Spring 2008

CEG 463/663-01: The Personal Software Development Process

John A. Reisner
Wright State University - Main Campus, john.reisner@wright.edu

Wright State University

CEG 463/663: The Personal Software Development Process

Spring Quarter, 2008

Course Description

In this course, you will learn about more about one particular way to address some of the challenges and issues associated with successful software development. Specifically, you will learn about (and use) the *Personal Software Process* (PSP), designed to help individual software practitioners become more adept at their craft through the use of project planning, project tracking, defect analysis, review and verification activities, software measurement, and process management. This course—and the PSP—are somewhat unique in that they aim to help software engineers become more successful, not by examining issues associated with *large-scale* development (as is the case with many software engineering courses), but by scaling *down* the software project efforts. These small projects are designed to provide participants with an opportunity to examine their own practices, strengths, and weaknesses at a minute level of detail. The findings from this analysis are meant to provide a foundation from which one can better succeed once participating with a team of practitioners striving to build a large-scale software system on-time and within budget.

Course Goals, Textbook, and Other Prerequisites

The course textbook is ***A Discipline for Software Engineering***, by Watts S. Humphrey, published by Addison-Wesley, 1995. This is a required textbook for this course.

Prerequisites: CEG 460 or equivalent. Moreover, this class has weekly programming assignments, so students should be skilled in at least one high-order programming language, being able to write, compile and run programs in this language without any outside help.

In this course, students will be writing several computer programs. They will be expected to write detailed plans and estimates prior to writing this software, and track their time during the effort, so that actual work data can be compared with initial planning estimates. This planning and analysis are the mainstays of the course; they will be used so that students can evaluate and improve their own software engineering capabilities. By learning to hone these skills in an academic setting, students can theoretically decrease the amount of trial-and-error discovery occurring in the workplace, when such lessons are much more costly to learn.

Learning Outcomes

By the conclusion of this course, students should be able to

- Explain the Personal Software Process (PSP)
- Describe the goals of the PSP
- Explain why the PSP can lead to improved quality and better schedule estimation
- Use aspects of the PSP to quantitatively evaluate software quality
- Use the PSP to build software and establish personal baseline metrics
- Plan software development activities in a consistent manner
- Build software according to their documented plans
- Become more motivated to strive toward producing high-quality software products
- Become more proficient in making more accurate personal estimates

Course Format

This course will be taught in a collaborative manner—meaning that, during class time, much material will be discussed among the class, rather than presented in a strict lecture format. Students will be expected to have done any readings, research, or homework assigned prior to the lecture, so that they will be able to contribute to the discussion in an informed, intelligent, and constructive manner. The lessons you learn during your project exercises should be shared with your fellow classmates, in order to enrich the educational experience for all registered students. Additionally, WebCT will be used to disseminate related reading materials, and WebCT's discussion board will be used as a way to report progress, and to promote out-of-class discussions to relevant topics.

Laptop Policy

Use of laptops during class time is prohibited. (Too often, when a student uses a laptop during lecture, it is being used for something *other* than note-taking). I make concerted efforts each week to prepare an interesting lecture; I expect students to do their best to remain interested. I also want students to contribute with their own opinions and ideas, so it's best to eliminate unnecessary distractions.

Course Grading

Course activities will be weighted as follows:

20%	Weekly Programming Assignments – Software Quality
40%	Weekly Programming Assignments – Planning, Measurements, and Documentation
20%	Midterm Exam
20%	Final Exam

As previously mentioned, this course has weekly programming assignments. These assignments are worth 60% of the grade; but the quality of the software itself only accounts for just a fraction of this portion. The weekly planning and documentation—along with the associated metric collection—are more heavily weighted than the software itself. The two exams make up the rest of the course grade. The final exam will be cumulative.

In this class, much of the grading needs to be done subjectively. Satisfactory work is typically given a grade of 90. This 90 does not mean that you have "lost 10 points;" instead, it means you are receiving ample credit for satisfactory work. By assigning a grade of 90, I am then able to distinguish between work that is "good" and work that is "very good" or "excellent." (Better-than-satisfactory work is graded above a 90; truly superior work may earn a 100).

If submitted work indicates either a lack of understanding of basic concepts, or an apparent apathetic carelessness, then it will be graded as Unsatisfactory, and a numeric grade will be assigned accordingly. If the problem appears to stem form a misunderstanding the basic ideas, then I will usually provide some personal feedback, with the aim of helping you understand the material better.

Judging the relative "goodness" of a computer program (and perhaps more significantly, of a program with its accompanying write-up, test plan, and other documentation) is much like judging a figure skating routine. How do the Olympics deal with judging subjective competitions? By having a number of judges, whose scores are averaged (often after throwing out the high and low scores). Similarly, the most fair way to grade work in this class would be to use a panel of graders (however, as of now, I've yet to enlist the volunteer help of four faculty members willing to assist me). Still, after examining a dozen or so assignments on the same topic, I generally get a pretty good idea of which submissions are better prepared than others. The ones that are "more than satisfactory" receive grades such as 92, 95, or 97, while the truly superior works will receive an E (100). Again, don't ask me what was "wrong" if your grade is a 90; a 90 means you completed the assignment in a satisfactory manner.

I also reserve the right to deduct points for late assignments, depending upon how late the work was turned in, how much advanced notice I was given about when I could expect the work, and any extenuating circumstances that may have applied.

