

Software Engineering

(SOFTWARE ENGINEERING (4+0) - 30866 - ECCS 464 - 01)

Spring Quarter 2008-09

Ohio Northern University

Prof. Retterer

Syllabus

Catalog Description: The methodologies used to design, create, evaluate, and maintain software systems, including coverage of several modern methodologies with emphasis on one. A project written in a modern software development environment will be developed. Prerequisite: ECCS 268. Offered Spring Quarter.

Meeting Times: MTThF 10-10:50

Contact Info: www.proffretterer.org

The Course: "Software Engineering" is not a particularly good name for the material we will cover this quarter but it is the name that has been adopted for a wide variety of technologies used to develop software systems. This course focuses on just a few of these technologies including object oriented software engineering, Agile Model Driven Development, UML and Patterns.

Activities: You will perform tasks during the quarter including, but not limited to:

- writing and drawing design specifications of various kinds and in various forms.
- building a software system from your design
- doing various kinds of research
- writing, testing and evaluating various kinds of software for a variety of purposes
- taking oral exams
- taking written exams
- participating in discussions

Grades: Your grade will be based on the quality of the products you produce, your attendance, participation and behavior in class, and your performance on the midterm exam(April 17, 2009) and final exam (Thursday, May 21, 2009 8-10AM). I will provide you with regular progress reports to document my analysis of your work and several times through the quarter the reports will also tell you what grade you would receive at that point and why you would receive that grade. There will be points attached to each assignment. Your grade will be (roughly) determined based on these points, but your final product is more important to than the number of points you have achieved at any specific landmark along the way. At this writing, each student in this class has an A.

Text: There is no text for this class. There is an extensive reading list provided on the website.

Assignments:

This class will be composed of topics from agile model-driven development and other areas as necessary. The Agile approach is important in part because it allows you to do design the way your heart and brain tell you it should be done (maybe the way you do it naturally) but to do it in such a way that you can justify your steps, allow your creative juices to flow and carry you where they will all the time providing you with enough structure and organization to keep you on task. With the agile model, you won't necessarily create every document that is described in a more prescriptive approach and you won't feel guilty about it. Also you won't feel guilty when some of your documents are drawn on the back of an envelope and others are created in a very formal way. You will make those decisions along the way. The assignments this quarter are designed to give you a feeling about what is important and what is not, about what activities are useful and in what context they are useful and to help you learn how to build systems (of anything) without compromising quality or cost while allowing spontaneity and creativity to guide part of your design. We will also talk about when different styles of design, documentation, presentation and discussion are appropriate and when they are not. And, it will hopefully show you ways to professionally and responsibly create great software systems and enjoy it all the way through.

Software Development without stress is much more fun and at least as productive as Software Development guided by one-size-fits-all rules and artificial conformance to Life-Cycle specifications.

Rules for Handing in Assignments: You must hand in the following for each assignment:

- Cover sheet (described elsewhere)
- An appropriate document complete describing your solution to the assignment.

These items must be organized in the order stated above and must be presented in an easy to read format. It is your job to convince me that:

- You did the work yourself
- The work was done properly and why you consider it to be proper for the assignment
- The work was done completely or why it was not
- You are capable of describing work you did
- You are capable of describing how you used the work of others (if appropriate)
- You know how to create a professional looking submission even if it contains handwritten documents or documents that are crinkled or documents that are ready for the printer.
- The work was done on time.

Software Engineering

Reading List

Object-Oriented Software Engineering, Using UML, Patterns and Java™, Bruegge & Dutoit, Pearson (a Prentice Hall company), Second Edition, 2004. (I expect you to own your own copy of this book)

The Object Primer, Third Edition, Agile Model-Driven Development With UML 2.0, Scott W. Ambler, Cambridge University Press, 2004 (ISBN 13-978-1521-54018-6) Written by the founder and though leader of the Agile Modeling methodology and community. A good author with practical and realistic ideas about the world of software development.

