

07115



Minnesota State University, Mankato HOLD and CLEAR buttons only compatible with Acrobat V. 4 and 5
Curriculum Proposal

Please type or select the requested information. Print completed forms, add appropriate paper attachments, and route through MSU's curricular process for recommendations and decisions.

(Check all that apply):

College: Science, Engineering and Technology Undergraduate
 Department: Computer and Information Sciences Graduate ✓
 Program: Information Systems (ISYS) CIP # 11.040100
 Type of Change: COURSE PROPOSALS
 Proposed: New Course

Title Current: _____
 Title Proposed: Software Quality Assurance and Testing
 24-Char. Abbrev: Software QA and Testing

Proposal #	124
Effective Date of Change:	
Academic Year	06-07
(For Office Use Only)	

Course Designator and Number	Number of Credits
ISYS 4/580	4
(if applicable)	

Include a course or program description for the Bulletin (30-40 words maximum for courses, 100 for programs):
 Developing quality software, assessing and maintaining software quality. Software testing at unit, module, subsystem, and system levels. Automatic and manual generation of test data, static and dynamic analysis, functional testing, inspections, and reliability assessment.
 Pre: ISYS 380 or IT 380
 Fall, Spring

Rationale or Justification for change:
 Course is required in new Information Systems (ISYS) curriculum.

*****For General Education or Cultural Diversity Courses Only*****

GE Category #	GE Category Name (Maximum of 3 Categories)
N/A	
N/A	
N/A	

For Writing Intensive Courses, attach a description of the kind and quantity of writing.
 For Upper Division Courses, include a description of the respects in which it is broad and general rather than narrow and specific, and so suitable as GE.

Attach paper copies of the following:
 a. Syllabus or course outline.
 b. Course's student learning outcomes associated with each GE competency or CD designation.
 c. List of strategies to be used to assess students' achievement of each GE competency or CD designation.

Cultural Diversity Course:
 (Please check one.)
 Core (At least 75% devoted to topics of race, gender, sexual orientation, age, class, and disabilities as they occur in United States Society.)
 Related (At least 25% devoted to the above topics or to a global perspective on topics related to African American, Asian, Hispanic, and Native American inhabitants of the United States.)

*****For New Courses*****

(Check all that apply):

Instructional Type: Lecture
 Grading Format: Grade P/N
 Course will be offered:
 Fall Semester
 Spring Semester
 Summer Session

Course is an elective.
 Course is required for program Information Systems (ISYS)
 Pre- or Co-requisites: ISYS 380 or IT 380
 Other courses are being changed or eliminated. (Explain.) _____

Course content or title is similar to courses in other departments. (Attach copy of letter of agreement with other program(s) contacted. Indicate the nature of the discussions and/or resolution of differences or potential conflicts.)

Attach paper copies of the following:
 a. Syllabus or course outline.
 b. Course's student learning outcomes.
 c. A list of resources required to offer and support this course.
 d. A description of how teaching this course will affect department staffing.
 e. If 400/500 level course, an explanation of added expectations of graduate students.

Proposal: ISYS 4/580: Software Quality Assurance and Testing

Catalog Description

This course focuses on the processes, methods and techniques for developing quality software, assessing software quality, and maintaining software quality. Software testing processes at the unit, module, subsystem, and system levels is discussed. Testing methods covered include: automatic and manual generation of test data, static vs. dynamic analysis, functional testing, inspections, and reliability assessment.

Prerequisites

- ISYS 380(4) Systems Analysis and Design (or IT 380)

Topics

Knowledge Unit 1. *QA Fundamentals*

- Goal: to be able to demonstrate application-level mastery of fundamental concepts and terminologies related to QA and the relationship between QA, QC and testing; benefits of the overall QA process for both the development and testing teams; why the essential knowledge and discipline is required for all phases of the development life cycle
- Approximate classroom hours: 2
- Mastery level 3: Application
- Objectives:
 - 1.1 construct a QA plan for a hypothetical organization
 - 1.2 construct a QC plan for a hypothetical organization
 - 1.3 dramatize the benefits of the overall QA process for both the development and testing teams

Knowledge Unit 2. *Testing Life Cycles*

- Goal: to be able to demonstrate comprehension-level mastery of the Systems Development Life Cycle (SDLC), Traditional Project Life Cycle and the Product Management Life Cycle; phases, tasks, inputs, deliverables or outputs for each Life Cycle in detail; relationship between the Testing Process, Testing Life Cycle and the SDLC, and how, in some instances, it can overlap multiple phases of the SDLC
- Approximate classroom hours: 2
- Mastery level 2: Comprehension
- Objectives:

