

EASTERN MICHIGAN UNIVERSITY
DIVISION OF ACADEMIC AFFAIRS

**REQUEST FOR INCLUSION OF A COURSE IN THE
GENERAL EDUCATION PROGRAM:
EDUCATION FOR PARTICIPATION IN THE GLOBAL COMMUNITY**

DEPARTMENT/SCHOOL: _____ COMPUTER SCIENCE _____ COLLEGE: _____ ARTS AND SCIENCES _____
DEPARTMENT CONTACT: _____ WILLIAM MCMILLAN _____ CONTACT PHONE: _____ 7-0110 _____
CONTACT EMAIL: _____ WCMILLAN@EMICH.EDU _____

1. Subject Code, Number, and Title: _____ COSC 481 Software Engineering and Senior Project _____

2. Credit Hours 3

3. Course Description

This capstone course surveys the fundamentals of software engineering, including requirements analysis, disciplined implementation and evaluation. Students will work on a semester-long software project that employs principles learned in other computer science classes.

4. This course is (check one):

- an existing course with no revisions (need not go through the input system)
- an existing course with revisions (attach this form to Request for Course Revision form)
- a new course (attach this form to Request for New Course form)

5. Check the General Education requirement this course is intended to meet. If the course is to be proposed for more than one requirement, submit a separate form for each one.

- Effective Communication**
- Quantitative Reasoning (*QR designation*)**
- Writing Intensive (*WI designation*)**
- Perspectives on a Diverse World**
 - Global Awareness
 - U.S. Diversity
- Knowledge of the Disciplines**
 - Arts
 - Humanities
 - Science
 - Social Science
- Learning Beyond the Classroom (*LBC designation*)**

- Self and Well Being
- Community Service, Citizenship, and Leadership
- Cultural and Academic Activities and Events
- Career and Professional Development
- International and Multicultural Experience
- Undergraduate Research

6. Rationale. Provide a concise, clear, jargon-free explanation of why this is a General Education course and how it fits into this specific area of the program. (The rationale should explain to students why they are taking the course. It should address both why it is part of the General Education program and why it fits into the particular category.) This rationale should appear on the general course syllabus provided here and should be included in specific course syllabi given to students.

Courses that fulfill the writing intensive (WI) designation in EMU's general education program should immerse students in discipline-specific communication, teaching them how specialists in their area of study plan, execute, and evaluate documents and other media in professional contexts. Besides teaching specific skills in writing, carrying out research, and related activities, the intention should also be to sensitize students to a host of issues relevant to successful practice that might otherwise escape their notice. This course, COSC 481 Software Engineering and Senior Project, is an introduction to real-world software development that is required of all computer science majors. It requires a series of written reports in which students communicate in a manner consistent with established software engineering practice. In a sense, the course is primarily centered on communication, though it includes purely technical content as well. The documents that students submit include a requirements document, which serves as a description of the clients' software needs, a design document, which specifies at various levels of detail the architecture of the system, and a report of the verification and validation of the system implemented. The audiences for these reports vary from clients who have little or no technical background to project managers to very technically savvy—and critical—software developers and quality assurance specialists. Communication in this realm is only partially (and sometimes not primarily) verbal. In software development, graphical notations are very important for communicating with clients and with technical audiences. This course introduces (or reinforces) several standard graphical notations for the specification and design of software.

7. Clearly and concisely explain how this course meets each of the General Education outcomes for the requirement checked in number five (all outcomes should be addressed). To do this, (a) list the General Education outcomes for the requirement and explain how the course meets each outcome; and (b) explain, in general terms, the method(s) of evaluation to be used in the course and how these methods assess the degree to which students have met the General Education outcomes for this requirement.

Develop and employ successful, flexible writing and reading strategies that support sustained inquiry in a discipline.

