CS 30700: Software Engineering

Lecture 1: Course Introduction, Software, Software Engineering, and Ethics

Prof. Jeff Turkstra
Lecture 01

- Objectives
- Course policies
- Software
- Software Engineering
- Ethics
Instructor

Prof. Jeff Turkstra
HAAS 128
jeff@purdue.edu
49-63088
Office Hours TBD
Project coordinators

Keehwan Park – park451@purdue.edu
Hasini Gunasinghe - huralali@purdue.edu
Shreya Inamdar – inamdar@purdue.edu
Akhil Israni – isrania@purdue.edu
Ayush Jain – jain207@purdue.edu
Abid Kaisani – akaisani@purdue.edu
Aviral Mansingka – amansin@purdue.edu
Harsh Parakh – hparakh@purdue.edu
Nihanshu Purohit – npurohit@purdue.edu
Anoop Ranganathan Santhosh – asanthos@purdue.edu
Yashkumar Shiroya – yshiroya@purdue.edu
Tori Shurman – vshurman@purdue.edu
Course schedule

- On the website
  - http://courses.cs.purdue.edu/cs30700:fall17:start
- Along with Syllabus
- Lectures may be different, roughly same schedule
- Note the deadlines
Teams

- First deadline: August 24, Team Assignments due
  - Email them to jeff@purdue.edu
  - If I don’t receive an email, I will assume you are withdrawing from the course
Course description

- Introduce fundamental principles, techniques, and tools used in the design of modern industrial-strength software systems
- Provide an opportunity to work in small teams
- Assist in sharpening of documentation and presentation skills
Objectives

- To understand the software development process
- To understand the tradeoffs between current software life cycle models
- To use current tools and methods to plan, analyze, design, test, measure, and manage software projects
To learn about and use version control systems
To understand that good people are one of, if not the most important, requirements for successful projects
To learn how to work on a team project
Homework

- Given as needed to support class material
- All assignments will be given out in class
- Due on the date given on the assignment
Quizzes

- Unannounced 5 to 10 minute quizzes
- Due in lecture
  - 8.5” x 11” standard sheet of paper
- Score of 0 if absent
- Lowest score will be dropped
# Project

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Charter</td>
<td>5%</td>
</tr>
<tr>
<td>Requirements</td>
<td>15%</td>
</tr>
<tr>
<td>Design Document</td>
<td>15%</td>
</tr>
<tr>
<td>Sprint 1 Planning Document</td>
<td>7%</td>
</tr>
<tr>
<td>Sprint 1 Review</td>
<td>10%</td>
</tr>
<tr>
<td>Sprint 1 Retrospective</td>
<td>3%</td>
</tr>
<tr>
<td>Sprint 2 Planning Document</td>
<td>7%</td>
</tr>
<tr>
<td>Sprint 2 Review</td>
<td>10%</td>
</tr>
<tr>
<td>Sprint 2 Retrospective</td>
<td>3%</td>
</tr>
<tr>
<td>Sprint 3 Planning Document</td>
<td>7%</td>
</tr>
<tr>
<td>Sprint 3 Review</td>
<td>15%</td>
</tr>
<tr>
<td>Sprint 3 Retrospective</td>
<td>3%</td>
</tr>
</tbody>
</table>

**Includes Final Project Presentation and Demonstration**

* Subject to change!
Multipliers

- Two team evaluations
  - after Sprint 1 and after Sprint 3
- Each member rates the others
  - See website for format
- Assign an average of ten points per member
- Everyone did their job and contributed? Everyone gets 10.
- Include reasons for your ratings
Factors

- Contribution – did they contribute productively to team discussion and work?
- Reliability – did they get the work done on time and as promised?
- Respect – did this person encourage others to contribute their ideas, did they listen well?
- Flexibility – was this person flexible and helpful when disagreements occurred?
Your multiplier

