CS 30700: Software Engineering

Lecture 1: Course Introduction, Software, and Software Engineering

Prof. Jeff Turkstra
Lecture 01

- Objectives
- Course policies
- Software
Instructor

Prof. Jeff Turkstra
HAAS 128
jeff@purdue.edu
49-63088
MWF 12:30pm – 1:30pm
TTh 3:00pm - 4:20pm
Project coordinators

Keehwan Park – park451@purdue.edu
Hasini Gunasinghe - huralali@purdue.edu
Anoop Ranganathan Santhosh – asanthos@purdue.edu
Garrick Buckley – gbuckle@purdue.edu
Matthew Page – page38@purdue.edu
Harsh Pandey – hpandey@purdue.edu
Sahil Pujari – pujari@purdue.edu
Asmaa Sallam – asallam@purdue.edu
Tori Shurman – vshurman@purdue.edu
Course schedule

- On the website
- Lectures may be different, roughly same schedule
- Note the deadlines
Teams

- First deadline: **January 13**, 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
  - Via email – jeff@purdue.edu
  - 8.5” x 11” standard sheet of paper
- Score of 0 if absent
- Lowest score will be dropped
Project

Project Charter ........................................5%
Requirements ........................................15%
Design Document .....................................15%
Sprint 1 Planning Document ......................7%
Sprint 1 Review ........................................10%
Sprint 1 Retrospective ..............................3%
Sprint 2 Planning Document ......................7%
Sprint 2 Review ........................................10%
Sprint 2 Retrospective ..............................3%
Sprint 3 Planning Document ......................7%
Sprint 3 Review ........................................15%
** Includes Final Project Presentation and Demonstration
Sprint 3 Retrospective ..............................3%
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:
  \[
  \frac{(11.2+10+9.4+9.8)}{40} = 1.01
  \]

  Second eval: 10.2, 11.1, 10.8, 10.7
  \[
  \frac{(10.2+11.1+10+10.7)}{40} = 1.07
  \]

  Final multiplier:
  \[
  \frac{(1.01 + 1.07)}{2} = 1.04
  \]
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

- 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:spring17:samples

- Heart rate deceleration analysis software to help researchers in autism, speech, and nutritional studies
Questions and contact

- Piazza
  - https://piazza.com/purdue/spring2017/cs30700
- Email
  - http://courses.cs.purdue.edu/cs30700:spring17:contacts
Slides

- 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