- 2.1 explain how system projects begin and how projects are reviewed
- 2.2 explain product management life cycle phases, inputs and outputs
- 2.3 explain quality assurance and test activities performed in typical system development projects during the SDLC phases
- 2.4 explain the overall testing life cycle and how it covers multiple phases of the SDLC
- 2.5 explain the purpose of a mission statement
- 2.6 explain the SDLC structured approach to designing and building application systems

Knowledge Unit 3. *Tracking Software Changes*

- Goal: to be able to demonstrate comprehension-level mastery of issues and terminology related to tracking software changes; why managing software changes is a critical part of the systems development and quality assurance functions; change control, scope creep, version control and configuration management
- Approximate classroom hours: 3
- Mastery level 2: Comprehension
- Objectives:
 - 3.1 define "change-control" and explain how it is used to monitor and manage software changes
 - 3.2 define "scope creep" and explain ways to prevent it
 - 3.3 explain how version control manages multiple versions of software applications
 - 3.4 explain system configuration management procedures

Knowledge Unit 4. *Reviews and Inspections*

- Goal: to be able to demonstrate comprehension-level mastery of issues and terminology related to non-execution-based review of software; code reviews, code walkthroughs, inspection techniques, static code analysis tools
- Approximate classroom hours: 2
- Mastery level 2: Comprehension
- Objectives:
 - 4.1 explain the evolution of the inspection process
 - 4.2 explain the similarities between reviews and inspections
 - 4.3 explain the value that occurs when inspections are used
 - 4.4 explain what inspections are designed to accomplish
 - 4.5 explain what reviews are designed to accomplish
 - 4.6 explain who uses inspections and why
 - 4.7 list and explain test products that can be reviewed
 - 4.8 list and explain the roles and responsibilities of review participants
 - 4.9 list, define, and explain review critical success factors

Knowledge Unit 5. *Principles of Testing*

- Goal: to be able to demonstrate analysis-level mastery of the history and fundamentals of testing with specific emphasis on software testing; strategies, categories and types of testing that can occur within the testing process; designing and executing, defined by measurement goals, data collection, software reliability and quality
- Approximate classroom hours: 7
- Mastery level 4: Analysis
- Objectives:
 - 5.1 simplify a set of testcases by removing redundant testcases

Knowledge Unit 6. *Defects and Problem Reporting*

- Goal: to be able to demonstrate comprehension-level mastery of the identification, classification and reporting of errors as early as possible in the development life cycle
- Approximate classroom hours: 7
- Mastery level 2: Comprehension
- Objectives:
 - 6.1 explain the concept of severity and priority as they relate to defects
 - 6.2 explain the fundamental principles of reporting defects
 - 6.3 explain what items are included in a problem report
 - 6.4 explain why software has defects

Knowledge Unit 7. *Automated Testing*

- Goal: to be able to demonstrate application-level mastery of how automated testing tools help organizations optimize and accelerate the delivery of applications; purchasing, setting-up and executing an automated test environment; review of automated testing methods that allow for the quick capture and reuse of quality activities to share and repeat throughout the testing life cycle
- Approximate classroom hours: 10
- Mastery level 3: Application
- Objectives:
 - 7.1 construct an automated test framework for a specific problem domain in a hypothetical organization
 - 7.2 utilize an automated testing tool to write a suite of tests

Knowledge Unit 8. *Project Documentation*

- Goal: to be able to demonstrate application-level mastery of the systematic arrangement of information about a project during the course of the process; the organizational agreement to fund the project and provide resources; the manner in which the project will be managed and the governance surrounding the project; the plan for each phase
- Approximate classroom hours: 7
- Mastery level 3: Application

- Objectives:
 - 8.1 construct a project document for a hypothetical situation based on a requirements specification

Knowledge Unit 9. *Requirements Documentation*

- Goal: to be able to demonstrate application-level mastery of how information contained within the documented requirements drives the Systems Development Life Cycle activities and these activities are essential for the software development and Quality Assurance processes; that a requirements definition is a documented agreement between the customer and the development team that explicitly describes the product to be developed
- Approximate classroom hours: 5
- Mastery level 3: Application
- Objectives:
 - 9.1 change a moderately complex requirements specification in response to a simulated user request, and predict the effect of the change on the overall time and budget estimates of the project

