Faculty Teaching Guide

CMIS 330
Software Engineering Principles and Techniques

About CMIS 330

Listed below are important concepts and highlights about the nature of this course and its students.

- Students should have completed an object-oriented programming course prior to taking this course.
- The focus of this course is on software life cycle documents and their completion.
- Teamwork is required for at least one of the documents.
- A skeletal SS should be provided by the instructor, or the student could be directed to provide a combination SS + SRS. The system context diagram and level 1 DFD would provide the foundation for the SRS and subsequent documents.

Using this Guide

This guide is designed to help faculty members teach the course effectively as well as to help the department maintain the appropriate alignment with the program curriculum. The guide and the course syllabus should be provided to faculty members teaching the course and should be regularly reviewed and updated. When you have suggestions and recommendations for the guide, see the corresponding 999 conference discussion.

How can you use this guide to help you teach your course? It's a long-term resource to consult as you progress, but here are some immediate ideas to:

- **Plan the flow of your class—**Look at the assignments and activities in the guide—they are extended descriptions of what you will find in the model syllabus, and will help you think about timing.

- **Introduce and explain your course to students—**Deliberately point to the intended course outcomes—you'll find them right after the course description. Also be sure to explain the level and place of the course and the kind of activities the students will experience. Students need to understand the focus and purpose of their learning.

- **Choose topics for class discussions—**Besides the outcomes, look at the Learning Activities section, which may give you ideas for discussions or techniques. The Required Concepts, Skills, and Issues to Be Covered section will also remind you of content to present to achieve the outcomes.

- **Explain class assignments—**Sample language is provided for major assignments, as well as assistance with grading weights and criteria. Feel free to cut and paste this language into your explanations. The assignments are designed to fit coherently with the course design. Make sure you specify the course outcomes for each assignment. This will show students there is purpose in the assignment.

- **Monitor student progress—**The course design and guide are planned for a sequence to achieve the learning outcomes. Look at how assignments are split up or staged in the syllabus and the suggestions in the guide for evaluating student work. You may want to institute subprojects, such as deadlines for drafts or pieces of larger projects, to let you know if students are ready for the big assignments.

- **Make sure you are teaching at the right level—**Besides the explanation of the place and level of the course at the beginning of the guide, review the Bloom's taxonomy chart at the end, which gives a quick picture of the cognitive level for this course. It can help you make sure you aren't teaching an introductory course at a level beyond your students' abilities or, conversely, teaching an advanced course at too low a level.

Course Number and Title

CMIS 330: *Software Engineering Principles and Techniques*

Course Description

Prerequisite: CMIS 115, CMIS 125, or CMIS 141. A study of software engineering from initial concept through design, development, testing, and maintenance of the product. Discussion covers software development life-cycle models. The goal is to analyze, customize, and document multiple processes to solve information technology problems. Topics include configuration management, quality, validation and verification, security, human factors, and organizational structures. Students may receive credit for only one of the following courses: CMIS 330 or CMIS 388A.

Intended Course Outcomes

List the learning outcomes for this course (phrased as "After completing this course, students should be able to ...").

1. describe and perform the key development activities in any software life cycle
2. select, tailor, and apply processes and products in order to solve IT problems
3. communicate and document all phases of a software development life cycle process to internal and external stakeholders
4. assess relevant software industry trends and best practices to judge their applicability to a given IT problem

Position of Course in Curriculum

- **Role in Program and Degree Requirements**

How does this course relate to requirements (e.g., required course, one of several choices for a specific requirement, supplemental course for major, elective)?

This is a 300-level elective course for students wanting to concentrate on software engineering and life cycle concepts.

- **Program Outcomes to Which this Course Maps**

To which program-level outcomes does this course contribute? How does it contribute?

P2. Design, develop, implement, secure, and maintain software applications that meet user requirements, using current best practices and tools for all application interfaces and domains.
P4. Plan, manage, and provide appropriate documentation and communication through all phases of the software development life cycle to ensure successful implementation of an information technology (IT) project that is on time and within budget.
P5. Identify, learn, and adapt to local and global IT trends, technologies, legalities, and policies, as well as appropriately communicate their impact to key stakeholders.
P6. Work independently or as an effective member of an application development team to determine and implement systems that meet customer requirements.

Students will develop a software life cycle document using best practices and current trends, and working within a collaborative team.

- **Relation to Hallmark Competencies, if Appropriate**

Hallmarks are the core competencies that UMUC assures each student will accomplish by the end of their degree. The hallmarks are introduced, reinforced, and emphasized in general education courses as well as the program. Which hallmarks are addressed in this course (effective writing, information literacy, technology fluency, critical thinking, ethics, quantitative literacy, scientific literacy, historical and cultural perspectives)?

H1. Historical and Cultural Perspectives

H2. Written Communications

H4. Ethics

H6. Technology Fluency

H8. Quantitative Literacy

- **Level of Course (100, 200, 300, or 400)**

Why is the stated level appropriate for this course?

