

**University of South Florida**  
**College of Business Administration**  
**Information Systems and Decision Sciences**  
**Seminar on Software Testing**

**Spring Semester 2005 - Syllabus**

**Instructor:**

Professor Alan R. Hevner

Office: CIS 2076

Phone: 974-6753, Email: [ahevner@coba.usf.edu](mailto:ahevner@coba.usf.edu)

Office Hours: Monday 5:00 – 6:00 and by appointment at the Tampa office

Web: [www.coba.usf.edu/hevner](http://www.coba.usf.edu/hevner)

**Course Objectives:**

The software development industry strives to produce high quality, reliable software system products and services. The software testing process verifies the quality of software systems before they are released into the field. It is well known that one cannot test quality into software. Software quality is predicated on effective development and verification processes for requirements, specification, design, and implementation. Testing must be an integral component of all development processes to ensure an adequate level of quality. This course will survey and analyze the best practices in industrial testing groups. New research ideas for improving the testing process will be explored. Students will gain practical experience with both functional (black box) and structural (clear box) testing methods via assignments. Automated testing tools will be an important part of the educational experience. The goal is for all students to come away with an in-depth understanding of software testing practice and research in the field.

**Course Texts:**

*Required:*

- Paul C. Jorgensen, **Software Testing: A Craftsman's Approach**, 2<sup>nd</sup> Edition, CRC Press, Inc., Boca Raton, 2002.
- Cem Kaner, Jack Falk, and Jung Quoc Nguyen, **Testing Computer Software**, 2<sup>nd</sup> Edition, John Wiley & Sons, Inc., New York, 1999.

*Recommended:*

- David Astels, **Test-Driven Development: A Practical Guide**, Prentice-Hall PTR, Upper Saddle River, NJ, 2003.
- Kent Beck, **Test-Driven Development: By Example**, Addison-Wesley, Inc., Boston, MA, 2003.
- Robert Binder, **Testing Object-Oriented Systems: Models, Patterns, and Tools**, Addison-Wesley, Inc., Boston, MA, 2000.
- Rick Craig and Stefan Jaskiel, **Systematic Software Testing**, Artech House Publishers, Norwood, MA, 2002.
- Cem Kaner, James Bach, and Bret Pettichord, **Lessons Learned in Software Testing: A Context-Driven Approach**, John Wiley & Sons, Inc., New York, 2002.
- Brian Marick, **The Craft of Software Testing**, PTR Prentice-Hall, Upper Saddle River, NJ, 1995.

- James Whittaker, **How to Break Software: A Practical Guide to Testing**, Addison-Wesley, Inc., Boston, MA, 2003.

**Course Outline:** (Craft = Jorgensen Text, TCS = Kaner et al. Text, DL = Web Downloads)

This seminar outline is subject to change as we progress through the semester.

<b>Date</b>	<b>Topic</b>	<b>Readings</b>
Jan. 10 Study Sheet 1	Introduction to Software Testing <ul style="list-style-type: none"> <li>▪ Example Test Series</li> <li>▪ What is Software Testing?</li> <li>▪ Definitions and Taxonomies of Testing Terms</li> </ul>	Craft 1 TCS 1, 2, 4 DL 1
Jan. 24 Study Sheet 2	Testing Foundations <ul style="list-style-type: none"> <li>▪ Testing in the Software Development Life Cycle</li> <li>▪ Why is Testing So Hard?</li> <li>▪ Testing Problems</li> <li>▪ Mathematical Foundations</li> </ul>	Craft 2-4 TCS 3 DL 2
Jan. 31 Study Sheet 3	Software Verification Techniques <ul style="list-style-type: none"> <li>▪ Formal Technical Reviews</li> <li>▪ Inspections</li> <li>▪ Software Quality Attributes</li> </ul> Problem Report Systems <ul style="list-style-type: none"> <li>▪ Issuezilla</li> </ul>	TCS 5-6 DL 3-7
Feb. 7 Study Sheet 4	Functional Testing Methods I <ul style="list-style-type: none"> <li>▪ Design of Test Cases</li> <li>▪ Boundary Values</li> <li>▪ Equivalence Classes</li> <li>▪ Decision Tables</li> </ul>	Craft 5-7 TCS 7
Feb. 14 Study Sheet 5	Functional Testing Methods II <ul style="list-style-type: none"> <li>▪ Paradigms of Functional Testing</li> <li>▪ Summary of Functional Testing</li> </ul>	Craft 8
Feb. 21 Study Sheet 6	Structural Testing <ul style="list-style-type: none"> <li>▪ Control Flow Path Testing</li> <li>▪ Data Flow Path Testing</li> <li>▪ Coverage Metrics</li> <li>▪ Summary of Structural Testing</li> </ul>	Craft 9-11
Feb. 28 Study Sheet 7	Integration and System Testing <ul style="list-style-type: none"> <li>▪ Integration Methods</li> <li>▪ System Threads</li> </ul>	Craft 12-15
Mar. 7 Study Sheet 8	Mid-Term Examination Review Automated Testing Tools Orientation	DL 8-11
Mar. 14	Spring Break	
Mar. 21	Mid-Term Exam	

