Student Originated Software
Program Syllabus -- Fall 2001-2002

Monday	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
10-12 Lectures OOP & OOAD LIB 1316	10-12 Seminar LIB 2218 LIB 2204	10-12 Lectures OOP & OOAD LIB 1316	9-10 Kate's Office Hour 10-12 Pgmg Help Session (ACC)	Preparation Day
1-4 OOAD Workshop LIB 1316	1-2 Project Teams 2-5:00 OOP Lab (ACC)		1-2 Judy's Office Hour 1-2:15 Project Teams 2:30-4 Visitors & Projects LIB 1316 available til 5pm	

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 the technical skills necessary for the analysis, design and programming of software systems.
- Understand issues behind difficulty inherent 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.

Program components (Project, OOAD, OOP, Case Study, and OOP) are designed to meet the above learning objectives; see individual syllabi for details. The primary vehicle for learning how to write software is a year-long software project for an identified real-world customer (or identifiable user community). Most teams will follow a development schedule similar to the following: Fall: identify a viable project and "customer", perform a preliminary systems analysis and feasibility study. Winter: develop and evaluate a working prototype with user guides, systems analysis 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.

Planned Credit Distribution

Fall	**Winter**	**Spring**
4: OOP	2: Software Project Management	4: advanced technical seminar
4: OOAD	2: User Interface Design	contemporary topic in SE,
2: SE Case Study	or 4: Database Systems	e.g., XML, Analysis Patterns
	or 4: Project-team Independent Study	
4: Seminar: software industry	4: Seminar (Current SE Topics)	4: Seminar and Lecture (Working as an SE'r; Classic CS Problems)
2: Project Proposal	8: Project: Design & Implementation	8: Project: Implementation & Testing

In the **fall**, we will concentrate on object-oriented analysis, design and programming, and an introduction to analysis and design and software engineering through a case study. For seminar, we will consider the nature of software systems -- history, market, culture, and discipline. All students must take the program full time in the fall, except a few part-time students who work full time. In **winter**, we will likely have a technical seminar on persistent and domain-specific languages and two five week sessions on Software Project Management and User

Interface Design. In **spring**, seminar will probably offer perspectives on jobs and working, and a lecture series on classic problems in computer science, thus examining the software project experience within a broader context of what it means and what it's like to work in this industry. The technical component will focus on some timely topic in software development such as design patterns. **Students are discouraged from taking more than 16 credits at any time during the year.**

Fall Quarter Books Tim Budd, *Understanding Object Oriented Programming in Java*; Arnold, Gosling & Holmes *The Java Programming Language*; Martin Fowler, *UML Distilled*; Quatrani, *Visual Modeling with rational Rose 2000 and UML*. Seminar books (in order): *Code Complete. Close to the Machine. Things That Make Us Smart. Mythical Man Month. Insanely Great. The Microsoft Edge. Cathedral and the Bazaar. Where Wizards Stay Up Late.*

Visitor's Lecture: Thursdays, we will have a session where we work together to organize project work or hear from and talk to a working professional 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.

Papers and Exams: The program will involve different kinds of writing, including regular 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 (in more ways than one!), 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 several individual and team conferences with you during the year to discuss your work. If you find you need or desire more evaluation than the considerable amount you should be getting through the routine functioning of the program (comments from faculty and fellow students, both written and spoken, on your work, both written and spoken), feel free at any time to make an appointment with your seminar leader to talk about how you are doing.

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 keep all program handouts and all program 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

Judy Cushing: Judy came to Evergreen anxious 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 Universite de Bordeaux, University of Texas Health Science Center at Dallas, the U.S. Public Health Service and several small startups. Recently, she was on leave from Evergreen, at the Oregon Graduate Institute in Portland working scientific database research with David Maier. She continues work in that area.
LAB I, 3006, (after September 20, 2065) 866-6000 x 6652, judyc@evergreen.edu.

Kate Cunningham: Kate arrives at Evergreen this year to teach SOS. For the past six years she held technical management positions in internet start-ups working on systems such as E-Commerce and Payment Processing, Membership Services, Search Engines, Localization and Translation, Digital/Physical Good Fulfillment, Content Management, Internet/Intranet Sites, Data integration with legacy systems, and XML with wireless devices. Earlier, she spent 10 years as a programmer in the corporate world. SEM 4165, 866-6000 x5056, cunnningk@evergreen.edu.