This is a 300-level, upper-level course requiring knowledge learned in prerequisite courses. It includes analysis and evaluation of current trends and best practices to create system life cycle documents.

- **Relation to or Sequencing with Other Courses**

How does this course relate to other courses (e.g., to prerequisites; as a prerequisite; where the course will fall in a recommended sequence; role as part of a group of courses)?

Students should have completed at least one object-oriented programming course prior to taking this course. Students may also take CMIS 455, CMIS 460, and CMIS 465 after taking this course to further enhance their software engineering skills.

- **Importance of the Course to Other Majors and/or Disciplines**

How does this course contribute to programs outside this major (e.g., requirement, related requirement or recommended elective for another major, fulfills general education requirement)?

This course is primarily for CMIS and CMSC majors. However, students in other disciplines are welcome to take this course, provided they have the prerequisites.

Approach to Course

- **Required Concepts, Skills, and Issues to Be Covered**

List the concepts, skills, and issues that must be included as central to the course (versus optional material that may be included depending on time and instructor/student interest).

As part of the redesign process, the following concepts, skills, and issues were identified as important to student success. Faculty members should design the overall course, reading assignments, weekly discussions, activities, and projects to address these in an appropriate manner to ensure that students are prepared to accomplish the graded activities. These have been arranged in a suggested weekly sequence; however, faculty should align with assigned readings and in a way that best supports the assignments and course activities.

Date	Assignments	Due Date
Week 1	**Concepts** • software within the system context • SDLC • customer input • what are requirements? • "what" vs. "how" **Skills** • analysis **Issues** • lack of/insufficient coding skill	
Week 2	**Concepts** • documentation standards • requirements elicitation and analysis • communications **Skills** • writing requirements, statements • requirements analysis • allocating requirements **Issues** • inability to think abstractly • writing skills	

Week 3	**Concepts** • software development paradigms and special challenges **Skills** • decomposing requirements **Issues** • time management	
Week 4	**Concepts** • functional vs. object-oriented analysis • UML **Skills** • writing a design document	
Week 5	**Concepts** • translation of requirements into design **Skills** • comparing/contrasting alternative designs **Issues** • group dynamics	
Week 6	**Concepts** • testing strategies **Skills** • designing test environments	
Week 7	**Concepts** • software development • project planning **Skills** • designing test cases that verify and validate requirements	
Week 8	**Concepts** • software estimation models/algorithms • task breakdown and project scheduling **Skills** • planning software development projects that satisfy the customer's requirements while being on time, within budget, and the desired quality	

Assessment and Learning Activities

- **Assessments, Projects, and Assignments**

List and describe the planned projects/assignments for this course that will fulfill the course outcomes and allow for appropriate assessment. Mark mandatory items with an asterisk (*). If an item is included in the program plan for assessment of a program-level outcome or a hallmark outcome, mark and specify the assessment activity to be required for that purpose as well (e.g., common final exam).

- conferences*
- final project: software development plan (SDP)*
- software requirements specification (SRS) document*
- software design document (SDD)*
- software test specification (STS) document (team project)*

Faculty are free to slightly modify the assignments' contributions to the final grade by plus or minus 5 percent. For example, weekly conferences are listed as 15 percent in the syllabus. Faculty are free to modify this level, provided they stay in the range between 10 percent and 20 percent.

Project Descriptions

To reinforce the concepts of the software development process, the course project entails submitting software development products. These products are documents that are generated during the course of software development. One key software development product that you will *not* submit is the program source code. Refer to the course schedule for project submission due dates.

The documents to be developed in this class are

- software requirements specification (SRS) document
- software design document (SDD)
- software test specification (STS) document
- software development plan (SDP)

IEEE standard reference manuals will be posted under Reserved Readings to provide suggested outlines for each of the project documents. Other published outlines for these documents, such as MIL-STD-498, are also acceptable. Because of the compressed course schedule, each of these documents should be on the order of six to 10 typed pages or their electronic equivalent. Technical content rather than document size is the factor that will be considered in grading. Graphical content can help satisfy the page requirement.

I will post conference topics listing the minimum required information for each document. Keep all the documents as short as possible, focusing on the detail with respect to the candidate IT system described in the SS.

In light of the high volume of information in the typical SDD and STS, you may submit a representative sample of the expected content of these documents—that is, a portion of the module or object design and a portion of the software test cases for the SDD and STS, respectively.

You are encouraged to use spell-checking and grammar-checking software to help you improve your writing. Well-written projects with adequate technical content will always get favorable grades.

The following tables provide the grading rubric for each of the software development lifecycle (SDLC) products/documents.