Study Sheet 9	Automated Testing Tools Lab	
Mar. 28	Automated Testing Tools <ul style="list-style-type: none"> <li>▪ Demonstrations of Tools</li> <li>▪ Hands-on Laboratory</li> </ul>	
Apr. 4 Study Sheet 10	Guest Speakers from Industry on Software Testing	Speaker Handouts
Apr. 11 Study Sheet 11	NISTP Research <ul style="list-style-type: none"> <li>▪ Computational Intelligence Tool (Guest Speaker: Tom Barr)</li> <li>▪ Genetic Algorithm Testing (Guest Speaker: Don Berndt)</li> </ul>	DL 12-14
Apr. 18 Study Sheet 12	Security Testing Issues The Testing Organization <ul style="list-style-type: none"> <li>▪ Organizational Structures</li> <li>▪ Testing Planning</li> <li>▪ Managing a Testing Group</li> <li>▪ Legal Issues</li> </ul>	TCS 12-15 DL 15-17
Apr. 25 May 2 Study Sheet 13	Student Testing Portfolio Presentations	

### Study Sheets:

Study Sheets for all class sessions can be downloaded from Blackboard. An outline of session topics and discussion questions are provided. (Since I continually revise all study sheets during the semester, not all study sheets will be available at the beginning of the semester.) A limited number of recommended papers and texts will appear on the Study Sheets. In our discussions, I will reference material from these sources.

### Feedback Sheets:

At the end of each class session, each student will complete a one-minute Feedback Sheet over the material covered that session. Completion of these sheets will support the goal of continuous quality improvement of the course materials.

### Grading Policy:

The Plus/Minus grading system will be used in this course.

Class Participation	10%
Mid-Term Exam	35%
Testing Assignments	55%
• Assignment #1	5%
• Assignment #2	10%
• Assignment #3	10%
• Assignment #4	10%

• Assignment #5	10%
• Assignment #6	10%

### Discussion Questions:

At the completion of each class session, a set of discussion questions will be posted on Blackboard. Participation in the discussions will enhance the student's learning experience. A portion of the class participation grade will be based on responses to the discussion questions.

### Incompletes:

Only in **rare cases**, such as serious illness, will an Incomplete be given. An Incomplete must be requested in writing giving the reason for the request and all appropriate documentation.

### Downloads:

Download the following documents from Blackboard. The documents are to be read by class on the date assigned.

