

Software Quality Analysis

OMSE 525, Fall 2006

Instructor: Kal Toth, ktoth@omse.pdx.edu

This is a preliminary syllabus for two fall 2006 offerings of this OMSE 525. Both a face-to-face and an on-line version of the course are scheduled for the fall. An updated syllabus for each version of the course will be posted by early September.

The on-line version of this course will involve weekly on-line discussions while the face-to-face version will conduct similar discussions in class. Both versions will require completing take-home assignments. Course materials will be available on-line.

Course Description

This course addresses processes, methods, and techniques for developing quality software, for assessing software quality, and for maintaining the quality of software. This course explores tradeoffs between software cost, schedule, and quality as well approaches for integrating quality into the software development process. Quality methods including formal review and inspection; test planning and testing; software maintenance; and software safety are covered.

Prerequisites

OMSE 500 Principles of Software Engineering (or equivalent) plus two or more years of experience in software development or software maintenance.

Aims and Objectives

Quality cannot be added to software after it is developed, but must be attended to throughout development. Real knowledge of software quality requires rigorous analysis -- it is easy to be fooled. Quality software can be destroyed very easily during "maintenance." This course describes software quality and introduces some of the primary techniques for attaining and assessing it. Upon completing this course the student should be able to:

- Find the balance between quality and risk as to how much software quality is "good enough."
- Evaluate the organization and procedures needed to attain a given level of quality.

- Participate in inspections/reviews of software products.

- Develop a system test plan for a software project

- Develop a quality-improvement plan for software products.

- Choose appropriate testing methods for each phase of development.

- Control maintenance activities so that quality is not compromised.

- Understand safety issues of critical software.

- Understand what aspects of the development process contribute to product quality.

The Instructor

Dr. Kal Toth is an Associate Professor in the Maseeh College of Engineering and Computer Science at Portland State University in Portland Oregon. He is the Associate Director of the Oregon Master of Software Engineering (OMSE) program and is also teaching software engineering courses in this program. Kal is conducting research in the field of information security. He has over 25 years of industry experience with large, medium and small companies including Hughes Aircraft of Canada, CGI Group, Datalink Systems Corp, Intellitech Canada Ltd. and various Canadian federal government agencies (notably Defence, Transport, and Communications). Kal has a Ph.D. in Computer Systems Engineering

from Carleton University (Ottawa) and is a P.Eng. (British Columbia) with a software engineering designation.

Required Textbooks and Resources

Jeff Tian, *Software Quality Engineering* (SQE), Wiley-Interscience, 2005; ISBN 0-471-71345-7.

Additional readings and notes will be distributed. Each week, all assigned readings are to be completed before embarking on discussions and assignments for that week.

Workload Distribution

To succeed in this course, students are expected to devote 9-12 hours of study per week as follows:

- Review the required textbooks, lecture notes and reference materials (5+ hrs/week)
- Participate in discussions and complete assignments (5+ hrs/week).

You are expected to complete the readings and lecture materials, and respond critically to the related discussion questions and assignments. Guidelines and structure for discussions and assignments will be provided by the instructor.

Policy on Student Collaboration and Academic Integrity

You are encouraged to discuss the course material and the assignments with other students, but all assigned work must be done individually unless the instructor explicitly tells you otherwise. You are expected to read and review the policy on academic integrity which will be provided by the instructor. Please contact the professor if you have any doubts about the propriety of your course activities.

Course Calendar

Week	Reading/ Topics and Activities
1	SQE 1-4: Introduction; Characterizing software quality: properties of the product and the process; Quality improvement process, verification and validation; Developer's organizational structure; Product quality expectation; Defect prevention, reduction, and containment.
2	SQE 5-6, 14: Glen W. Russell, Experience with Inspection in Ultralarge-Scale Developments (<i>IEEE Software</i> , Jan 1991). Hamlet and Maybee, Properties of Good Specifications (excerpt from <i>The Engineering of Software</i> , Addison-Wesley, 2001). Quality engineering and planning; Fault models through development; Requirements quality; Inspections; Form inspection teams; Distribute Requirements Document for inspection
3	SQE 7, 8, 21: Requirements document for inspection; Software Formal Inspections Guidebook, NASA-GB-A302 (and requirements checklist). ;Adam Porter et al., A review of software inspections. In-class inspection exercise (1 hour); Debrief and discuss exercise; Test activities and planning; Introduction to testing
4	SQE 9 – 11: Hamlet and Maybee, Software Testing, Systematic Testing; Thomas Ostrand and Marc Balcer, The Category-partition Method (<i>CACM</i> , Jun 1988); Functional testing; Flow control testing; Risks associated with poor software quality; Cost/schedule/quality tradeoffs; Tracking quality through the development process
5	SQE 12: Hamlet and Maybee, Test Plans; Integration, System, Acceptance testing; Interface design for test; Testability
6	SQE 13, 18, 20: David Card, Learning from Our Mistakes with Defect Causal Analysis (<i>IEEE Software</i> , Jan 1998); Defect prevention; Quality metrics; Static analysis; Quality and process improvement
7	SQE 19: Terry Bollinger and Clement McGowan. A Critical Look at Software Capability Evaluations. (<i>IEEE Software</i> , Apr 1991); Watts Humphrey and Bill Curtis. Comments on 'A Critical Look' (<i>ibid</i>); Watts Humphrey. Characterizing the Software Process: A Maturity Framework (<i>IEEE Software</i> , Mar 1988); Software processes; Capability Maturity Model and quality models; ISO 9000 ; Root-cause Analysis; Extreme Programming; Open-source Dev.
8	SQE 22; Edward H. Bersoff and Alan M. Davis, Impacts of Life Cycle Models on Software Configuration Management (<i>CACM</i> Aug 1991); Conf Mgt; Quality after release; Tracking field performance; Maintenance; Regression testing; Large software systems and quality
9	SQE 16; David Parnas et al. Evaluation of Safety-Critical Software (<i>CACM</i> , 1990). Nancy Leveson. Medical Devices: The Therac-25 (excerpt from <i>Software: System Safety and Computers</i> , Addison-Wesley, 1995); Ariane 5, Flight 501 Failure, Rep Inquiry Board; SW safety
10	SQE 15, 17; Harlan Mills. Cleanroom Software Engineering (<i>IEEE SOFTWARE</i> , Sep 1987). John Musa. Tools for Measuring Software Reliability (<i>IEEE Spectrum</i> , Feb 1989); C. A. R. Hoare, Programming: Sorcery or Science? (<i>IEEE SOFTWARE</i> , Apr 1984); The future of software quality; Formal methods Quality verses cost; Model checking; Quality ownership