Rubrics

I will use the following rubric to objectively grade the SRS:

Essential Contents	**Unacceptable** **(0–6 pts.)**	**Acceptable** **(7–12 pts.)**	**Distinguished** **(13–14 pts.)**
functional/class analysis	- Functions/objects listed in the SS are essentially repeated, with no elaboration or decomposition	- Functions/objects are elaborated/decomposed from the SS	- Engineering rationale for the decomposition is provided
requirements derivation from the SS	- SS requirements are repeated with no decomposition	- SS requirements are decomposed in a rational manner - Discussion of the hierarchical (parent-child) relationship between the SS and SRS requirements is given	- Engineering rationale is provided for the derived requirements
behavioral analysis	- Only a textual discussion is provided describing the observable behavior of the software with respect to its external stimuli - - Discussion does not address all	- State transition diagram, control flow diagrams (CFDs), logic tables, flowcharts, object collaboration diagram, sequence charts, and other graphic notations are provided to describe the behavior of the system based on	- Multiple types of notation are included to provide several views of the behavior of the system based on stimuli over time - Narrative text supports the

	external stimuli identified in the context diagram	external stimuli	graphic notation to provide an in-depth view of the desired software behavior
data analysis	- Data elaboration is incomplete or absent	- Data elaboration is at least one layer deep, demonstrating an understanding of the problem statement and proposed IT system - Data dictionary is included	- Data elaboration is more than one layer deep, with a corresponding data dictionary
human-computer interface (HCI) analysis	- HCI is not discussed	- HCI is discussed, but SRS does not differentiate among the types of system users	- HCI is discussed, and the types of users are clearly differentiated
requirements tracing from the SS	- No requirements tracing to the SS is included	- A high-level tracing of SRS requirements to SS requirements is provided	- A low-level tracing of SRS requirements to SS requirements is provided
formal document structure based on an industry standard, including title page, TOC, references	- SRS has little formal structure - No references are provided	- SRS has little formal structure - References are provided	- SRS has formal structure based on an SRS standard - Complete list of references is provided, including SS and SRS standard

I will use the following rubric to objectively grade the SDD:

Essential Contents	**Unacceptable** **(0–8 pts.)**	**Acceptable** **(9–14 pts.)**	**Distinguished** **(15–17 pts.)**
system (architectural) design	- Textual description of the system architecture is provided (this can be a repeat of	- Both a graphic and a textual description of the system architecture are provided (these can	- Discussion of custom versus commercial system components is provided

	the architectural description from the SS)	be repeats of the architectural description from the SS)	
procedural design	- Only names are given to functions/methods, without elaborations of the internal logic	- UML diagrams, flowcharts, sequence diagrams, structured English, or other formal notations are provided to elaborate on the functions/methods	- Graphic procedural design is supported with explanatory text and engineering rationales
interface design	- Only a qualitative discussion of the external interfaces is provided - No internal system interfaces are discussed	- External interfaces are defined with respect to the specific protocols and standards that they will employ	- Both internal and external interfaces are defined in terms of the specific protocols and standards that they will employ
data design	- Data design is simply a repeat of the information from the SRS, or is absent	- An entity-relationship (E-R) diagram is provided	- In addition to an E-R diagram and other graphic design notations, UML/class diagrams are included
requirements tracing from the SS/SRS	- No tracing of design components to the requirements is provided	- High-level tracing of design components to sections of the SRS is provided	- Low-level tracing of design components to individual requirements of the SRS is provided
formal document structure based on an industry standard, including title page, TOC, references	- SDD has little formal structure - No references are provided	- SDD has little formal structure - References are provided	- SDD has a formal structure based on an SDD standard - Complete list of references is provided, including SS and SRS and SDD standards

I will use the following rubric to objectively grade the STS:

Essential Contents	Unacceptable (0–8 pts.)	Acceptable (9–14 pts.)	Distinguished (15–17 pts.)
test approach	- There is little to no discussion of the overall test strategy, e.g., ?big-bang? vs. integration testing	- High-level discussion of the overall test strategy is provided	- Detailed discussion of the test strategy is provided that is consistent with the SDLC process employed, e.g., linear vs. incremental - SDLC process is identified in the SPP
test environment description	- Little to no test environment description is given	- Test environment is described, but description is not comprehensive	- Detailed discussion of the test environment is provided, with HW/SW resources, tools, test data, infrastructure, and test driver elements required for high-fidelity testing
white box test case(s)	- Test case(s) are not distinguished as being either white or black box	- High-level and qualitative description of the test case is provided - Little detail is given on test data, tools, and expected results	- Detailed description is included of the test case, identifying the test data, test steps, and expected results - Pass/fail criteria are provided
black box test case(s)	- Test case(s) are not distinguished as being either white or black box	- High-level and qualitative description of the test case is provided - Little detail is given on test data, tools, and expected	- Detailed description is included of the test case, identifying the test data, test steps, and expected results

		results	- Pass/fail criteria are provided
requirements tracing from the SS/SRS	- There is no discussion of the need to connect test cases with requirements verification (white box) and validation (black box)	- Qualitative discussion is provided that conveys the understanding that the STS must be driven by the SS and SRS requirements	- Requirements IDs or other unique identifiers from the SS/SRS are included in the test case descriptions
formal document structure based on an industry standard, including title page, TOC, references	- STS has little formal structure - No references are provided	- STS has little formal structure - References are provided	- STS has formal structure based on an STS standard - Complete list of references is provided, including SS and SRS, SDD, and STS standards

