

**Student Originated Software  
Program Syllabus -- Fall 2003-2004**

| <b>Monday</b>                          | <b>Tuesday</b>  | <b>Wednesday</b>                      | <b>Thursday</b>  | <b>Friday</b>      |
|--|---|---------------------------------------|--|--------------------|
| 10-12:30 Lectures<br>OOP & OOAD<br>LH5 | 10-12 OOP lab<br>ACC  | 11-1:15 Lectures<br>OOP & OOAD<br>LH1 | 10-12 Seminar<br>LIB 1505<br>LIB 1507  | Preparation<br>Day |
| 1-4 OOAD Workshop<br>LIB 3500          | 1-3:00 Visitor/<br>Project Org<br>Lab I 1047<br>3-5:00 Team Mtg &<br>Open Lab (ACC) |                                       | 1-3:00 Project Org<br>LIB 1505, 1507<br>3-5:00 Team Time<br>Open Lab / (ACC) |                    |

Even the best efforts of computer users and software engineers have not alleviated critical software development problems: most software is late and over-budget, does not meet user needs or expectations, or is socially irresponsible. The “software engineering problem” is not just a matter of technology, but a problem of organization, psychology, artistic design, group dynamics and culture. In addition considerable knowledge and understanding of the application area is required to design, implement, deploy, support and maintain a successful system. Evergreen's *Student Originated Software* program is designed to address these issues and to prepare **students who already have learned the fundamentals of computer science (i.e., Data to Information or equivalent)** to face these problems.

We expect that, by the end of the academic year, successful students in the program will:

- Understand and gain practical experience in the software development life cycle.
- Learn technical skills necessary for the analysis, design and programming of software systems.
- Understand issues behind inherent difficulties in writing software, including the socio-cultural, political, and ethical milieu in which that software is written and used.
- Be able to work as part of an inter-dependent team.
- Be able to present technical material verbally and in written form to both large and small groups.
- Have completed a software project to use a jumping off place for work and further learning.

The primary vehicle for learning how to write software is a year-long **project** for an identified real-world customer or community, or around a particular focused topic such as compiler construction. Most teams will follow a development schedule similar to the following: Fall: identify a viable project, perform a preliminary analysis and a feasibility study. Winter: develop and evaluate a working prototype with user guides and design documents. Spring: refine the prototype; complete the programming; finalize user's guides, maintenance plans and installation; and evaluate the final system. Projects will be completed by the last week of spring quarter and demonstrated at a software fair. **Except for extreme and unusual circumstances, we expect that all students will work in a project team.**

Tuesdays and Thursdays, we will have sessions where we work together to organize project or hear from and talk to working professionals in the field. We will invite a range of folks; if you have suggestions for particular guests or for particular kinds of people, please let us know.

For **seminar**, we will consider the nature of software systems -- history, market, culture, and discipline.

**Program technical components** are designed to meet the program's learning objectives and to prepare students for project work; see individual program component syllabi for details. In the **fall** (OOAD, OOP, Case Study), we will concentrate on object-oriented analysis, design and programming, and an introduction to analysis and design and software engineering through a case study. In **winter**, students will choose between two technical components (likely database systems and parallel algorithms or systems programming (experience with the system call interface with an emphasis on process control, synchronization, and networking). Winter seminar will focus on software engineering practicing, incl. the “human” aspects of software. In **spring**, seminar and program lecture will be coordinated with the PLATO lecture series on the *Semantic Web*. A spring technical seminar will focus on a timely topic in

software development such as design patterns or domain specific languages. In winter and spring, a student team can propose to faculty learning a particular skill or gaining knowledge needed for their project (in lieu of a quarter's a technical component).

#### **Fall Quarter Books:**

- **Java:** Tim Budd's *Understanding Object Oriented Programming in Java*; And a Java reference such as Arnold, Gosling & Holmes' *The Java Programming Language*.
- **OOP Programming Practices:** Kent Beck's *Smalltalk Best Practice Patterns*
- **OOAD:** Perditia Steven's *Using UML*.
- **Seminar:** Brooks' *Mythical Man Month*, Norman's *Things That Make Us Smart*. Lohr's, *GoTo*; Raymond's *Cathedral and the Bazaar*. Other readings assigned on line or hard copy.

**Papers and Exams:** The program will involve different kinds of writing, including fairly standard academic papers, documents of the sort that software designers and developers are routinely expected to produce, and short pieces focused on helping you think about, apply or clarify the program's materials and experiences. There will also be a couple of synthesizing exams. Be aware also that the programs you write are meant to be read by human beings, and as such represent examples of your writing. Communication, written and oral, is a critical part of your education as a software developer.

**Faculty Feedback on your work:** We expect to hold individual and team conferences during the year to discuss your work. If you find you need more than the considerable time you will be getting through the routine functioning of the program, feel free to make an appointment with your seminar leader.

**Program Portfolio (and Year-Long Project Notebooks):** Keep track of all your work; we will want to see it again to prepare for evaluation conferences. Buy a notebook, a binder, or a portfolio immediately in which you can organize coherently program handouts and your work for the year. Don't throw anything away until the year is over. There will be no basis for writing your evaluations unless you can resubmit all your work in an organized fashion. If you start being organized right from the start, you won't lose anything, and you will have no scrambles or frantic searches when evaluation week arrives.

**Conclusion:** As we work together over the next year, we hope to help you form a community of inquiry. We expect to work hard, to learn a lot, and to have a terrific time together in the process.

#### **Program Faculty:**

**Sherri Shulman** came to Evergreen to teach computer science, with an emphasis on programming languages and operating systems. Before graduate school (Illinois Institute of Technology, Rutgers, Oregon Graduate Institute) she worked at Bell Laboratories, North American Phillips, and EMI. Sem 3163, 867-6721, [sherri@evergreen.edu](mailto:sherri@evergreen.edu).

**Judy Cushing** came to Evergreen to teach software development within the context of the liberal arts. Before Evergreen, she worked in a variety of software development and support positions for industry (IBM, Texas Instruments, start up firms), Cornell University and Université de Bordeaux, University of Texas Health Science Center at Dallas, the U.S. Public Health Service and several small startups. Recently, she spent four years at Oregon Graduate Institute working scientific database research. She continues work in that area. Lab I, 1003, 867-6652, [judyc@evergreen.edu](mailto:judyc@evergreen.edu). See also <http://grace.evergreen.edu/~sherri> <http://academic.evergreen.edu/j/judyc>

### Planned Credit Distribution

#### Fall

4: OOP  
4: OOAD  
2: SE Case Study  
4: Seminar: Software industry  
2: Project Proposal

#### Winter

4: Parallel algorithms or Adv. Sys. Prg'g  
or 4: Database Systems  
or 4: Project-team Independent Study  
4: SE Topics & Team Theory  
8: Project: Design & Implementation

#### Spring

4: advanced technical seminar  
contemporary topic in SE,  
e.g., XML, Analysis Patterns  
4: The Semantic Web  
8: Project: Implementation & Testing

All students must take the program full time in fall, except with on faculty approval or part-time students who work full time. Students are discouraged from taking more than 16 credits in any one quarter.