in software engineering.

- **Processes**: understand the processes of software development.
- **Techniques**: learn theories and techniques developed for increasing productivity of development and quality of products, and
- **Methods**: understand industrial practices and methodologies.

2

Class Objectives

- A semester-long project designed to give you hands-on experience with software engineering principles. You will,
  - Practice skills in a team environment on
    - Requirement solicitation and engineering
    - Architecture and functional design
    - Verification and validation
    - Project management
  - Gain proficiency in technical writing by producing the documentation required for the project.
- To learn, to think, and to have fun.

3

Software Engineering Courses EECS
CptS322 → CptS323 →CptS422

- **CptS 322 Software Engineering Principles I**
  - Introduction to software engineering
  - Emphasizes on processes, techniques and methods in software engineering
  - Covers :
    - the main activities of building software
      - requirements engineering,
      - system architecture and design,
      - system construction, and
      - deployment and maintenance
    - the elements that are integral to those activities
      - quality assurance,
      - risk management ,
      - process engineering and
      - project management.

4

Software Engineering Courses at EECS
CptS322 → CptS323 →CptS422

- **CptS 323 – Software Design**
  - Emphasizes on software design techniques and methodologies
    - object oriented design and analysis using UML,
    - design patterns.
  - Covers the main activities of software design process in detail
    - requirements engineering,
    - system architecture and design,
    - system construction.

5

Software Engineering Courses at EECS
CptS322 → CptS323 →CptS422

- **CptS 422 Software Engineering Principles II**
  - Emphasizes on software testing
  - Covers :
    - a wide spectrum of software testing techniques
      - white-box,
      - black-box, and
      - model-based techniques.
    - evaluating and maintaining the quality of software programs
    - applying CASE testing tools to aid in detecting software defects

6

Expected Work

- Prerequisites: CptS 222/223
- Homework assignments
  - Will be announced in the class and posted on the website with their respective due dates.
  - Unless posted, shall be edited and submitted electronically via ANGEL.
    - Acceptable formats: Word, Powerpoint, PDF, OpenDocument, Postscript, GIF or JPEG.
    - Homework submission deadline is the midnight of its posted due date.
  - Late penalty
    - Advanced notice to the instructor is required for late submission.
    - 10% reduction of points for the late homework for up to 1 week with prior request for extension.

Expected Work

- Project
  - Progress of the project is marked by a series of milestone, at each milestone, you may be required to submit required software artifacts to demonstrate your progress ;
    - Example of software artifacts: project plans, written progress reports, design documents, code, test cases, etc.
  - Your project score will be given as a team: everyone in the team gets the same grades.
- Grading
  - 20% Homework, 20% Midterm, 20% Final
  - 40% Project

9

Policies

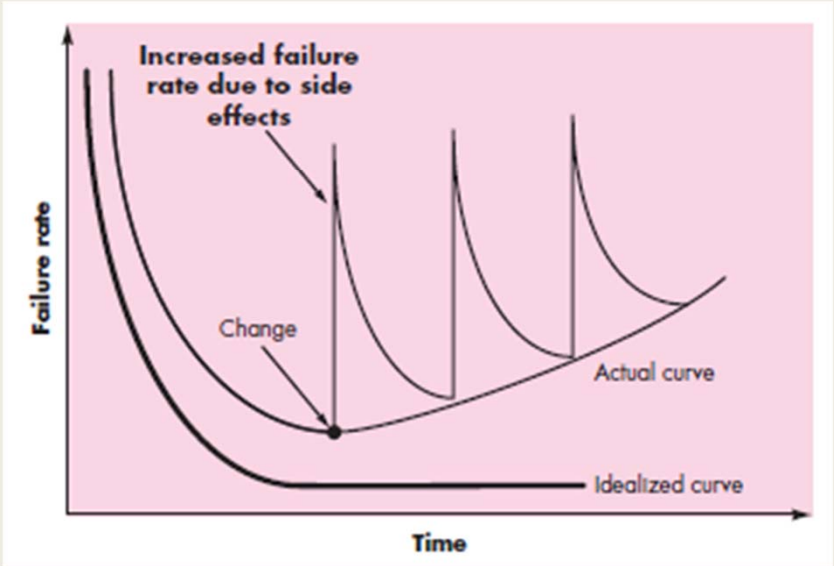
- Academic Integrity
  - Will strictly follow the academic integrity standard per university policy. No exception.
  - Use your common sense and if you have any doubt about this policy, let us address ASAP.
  - Make good use of office hours to gain assistance with homework or projects.
- Disability Accommodation
  - Some paper work with Access Center.

