Spring 2010

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, 2010

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 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. Participants have 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.

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.

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. **NO LAPTOPS IN CLASS DURING LECTURE TIME.**

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.

Course Grading

Course activities will be weighted as follows:

50%	Weekly Programming Assignments – Planning, Measurements, and Documentation
20%	Midterm Exam
20%	Final Exam
10%	Overall quality of final program and turn-in

As previously mentioned, this course has weekly programming assignments. These assignments are worth 50% of the grade; but the quality of the software itself only accounts a small fraction of this grade. The weekly planning and documentation—along with the associated metric collection—are the primary means of assessing these assignments.

The two exams make up 40% of the course grade. The final exam will be cumulative.

The overall quality of the final program, which will be demo'd in class, makes up the last 10% of the grade.

Final course grades will be assigned at the instructor's discretion, after all grades have been calculated. Grades over 93 will be A, over 86 will be B, over 77 will be C, over 70 will be D, **although this scale can be (and typically is) curved**.

How to Do Well (or Poorly) in this Course

- Keep up with your work. There is a lot of work in this course, every week. Don't wait until the week is almost over before you begin your weekly assignments.
- There is a lot of programming in this class, but, remember, this not a programming class. Don't spend the bulk of your time programming, and then turn in sloppy PSP worksheets. Use the programming effort to learn about the PSP and its potential benefits.
- Emphasize planning early. Generally speaking, students who don't do look-ahead design from the outset have the most problems in the latter part of the project.
- Save some time for analysis. When you're done with your PSP worksheets, study them, analyze them, and see what conclusions you can derive.

Grading Assignments

Homework (programs) for this class will be ordinarily evaluated on a three-tier scale: Satisfactory, Unsatisfactory, or Exemplary. If your submission is Satisfactory, then your grade will be a 90. Don't think of a 90 as "losing 10 points;" think of it as getting ample credit for satisfactory work. When assignments are exceptionally well prepared, reflecting much insight, understanding and effort, that work is graded higher.

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 I think the problem lies with misunderstanding the basic ideas, then I will usually provide some personal feedback, with the aim of helping you understand the material better.

In short, work that is "more than satisfactory" will be graded above 90, while the truly superior works may receive 100 points. 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.

Course Schedule (subject to change)

Week	Lesson	Date	Lesson Focus	Assigned Reading	Programming Assignment	PSP No.
1	1	Tue Mar 27	Intro to PSP	Chapters 1 & 2	**1** Roll a pair of dice; have multiple tokens traverse a 40-space board with these dice; allow extra turns with doubles.	PSP 0
	2	Thu Mar 29	Planning - Process	Chapter 3		
2	3	Tue Apr 7	Planning - Size	Chapters 4 & 5	**2** Name Monopoly squares; add ownership; allow players to acquire unowned properties (assume ownership is acquired upon initial landing)	PSP 0.1
	4	Thu Apr 9	Planning - Estimates	Chapter 6 (thru Section 6.5)		
3	5	Tue Apr 14	Planning - Tracking	Chapter 6 (Section 6.6 to end)	**3** Add money, charge for property, charge for basic rent, add $200 for passing GO, pay Luxury & Income Taxes	PSP 0.1
	6	Thu Apr 16	Measuring - Goals	Chapter 7 (thru Section 7.4)		
4	7	Tue Apr 21	Measuring - Data	Chapter 7 (Section 7.5 to end)	**4** Recognize monopolies; allow improvements (houses and hotels); charge adjusted rent, to include for railroads and utilities.	PSP 1
	8	Thu Apr 23	Reviews - Design	Chapter 8 (thru Section 8.6)		
5	9	Tue Apr 28	Reviews - Code	Chapter 8 (Section 8.7 to end)	**5** Add auctioning capability when players elect to not buy property; incorporate Jail rules.	PSP 1.1
	10	Thu Apr 30	NO LESSON	MIDTERM EXAM		
6	10	Tue May 5	Quality - Strategy	Chapter 9 (thru Section 9.5)	**6 Week to refactor, catch-up, or get-ahead.**	PSP 1.1
	11	Thu May 7	Quality - Defects	Chapter 9 (Section 9.6 to end)		
7	12	Tue May 12	Design	Chapter 10	**7** Add functionality for all Chance and Community Chest cards	PSP 2
	13	Thu May 14	Scaling - Abstraction	Chapter 11 (thru Section 11.4)		
8	14	Tue May 19	Scaling – PSP3	Chapter 11 (Section 11.5 to end)	**8** Allow players to mortgage and unmortgage properties	PSP 2.1
	15	Thu May 21	Design Verification	Chapter 12		
9	16	Tue May 26	S/W Process - Defined	Chapter 13 (thru Section 13.5)	**9 & 10** Incorporate bankruptcy and end-of-game rules; debug, so that product is ready for in-class demo.	PSP 3
	17	Thu May 28	S/W Process - Evolved	Chapter 13 (Section 13.6 to end)		
10	18	Tue Jun 2	Using PSP	Chapter 14 (thru Section 14.4)		
	19	Thu Jun 4	Your Future	Chapter 14 (Section 14.5 to end)		

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, provided it is a compiled language. Note: the game need not be one with a graphical user interface and display; the state of the game can be displayed in text format.