

CS 350: Introduction to Software Engineering

Slide Set 1
C. M. Overstreet
Old Dominion University
Spring 2005

Lecture overview

- Announcements
- Class expectations
 - Structure, grading, project, logistics, semester structure
- PSP: what's it about
- TSP: what's it about

CS 350: Intro to SE

Texts: 1) *Introduction to the Personal Software Process*, Watts S. Humphrey, Addison-Wesley, 1997.
2) *Introduction to the Team Software Process*, Watts S. Humphrey, Addison-Wesley, 2000 (not used till we finish PSP text).
3) *Guidance and Control Software Development Specification*, Ed Withers (RTI), Bernice Becher (Lockheed), NASA Langley Research Center, Hampton, VA 23681 (available at Monarch Copy Center, Webb Center)

Miscellaneous Class Information

- Class material is available on web (www.cs.odu.edu/~cmo under cs350)
- Class is time consuming
 - But more time spent on process than coding
- Reading assignments:
 - PSP, chapters 1, 2, & 3.

Recitations - 1

- Recitations meet on Fridays
 - But not this Friday!

Recitations - 2

- Recitations:
 - You must register for one
 - You must attend
 - Will be used for:
 - Group meetings (later)
 - Project discussions
 - Covering some software tools
 - Reviews of class performance on programming
 - Your data compared to class averages
 - Answering questions

Announcements

- Check www.acm.org/technews/current/homepage.html
 - Industry trends
 - Microsoft's Sender ID technology at odds with open-source licenses
 - E-books taking off?
 - Supercomputing making a comeback?
 - Fewer students enrolling in IT fields
 - Top online computing degrees
- You must have a CS dept account:
 - Go to www.cs.odu.edu, pick **Online Services**, then select **Account Creation**.

General Information

- Prerequisites
 - CS 361 or CS 330
 - UNIX exposure
- Helpful background
 - Some simple statistics (but we'll cover in class as needed)

Course Overview

- Activities:
 - 3 programming assignments, PSP based
 - 1 team project, TSP based
 - 2 in-class exams & comprehensive final
 - recitation/class assignments
- Grading:

● Individual projects:	25%
● Team project:	20%
● In-class exams:	25%
● Recitation/class assignments:	10%
● Final exam:	20%

From the Syllabus

- Honor code
- Lateness policy
- Special needs
- Read the syllabus!

Slides

- Available before class on Web
- Slides are OUTLINE only.
 - For content, read textbooks, assigned readings
 - For content, come to class
 - If class is missed, get GOOD notes from class member

Structure/purpose of class projects

- Not about programming
 - Focus is on the process used to develop software products. Learning the process involves using it to development some software.
- Four individual projects involving coding
 - However you will spend more time on process steps than coding
 - Industry data: of total project time, 15% is coding
 - Emphasize your Personal Software Process
- One team project
 - Emphasizes the [Team Software Process](#)

Course Objectives 1: PSP

- Introduce you to a process-based approach to developing software
- Show you how to measure and analyze your **Personal Software Process (PSP)**
- Improve your software development skills:
 - faster development
 - fewer errors (i.e. better software)
 - more predictable (more accurate estimates of time required to complete a project)
- Show you how to use data to improve your personal performance

Course Objectives 2: TSP

- Working on teams requires specific skills
- TSP goals:
 - Understand how to build teams
 - Understand different team roles
 - Understand how to work on teams

Quick Survey (Quiz 1!)

- What's best prog. language?
 - a. C++
 - b. Java
 - c. Perl
 - d. Visual Basic
 - e. Other
- You prefer to use:
 - a. C++
 - b. Java
 - c. Other
- Your largest program:
 - a. < 500 loc
 - b. > 500 & < 1 kloc
 - c. > 1 & < 10 kloc
 - d. > 10 & < 100 kloc
 - e. > 100 kloc
- What's a kloc?
 - a. comments inc.?
 - b. declarations?
 - c. only exec. stmts?
 - d. number of CRs?

Mail to cmo@cs.odu.edu

PSP Chapter 1 Overview

- PSP overview, intro
 - Costs and benefits
 - History: capability maturity model (CMM)
 - The CMM and the PSP
- Time management
 - Past can help predict future sometimes
- Tracking time
 - Use past time used to predict future time needs

Difference in CS and SE - 1

- SE:
 - Mgmt: How do you predict costs, project time, determine if on budget & schedule?
 - Economics: What's the cheapest way to build it?
 - Reliability, etc: How do you make it reliable?
- CS:
 - What can computers do?
 - How do you make computers do things efficiently?
- Some people think CS is part of SE, others that SE is part of CS. Which is correct?

Other Differences

- Frequent student view: I assume (or hope) the code I wrote works.
- Typical professional view: I assume the code doesn't work (no matter who wrote it).
 - Someone (usually me) must prove it does before I let it mess other things up.
 - In many organizations, correctness of a new component must be demonstrated before it is incorporated into project base.

SE emphasis on metrics & data

- It's not engineering if you can't measure and predict.
- You can't predict if without data!

PSP composed of this week's "best-known practices"

- May be different next year; probably very different in 10 years as we learn more.
- Approaches similar to PSP are widely advocated, and often used in industry.
- Past perception: most software organizations use poor practices resulting in overly expensive, late, and unreliable software.
 - This must change otherwise more software jobs will move to India!
- Now required of DoD software contractors.

