# Course Overview

Steven J Zeil

February 13, 2013

## Contents

# 1 Basic Course Information

## 1.1 Objectives:

**Objectives:**

Support software development in Dept Research Projects

- exploratory mode rather than formal requirements
  - frequent changes to explore branches and/or recover from dead ends
- developed by an individual or small team
  - complete turnoever every few years
- may need to be packaged on short notice

. . . . . .

Software development in research projects is, of necessity, different from development in the corporate world. Research software development rarely has an opportunity for a separate, organized requirements analysis phase. It is more instead more exploratory, more evolutionary.

But that does not mean that all development discipline should be abandoned. Individual developers need to make efficient use of their time and effort. Projects need to plan for the future moment when that code base may need to be published or when the responsibility of further development must be passed to a new group of student researchers.

This course will explore lessons and tools offered by the major successful open-source software efforts. The focus will be on technqiues that work well with projects of 1-5 developers.

This first offering in Spring 2013 is intended to supplement an ongoing redesign of the CS350 Software Engineering course. This course will explore topics at a more advanced level, and may be used as a vehicle to develop course modules that can support future offerings of CS350.

**Course Themes**

- Look at best practices from the open source world



- Automate best practices
  - Make it more trouble and time consuming to do things wrong than to do them right

. . . . . .

**Areas of Emphasis**

- Test-Driven Development
- Build Management
- Version control
- Configuration Management
- Documentation Management

. . . . . .

Test-Driven development is exemplified by the philosphy of "write the tests first', *then* design and write the code." This is easiest to justify when fixing bugs. You need to have a test handy that shows the bug causing the program to fail, so that you can run it (again and again) while you try to fix the bug. How else will you know that you have it fixed?

But test-driven development is really about how to do the initial design. Programmers are lazy. If they write the code first, they often skimp on the testing because that seems like too much work for code that they are "sure" is correct (programmers are optimists – lazy optimists).

But if the tests are already there, and if it's easy to run them (or, even better, hard not to run them), then programmers will actually do the testing o na regular basis.

And thinking about tests for special /boundary cases, etc., often helps you remember those cases when later doing the deisgn and coding.

Build management is about making sure that you and other cans build the system easily. A good build manager will support TDD, documentation processes, and many of the other things we will be exploring.

Version control is the ability to track changes in the software. For teams, version control should help prevent teams members from killing each other's code with simultaneous, conflicting changes. For individuals, good version control let's you explore ideas, then back up over the ones that turn out to be undesirable.

Configuration management actually addresses two related problems:



- How do we cope with importing third-party libraries that are themselves changing and have version depencies among themselves?
- How do we cope with dependencies of our own software upon the underlying platform?

Documentation management is about integratign documentation into the development process, minimizing the “chore”.

**Basic Course Information**

**Meets:** Tues & Thurs 3:00-4:15, Dragas 2111
**Website:** https://secweb.cs.odu.edu/~zeil/cs795SD/s13/

. . . . . .

**Required Text:**

- Zeller, Essential Open Source Toolset, Wiley, 2005, 0-470-84445-0
- Lots of readings from the web

. . . . . .

## 2 Course Policies

**Due Dates and Late Submissions:**

Late assignments and projects will not normally be permitted.

Exceptions will be made only in situations of unusual and unforeseeable circumstances beyond the student’s control, and such arrangements must be made prior to the due date in any situations where the conflict is foreseeable.

. . . . . .

**Academic Honesty:**

Everything turned in for grading in this course must be your own work.

Students are expected to conform to academic standards in avoiding plagiarism.

. . . . . .



**Grading:**

- Assignments: 50%
- Semester project: 50%

. . . . . .

**Semester Project**

The semester project may take one of several forms:

- A term paper providing a critical comparison of a selection of tools or tehcniques for accomplishing a common task
- An experience report, in which a student affiliated with a research project in the Dept demonstrates the application of one or more of the tools and techqniues covered in this coruse to the software maintained by that project and evaluates the impact of this new approach.
- A course module (slides plus labs and/or assignments) suitable for use in a CS350 level course for a tool or technique not currently employed there.

All term projects must be pre-approved by the instructor as to suitability of the subject matter and scope of the project. An oral presentation of the project to the CS795/895 class will be required.

. . . . . .