Knowledge Unit 10. *Requirements-Based Testing*

- Goal: to be able to demonstrate application-level mastery of requirements and their relationship to the testing and test process; how requirements incorporate test conditions and functional checklists to verify correct system functions
- Approximate classroom hours: 7
- Mastery level 3: Application
- Objectives:
 - 10.1 utilize a requirements specification to write a suite of testcases

Knowledge Unit 11. *Standard Test Documentation*

- Goal: to be able to demonstrate analysis-level mastery of test documentation as a tool and how it is used to administer and maintain the testing process along with how it aids in the planning, monitoring, and managing of the testing phases, the three primary layers of test documentation: test plan definition, test plans, and test cases
- Approximate classroom hours: 3
- Mastery level 4: Analysis
- Objectives:
 - 11.1 discuss how and when a test plan should be developed and how the test plan guides the testing effort

Knowledge Unit 12. *Continuous Improvement*

- Goal: to be able to demonstrate application-level mastery of organizational systems that determine how the systems might be improved

- Approximate classroom hours: 7
- Mastery level 3: Application
- Objectives:
 - 12.1 apply systems, decision and quality theory and information systems development techniques and methodologies to initiate, specify and implement a relatively complex multi-user information system originating in a quality conscious organization involved in continuous improvement of its processes
 - 12.2 implement physical flows as well as a complete work flow at an enterprise or multi-department level
 - 12.3 utilize development methodologies compatible with the concept of process of continuous improvement

Additional topics may also be covered based on time and student interest.

Graduate Students


Students taking the 500-level version of this course are required to perform beyond expectations of undergraduate students by completing one or more of the following, at the discretion of the instructor:

- A term paper that summarizes and critiques an article from a scholarly journal in the area of software testing and software assurance.
- A project that implements advanced ideas in software testing and software assurance.
- A presentation about an advanced area in software testing and software assurance, or that presents the student's term paper or project.
- Some other activity that demonstrates grasp of the material beyond what is expected of undergraduates.

Instructional and Library

Resources currently in place within the department and the University Library will support this new course. No new resources are required.

Staffing

This course will be able to be staffed by the faculty that have been designated in the proposed Department of Information Systems and Technology ~~by the Dean of the College of Science, Engineering, and Technology, Dr. John Frey.~~  This course will be cross-listed between the "Information Systems" and "Information Technology" programs in the new department. This course will not need assistance from faculty of the new Computer Science Department.

Possible Textbook(s)

Craig, Jaskiel , *Systematic Software Testing* , Artech House Computer Library, 2002 (ISBN: 1580535089)

Craig, Jaskiel, *Software Testing: A Craftsman's Approach* (2nd edition) , CRC Press, 2002 (ISBN: 084937345X)

Culbertson, Brown, Cobb, *Rapid Testing* , Prentice-Hall, 2002 (ISBN: 0-13-091294-8)

Henry, Hanlon, *Software Quality Assurance* , Prentice Hall, 2007 (ISBN: 0131435817)

Lewis, *Software Testing and Continuous Quality Improvement* (2nd edition) , CRC Press, 2004 (ISBN: 0849325242)

Utting, Legeard, *Practical Model-Based Testing: A Tools Approach* , Morgan Kaufmann, 2006 (ISBN: 0-12-372501-1)

Young, Pezze, *Software Testing and Analysis: Process, Principles and Techniques* , Wiley, 2007 (ISBN: 0-471-45593-8)

Syllabus for ISYS 4/580: Software Quality Assurance and Testing

Instructor

Name: Prof. Sample Faculty
Office: 200 Wissink Hall
Department: Information Systems and Technology, Minnesota State University, Mankato
Office hours: Monday through Friday from 1:00 to 4:00 pm
Phone: 507-389-1212
Email: sample.faculty@mnsu.edu
Course page: <https://d21.mnsu.edu/>

Meeting

MTRF 10-11

Catalog Description

This course focuses on the processes, methods and techniques for developing quality software, assessing software quality, and maintaining software quality. Software testing processes at the unit, module, subsystem, and system levels is discussed. Testing methods covered include: automatic and manual generation of test data, static vs. dynamic analysis, functional testing, inspections, and reliability assessment.

Prerequisites

- ISYS 380(4) Systems Analysis and Design (or IT 380)

Topics

The following content areas will be covered.