Unpleasant Facts of life:

- Some PSP aspects I don't like. Some I don't believe.
- You may not either, but after this course, you should be knowledgeable.
- Future of software development?
 - India has bright, well-trained software developers, earn \$15k rather than \$80k per year
- It's all about costs, predictability, and quality!
- In PSP
 - If you can find a better way to reduce costs, improve quality and increase predictability – and can prove it works, you should use it.
 - And you can make a lot of money!!!!

Process!

- Current industry belief: the process used to develop software has significant impact on quality and costs.
- Things like programming language or design notation mostly don't
 - C++, C#, PHP, UML will be replaced
- So get the **process** right!

PSP Principles - 1

- The quality of a software system is governed by the quality of its worst components.
- The quality of a software component is governed by the individual who developed it.
- This is governed by that person's:
 - Knowledge
 - Discipline
 - Commitment

Another Warning:

- All people **hate** data collection about their activities!
- Purpose of PSP forms is to
 - Collect data
 - Guide process
 - "Enforce" process
- Must have the data!
 - Measure quality
 - Measure productivity
 - Help develop future projects budgets, schedules

PSP Principles - 2

- As software professionals you should know your own performance.
- You should measure, track, and analyze your work.
- You should learn from your performance variations.
- You should incorporate these lessons in your personal practices.

PSP Subgoals

- Clean compile on first compile
- Successful execution of all test data on first run
- Why?
 - Faster (less of your time spent)
 - Clean compile indicates careful code review was performed
 - Better (testing often weak)
 - Code reviews find bugs not found through testing

With a Stable PSP

- You can
 - Estimate and plan your work
 - Meet your commitments
 - Resist unreasonable commitment pressures
- You will also
 - Understand your ability
 - Be better able to improve
 - Know if you are improving

A PSP Also Provides

- A proven basis for developing and practicing industrial-strength personal disciplines
- A discipline that shows you how to improve your personal process
- The data to continually improve the productivity, quality, and predictability of your work

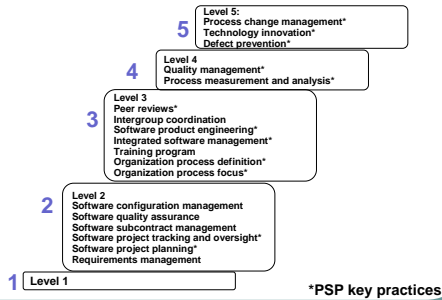
What is a PSP?

- A personal process for developing software
 - Defined steps
 - Forms
 - Standards
- A measurement and analyses framework to help you characterize your process
- A defined procedure to help you improve your performance (time, quality)

The CMM and the PSP - 1

- The capability maturity model (CMM) was developed by the Software Engineering Institute (SEI) at CMU with the help of leading software groups.
- The CMM characterizes effective large-scale software practices. DoD oriented.
- The PSP:
 - Applies the CMM
 - But is for individual work

The CMM and PSP - 2



The CMM and the PSP - 3

- The CMM provides a framework for effective process management.
- It assumes that the software professionals will follow disciplined personal methods.
- The PSP provides the framework for disciplined individual work.
- It assumes effective process management.

Why CMM (& PSP)?

- Too many missed software deadlines
 - Too often by a factor of 3
- Poor quality software
 - Widely used metrics
 - Errors/kloc, or
 - Customer observed errors per month
 - Some software failures have killed people (but luckily not often)
- Most other engineering areas seem to do better
 - Why not with software?

The PSP Metaprocess

- A process consists of a defined sequence of a steps.
- For PSP, each step consists of:
 - A set of entry criteria - step cannot start until all entry conditions are satisfied
 - A sequence of activities - this is the work to be done in this step
 - A set of exit criteria - you're not done with the step until these are satisfied

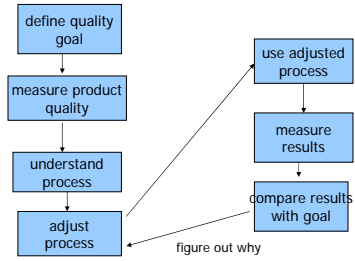
PSP Overview - 1

- The PSP is introduced in steps
- You write small programs to practice PSP principles
- You gather and analyze data on your work
- You use these and analyses to improve your work

PSP Objectives

- Better planning
 - What steps and how long each will take
- Work according to plan
 - Meet deadlines
- Improve quality
 - Fewer bugs in software

Goal: Improve SE Process



Time Management

Base time estimates on real data

- People remember how time is spent poorly
 - Difficult activities seem to take longer, fun activities less
- For better accuracy, need time logs
 - May seem like overkill
 - Get used to it! In many orgs, people must log their time so project charges are documented

Logging time

- Categorize activities by type
- Record time spent in each activity
- Record in a standard way
- Make it as easy as possible to log time

Engineering notebook

- In many organizations, professionals rely on an "engineering notebook."
- Lots of different ways of doing this; textbook has example formats
- Spiral binders handy for this. Maybe PDAs
- Can be used for many class related topics (notes, ideas for proj. solutions)
- But now, focus is for your time log

Forms? Yuck!!!!

- Use what's in the text
 - Since they are based on real experience involving lots of people
- After you have experience in these,
 - Feel free to improve since
 - We're all different
 - Have different needs
- Nice to have things in common format
