

# Software Implementation and Testing

*OMSE 535 Spring 2006 Syllabus*

## ***Instructor***

*Manny Gatlin*

*Oregon Master of Software Engineering*

<mailto:gatlinm@omse.org>

---

## **About This Course**

OMSE 535, *Software Implementation and Testing*, is a course of the [Oregon Master of Software Engineering](#) program.

---

## **Prerequisites**

OMSE 500 (Principles of Software Engineering) is a required prerequisite unless special arrangements have been made with the instructor.

---

## **Course Description**

This course is an introduction to principles of implementing and testing computer software. Implementation topics include: coding style, packaging principles, reuse, creating testable and maintainable code. Testing topics include: code reviews, specification-based (black box) testing, structural (clear box) testing, test adequacy, and test administration.

---

## **Aims and Objectives of the Course**

On completion of this course students will understand techniques and principles for developing and testing maintainable code. A successful student should be prepared to join an industrial implementation or testing group.

Students should:

- Understand the properties of testable, maintainable code, and be able to inspect for them.
  - Understand the differences among inspection, testing to find defects, and reliability testing, and know when to apply each technique.
  - Understand specification-based and structural testing methods, their strengths and weaknesses.
  - Be familiar with tools for test management, capture-replay, and program analysis, and know the tradeoffs that determine the use of such tools.
- 

## Methods and Procedures

To succeed in this course students will:

- A. Attend lectures and discussions: 3 hours per week
  - B. Read assignments from texts and references
  - C. Participate in class discussions
  - D. Complete a series of course assignments
  - E. Pass examinations
- 

## Measuring Student Progress

Activity	Points
Exams	30
Class Projects	40
Participation	30
<b>Total Points</b>	<b>100</b>

---

## Course Materials

### Required Texts:

- *Code Complete 2nd ed.*, Steve McConnell, Microsoft Press, 2004, ISBN 0-7356-1967-0
- *The Complete Guide to Software Testing, 2nd ed.*, Bill Hetzel, Wiley-QED, 1988, ISBN 0-471-56567-9

### Recommended Text:

- *The Art of Software Testing*, Glenford Myers, Wiley, 1979, ISBN0-471-04328-1
- 

## Coursework

### Class Projects

There are 4 projects to be completed during the term:

1. Research and Discussion of Development Tools
2. Creation of an Annotated Programming Standard
3. Review and Analysis of Code
4. Development of Unit Test Plan and Test Cases

The projects are individual assignments for each student. Students will be expected to deliver completed projects during this term that are their own work. Note that scheduled discussions regarding the projects will occur in sessions 7-10 / Modules 2&3, so completion of the project elements will be required prior to the scheduled discussions. See the class project instructions for a complete description of the projects and deliverables.

### Instructions

You are expected to be in class for all sessions and to be on time. You should read assignments prior to class so that you can get maximum benefit from each session; the readings and projects will require more time than you think. If you are forced to miss class due to work or illness, please make arrangements with your instructor before you miss so that we can work out a plan to make up missed sessions. Contact your instructor as soon as possible if you need help or have any questions.

### Policy on Student Collaboration and Academic Honesty

You are encouraged to discuss the course material and the assignments with other students, but ***all assigned work must be done by individual students unless you are explicitly told otherwise by the course professor***. You will receive a document in class detailing OMSE policy on academic honesty. Please contact me, your professor, if you have any doubts about the propriety of your course activities.

### Class Participation

Your class participation grade will be based ***in part*** on your attendance and participation in the class discussion periods. It is your responsibility to meet the requirements of class participation; please make arrangements with me early in the term if you feel that you require an alternative means of participating actively in the course.

We will devote part of each class period to the discussion of questions related to the weekly topic. You are expected to read the questions and prepare a response suitable for discussion during the class period. The weekly questions will be available on the course website.

## Course Schedule

### Readings Key:

- CC = *Code Complete, 2nd. ed.* (McConnell)
- ST = *The Complete Guide to Software Testing, 2nd ed.* (Hetzel)
- [AuthYY] = Reading indicated in the readings section

<i>Session</i>	<i>Topics</i>	<i>Code Complete</i>	<i>Software Testing</i>	<i>Other Readings</i>	<i>Assignments Due</i>
4/3	Writing Good Software	2,4,5,7		[ <a href="#">Green97</a> ]	
4/10	Modules	6		[ <a href="#">Parnas72</a> ]	
4/17	Coding Techniques	8-19			
4/24	Testability-Reuse-Reviews	22,24,32		[ <a href="#">NASA</a> ][ <a href="#">Marick92</a> ]	Project I Approval
5/1	Debugging and Performance	25,26			
5/8	Introduction to Testing		1-4		Take Home Exam 1
5/15	Unit Testing		7	[ <a href="#">OstBal88</a> ]	Project II
5/22	Software Testing		8,10	[ <a href="#">Hamlet94</a> ]	Project III
5/29	Reliability			[ <a href="#">ButFin93</a> ]	Project IV
6/5	Formal Methods / Class Presentations				Project I
6/12	<i>No class</i>				Take Home Exam 2

## Readings

[[ButFin93](#)]

Ricky Butler and George Finelli, The infeasibility of quantifying the reliability of life-critical real-time software, *IEEE Trans. Soft. Eng.* SE-16 (Jan, 1993), 2-12.  
<http://citeseer.nj.nec.com/cache/papers/cs/21388/http:zSzzSzarchive.larc.nasa.govzSzshemeshzSzpaper-nonqzSznonq-tse.pdf/butler93infeasibility.pdf>

[[Hamlet94](#)]

Dick Hamlet, Random testing, in J. Marciniak, ed., *Encyclopedia of Software Engineering*, Wiley, 1994, 970-978.  
<http://citeseer.nj.nec.com/cache/papers/cs/3498/ftp:zSzzSzftp.cs.pdx.eduzSzpubzSzfacultyzSzhamletzSzrandom.pdf/hamlet94random.pdf>

[[Green97](#)]

Roedy Green, How to write unmaintainable code (website), 1997-2002 (orig.)  
<http://mindprod.com/unmain.html>

[[Marick92](#)]

Brian Marick, A question catalog for code inspections, Testing Foundations, 1992. <http://www2.umassd.edu/SWPI/TechnicalReview/inspect.pdf>

[[NASA](#)]

NASA, *Software Formal Inspection Guidebook*, NASA-GB-A302, 1993, (with C code inspection checklist).  
<http://satc.gsfc.nasa.gov/fi/gdb/fi.pdf>

[[OstBal88](#)]

Thomas J. Ostrand and Marc J. Balcer, The category-partition method for specifying and generating functional tests, *Comm. of the ACM* 31 (Jun, 1988), 676-686.  
<http://www.math.tau.ac.il/~msagiv/courses/testing/seminar3.ppt>

[[Parnas72](#)]

D. Parnas, On the criteria to be used in decomposing systems into modules, *Comm. of the ACM* 12 (Dec, 1972), 1053-58.  
<http://www.acm.org/classics/may96/>

---