1. QA Fundamentals (about 2 hours)
2. Testing Life Cycles (about 2 hours)
3. Tracking Software Changes (about 3 hours)
4. Reviews and Inspections (about 2 hours)

5. Principles of Testing (about 7 hours)
6. Defects and Problem Reporting (about 7 hours)
7. Automated Testing (about 10 hours)
8. Project Documentation (about 7 hours)
9. Requirements Documentation (about 5 hours)
10. Requirements-Based Testing (about 7 hours)
11. Standard Test Documentation (about 3 hours)
12. Continuous Improvement (about 7 hours)

Additional topics may also be covered based on time and student interest.

Objectives

By the end of this course, you should be able to

- construct a QA plan for a hypothetical organization
- construct a QC plan for a hypothetical organization
- dramatize the benefits of the overall QA process for both the development and testing teams
- explain how system projects begin and how projects are reviewed
- explain product management life cycle phases, inputs and outputs
- explain quality assurance and test activities performed in typical system development projects during the SDLC phases
- explain the overall testing life cycle and how it covers multiple phases of the SDLC
- explain the purpose of a mission statement
- explain the SDLC structured approach to designing and building application systems
- define "change-control" and explain how it is used to monitor and manage software changes
- define "scope creep" and explain ways to prevent it
- explain how version control manages multiple versions of software applications
- explain system configuration management procedures
- explain the evolution of the inspection process
- explain the similarities between reviews and inspections
- explain the value that occurs when inspections are used
- explain what inspections are designed to accomplish
- explain what reviews are designed to accomplish
- explain who uses inspections and why
- list and explain test products that can be reviewed
- list and explain the roles and responsibilities of review participants
- list, define, and explain review critical success factors
- simplify a set of testcases by removing redundant testcases
- explain the concept of severity and priority as they relate to defects
- explain the fundamental principles of reporting defects
- explain what items are included in a problem report

- explain why software has defects
- construct an automated test framework for a specific problem domain in a hypothetical organization
- utilize an automated testing tool to write a suite of tests
- construct a project document for a hypothetical situation based on a requirements specification
- change a moderately complex requirements specification in response to a simulated user request, and predict the effect of the change on the overall time and budget estimates of the project
- utilize a requirements specification to write a suite of testcases
- discuss how and when a test plan should be developed and how the test plan guides the testing effort
- apply systems, decision and quality theory and information systems development techniques and methodologies to initiate, specify and implement a relatively complex multi-user information system originating in a quality conscious organization involved in continuous improvement of its processes
- implement physical flows as well as a complete work flow at an enterprise or multi-department level
- utilize development methodologies compatible with the concept of process of continuous improvement

Students with Disabilities

Every attempt will be made to accommodate qualified students with disabilities. If you are a student with a documented disability, please see me as early in the semester as possible to discuss the necessary accommodations, and/or contact the Disability Services Office at (507) 389-2825 (V) or 1-800-627-3529 (MRS/TTY).

Textbook

This course will use one more more of the following textbooks:

Craig, Jaskiel , *Systematic Software Testing* , Artech House Computer Library, 2002 (ISBN: 1580535089)

Craig, Jaskiel, *Software Testing: A Craftsman's Approach* (2nd) , CRC Press, 2002 (ISBN: 084937345X)

Culbertson, Brown, Cobb, *Rapid Testing* , Prentice-Hall, 2002 (ISBN: 0-13-091294-8)

Henry, Hanlon, *Software Quality Assurance* , Prentice Hall, 2007 (ISBN: 0131435817)

Lewis, *Software Testing and Continuous Quality Improvement* (2nd) , CRC Press, 2004 (ISBN: 0849325242)

Utting, Legeard, *Practical Model-Based Testing: A Tools Approach* , Morgan Kaufmann, 2006 (ISBN: 0-12-372501-1)

Young, Pezze, *Software Testing and Analysis: Process, Principles and Techniques* , Wiley, 2007 (ISBN: 0-471-45593-8)

Additional readings may be assigned by the instructor.

Grading

Your course grade will be based on:

[Varies by faculty]

If you receive 90% or more of the possible points, you are guaranteed an A, 80% a B, et cetera. However, a score just below the grade cutoff will not necessarily earn the higher grade. Therefore, you should try to attain a score well above the cutoff to achieve the grade you want.