The students in COSC 481 use a variety of methods in carrying out research. They need to find and read background material for the software engineering project, including information about the *application domain* (ranging from computer games to software that controls industrial processes), *new computing tools to be employed* (e.g., programming languages, delivery platforms, development environments), and *technical*

background (in such areas as networking, graphics, web technologies, and others). This information is obtained from interviews with a client, trade publications, textbooks, consultation with students and faculty, various web-based resources, and from the use of relevant software tools. Much of the learning is done independently or in small groups. Some of the information obtained shows up in the reports students submit, but much of it facilitates the work of the project without being explicitly included in submissions. In doing this background research students learn general skills that will serve them well in further courses and in the workplace.

Please note that a subsection has been added at the end of Section 7 to address teaching methods for these skills and others.

Use writing strategies that achieve the purpose(s) for writing and address the expectations of audience(s) within a disciplinary context.

The audiences for, and purposes of, the project increments are as follows:

Requirements document: client, designers (secondary: developers, quality-assurance specialists, maintainers)
The purpose of developing the requirements document is to understand the needs of the client. This is an iterative process. The result serves as an agreement with the client regarding the functionality of the software, communicates it to the designers, and provides reference material for maintainers. This document may make reference to published or web-based resources, and includes results of interviews with a client.

Design document: developers (secondary: quality-assurance specialists, maintainers)
The design document is like a set of blueprints for a building, telling the builders (the programmers) exactly how to build the system. The design is expressed in high level terms (the “bird’s eye view”) and in detail. Students may use design patterns that are described in published sources and may incorporate standard solutions such as query language interpreters or e-commerce tools that they have encountered in other classes or available resources.

Verification & validation report: project leader, client (secondary: maintainers)
The verification and validation document relates the results of a spectrum of tests carried out on the system, ranging from code inspections to user tests of the finished system. Its purposes include demonstrating the correctness of the system and giving maintainers foundations for further evaluation as changes are made over time. The background necessary here comes from the textbook, lectures, and external sources available from the library or the Internet. (This last part of the project also includes the actual code of the system, but this is not being proposed here as a part of the writing that qualifies this course for a WI designation)

The students in COSC 481 are coached on how to target these documents to the appropriate audiences and on the details of the communication techniques employed. (See *Teaching Methods* below.)

Formulate research questions and employ strategies for researching and responding to those questions.

The questions that students formulate in COSC 481 include those about the application domain and about the desired characteristics of the system such as: What are the data to be used in the system? What operations should be carried out on the data? How should results be displayed to users? What are the abilities of the users? What processes in the domain parallel the operation of the system? To answer these, the student interviews a client. One alternative to using an external client is to have the instructor define the system; he or she acts like a product development manager or a client. Besides interviews, the students may carry out

some informal marketing research such as asking acquaintances about desirable features for software that might be distributed to a wide population of users.

Questions about software tools, their availability, and how well those tools work with other tools include: What tool does X? What does tool Y do? How well does tool X work with existing system S? How does one acquire tool X and how much does it cost? Will tool X be supported over the planned life of the system? How does one do task T with tool X? (The last question is asked and answered many dozens of times in the course of system development.) To answer these sorts of questions, the student does web-based research, talks to knowledgeable students and faculty, reads reviews in trade publications, employs the tools' help systems, and experiments with tools, often using trial or free versions of tools under consideration.

Perhaps the most interesting questions relate to the creative aspects of the course: What software structure is best for this system? How can I make one general software component serve instead of two or more specific ones? The inventive aspects of designing and describing software help contribute to the student's education in writing and related processes.

Use discipline-specific genres to communicate information.