I will use the following rubric to objectively grade the SDP:

Essential Contents	**Unacceptable** **(0–5 pts.)**	**Acceptable** **(6–10 pts.)**	**Distinguished** **(11–12 pts.)**
problem statement	- Problem statement is not included	- Problem statement is included, with some elaborating discussion to provide context for IT system	- Problem statement is included, with a detailed discussion of the boundaries of the IT system, with engineering rationale
project scope	- No project scope discussion is provided	- Project scope is discussed, but this section essentially reiterates the problem statement	- Project scope is clearly discussed and clearly linked to problem statement, with engineering rationale
actors and responsibilities	- There is limited to no discussion of the roles and responsibilities of the key actors in	- Only the project manager and software engineers are discussed	- The project manager, software engineers, testers, QA, CM, V&V, and client are

	the project		discussed
project effort and duration estimates	- Only a qualitative discussion of effort and duration is provided	- A quantitative analysis of project effort and duration is provided based on techniques discussed in the text or other valid references	- Multiple quantitative methods for effort and duration are provided to demonstrate the need to show variations and rationales for the best-estimate
project schedule with work breakdown	- No Gantt chart or other graphic depiction of the SDLC schedule is given - There is no correlation of the schedule and the duration and effort estimates - There is no decomposition of high-level SDLC tasks into lower-level sub-tasks, i.e., work breakdown	- Gantt or other chart depicting critical SDLC tasks over time is provided - Project timeline is consistent with the project duration estimate - SDLC tasks are insufficiently or not at all decomposed	- Schedule correlates with project effort and duration estimates - SDLC task decomposition, i.e., work breakdown, within schedule is detailed, and incorporates overarching CM and QA tasks.
staffing	- No estimate is made of staff size based on effort and duration estimates - There is only a limited discussion of the required staffing, with no rationale	- Staff size estimate is quantitative but not based on the methods provided in the text or other references	- Detailed discussion of staffing is provided using formal estimation models, e.g., COCOMO, that are based on the text or other valid references
resources, dependencies, and project specifics	- Little to no discussion of project resources,	- Discussion includes resources, besides staff, that are	- Comprehensive discussion itemizes all resources

	including staff, is provided - No discussion of project dependencies is given	required for the successful completion of the project - Project dependencies are addressed	required for the successful completion of the project - SDLC process is selected - Programming language is identified with rationale - Project dependencies are discussed
risk assessment	- No risk assessment is provided	- Project risks are discussed qualitatively, with no analysis or ranking of risks	- Project risks are presented quantitatively and ranked according to techniques described in the text or other valid references - Risk mitigation is discussed for the highest-ranked risks
formal document structure based on an industry standard, including title page, TOC, references	- SDP has little formal structure - No references are provided	- SDP has little formal structure - References are provided	- SDP has formal structure based on an SDP standard - Complete list of references is provided, including problem statement and SPP standard

Faculty may decide which of the documents makes the most sense for a team project. It is recommended not to be the first document, because some group dynamics are needed and adjustments for the course.

- **Learning Activities**

List and describe the learning activities for this course that will address learning outcomes and fulfill the principles of the SUS learning model (e.g., working in study groups or learning collaboratively in small groups; investigating and discussing a resource; responding to material or taking a quiz in an online course module; reviewing and commenting on another student's draft; revising a draft based on specific criteria). These may or may not be graded activities. Specify required learning activities as well as suggested default learning activities that can be used at the instructor's option.

Weekly conferences should be related to the concepts and allow students to comment on one another's work and contributions.

Certification organizations can and should be addressed by the instructor in conference discussions and/or homework assignments.

- **Cognitive Level and Assessing Course Outcomes**

Use the table on the following page to indicate the level at which each course outcome will be addressed and assessed in the course as per Bloom's taxonomy. (See more on Bloom's taxonomy below the table.) Mark the appropriate cell with an X to indicate the level for the outcome.

Course Outcomes Mapped to Bloom's Taxonomy

Bloom's taxonomy is a tool for classifying the cognitive demand level of instructional activities or questions. As one moves through the hierarchy from knowledge to evaluation, the activities and questions require increasingly higher-level thinking skills. See the list below this table for a summary of Bloom's taxonomy and examples of verbs corresponding to each cognitive level.