Exams

The exams will cover reading assignments, lectures, and class discussion. It is your responsibility to remember the exam schedule. If you forget to attend an exam or are more than ten minutes late for the exam, you must forfeit the grade for that exam.

You may take the exam at an alternate testing time if you participate in a university-sponsored activity that requires your attendance. You must arrange with the instructor at least a week ahead of the exam date.

If you miss an exam because of illness or family emergency, you may arrange with the instructor for a makeup exam. You must produce written proof of the reason you cannot take a test at the normally scheduled time.

Graduate Students

If you are taking the 500-level version of this course, you must complete one or more of the following additional requirements, at the discretion of the instructor:

- A term paper that summarizes and critiques an article from a scholarly journal in the area of software testing and software assurance.
- A project that implements advanced ideas in software testing and software assurance.
- A presentation about an advanced area in software testing and software assurance, or that presents the student's term paper or project.

- Some other activity that demonstrates grasp of the material beyond what is expected of undergraduates.

Programming Assignments

[Varies by faculty]

Homework

[Varies by faculty]

Class Policies

Attendance

[Varies by faculty]

Late Policy

[Varies by faculty]

Academic Honesty

Please be aware that the University's policy for Academic Honesty appears in the Student Handbook. Each student is expected to have read this material. If you do not understand what is meant by this policy, or if you are confused by terms such as plagiarism, cheating, or collusion, please discuss this policy with me, your advisor, or another faculty member as soon as possible. I absolutely require that each student in this class will fulfill his or her academic obligations in a fair and honest manner.

Anything that you observe in other students that is of questionable integrity should be brought to my attention. You may do so anonymously if you desire (e-mail works fine for this).

For the writing of papers and program design documents, it is quite easy to define cheating in terms of traditional definitions of plagiarism; however, for the writing of computer programs, the distinction is not as obvious to many students. It is easy to use the English paper comparison when thinking about what is appropriate and what constitutes dishonest academic work when writing computer programs. Like writing a paper, you may discuss general ideas with fellow students.

You must write each programming assignment yourself. It is acceptable to discuss logic and other strategies such as the number of variables or methods, but it is NOT acceptable to show another student even a single line of your program until after the due date.

Protect your programming assignments. If another student obtains a copy of your program even though you are unaware of the infraction, you are still guilty of collusion. Do not leave an ACC workstation unattended, even for a little while. Other students monitor may your patterns and, while you are out for even a few seconds, walk up to your workstation and email a copy of your program to themselves! You should also be careful where you leave paper copies of your program. Do not leave files on ACC computers. A trash receptacle anywhere near the ACC is likely to be rummaged. Make sure you log out of the course web site when you are done using it, close all browser windows, and log out of the system.

I strongly suggest you consult your student handbook or talk with me if you are unsure as to what is acceptable academic behavior. The consequences are quite severe. Academic misconduct will automatically result in my informing Judicial Affairs, a division of Student Affairs, of the misconduct. This misconduct usually results in a failing grade for the course. (And it can be worse.)

Specific items that will be considered cheating on programming assignments are:

- Turning in work done by somebody else as your own (with or without that person's consent). This includes turning in a copy of something that can be mechanically transformed into a copy of someone else's work. Do not try to disguise cheating by simply modifying someone else's work and calling it your own. I use software that detects this type of cheating.
- Allowing someone else to turn in your work as his or her own work. This includes allowing fellow students access to your copy, even without your knowledge or consent.
- Using a solution developed by a student in a previous semester or another section.

Errors in Grading

If something has been graded incorrectly, or if a grade has been recorded incorrectly, you must request a correction no later than one week after the grade has been posted. Course grades are final, and cannot be changed unless there has been a substantial error.

Grades are based on the quality of your work and on how well you are prepared for class. While working hard is admirable, your grade will not be based on how much time you spent working on an assignment or preparing for an examination.

Weather and Other Problems

In the event inclement weather conditions or other problems cause class not be held on a given day, any work due for that day will be due at the next class meeting. It will not cause any other changes in the schedule. Weather-related closings will be made by the university and announced on the Twin Cities and local media. You can call the MSU WeatherLine at 2463 for weather-related closing or cancellation information.

Classroom Etiquette

Please turn off or silence your cell phone or pager while in class. If you are expecting an urgent call (such as from your sick child):

- Use the "silent" mode of your cell phone or pager.
- Sit right next to the door.
- Leave the room as quietly as possible when you receive the call, so you do not disturb other students.