If the prose of software engineering qualifies as a genre, it is a straightforward, generally colorless one. One tries to be as accurate and succinct as possible. In prose that is intended to be read by a client, the technical jargon is minimized, while in the prose intended for technical staff such as programmers, a very different vocabulary may be employed. There are principles in determining the content of the documents being written, such as the avoidance of implementation details in requirements documents and the use whenever possible of quantitative targets for system performance. In addition, some prose may be highly structured, e.g., heavily nested outlines to specify system functions at different levels of detail, or recipe-like lists of actions anticipated in the execution of "use cases". Besides prose, software engineering students use various diagrams to show how data are structured, the relationships between software objects, planned deployment of the system, data flows between different functional components, message traffic over time between entities, transitions of the system between states as events occur, and others. Students in this course are expected to write in a manner consistent with typical practices in software engineering.

Understand conventions for communicating, disseminating, and interpreting information within a discipline.

The conventions that govern the syntax and semantics of diagramming notations in software engineering are well defined for all but a few techniques. There are accepted methods for creating entity-relationship diagrams, data flow diagram, the many diagrams of the Unified Modeling Language, finite state diagrams, and many others. These conventions typically define how verbal labels are to be employed (parts of speech, for example). In addition, there are clear rules for writing specifications in formal logic and some other textual forms of communication. In the realm of pure prose, industry conventions and standards are usually provided more in terms of desired outlines than details of vocabulary or syntax. For the techniques employed in this course, students are taught and expected to follow standard conventions.

Evaluation

Students' written work is evaluated across several dimensions. Attached to the syllabus are the project assignments and the evaluation sheets that one faculty member has used in this course. The entire semester project usually fills a notebook that ranges from ½ to 2 inches thick. The longer submissions generally include printouts of screen shots and the actual code of the system implemented, which we do not consider to be "writing" for the purposes of this proposal. On average, the total number of pages of actual finished writing and diagrams for the whole term project is about 50. It is not unusual to get projects that are well over 100 pages.

To the extent possible, various aspects of the project submissions are evaluated separately, e.g., anticipated scenarios of use, data descriptions, dynamics (behavior) of the system, reports of code inspections. Scores depend on the clarity of communication, its completeness, the technical correctness of any specialized forms of communication employed such as the Unified Modeling Language, and adherence to styles required by the instructor or any industrial standard being used in the course such as the Institute of Electrical and Electronics Engineers' standard number 830 for requirements documents.

In group projects, it is not always clear how much each student has contributed to the creation of submitted projects, though the goal is to ensure that each student achieve a satisfactory level of skill in written communication. To assess each student's capabilities in communication, this course does the following. Students submit records that state how much time each student spent on each facet of the project. This allows the instructor to determine if each student has gained adequate experience, and has taken sufficient responsibility for, different kinds of writing. Second, the course requires students in a group to rate one another's contributions in the creation of the projects, providing an impetus to students' making fair and reasonable contributions. Third, exams require students to produce the same kinds of writing as is required for projects, thus requiring individuals to demonstrate, on a smaller scale, targeted communication skills. Fourth, students participate in in-class exercises and presentations in which the instructor can observe and interact with individual students, determining their levels of competence with different kinds of communication and providing advice and coaching.

To link evaluations to the WI outcomes, page 9 of this proposal, right after the syllabus, explains annotations of the project assignment sheets that attempt to make these connections.

Teaching Methods

To teach students how to compose and write a document in this area, we provide examples, have them read the textbook (which has its own examples and discussions), and give feedback on in-class and homework exercises. A homework assignment or in-class exercise might, for example, ask students to write up system constraints, test plans, or create a state diagram for a hypothetical system. These short tasks serve as practice for the documents of the term project. There are standard outlines for software engineering documents that are available on the Internet and in textbooks. Usually the instructor asks students to use outlines that are based on standard ones, but that are tailored to the purposes of the project and the course. (In fact, one important point of the course is to *not* be a slave to form, but rather to ensure that the intended communication is successful.) Most of us who teach this course have used the submit-edit-rewrite cycle for the main project only sparingly, e.g., when a project increment is seriously defective, but our plans are to be more systematic about giving feedback that leads to revisions and resubmissions.