	Cognitive Demand Level – Based on Bloom's Taxonomy					
Course Outcome	**Knowledge (Lowest)**	**Comprehension**	**Application**	**Analysis**	**Synthesis**	**Evaluation (Highest)**
1. describe and perform the key development activities in any software life cycle		X				
2. select, tailor, and apply processes and products in order to solve IT problems				X		
3. communicate and document all phases of a software development life cycle process to internal and external stakeholders			X			
4. assess relevant software industry trends and best practices to judge their applicability to a given IT problem				X		

Bloom's Taxonomy Terms

Source: Wyatt, A. T. (2001). *Bloom's taxonomy*. Retrieved September 11, 2006, from http://cs1.mcm.edu/~awyatt/csc3315/bloom.htm

Level	Type of Activity or Question	Verbs Used for Outcomes
Lowest level	**Knowledge**	define, memorize, repeat, match, record, list, recall, name, relate, collect, label, specify, cite, enumerate, recite, tell, recount
	Comprehension	restate, summarize, differentiate, discuss, describe, recognize, explain, express, identify, locate, report, retell, review, translate, paraphrase
	Application	exhibit, solve, manipulate, interview, simulate, apply, employ, use, demonstrate, dramatize, practice, illustrate, operate, calculate, show, experiment
Higher levels	**Analysis**	interpret, classify, analyze, arrange, differentiate, group, compare, organize, contrast, examine, scrutinize, survey, categorize, dissect, probe, create an inventory, investigate, question, discover, inquire, distinguish, detect, diagram, chart, inspect
	Synthesis	compose, set up, plan, prepare, propose, imagine, produce, hypothesize, invent, incorporate, develop, generalize, design, originate, formulate, predict, arrange, assemble, construct, create
	Evaluation	judge, assess, decide, measure, appraise, estimate, evaluate, rate, deduce, compare, score, value, predict, revise, choose, conclude, recommend, determine, criticize, test

Appendix A

CMIS 330: *Software Engineering Principles and Techniques*

Course Syllabus

Faculty Contact Information

Course Materials

Author	Required Text	Publisher	ISBN
Pressman	*Software Engineering: A Practitioner's Approach*, 7th ed.	McGraw-Hill	9780073375977

Reserved Readings

IEEE Std 1233-1998: IEEE Guide For Developing System Requirements Specifications
IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
IEEE Std 1016-1998: IEEE Recommended Practice for Software Design Descriptions
IEEE Std 829-1998: IEEE Standard for Software Test Documentation
IEEE Std 1058-1998: IEEE Standard for Software Project Management Plans

Course Description

Prerequisite: CMIS 115, CMIS 125, or CMIS 141. A study of software engineering from initial concept through design, development, testing, and maintenance of the product. Discussion covers software development life-cycle models. The goal is to analyze, customize, and document multiple processes to solve information technology problems. Topics include configuration management, quality, validation and verification, security, human factors, and organizational structures. Students may receive credit for only one of the following courses: CMIS 330 or CMIS 388A.

Course Outcomes

After completing this course, you should be able to

- describe and perform the key development activities in any software life cycle
- select, tailor, and apply processes and products in order to solve IT problems
- communicate and document all phases of a software development life cycle process to internal and external stakeholders
- assess relevant software industry trends and best practices to judge their applicability to a given IT problem

Course Introduction

This course is a survey of topics in software engineering, focusing on the development of software for large computer systems. Because of its complexity, software for large systems is the most difficult to build and requires teams of developers. Each developer must possess skills in problem analysis, system and software design, and test planning and test case design. In other words, each member of the team must be a software engineer. Although new by engineering standards, software engineering has an influence on our personal and professional lives as pervasive as that of computing systems.

We will learn about the discipline of software engineering by studying the software engineering development process and the software product. Process and product are really two sides of the same coin. The goal of the process is the product, and there is no product without the process. To learn about the different roles of software products, we will study software development processes that vary from one another in terms of sequence and frequency of basic tasks. The basic tasks are invariant, as shown in this diagram of one of the predominant software development processes, linear-sequential:

Finally, software process and product exist within the larger framework of a software methodology. A *software methodology* is essentially a fundamental view of software. The methodologies we will cover are structured (also referred to as *functional*) and object-oriented. Both methodologies guide the software engineer in the performance of the tasks of software development.

Grading Information and Criteria

You are responsible for the following graded items:

Conference participation	15%
Software requirements specification (SRS) document	20%
Software design document (SDD)	20%
Software test specification (STS) document (team project)	20%
Final project: software development plan (SDP)	25%
Total	**100%**

The grading scale, based on 100 points, is as follows:

A =	90-100
B =	80–89
C =	70–79
D =	60–69
F =	0–59

Participation

By registering for a Web-based course, you have made a commitment to participate in course conferences as well as other online activities. Plan to participate regularly. Participation for this course is defined as proactive discussion in weekly conferences and discussion questions. This requires you to actively reflect on weekly readings and to develop original ideas in your responses. You are expected to demonstrate critical thinking and your understanding of the content in the assigned readings as they relate to the issues identified in the conference discussion.

You are expected to respond to a main topic each week and read other student posts. You are encouraged to respond to other students as well as to your instructor. Note that your online conference participation counts significantly toward your final grade.

When communicating with others in this class, always work to be respectful.