Final course grades: A = 92 and above; B = 85 thru 91; C = 75 thru 84; D = 60 thru 74; F = 59 or less; however, this scale may be (and frequently is) curved in favor of the students.

Instructor Contact Info

John Reisner (Office Hours by Appointment)
Work Phone: 255-3636 x7422 (Wright-Patterson AFB)
email: **john.reisner@wright.edu** (if you want a timely response, please CC: **john.reisner@afit.edu**)
→ or use WebCT email tool

The instructor is an adjunct faculty member. Most contact will be done via WebCT, or in after-class discussions. Other meetings can be arranged.

If, at any time, you are having trouble accessing course materials via WebCT, please send me an email immediately. The sooner I am aware of a problem, the sooner I can fix it. Because I have the instructor's view of WebCT, I sometimes mistakenly believe materials have been posted when in fact students cannot access them. Your support in this matter is greatly appreciated.

Course Schedule (subject to change)

Week	Lesson	Date	Lesson Focus	Assigned Reading	Programming Assignment	PSP No.
1	1	Tue Mar 27	Intro to SW Eng & the PSP	Chapters 1 & 2	1 Roll a pair of dice and sum their values. Add a "doubles flag" and a "doubles counter." **Note**: This program is easy, but the testing is difficult!	PSP 0
	2	Thu Mar 29	Planning - Process	Chapter 3		
2	3	Tue Apr 3	Planning - Size	Chapters 4 & 5	2 Use the dice to have tokens traverse a Monopoly board; have multiple players traverse the board; allow extra turns with doubles	PSP 0.1
	4	Thu Apr 5	Planning - Estimates	Chapter 6 (thru Section 6.5)		
3	5	Tue Apr 10	Planning - Tracking	Chapter 6 (Section 6.6 to end)	3 Add $1500 start money; add property ownership (allow players to purchase unowned properties if they have sufficient funds)	PSP 0.1
	6	Thu Apr 12	Measuring - Goals	Chapter 7 (thru Section 7.4)		
4	7	Tue Apr 17	Measuring - Data	Chapter 7 (Section 7.5 to end)	4 Have players pay Luxury & Income Taxes, plus basic rents on owned properties, utilities and railroads; add $200 for passing GO	PSP 1
	8	Thu Apr 19	Reviews - Design	Chapter 8 (thru Section 8.6)		
5	9	Tue Apr 24	Reviews - Code	Chapter 8 (Section 8.7 to end)	5 Recognize monopolies; allow improvements (houses and hotels); charge adjusted rent as appropriate	PSP 1.01
	10	Thu Apr 26	NO LESSON	MIDTERM EXAM		
6	10	Tue May 1	Quality - Strategy	Chapter 9 (thru Section 9.5)	6 Add functionality for all Community Chest cards; add jail rules (into jail and out of jail)	PSP 1.1
	11	Thu May 3	Quality - Defects	Chapter 9 (Section 9.6 to end)		
7	12	Tue May 8	Design	Chapter 10	7 Add functionality for all Chance cards; allow players to sell houses and hotels back to the bank	PSP 2
	13	Thu May 10	Scaling - Abstraction	Chapter 11 (thru Section 11.4)		
8	14	Tue May 15	Scaling – PSP3	Chapter 11 (Section 11.5 to end)	8 Incorporate trading and auctioning capabilities	No turn-in: multiple iterations for PSP 3
	15	Thu May 17	Design Verification	Chapter 12		
9	16	Tue May 22	S/W Process - Defined	Chapter 13 (thru Section 13.5)	9 Allow players to mortgage and unmortgage properties; incorporate bankruptcy rules	PSP 3 for Wks 8 & 9 due on Thursday
	17	Thu May 24	S/W Process - Evolved	Chapter 13 (Section 13.6 to end)		
10	18	Tue May 29	Using PSP	Chapter 14 (thru Section 14.4)	10 Get software product ready for in-class demo – be ready to demonstrate ALL functionality on demand!	
	19	Thu May 31	Your Future	Chapter 14 (Section 14.5 to end)		

Rather than using the assignments provided in the text, students will write a software program that plays the game of Monopoly. Each weekly assignment will add to the previous week's work. Students can use the language of their choice. The game need not be one with a graphical user interface and display; the state of the game can be displayed in text format. NOTE: Do **not** "program ahead!" (You may look ahead for design purposes, but do not code any more than what is required to incorporate the functionality prescribed for each week).
MOST IMPORTANT: Do NOT start any programming or design until you have completed your Project Plan *first*.

Weekly Turn-ins

Each week, before the start of the Tuesday class, you will be expected to turn in the following, **in this order**:

1. **PSP Worksheets** (the number of sheets will vary depending upon PSP version):
   a. Project Plan – This should be your "cover sheet." Include **TWO** copies of *this* worksheet each week.
   b. Time Recording Log, Defect Recording Log, other PSP sheets (one copy of each of these worksheets).
2. **Other documentation**: (weekly project description, enumerated requirements, design sketches, assumptions, overviews, summaries, clarifications, lessons learned, PIPs, etc.)
3. **Test Plan**: Test Cases & Results; to include Sample Test Runs & Screen Captures
4. **Source Code Listing**: A complete code listing for the entire program to date.