- Every day past the deadline, your multiplier decreases by 0.05
- Example
  5 people: 11.2, 10, 9.4, 9.8
  Your multiplier:
  \[(11.2 + 10 + 9.4 + 9.8)/40 = 1.01\]
  Second eval: 10.2, 11.1, 10.8, 10.7
  \[(10.2 + 11.1 + 10 + 10.7)/40 = 1.07\]
  Final multiplier:
  \[(1.01 + 1.07)/2 = 1.04\]
Evaluation

- Confidential
  - At no point will any person be told any other member’s ranking of them or anyone else

- Questions
  - Contact Prof. Turkstra
Grading rubrics

- Available on course website
  - [http://courses.cs.purdue.edu/cs30700:fall17:rubrics](http://courses.cs.purdue.edu/cs30700:fall17:rubrics)

- Subject to change with advanced notice

- Sample documents from previous semesters
Grades

Project  65%
Midterm Exam  25%
In-Class Quizzes  5%
Homework  5%

Grading issues will be addressed as they occur. Not at the end of the semester.
Sample projects

- Also on the website
  - http://courses.cs.purdue.edu/cs30700:fall17:samples
Questions and contact

- **Piazza**
  - [https://piazza.com/purdue/fall2017/cs30700](https://piazza.com/purdue/fall2017/cs30700)

- **Email**
  - [http://courses.cs.purdue.edu/cs30700:fall17:contacts](http://courses.cs.purdue.edu/cs30700:fall17:contacts)
Based heavily on “Object-Oriented Software Engineering: Practical Software Development using UML and Java”
- http://www.lloseng.com/
- Prof. Fred Mowle’s software engineering slides
Software

“Today the ‘software’ comprising the carefully planned interpretive routines, compilers, and other aspects of automative programming are at least as important to the modern electronic calculator as its ‘hardware’ of tubes, transistors, wires, tapes, and the like.”

Software

- Computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system.
The Nature of Software...

- **Software is intangible**
  - Hard to understand (or even explain) the development effort required

- **Software is easily reproducible**
  - Cost is in its *development*
    - in other engineering products, manufacturing often the costly stage

- **Software development is labor-intensive**
  - Difficult to automate
- Untrained people can hack something together
  - Often quickly
  - Quality problems are hard to determine
- Software is easy to modify
  - People make changes without fully understanding effects
- Software does not ‘wear out’
  - It *deteriorates* by having its design changed:
    - Erroneously, or
    - In ways that were not anticipated
Today

- Demand for software is high and rising
- Producing quality software is (still) a challenge
- Software should be ‘engineered’
  - Not just hacked together
Types of Software

- **Applications**
  - Business, CAD, databases, spreadsheets, simulations, video games, word processors
  - Compilers, middleware

- **System**
  - Operating systems
  - Device drivers
  - Privileged utilities
- **Embedded**
  - Firmware
  - Microcode
- **Real time**
  - Control and monitoring
  - Must react within some $\Delta T$
  - Often safety critical
Development

- Custom
  - For a specific customer

- Generic
  - Sold on open market
  - Commercial Off the Shelf (COTS)
  - “Shrink-wrapped”

- Embedded
  - Tied closely to hardware
  - Harder to modify
Example

- Microfluidics
  - Firmware on PICs
  - Device driver for I²C
  - Operating system for BeagleBoard
    - Linux
  - Userland API that relied on I²C
  - Application environment that invoked API
Software Engineering

IEEE: (1) the application of a systematic, disciplined, quantifiable approach to the development, operation, maintenance of software; that is, the application of engineering to software. (2) the study of approaches as in (1)
Solving Problems

- The **goal** of software engineering
- Within constraints
  - Cost
  - Time
  - Customer
  - Others
- And with rigor
Solutions

- Require effective communication
  - “I know you believe you understood what you think I said, but I am not sure you realize that what you heard is not what I meant…”

- Sometimes one can
  - Assemble from existing components
    - Not always build from scratch
    - Avoid NIH - “Not Invented Here”
  - Buy, not assemble or build
Systematic development and evolution

- An engineering process involves applying well-understood techniques in an organized and disciplined way
- Many practices have been formally standardized
- Most development is evolutionary
Why?