Other Information

Project Descriptions

To reinforce the concepts of the software development process, the course project entails submitting software development products. These products are documents that are generated during the course of software development. One key software development product that you will *not* submit is the program source code. Refer to the course schedule for project submission due dates.

The documents to be developed in this class are

- software requirements specification (SRS) document
- software design document (SDD)
- software test specification (STS) document
- software development plan (SDP)

IEEE standard reference manuals will be posted under Reserved Readings to provide suggested outlines for each of the project documents. Other published outlines for these documents, such as MIL-STD-498, are also acceptable. Because of the compressed course schedule, each of these documents should be on the order of six to 10 typed pages or their electronic equivalent. Technical content rather than document size is the factor that will be considered in grading. Graphical content can help satisfy the page requirement.

I will post conference topics listing the minimum required information for each document. Keep all the documents as short as possible, focusing on the detail with respect to the candidate IT system described in the SS.

In light of the high volume of information in the typical SDD and STS, you may submit a representative sample of the expected content of these documents—that is, a portion of the module or object design and a portion of the software test cases for the SDD and STS, respectively.

You are encouraged to use spell-checking and grammar-checking software to help you improve your writing. Well-written projects with adequate technical content will always get favorable grades.

The following tables provide the grading rubric for each of the software development lifecycle (SDLC) products/documents.

Rubrics

I will use the following rubric to objectively grade the SRS:

Essential Contents	Unacceptable (0–6 pts.)	Acceptable (7–12 pts.)	Distinguished (13–14 pts.)
functional/class analysis	- Functions/objects listed in the SS are essentially repeated, with no elaboration or decomposition	- Functions/objects are elaborated/decomposed from the SS	- Engineering rationale for the decomposition is provided
requirements	- SS requirements	- SS requirements are	- Engineering

derivation from the SS	are repeated with no decomposition	decomposed in a rational manner - Discussion of the hierarchical (parent-child) relationship between the SS and SRS requirements is given	rationale is provided for the derived requirements
behavioral analysis	- Only a textual discussion is provided describing the observable behavior of the software with respect to its external stimuli - - Discussion does not address all external stimuli identified in the context diagram	- State transition diagram, control flow diagrams (CFDs), logic tables, flowcharts, object collaboration diagram, sequence charts, and other graphic notations are provided to describe the behavior of the system based on external stimuli	- Multiple types of notation are included to provide several views of the behavior of the system based on stimuli over time - Narrative text supports the graphic notation to provide an in-depth view of the desired software behavior
data analysis	- Data elaboration is incomplete or absent	- Data elaboration is at least one layer deep, demonstrating an understanding of the problem statement and proposed IT system - Data dictionary is included	- Data elaboration is more than one layer deep, with a corresponding data dictionary
human-computer interface (HCI) analysis	- HCI is not discussed	- HCI is discussed, but SRS does not differentiate among the types of system users	- HCI is discussed, and the types of users are clearly differentiated
requirements tracing from the SS	- No requirements tracing to the SS is included	- A high-level tracing of SRS requirements to SS requirements is provided	- A low-level tracing of SRS requirements to SS requirements is provided
formal document structure based on an industry standard, including title page, TOC, references	- SRS has little formal structure - No references are provided	- SRS has little formal structure - References are provided	- SRS has formal structure based on an SRS standard - Complete list of references is provided, including SS and SRS standard

I will use the following rubric to objectively grade the SDD:

Essential Contents	Unacceptable (0–8 pts.)	Acceptable (9–14 pts.)	Distinguished (15–17 pts.)
system (architectural) design	- Textual description of the system architecture is provided (this can be a repeat of the architectural description from the SS)	- Both a graphic and a textual description of the system architecture are provided (these can be repeats of the architectural description from the SS)	- Discussion of custom versus commercial system components is provided
procedural design	- Only names are given to functions/methods, without elaborations of the internal logic	- UML diagrams, flowcharts, sequence diagrams, structured English, or other formal notations are provided to elaborate on the functions/methods	- Graphic procedural design is supported with explanatory text and engineering rationales
interface design	- Only a qualitative discussion of the external interfaces is provided - No internal system interfaces are discussed	- External interfaces are defined with respect to the specific protocols and standards that they will employ	- Both internal and external interfaces are defined in terms of the specific protocols and standards that they will employ
data design	- Data design is simply a repeat of the information from the SRS, or is absent	- An entity-relationship (E-R) diagram is provided	- In addition to an E-R diagram and other graphic design notations, UML/class diagrams are included
requirements tracing from the SS/SRS	- No tracing of design components to the requirements is provided	- High-level tracing of design components to sections of the SRS is provided	- Low-level tracing of design components to individual requirements of the SRS is provided

formal document structure based on an industry standard, including title page, TOC, references	- SDD has little formal structure - No references are provided	- SDD has little formal structure - References are provided	- SDD has a formal structure based on an SDD standard - Complete list of references is provided, including SS and SRS and SDD standards

I will use the following rubric to objectively grade the STS:

Essential Contents	**Unacceptable** **(0–8 pts.)**	**Acceptable** **(9–14 pts.)**	**Distinguished** **(15–17 pts.)**
test approach	- There is little to no discussion of the overall test strategy, e.g., ?big-bang? vs. integration testing	- High-level discussion of the overall test strategy is provided	- Detailed discussion of the test strategy is provided that is consistent with the SDLC process employed, e.g., linear vs. incremental - SDLC process is identified in the SPP
test environment description	- Little to no test environment description is given	- Test environment is described, but description is not comprehensive	- Detailed discussion of the test environment is provided, with HW/SW resources, tools, test data, infrastructure, and test driver elements required for high-fidelity testing
white box test case(s)	- Test case(s) are not distinguished as being either white or black box	- High-level and qualitative description of the test case is provided - Little detail is given on test data, tools, and expected results	- Detailed description is included of the test case, identifying the test data, test steps, and expected results - Pass/fail criteria are provided
black box test	- Test case(s) are	- High-level and	- Detailed

case(s)	not distinguished as being either white or black box	qualitative description of the test case is provided - Little detail is given on test data, tools, and expected results	description is included of the test case, identifying the test data, test steps, and expected results - Pass/fail criteria are provided
requirements tracing from the SS/SRS	- There is no discussion of the need to connect test cases with requirements verification (white box) and validation (black box)	- Qualitative discussion is provided that conveys the understanding that the STS must be driven by the SS and SRS requirements	- Requirements IDs or other unique identifiers from the SS/SRS are included in the test case descriptions
formal document structure based on an industry standard, including title page, TOC, references	- STS has little formal structure - No references are provided	- STS has little formal structure - References are provided	- STS has formal structure based on an STS standard - Complete list of references is provided, including SS and SRS, SDD, and STS standards

I will use the following rubric to objectively grade the SDP:

Essential Contents	**Unacceptable** **(0–5 pts.)**	**Acceptable** **(6–10 pts.)**	**Distinguished** **(11–12 pts.)**
problem statement	- Problem statement is not included	- Problem statement is included, with some elaborating discussion to provide context for IT system	- Problem statement is included, with a detailed discussion of the boundaries of the IT system, with engineering rationale
project scope	- No project scope discussion is provided	- Project scope is discussed, but this section essentially reiterates the problem statement	- Project scope is clearly discussed and clearly linked to problem statement, with engineering rationale
actors and responsibilities	- There is limited to no discussion of the roles and responsibilities of	- Only the project manager and software engineers are discussed	- The project manager, software engineers, testers, QA, CM, V&V, and

	the key actors in the project		client are discussed
project effort and duration estimates	- Only a qualitative discussion of effort and duration is provided	- A quantitative analysis of project effort and duration is provided based on techniques discussed in the text or other valid references	- Multiple quantitative methods for effort and duration are provided to demonstrate the need to show variations and rationales for the best-estimate
project schedule with work breakdown	- No Gantt chart or other graphic depiction of the SDLC schedule is given - There is no correlation of the schedule and the duration and effort estimates - There is no decomposition of high-level SDLC tasks into lower-level sub-tasks, i.e., work breakdown	- Gantt or other chart depicting critical SDLC tasks over time is provided - Project timeline is consistent with the project duration estimate - SDLC tasks are insufficiently or not at all decomposed	- Schedule correlates with project effort and duration estimates - SDLC task decomposition, i.e., work breakdown, within schedule is detailed, and incorporates overarching CM and QA tasks.
staffing	- No estimate is made of staff size based on effort and duration estimates - There is only a limited discussion of the required staffing, with no rationale	- Staff size estimate is quantitative but not based on the methods provided in the text or other references	- Detailed discussion of staffing is provided using formal estimation models, e.g., COCOMO, that are based on the text or other valid references
resources, dependencies, and project specifics	- Little to no discussion of project resources, including staff, is provided - No discussion of project	- Discussion includes resources, besides staff, that are required for the successful completion of the project	- Comprehensive discussion itemizes all resources required for the successful completion of the project

	dependencies is given	- Project dependencies are addressed	- SDLC process is selected - Programming language is identified with rationale - Project dependencies are discussed
risk assessment	- No risk assessment is provided	- Project risks are discussed qualitatively, with no analysis or ranking of risks	- Project risks are presented quantitatively and ranked according to techniques described in the text or other valid references - Risk mitigation is discussed for the highest-ranked risks
formal document structure based on an industry standard, including title page, TOC, references	- SDP has little formal structure - No references are provided	- SDP has little formal structure - References are provided	- SDP has formal structure based on an SDP standard - Complete list of references is provided, including problem statement and SPP standard

Academic Policies

Academic Integrity

UMUC is an academic community that honors integrity and respect for others, and it is expected that, as a member of this community, you will maintain a high level of personal integrity in your academic work at all times.