1. Cem Kaner, "The Impossibility of Complete Testing," Law of Software Quality Column, **Software QA**, Vol. 4, No. 4, 1997.
2. James Whittaker, "What Is Software Testing? And Why Is It So Hard?" **IEEE Software**, Vol. 17, No. 1, January/February 2000.
3. A. Porter, H. Siy, A. Mockus, and L. Votta, "Understanding the Sources of Variation in Software Inspections," *ACM Transactions on Software Engineering and Methodology*, Vol. 7, No. 1, January 1998, pp. 41-79.
4. B. Hungerford, A. Hevner, and R. Collins, "Reviewing Software Diagrams: A Cognitive Study," *IEEE Transactions on Software Engineering*, Vol. 30, No. 2, February 2004, pp. 82-96.
5. C. LeRouge, A. Hevner, R. Collins, M. Garfield, and D. Law, "Telemedicine Encounter Quality: Comparing Patient and Provider Perspectives of a Socio-Technical System," *Proceedings of the 37<sup>th</sup> Annual Hawaii International Conference on System Sciences (HICSS37)*, Hawaii, January 2004.
6. A. Mockus, R. Fielding, and J. Herbsleb, "Two Case Studies of Open Source Software Development," *ACM Transactions on Software Engineering and Methodology*, Vol. 11, No. 3, July 2002, pp. 309-346.
7. Issuezilla Users Manual
8. B. Halipern and P. Santhanam, "Software Debugging, Testing, and Verification," *IBM Systems Journal*, Vol. 41, No. 1, 2002.
9. H. Robinson, "Intelligent Test Automation," *Software Testing and Quality Engineering*, September/October 2000.
10. K. Zallar, "Are You Ready for the Test Automation Game?" *Software Testing and Quality Engineering*, November/December 2001.
11. Rational Robot User Guide and SQABasic Reference
12. T. Barr, "Architectural Overview of the Computational Intelligence Testing Tool CI-Tool," *Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering*, Tampa, 2004.
13. A. Watkins, D. Berndt, K. Aebischer, J. Fisher, and L. Johnson, "Breeding Software Test Cases for Complex Systems," *Proceedings of the 37<sup>th</sup> Hawaii International Conference on System*

*Sciences*, Big Island, Hawaii, 2004.

14. D. Berndt and A. Watkins, "High Volume Software Testing using Genetic Algorithms," *Proceedings of the 38<sup>th</sup> Hawaii International Conference on System Sciences*, Big Island, Hawaii, 2005.
15. C. Cohen, S. Birkin, M. Garfield, and H. Webb, "Managing Conflict in Software Testing," *Communications of the ACM*, Vol. 47, No. 1, January 2004.
16. Cem Kaner, Elisabeth Hendrickson, and Jennifer Smith-Brock, "Managing the Proportion of Testers to (Other) Developers," *Pacific Northwest Software Quality Conference*, Oct. 2001.
17. Brian Marick, "Classic Testing Mistakes," *Software Testing, Analysis, and Review (STAR) Conference*, Oct. 1997.

### **Academic Integrity / Plagiarism:**

Academic dishonesty may take several forms, including plagiarizing, collaborating with others without acknowledging their work, submitting work that you did previously or that you are doing for another class (unless I give you permission to do this), falsifying results or data, interfering with another student's work, and copyright violation. When in doubt, ask me for clarification. Some of the best testing has been done by collaborating testers. You are welcome to test and to do assignments in groups, or to work alone. Of course, if three people work together, I expect them to submit three times as many bugs, or to submit work that is in some way, three times as good or as extensive as I would expect from a person working alone. Bug reports should be submitted by the person who did the majority of the work to find that bug (or to develop / design the test that found that bug). If more than one person contributed to the design or the bug finding, name the other people at the start of the bug description.

You may not collaborate on the mid-term examination. The usual rules governing cheating in closed book tests and exams will be applied.

### **Disability Assistance:**

Any students who feel they need assistance to attend and participate in this class must contact USF's Office of Student Disability Services (974-4309) and request a review and authorization if appropriate for necessary support. Please note that USF regulations state that only students who are registered for this class may attend and be present in the classroom while presentations are being made.

### **National Institute for Systems Test and Productivity (NISTP):**

The mission of the federally-funded National Institute for Systems Test and Productivity (NISTP) is to work with industry, government, academe and the other allied Institutes to identify, synthesize, improve, and disseminate critical systems productivity and test practices and tools that will improve the development and quality of software-intensive systems in industry and government. In order to rapidly achieve the significant payoffs needed, the Institute's research is focused finding innovative technical approaches and building intelligent tools utilizing computational intelligence and end to end testing methods. The field of Computational Intelligence includes areas such as uncertainty management, applied fuzzy logic, data mining, machine learning, genetic algorithms, neural networks, clustering analysis, and case-based reasoning. The End-to-End (E2E) systems test methodology provides new and effective techniques for validating the completeness and quality of network-centric, large-scale systems. NISTP is hosted at the University of South Florida. NISTP partner institutions include Arizona State University, Drexel University, Ben-Gurion University, and the Boeing Corporation. This course is partially funded by NISTP.