Using UML, 2nd Edition, Perdita Stevens with Rob Pooley. Addison Wesley, 2006 (ISBN 0-32126-967-6). A little bit of the flavor of a textbook and a lot of the flavor of a reference book. Well written description of how to use UML (Unified Modeling Language) from the series deided by Booch, Jacobson and Rumbaugh. Grady Booch has been talking about object oriented software engineering and other software system design issues at least since the 1980's. His work and the work of his "followers" has been important to many developers. He has always supported realistic methods that recognize the need to avoid the straight jacket of prescriptive processes but that also recognize that structure and order are important in the design process. It is not a simple task to demand organization and structure while allowing ingenuity, creativity and autonomy but this crowd has done a good job of it. In fact, his praise for the next book in this list reads

"Software development is a human activity, although we sometimes try to deny that fact by wrapping high ceremony processes and tools around our teams which, if unleashed, can produce some truly amazing things."

Agile Project Management, Jim Highsmith, Addison Wesley (Pearson), 2004 (ISBN 0-321-21977-5). Highsmith is a principle supporter and evangelist of the Agile Revolution and a charter signatory of the Agile Manifesto. Understanding agile project management issues helps to understand how to participate on an agile based project development team.

Artful Making, What Managers Need to Know About How Artists Work, Rob Austin and Lee Devin. A fantastic book that describes issues related to software engineering and other artistic endeavors in the context of the theater. It is surprising how much the two have in common. According to Eric Schmidt (CEO of Google) the "point of view represents an important expression of the new ethos of management".

Agile Modeling, Effective Practices for eXtreme Programming and the Unified Process, Scott W. Ambler, Wiley & Sons, 2002 (ISBN 0-471-20282-7) An introductory book on Agile Modeling. Maybe a bit dated but still a good read.

Software Conflict 2.0, The Art and Science of Software Engineering, Robert L. Glass, developer.* books. Reprint and expansion of an earlier (1990) book titled Software Conflict: Essays on the Art and Science of Software Engineering. An interesting collection of essays on a variety of topics related to software development.

Design Patterns: Elements of Reusable Object-Oriented Software, by Gamma, Helm Johnson, Vlissides and Foreword by Grady Booch, Addison-Wesley, (ISBN 0-201-63361-2). The introductory treatment of the concept of Patterns in design. We all recognize parts of solutions to problems and solve those pieces the same way every time. This is not peculiar to software development but it is useful there as well. This book describes some of the most common patterns even using a pattern for the form of the descriptions. It is a classic in design and you should own it (only about \$50).

Elements of UML 2.0 Style, Scott W. Ambler, Cambridge University Press, 2005, (ISBN 0-521-61678-6). A great little handbook that will help you understand UML usage. The title is a take-off on another classic that you should own. It is called *The Elements of Style* by William Strunk and E.B. White and describes appropriate style for the writing of English. You should own both of those books. This one (*Elements of UML 2.0 Style*) is very well written, terse and easily understood. It is an essential reference if you intend to use UML. Costs about \$15 online.

Mythical Man-Month: Essays on Software Engineering, Frederick P. Brooks, Jr., Addison-Wesley, 1995, (ISBN 0-201-83595-9). This is a reprinting of the original book by the same title. The reprinting has 4 new chapters by Mr. Brooks with observations about how things changed over 20 years or so. The original chapters are about Mr. Brooks' experiences as project manager on the IBM/360 computer systems and their operating systems. Mr. Brooks is responsible for the statement that there is "No Silver Bullet" that will improve our handling of complexity in software development by an order of magnitude. Ask a hundred software developers to name THE classic work on software development and (I bet) 94 of them will say "The Mythical Man-Month". You should own this book.

The Practical Guide to Structured Systems Design, Meilir Page-Jones, Yourdon Press, 1980. One of the first books we used in this course – maybe 25 years ago. This is a great book that describes a pretty good way to build software. It describes some methods for describing problems and for describing solutions to those problems. It also talks about code design (as opposed to system design) and describes issues that help us understand and, in fact, measure the quality of the design of a black box, and another measure of the quality of the organization of multiple black boxes in a system. We will talk about these (and I'll use this book) when the time comes. These topics are not only useful in software design but also in other design venues that benefit from building subcomponents for later use or re-use.