Academic dishonesty is the failure to maintain academic integrity, and includes the intentional or unintentional presentation of another person's ideas or products as your own (plagiarism); the use or the attempt to make use of unauthorized materials, information, or study aids in any academic exercise; and the performance of work for another student (cheating). All academic work you submit during your time at UMUC must be original and must not be reused in other courses.

Turnitin.com

The university has a license agreement with Turnitin, an educational tool that helps identify and prevent plagiarism from Internet resources. I may use the service in class, either by requiring you to submit assignments electronically to Turnitin, by submitting assignments on your behalf, or by providing the option for you to check your own work for originality. The Turnitin Originality Report will indicate the amount of original text in your work and whether all the material that you quoted, paraphrased, summarized, or used from another source is appropriately referenced.

If you or I submit all or part of your assignment to the Turnitin service, Turnitin will by default store that assignment in its database. The assignment will be checked for any matches between your work and other material stored in Turnitin's database. If you object to the long-term storage of your work in the Turnitin database, you must let me know no later than two weeks after the start of this class.

You have three options regarding the storage of your assignment in the Turnitin database: 1) You can do nothing; your assignment will then be stored in the Turnitin database for the duration of UMUC's contract with Turnitin; 2) You can ask me to have Turnitin store your assignment only for the duration of the semester or term, then have your assignment deleted from the Turnitin database once the class is over; or 3) You can ask me to change the Turnitin settings so that your assignment is not stored in the Turnitin database at all.

Please note: I may use other services in addition to or in place of Turnitin to check your work for plagiarism.

Course Expectations

For an eight-week course, you should expect to spend about six hours per week participating in class discussions and activities (online or onsite) and two to three times that number of hours outside class in study, assigned reading, and preparation of assignments. Courses offered in shorter formats will require more time per week. You are expected to meet the same learning outcomes and perform the same amount of work in an online course as in an onsite course. Active participation is required in all online courses, and you should expect to log in to your online course several times a week.

The following links to important academic policies and other information are provided to help you as you complete your coursework at UMUC.

Policies and Procedures

- Policy and Procedures on Affirmative Action, Equal Opportunity, and Sexual Harassment—Nondiscrimination: It is the policy of UMUC that no student or employee of the university or contractor/vendor conducting business with the university may discriminate on the basis of race, religion, color, creed, sex (including sexual harassment), marital status, age, national origin, political affiliation, mental or physical disability, or sexual orientation. Individuals who believe they have been discriminated against because of any factor protected under this policy may file a complaint of discrimination.

- Information on Support for Disabled Students

- University System of Maryland Board of Regents' Policy on Academic Integrity

- UMUC's Policy on Academic Dishonesty and Plagiarism

- UMUC's Policy on the Grade of Incomplete, Grade Pending, and Withdrawal

- UMUC's Policy on the Code of Student Conduct

- UMUC's Policy and Procedures for Review of Alleged Arbitrary and Capricious Grading

For more information about student services and more general information, visit UMUC's website at http://www.umuc.edu.

Faculty Bio

Eight-Week Course Schedule

Date	Assignments	Due Date
Week 1	**Read:** • Pressman, chapters 1, 2.1–2.4, 2.5–2.9, 3 • IEEE standard 1233-1998, Guide For Developing System Requirements Specifications, sections 3–7, annex A • module 1 commentary **Do:** • conference topic discussions	
Week 2	**Read:** • Pressman, chapters 4, 5.1–5.8, 6.1–6.5 • IEEE standard 830-1998, Recommended Practice For Software Requirements Specifications, sections 3–4 • module 2 commentary **Do:** • conference topic discussions • submit software requirements specification (SRS) document	
Week 3	**Read:** • Pressman, chapter 8, 9.1–9.4, 9.6 • module 3 commentary **Do:** • conference topic discussions	
Week 4	**Read:** • Pressman, chapters 10.1, 10.2, 11.1–11.4 • IEEE standard 1016-1998, Recommended Practice for Software Design Descriptions, sections 5–6, annex A **Do:** • conference topic discussions • submit software design document (SDD)	

Week 5	**Read:** • Pressman, chapters 16, 17 • module 4 commentary **Do:** • conference topic discussions	
Week 6	**Read:** • Pressman, chapters 18.1–18.6, 19 • IEEE standard 829, Standard for Software Test Documentation, sections 3–11, annex A **Do:** • conference topic discussions • submit software test specification (STS) document (team project)	
Week 7	**Read:** • Pressman, chapters 26, 27 • IEEE standard 1058-1998, Standard for Software Project Management Plans • module 5 commentary **Do:** • conference topic discussions	
Week 8	**Read:** • Pressman, chapters 28, 22 **Do:** • conference topic discussions • submit final project: software development plan (SDP)	

Course Outcomes

1.	describe and perform the key development activities in any software life cycle
2.	select, tailor, and apply processes and products in order to solve IT problems
3.	communicate and document all phases of a software development life cycle process to internal and external stakeholders
4.	assess relevant software industry trends and best practices to judge their applicability to a given IT problem