

December 22, 2004

All future syllabi for this class will include specific information about undergraduate and graduate requirements.

## **CS 442-542 Syllabus -Fall 2004**

### **Software Development**

**Instructor:**

Kevin Reilly, Professor

**Class schedule:**

Mon & Wed 3:30-4:45 pm (435 Campbell Hall)

**Preq:**

CS303, minimum. Helpful information occurs in systems, theory and programming language courses; it would be good to have one or more these courses.

**Office Hours:**

Plan for time AFTER the class. **PI.** avoid BEFORE class unless you seek permission to meet. ..Other times, by appointment. .."Pre-Noon" is a good time. ...Location: 139 Campbell Hall, a lab across from my office (Room 141 ). (We may, later on, set aside some meeting times for help, esp., on project work.)

**Text:**

Schach, S. Object Oriented and Classical Software Engineering, (6th ed.), McGraw-Hill, Inc., 2005.

**Grading:**

There will be two equal-valued (40 pt.) midterm exams, the first near MidTerm, probably a little later (e.g., the week of Oct 18), and the second near the final, say, Dec 1. Tests will NOT be open-book, but you will be permitted a one-sheet (8.5" x 11") "Crib Sheet." (*This does not apply to any late examination! Expect a more difficult exam, also.*)

The exams typically will be configured so that G (542) students will be responsible for **ALL questions asked**. The UG (442) student's will be graded on the "best of" answers. For example, look for 50-point tests with UG's needing to answer for 40 points (or answer all but be graded on correct answers up to 40 points); meanwhile, G students sit for all 50 points, which means a harder test, since some of the questions beyond a core 40 points will be more difficult. (Scores adjust to 40 points for both groups of students, of course.)

The final exam period is primarily **PROJECT PRESENTATION TIME**; it's set for the regularly scheduled final exam time, Dec. 13th - 4:15. Projects can be presented, in part or in full, before this period, perhaps in (rare?) cases in the last days of the course. *Questions from you as listener* are welcome and will be noted for possible "bonus" points helpful *primarily* in border-line grade cases --- still much appreciated by past students. The project value is set at 20 points. Since project teams may be partially G and partially UG groups, we will use the **oral** exam part of this final exam to drill the G students with the most difficult questions. Since we encourage questions from attending students, their questions may be directed **first** to the **G** team members and **second** to **UG** students (for amplification, if felt needed).

**Course Material and Projects:**

The course will cover all the book in the sense that every chapter will receive a least one session's

worth of attention. We will proceed in sequence from Chap. 1 with a few remarks on the initial chapters in Part Two (to expedite prompt start-up on project work). There is approx. one chapter per week to cover and we will aim to keep on time, perhaps leaving some sections to student reading. NOTE: the book generally has good end-of-chapter summaries.

The student should focus on the scope and meaning and details of RADI for both test and project activity:

### **R = Requirements A = Analysis D = Design I = Implementation**

(Shoe-horn in the author's T =Testing during all the phases *and* perhaps as a separate phase. )

We will go thru **ALL** these phases as will your project, in the latter case each phase ending with formal documentation (per the book and instructions given in class), i.e., there will be four (4) write-up's involved and suggestions will be given to you as we proceed. Emphasis will be on brevity, use of diagrams, and other recommended procedures appropriate to a software engineering course. Offering **principal** documentation items on one page and offering an **opposing page of commentary** is our prime recommendation for a successful product in this course. **Commentary** might take the form of: **text** (e.g., explaining equations); **equations or pseudo codes** (e.g., related to "real" code); **data flow diagrams** in the simplified form your instructor will present in class; other more or less **casual diagrams** (sometimes referred to as "cartoons"); and **tables** (such as decision tables, e.g., to capture Osbert Oglevie's pricing algorithm, known at the start of the RADI effort from the non-automated system). In class, we may offer yet additional suggestions with usages of these and the ones just listed to help you see how to employ superb commentary in your work.

**Project topics promoted this term include** (each a team project with 3-4 member teams):

**1) Osbert Oglevie** (a partially completed example in the book, code available on the internet per the book's description): Several suggestions also occur in the book on how to complete the example and, even, to **add** entirely new departures (see examples in Chap 10-15). We highly encourage **your** VA ("Value Added). Modifications can be expected to require you to work in all four RADI areas along with testing;

**2) Ophelia's Oasis in the Amlet Desert:** Appendix A 's "Term Project," a hotel reservation system. This ostensibly is a more difficult project and a full R, A, D should be prepared with a good I demonstration, e.g., a significant portion backed by indicating how the "guts" of the I has been provided;

**3) A Jess project:** Jess is a Java-based Expert System Shell, and readily available materials support four (4) different examples of which the team can choose one (or more). You follow the *same* pattern as **1)** above, i.e., handle a base case (partly a "translation" from the discursive treatment in existing documentation) and seeking additions, etc. Again, code is provided for the base case so VA is v. important. the RADI scheme could be OO or Classical, or mixed, depending on your choices;

**or**

**4) An SQA Group project:** Here the team examines the R, A, D, and I documents from one or more teams (perhaps all four documents for one team and some selections from other teams). Getting info on the I phase can be difficult since I occurs near the course's end. Therefore, we will amend, as needed and appropriate, to concentrate on the first three phases. We look here for "schemes" on how to do SQA work (perhaps from existing CASE software, DB software, spreadsheets, etc.).

Ideas about teams will be presented in the class, e.g., related to each team "distributing" its

talent over the variety of skills needed: diagramming, programming, writing and talking, etc. Continue to THIMK: in project work, "Value Added" - is v. important. Organize for it ... emphasize it in writing & talking.