What is Software

- Software is (programs, data, and descriptive information)
- (1) instructions (computer programs) that when executed provide desired features, function, and performance;
- (2) data structures that enable the programs to adequately manipulate information and
- (3) documentation that describes the operation and use of the programs.

Wear vs. Deterioration

- Software does not wear out, yet it deteriorates



12

Software Diversity

- System software
- Application software
- Engineering/scientific software
- Embedded software
- Product-line software
- WebApps
- AI software

Software Diversity

- Legacy software
  - Longevity
  - Criticality
- Other categorization
  - Open world computing (pervasive, distributed)
  - Open source (free)
  - Ubiquitous computing (wireless)
  - Net-sourcing (web-centered)

WebApps

- Network intensiveness
- Concurrency
- Unpredictable load
- Performance
- Availability
- Data driven
- Continuous evolution
- Immediacy
- Security

15

Software: the changing nature

- Web-based systems and applications have evolved
- Mobile applications present new challenges as apps migrate to a wide array of platforms
- Cloud computing will transform the way in which software is delivered and the environment in which it exists
- Product line software offers potential efficiencies in the manner in which software is built

16

Software Engineering v.s. Ad hoc development

Software Engineering v.s. Ad hoc development

What *not* works:

hire a bunch of developers, plug them into a project, and then "set and forget", hoping that software will come out with decent quality;

What works:

- Clearly-communicated, unambiguous requirements. (Chapters 5-7);
- Clean and well-documented functional and architectural designs (Chapters 8-12);
- Adequate quality assurance (Chapters 17-19, 23);
- Project Management (Chapters 24).

18

How must people process and problem be managed during the software process

How to come up with reliable project plan

How to select a set of software engineering work tasks

Software Engineering v.s. Ad hoc development

You ought to have, (cont'd)

- Right contingent plan and risk management (Chapters 28)
- Realistic schedule (Chapter 27).
- And above all, a right development process (Chapters 2).
  - Having a right process built in is worth half of journey for quality software.
  - For other half, you need to have right techniques, methods, and tools.

19

Software risks involves uncertainty. Your project plan may go as planned or unexpected surprises, unw

What is Software Engineering

- Classic Definition (1969)

> *"The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines."*

- IEEE Definition (1993)

> *"Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is the application of engineering to software. (2) The study of approaches as in (1)."*

20

What is Software Engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software engineers should adopt a systematic and organised approach to their work and use appropriate tools and techniques depending on the problem to be solved, the development constraints and the resources available.

Software Engineering v.s. other Engineering Disciplines

- Software is designed, not manufactured
  - General principle of quality control such as six sigma are not working well in software development.
  - Component re-usability is challenging.
- Software is based on discrete math
  - Butterfly effect: small errors can have big consequences.
  - Over-engineering does not work well

Software Engineering v.s. other Engineering Disciplines

- Software is malleable
  - Can apply to a large variety of problems.
  - Development process shall also be adaptive for domain-specific applications.
- Software doesn't wear out
  - However, the changes to software may introduce new bugs while adding new features.

Summary

- Course logistics
- Software
  - definition
  - changing nature
  - domains (web apps)
- Software engineering
  - Comparison to ad-hoc development
  - definition