Some research techniques, such as searching for Internet-based information on software tools, are well-known to students who enter the course. Other skills, such as interviewing clients, are taught in class, discussed in handouts, and practiced in group exercises. In one method, the instructor acts as a client and the class as interviewers. After students have asked some questions, they invariably lose momentum, and the questions peter out. The instructor then points out important issues that students have not inquired about.

Research in trade and academic literature are demonstrated in class by showing searches in The Halle Library's online databases and in the archives of trade publications such as *Web Developer's Journal*, *Java World*, and *Dr. Dobbs Journal*. In addition, there are many online forums and email lists for developers who use various tools. Students who do not already use these resources are made aware of their existence and shown how to post questions.

8. Attach a syllabus (1-inch margins and 10-12 pt. font). The syllabus must include the rationale from #6 above and clearly reflect the outcomes and methods of evaluation detailed in #7 above.

Please submit all materials in electronic form.

Action of the Department/College

1. Department

Vote of department faculty: For 12 Against 0 Abstentions 0

William W. McMillan
Interim Department Head

10 April 2006
Date

2. College

College Dean

Date

Action of General Education Advisory Committee

Vote of General Education Committee: For _____ Against _____ Abstentions _____

Chairperson, General Education Advisory Committee

Date

Approval

Associate Vice-President for Undergraduate Studies and Curriculum

Date

**COSC 481 SOFTWARE ENGINEERING AND SENIOR PROJECT
Fall 20xx**

Purpose: This course teaches the student concepts and techniques for the development of complex software systems that meet real users' needs and that are correct, maintainable, and composed of reusable components. Software engineering techniques will be employed in a realistic group project.

Class meetings: Tues. and Thurs., 7:15-8:30 PM, 3xx Pray-Harrold.

Office: 511x Pray-Harrold

Office hours: Tues. and Thurs. 2:00-5:00 PM

Phone: 734-487-xxxx

Email: afacultymember@emich.edu

Textbook: Schach, S. R., *Object-Oriented and Classical Software Engineering*, 6th ed., McGraw-Hill, 2005.

Exams: The midterm and final will count for 50% of the grade and will be equally weighted.

Participation in class counts for 20% of the grade. Discussion questions about reading assignments will be distributed in advance, and students will make informal presentations related to their ongoing projects.

Project: There will be a group software project, for which groups will be assigned. The major parts will be 1) Requirements analysis, 2) Design, and 3) Implementation and evaluation. Separate handouts will describe the project in more detail. Lateness of project increments will be penalized by subtracting 5% of the value of the assignment for every business day (not class day) that it is late. Students will keep individual logs of time spent and work done which will be turned in at the same time as each project increment. Credit will be assigned to team members according to their levels of participation in the project. Groups will make informal progress reports in class. The project will count for 30% of the grade.

Rationale for Writing Intensive Designation in General Education: Courses that fulfill the writing intensive (WI) designation in EMU's general education program should immerse students in discipline-specific communication, teaching them how specialists in their area of study plan, execute, and evaluate documents and other media in professional contexts. Besides teaching specific skills in writing, carrying out research, and related activities, the intention should also be to sensitize students to a host of issues relevant to successful practice that might otherwise escape their notice. This course, COSC 481 Software Engineering and Senior Project, is an introduction to real-world software development that is required of all computer science majors. It requires a series of written reports in which students communicate in a manner consistent with established software engineering practice. In a sense, the course is primarily centered on communication, though it includes purely technical content as well. The documents that students submit include a requirements document, which serves as a description of the clients' software needs, a design document, which specifies at various levels of detail the architecture of the system, and a report of the verification and validation of the system implemented. The audiences for these reports vary from clients who have little or no technical background to project managers to very technically savvy—and critical—software developers and quality assurance specialists. Communication in this realm is only partially (and sometimes not primarily) verbal. In software development, graphical notations are very important for

communicating with clients and with technical audiences. This course introduces (or reinforces) several standard graphical notations for the specification and design of software.

