31 October 2018

Writing Across the Campus (WAC)

The Computer Science Department
The WAC Plan for the CSC3050 Software Engineering Course
Implementing the System Development Life Cycle (CDLC)

Goal: The goal of the intensive writing course in the Department of Computer Science is for the student to produce a well-written document. This document details the different stages of the software engineering life cycle as it applies to the software product developed by the student. The product may be a simple stand-alone program or a complicated Web application in one or more environments and languages.

The student must use all stages of the software engineering life cycle to produce a quality product. A quality product addresses most if not all of the needs of the customer. The software engineering life cycle methodology requires an extensive and well-written document in a narrative form.

The document is considered "extensive" if it addresses all stages of the software engineering life cycle. The document is "well-written" if it has a clear, straightforward narrative if it is readable, and logically organized. Also, the document demonstrates good composition with a logical-orderly sequence of observations. All mechanical aspects of writing are observed by the English language.

Course Criteria:

1. CSC 305 is the designated writing intensive course for Computer Science and Computer Information Technology majors. The written component of this course accounts for **forty percent** of the overall grade.

2. CSC 305 is limited to fifteen students.

3. CSC 305 is taught by full-time faculty.

4. The CSC 305 student must produce a document containing a minimum of fifteen double-spaced pages. A minimum of 5 of those pages must be written in paragraph format that includes a research component describing the various CDLC techniques.

5. The instructor establishes four checkpoints during the semester to confer with students concerning their project.

6. The instructor allows students the opportunity to revise their document during the four designated checkpoints throughout the semester.

7. The written work in this course is assessed according to the rubric outline listed below.

THE CSC3050 SOFTWARE ENGINEERING COUSE RUBRIC

CSC 305 exposes the student to the issues and problems of creating a larger software system. The software engineering lifecycle process acts as a framework guiding the student through this larger software maze. The student thinks more clearly and rigorously about the overall problem and its solution when forced to address each of the software lifecycle stages in written form.

The Six Stages of the System Development Life Cycle (CDLC)

1. System requirements and objectives.
2. Design of the software system.
3. Writing the program code.
4. Testing and debugging the program
5. Implementation of the program at the customer location.
6. Maintenance of the application to include updates and new versions.

General Writing Guidelines for the Entire Document APA Formatting. (6 points)
All students submitting a software project document must include a cover page, a table of contents, abstract, body, and reference page. Each stage of the CDLC above must be separated into its section.

- Section headers and sub-section are written in bold. (2)
- Line spacing is double. (2)
- All tables and figures are labeled. Figures and tables have a separate numbering system. The labels for figures appear below the actual figures while those for tables appear above the tables. (2)

Also, documentation of the different stages should follow the guidelines below.

1. Requirement Specification. (4 points) Paragraph Format
The requirements of the software system to be developed and implemented are documented in a requirement specification document or section. The following needs to be stated clearly:

- The client organization for which the software system is to be developed. (1)
- Purpose of the software system and the intended users of the software system. (1)
- The details of each function to be performed by the system. (1)
- Any specific software requirements by the organization for which the system is being developed and any specific user-interface required by the organization. (1)

2. Project Management Plan. (2 points) Paragraph Format

In this section, a detailed plan and schedule are presented that articulates the time and workforce required for completing each stage of the software project.

- The resources are justified by giving details in the plan. For example, if ten days and five persons are needed to implement the coding phase, then the plan states for each person the approximate number of lines of code to be written by the individual, the section of the software system for which it is written, and the projected time to develop it. (2)
- Software tools such as Microsoft Project may be used for creating the project plan.
- The use of tables is recommended for writing this section.
- The project plan should be updated as needed because, in practice, estimates may not be accurate all the time. Each updated plan is to be shown separately in this section.

3. Design Specification. (5 points) Paragraph Format

The following items are clearly described in this section:

- The design process used; e.g., Object-Oriented Design (OOD). (1)
- All identified system components and their functions. (1)
- All user-interfaces (diagrams are recommended). (1)
- Algorithms and pseudocode (if applicable) used in the detailed design. (1)
- Details of all the different software required by the system and the reasons for their selection. (1)

4. Program Code. (5 points) Syntax Format

For each program, part/whole of the code is copied and pasted in this section with a description of the environment and tools used and a brief description of the part of the system for which the code is written.

- Each program has a meaningful name. (1)
- At the beginning of the program, the programmer will list the date the program was written, its developer(s), and its functionality. (1)
- Each update to the program is documented below with its description by date. (1)

An example is given below:

```
Date: MM/DD/YYYY.
Written by: XXXX.
Description: This program develops the main menu of an ATM for a bank. It is written in
             Java in the Linux environment. The tools used are ZZZZ.
MM/DD/ YYYY: A variable aaaa was updated to bbbb.
```

- Variable and module names are declared and documented at the beginning. (2)
- Separate modules appear after the main module in the program.

5. Testing/Debugging Specification. (8 points) Tabular Format

A test plan must be presented, preferably in tabular form. This will contain:

- The system components to be tested. (2)
- The functionalities in each component to be tested. (2)
- The test cases for each programming method or algorithm. (2)
- The results of each test are documented, and it must be mentioned if it matches the expected outcome; if not, the reasons for the incorrect response must be documented as well. Debugging methods must be documented including tools used for the debugging process. (2)

6. Implementation Guide. (3 points) Paragraph Format

In this section, the implementation details are to be mentioned.

- Details of the implementation date and environment are essential. (1)
- One of the most important things to be detailed is the place where the software system will go into production. It may be the same or different from the test site. (1)
- Any problems found during implementation, debugging measures and the initial reaction of the users must be documented as well. (1)

7 Maintenance Guide. (5 points) Paragraph and Tabular Format

This section contains:

- Details concerning each unit of software and the required maintenance. (1)
- The time intervals during which maintenance is to be carried out. (1)
- Tools and any specialized knowledge required to perform maintenance. (1)
- If a unit fails, the corrective or debugging actions (if any) that can be taken must also be documented. (2) You can also implement a tabular form is needed.