- Software engineering techniques are needed because systems have constraints
  - Time constraints
  - Knowledge/capacity constraints
  - Finite resources
  - etc
- Teamwork and coordination are required
  - Key challenge: dividing up the work and ensuring that all parts work together properly
- Necessity for “high quality”
Software Engineering

- The term Software Engineering was coined in 1968
  - People began to suggest that the principles of engineering should be applied to software development
- Engineering is a licensed profession
  - In order to protect the public
  - Engineers design artifacts following well-accepted practices which involve the application of science, mathematics, economics, ...
  - Ethical practice is also a key tenet of the profession
- In many countries, software engineering does not (yet) require an engineering exam or license
Differences with other Engineering Fields

- Foundations are primarily in computer science
  - As opposed to the natural sciences
- Focus on discrete rather than continuous mathematics
- Concentrates on abstract/logical entities instead of concrete/physical artifacts
- No “manufacturing” phase
- “Maintenance” primarily refers to continued development, or evolution
  - Not conventional wear and tear
Commonalities

- Engineers proceed by making a series of decisions, carefully evaluating options, and choosing an approach at each decision-point that is appropriate for the current task in the current context.

- Appropriateness can be judged by trade-off analysis, which balances costs against benefits.
Engineers measure things, and when appropriate, work quantitatively; they calibrate and validate their measurements; and they use approximations based on experience and empirical data.
Engineers emphasize the use of a disciplined process when creating a design and can operate effectively as part of a team in doing so.

Engineers can have multiple roles:
- Research, development, design, production, testing, construction, operations, management, and others such as sales, consulting, and teaching.
Engineers use tools to apply processes **systematically**. Therefore, the choice and use of appropriate tools is key to engineering.

Engineers, via their **professional societies**, advance by the development and validation of principles, standards, and best practices.
Ethics

- A theory or system of moral values
- The principles of conduct governing an individual or group
WALLY, I DISCOVERED A DEADLY SAFETY FLAW IN OUR PRODUCT. WHO SHOULD I INFORM?

NO ONE. THE STOCK WOULD PLUNGE AND WE'D HAVE MASSIVE LAYOFFS. YOUR CAREER WOULD BE RUINED.

BUT MY NEGLIGENCE COULD CAUSE THE DEATHS OF A DOZEN CUSTOMERS.

THE FIRST DOZEN IS ALWAYS THE HARDEST.
Engineering ethics

- Well-developed area of professional ethics
  - Like medical and legal ethics
- Engineers today are expected to both learn about and live up to ethical standards as a condition of their membership in the profession
Ethics in Software Engineering

- Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force
- http://www.acm.org/about/se-code
Short version

“Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to health, safety and welfare of the public, software engineers adhere to the following Eight Principles:”
Public

Software engineers shall act consistently with the public interest.
Client and Employer

Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest
Product

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
Judgment

Software engineers shall maintain integrity and independence in their professional judgment.
Management

Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
Profession

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
Colleagues

Software engineers shall be fair and supportive of their colleagues.
Self

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.
Responsibility

- Software engineers have significant opportunities to do good or cause harm
  - Also to enable or influence others to do good or cause harm
- Software engineers must commit themselves to making software engineering a beneficial and respected profession
National Society of Professional Engineers

“to hold paramount the safety, health, and welfare of the public.”
Stakeholders

- **Users**
  - Those who *use* the software

- **Customers (Project Owners)**
  - Those who *pay* for the software

- **Software Developers (Software Engineers)**
  - Those who *create* and *maintain* the software

- **Development Managers (Project Coordinators)**
  - Those who *supervise* the software development process
Quality

- Conformance to requirements
  - Crosby
- Fitness for use
  - Japan
- I Know It When I See It
  - John Guaspari, 1985
- Value to someone
  - Weinberg
Japan

- Business is done by reputation, not always by contract
- Something goes wrong after release?
  - Lose key currency of business: credibility and reputation
Questions?