Schedule:

		<i>Topics</i>	<i>Chapters</i>	<i>Due</i>
Thurs	2-Sep	Introduction	1	
Tues	7-Sep	"	1	
Thurs	9-Sep	Life-Cycle Models	2	
Tues	14-Sep	"	2	
Thurs	16-Sep	Process overview	3	
Tues	21-Sep	"	3	
Thurs	23-Sep	Requirements	10	Project proposal
Tues	28-Sep	"	10	
Thurs	30-Sep	Specification	11	
Tues	5-Oct	"	11	
Thurs	7-Oct	Evaluation	6	
Tues	12-Oct	"	6	
Thurs	14-Oct	Modules and Objects	7	
Tues	19-Oct	"	7	Requirements
Thurs	21-Oct	Review		
Tues	26-Oct	Midterm		
Thurs	28-Oct	O-O Analysis	12	
Tues	2-Nov	"	12	
Thurs	4-Nov	Design	13	
Tues	9-Nov	"	13	
Thurs	11-Nov	Implementation	14	
Tues	16-Nov	"	14	Design
Thurs	18-Nov	Integration	14	
Tues	23-Nov	Resource Estimation	9	
Thurs	25-Nov	Thanksgiving		
Tues	30-Nov	Resource Estimation	9	
Thurs	2-Dec	Maintenance	15	
Tues	7-Dec	"	15	
Thurs	9-Dec	Review		Implementation & eval
Tues	14-Dec	Final		

Grading scale:

92%	A
90%	A-
88%	B+
82%	B
80%	B-

Linking Evaluation to Outcomes

In the project assignment sheets that follow this page, the WI outcomes addressed by each item being assigned are listed by number in { curly brackets } and highlighted in yellow.

Numbered outcomes:

1. Develop and employ successful, flexible writing and reading strategies that support sustained inquiry in a discipline.
2. Use writing strategies that achieve the purposes(s) for writing and address the expectations of audience(s) within a disciplinary context.
3. Formulate research questions and employ strategies for researching and responding to those questions.
4. Use discipline-specific genres to communicate information.
5. Understand conventions for communicating, disseminating, and interpreting information within a discipline.

It is difficult to specify clear links between which outcomes are assessed by what evaluation. Number 2, for example, is covered to some extent by every assignment. Some outcomes are more emphasized at certain points than others, though, and it is in this spirit that the links were specified below. Consideration of audience is more critical in the early stages of the project because of the need to communicate with a non-technical audience. In the later stages, the technical correctness of the form of the communication is more important. Research skills come into play most strongly in determining users' needs and then again in designing and evaluating the system.

Requirements Document

The requirements document is a complete description of the system from the point of view of the client. (The client may or may not be the end user.) It lays out the entire logical structure of the system, defines all data items that the user will see, describes the functioning of the system in detail, includes sample screen designs and reports, and provides other information necessary for the client to approve the system and for the software engineer to move on to the design.

1. Purpose of the system, scope { 2, 3 }
2. User population(s) { 1, 2 }
 - Description
 - Assumed abilities and skills
3. Domain { 1, 2, 3 }
 - Description
 - Glossary
4. Example scenarios of system use { 2, 4, 5 }
5. Functions and data, selecting from: { 2, 4, 5 }
 - Data-flow diagrams
 - Data Dictionary
 - State Diagrams
 - Narrative descriptions of functions
 - Specifications (assumptions-effects-exceptions)
6. Deployment plan, interactions with existing systems { 2, 4, 5 }
7. Prototypes (screen shots)
8. Constraints { 1, 2, 3 }

Size	Speed	Usability	Reliability	
Load	Portability	Security	Privacy	...etc.
9. Estimated size of the program { 3 }
 - number of functions (methods) and/or
 - number of lines of code
10. Records of client interviews { 1, 2, 3 }
11. Validation of requirements
 - sign-off by client
 - checks of consistency and completeness
12. Time records for each group member

The requirements document should be submitted as a professional-looking report, in a report binder or in a three-ring notebook *of an appropriate size*. Please, *no slide-on report binders*.