8. User Guide. (2 points) Paragraph and Bullet Format

The user guide should specify

- Intended users of the system and all functionalities intended for the users.(1)
- Each user interface menu and its usage method. (1)

Methodist University
Department of Computer Science
Spring 2018
CSC 3050 Software Engineering and UNIX Commands

Credit Hours: 3
Instructor: Terry C. House, Ph.D.
Class Location: M115
Class Time: **Tu-Thur 11:00 – 12:15PM**
Office Location: M 106
Office Hours: **MWF (3:00 PM to 4:30 PM) Tuesday Thursday: (3:15 PM to 5:00 PM)**
Email: thouse@methodist.edu
Phone: 910.630.7416

IMPORTANT: Students must check their Methodist student email account multiple times daily for any email from the instructor. If the instructor sends an email to everybody with important information about the class or to a student in particular and the student misses it because he/she did not check his/her email, then in either case, it is the student's responsibility.

Textbooks

UNIX the Ultimate Guide 3 edition 0-07-3376205

Author(s): Sumitabha Das
Publisher: McGraw-Hill

Course Description

An introduction to software engineering concepts using UNIX. Software engineering concepts will be used in the design and implementation of assignments and project.

Prerequisites

CSC 301. Students must have a grade of C or Java in the prerequisite.

Department Goals

Goal 1: Graduates who major in Computer Science (CSC) will be prepared to pursue successful careers in computing or a related field.
Goal 2: Graduates with majors in Computer Science (CSC) will be prepared to pursue advanced degrees in graduate school.
Goal 3: Students who complete the introductory course (CSC 100) in Computer Science will be able to function as computer literate individuals.
Goal 4: Graduates with majors in Computer Information Technology (CIT) will be prepared to pursue successful careers in computing or a related field.

Learning Objectives
Upon completion of the course, a student is expected to demonstrate competency in the following areas:

1. UNIX commands, that include: shell, pipes, scheduling, and scripts
2. Use of software engineering concepts in Java.
3. Use of software engineering concepts to design and implement a software project.

CSC 305 is a required course in the CSC and CIT curricula. The learning objectives for this course address department goals 1, 2 and 4.

Quality Enhancement Plan (QEP)
Methodist University has a **QUALITY ENHANCEMENT PLAN** (QEP), and we're asking students to—

Get Between the Covers!:

Improving Student Reading Skills by Developing a Culture of Reading

Evaluation
Learning outcomes of topical materials are measured by assignments and unit tests. The overall learning objectives are evaluated by a project and a comprehensive final examination.

There are class and homework assignments, 4 tests, a software development project, and one final examination. The tests and final exam will be a mix of essay and objective type questions. No library assignments are required for the course. However, a student is free to consult material from the library for this course if needed.

The weights for each are as follows:

Homework 20% = 5% x **6 assignments**

Exams 25% = 5% x **5 exams**

Project 40% = (**2 presentations, and Coding of program 20%) + (documentation 20%)**

Final exam = **15%**

The project should be written as per the specifications provided in the Software Engineering Project Writing Guide. Half of the total points in the project is allotted for the writing portion of the project.

Letter grades will be assigned as follows given the semester score x:

$90 \le x \le 100$	$80 \le x < 90$	$70 \le x < 80$	$60 \le x < 70$	$x < 60$

A	B	C	D	F

Policy pertaining to assignments and project

The due date for each assignment will be given when it is assigned in class. The project is due last week of class.. Any assignments or project submitted past the due date will not be accepted. NO EXCEPTIONS. In all cases, whether late or not, the acceptance of an assignment or project and deduction of points is at the discretion of the class instructor.

If you plan on waiting until the last minute to turn an assignment, you take the risk of encountering some unexpected events that may prevent you from turning it in. Don't wait until the last minute!

Policy pertaining to tests and exams

Tests are designed to be completed within a given time frame. Tests must be returned by the end of the class period. Otherwise, they will not be accepted. Make-up tests will only be given for students who have valid excuses like doctor's notes, police statements, and/or court excuses. Additional points may be deducted from a make-up test.

The university's published final examination schedule will be strictly followed. No deviation from that will be made. 10% of the points will be deducted for any make-up test given for the final examination.

In all cases, giving a make-up tests or deducting points lies at the discretion of the class instructor.

Cheating/Plagiarism Policy

Any form of cheating and plagiarism is strictly prohibited. Discussion of programming assignments should be carried out at high-level semantics. Never show another student your program code when you try to help him/her. Students are responsible to obey the Honor Code as stated in the MU student handbook. Any violation of the Honor Code will result in a grade of zero for the work and will be reported to the Honor Board of MU. A second offense in the same course will result in a grade of F for the course.

Attendance

Attendance will be kept. If a student is more than 10 minutes late for class for 5 days, he/she will be marked absent for one day. The determination of the 10 minutes is at the discretion of the instructor. A student is responsible for any material missed when he/she is late for class. If a student misses class any day, he/she will be responsible for the material that was taught in the class on the day he/she is absent. If a student leaves class without the permission of the instructor, he/she will be marked absent.

Students with Disabilities

Any student requiring accommodations due to one or more disabilities must bring the appropriate documentation to the Center for Personal Development no later than the first week of classes during the semester in which the accommodations are required. Together

the student and the Director of Disability Services will decide upon the accommodations to be implemented.

Disclaimer

The instructor reserves the right to update this course syllabus to enhance learning and/or to meet special needs of students.