6. Requirements Document Evaluation

- [_____ / 5] Purpose of the program, scope
- [_____ / 5] User population description
- [_____ / 5] Domain description (glossary, etc.)
- [_____ / 5] Scenarios
- [_____ / 9] Data dictionary or other data description.
- [_____ / 18] Data flow diagrams, state diagrams, use case diagrams, or other
- [_____ / 9] Service specifications: narrative and "assumptions-effects-exceptions" and main flows from use cases (if used).
- [_____ / 5] Deployment plans
- [_____ / 9] User interface prototype
- [_____ / 5] Constraints, non-functional requirements
- [_____ / 5] Estimated size of program (number of modules, lines of code)
- [_____ / 5] Interview records and validation of the requirements
- [_____ / 5] Overall magnitude and complexity
- [_____ / 5] Group member time records
- [_____ / 5] The requirements document should be submitted as a professional-looking report, in a report binder or in a three-ring notebook *of an appropriate size*.
- [_____ / 100] **Total**

Design

In this part, you describe the system in enough detail, and in a technical enough way, that someone can program it, demonstrate its correctness, reuse its components, and maintain it. As in the requirements, the structure, function, inter-relationships, and data are described. Here, though, it is *not* important that the client or user be able to read the documents. These are the “blueprints” that professionals use to create the system.

What to include:

Architecture

Class diagrams or structure charts { 3, 4, 5 }

Specifications of methods or functions

Assumptions-effects-exceptions for each method (as magnitude of the system allows) { 4, 5 }

Data communication

Object collaboration diagrams, data flow diagrams (more detailed than in requirements), or data values passed in a structure chart { 4, 5 }

Dynamics

State diagrams (more detailed than in requirements) and/or sequence diagrams { 4, 5 }

Test cases for each method or function { 1, 3, 4, 5 }

Evaluation of design

Consistency and completeness checks within design document and with requirements document (record of inspections and what was checked) { 4, 5 }

Time records for group members

Design Evaluation

Architecture

Class diagrams or structure charts

[____ / 20]

Specifications of methods or functions

Assumptions-effects-exceptions for each method (as magnitude of the system allows)

[____ / 20]

Data communication

Object collaboration diagrams, data flow diagrams (more detailed than in requirements), or data values passed in a structure chart

[____ / 10]

Dynamics

State diagrams (more detailed than in requirements) and/or sequence diagrams

[____ / 10]

Test cases for each method or function

[____ / 10]

Evaluation of design

Consistency and completeness checks within design document and with requirements document (record of inspections and what was checked)

[____ / 10]

Time records for group members

[____ / 5]

Professional presentation

[____ / 5]

Overall magnitude and complexity

[____ / 10]

[Total: ____ / 100]

Implementation and Evaluation

As the final part of your team's project, you submit the code you've written and the results of verification and validation.

Parts to include:

- Printouts of code, including comments. The style of this code affects the grade.
- Sample screens or outputs from runs.
- Verification and validation results. { 3, 4, 5 } The exact nature of the items included here depends on the V & V plans you made in the design phase. You might include things such as
 - Formal proofs of correctness
 - Inspection reports (of code, screens, design,...)
 - Results of path tests.
 - Results of partition tests.
 - Results of statistical tests.
 - Results of user tests.

(All dynamic test results should include specific test cases that were run, expected results, and actual results.)

- Individual team members' reports on what they did and how much time was spent on each facet.

Implementation Evaluation

Student or Group _____ Date _____

Complexity, size, amount completed (35) _____

Code style (25) _____

User interface (10) _____

Verification and validation results (25) _____

Physical form of submission (5) _____

General comments:

Overall (100) _____