RECEIVED
FEB 15 2001
LIBERAL STUDIES

LSC Use Only
Number: _____
Submission Date: _____
Action-Date: _____

UWUCC USE Only
Number: 01-12d ~~00-55d~~
Submission Date: _____
Action-Date: App. 10/9/01
Senate App 12/4/01

CURRICULUM PROPOSAL COVER SHEET
University-Wide Undergraduate Curriculum Committee

I. CONTACT

Contact Person Charles Shubra Phone 357-7917

Department Computer Science

II. PROPOSAL TYPE (Check All Appropriate Lines)

_____ **COURSE** _____
Suggested 20 character title

___ New Course* _____
Course Number and Full Title

_X_ Course Revision _COSC 319 Software Engineering Concepts_
Course Number and Full Title

RECEIVED
OCT - 3 2001
LIBERAL STUDIES

___ Liberal Studies Approval+ for new or existing course _____
Course Number and Full Title

___ Course Deletion _____
Course Number and Full Title

___ Number and/or Title Change _____
Old Number and/or Full Old Title

_____
New Number and/or Full New Title

___ Course or Catalog Description Change _____
Course Number and Full Title

Rev.

_____ **PROGRAM:** _____ Major _____ Minor _____ Track

___ New Program* _____
Program Name

___ Program Revision* _____
Program Name

___ Program Deletion* _____
Program Name

___ Title Change _____
Old Program Name

_____
New Program Name

RECEIVED
NOV 19 2001
LIBERAL STUDIES

III. Approvals (signatures and date)

Tim Wahls
Department Curriculum Committee

Gary L Bretthauer
Department Chair

College Curriculum Committee

College Dean

+Director of Liberal Studies (where applicable)

*Provost (where applicable)

Part II. Description of Curriculum Change

1. New Syllabus of Record

2. Summary of the proposed revision

Change the prerequisite for COSC 319, Software Engineering Concepts, from COSC 315 to COS 220 and COSC 310.

3. Justification for the revision

The original prerequisite for COSC 319 was meant to assure that students had experienced enough software development to engender a sufficient level of programming maturity. COSC 315, which was a second programming course using COBOL as the programming language provided such maturity. Since COSC 315 is being eliminated from the Computer Science curriculum, another set of prerequisites has been identified to assure an acceptable level of programming maturity. Since at least 1/3 of the material originally in COSC315 has migrated to COSC 220, it makes sense to include COSC 220 in the new prerequisites for COSC 319. However, since COSC 220 did not inherit all of the subject matter from COSC 315 an additional course has been included in the prerequisite. That additional course is COSC 310; COSC 310 will provide both the data structures and the additional programming maturity seen as necessary by the faculty.

4. Old Syllabus of Record

5. Letters of Support

COSC 319 Appendix A
Software Engineering Concepts
Syllabus of Record

I. Catalog Description Programming

3 credits
3 lecture hours
0 lab hours
(3c-0l-3sh)

COSC 319
Prerequisites: COSC 220 and 310 or permission of the instructor

Software engineering concepts include the collection of tools, procedures, methodologies and accumulated knowledge about the development and maintenance of software based systems. This course is strongly suggested for any student planning to take an internship in Computer Science. After an overview of the phases of the software lifecycle, current methodologies, tools and techniques being applied to each phase will be discussed in depth with localized exercises given to reinforce learning of concepts.

II. Course Objectives

This course will serve to broaden the student's understanding of the issues and latest developments in the area of software development and maintenance. To reach this goal, the following objectives need to be met:

1. Define the current state of software development and maintenance characterized as "the software crisis".

2. Understand the multidimensional aspect of software engineering, which is the current best attempt at solving the software crisis.

3. Become familiar with popular models of the software development and maintenance process.

4. Using the waterfall model, study the inputs, outputs, and processes present in each phase.

5. Study the core concepts present in several popular methodologies and be able to identify strengths and weaknesses of each.

6. Study existing CASE tools to be able to identify opportunities to automate tasks through the use of such tools.

7. Consider the issues and techniques present in confidence gaining measures residing in each phase of the software lifecycle.

8. Briefly investigate problems present in project management.

III. Course Outline

The following subjects will be addressed:

A.	Course Introduction and Administration	0.5 hours
B.	The Software Crisis and Software Engineering	3.0 hours
C.	The Software Life Cycle - A Model of Software Development	1.5 hours
D.	Requirements Analysis	1.5 hours
E.	Design Issues	3.0 hours
F.	Design Methodologies	6.0 hours
G.	Implementation Techniques	3.0 hours
H.	Development Tools	3.0 hours
I.	Software Quality	6.0 hours
J.	Generic Code and Automatic Code Generation	6.0 hours
K.	Programming Environments	3.0 hours
L.	Management of Software Development	3.0 hours
M.	Maintenance	3.0 hours

IV. Evaluation Methods

Grades will be determined by taking the weighted (to approximate the distribution of points below) point total and identifying where 90%, 80%, 70%, 60% of the total points lies.

Points and percentages are allocated as follows:

2 exams (including final)	30%	300 points (150 points each)
Papers	30%	300 points
Projects	30%	300 points
Homework	10%	100 points
TOTAL	100%	400 points

V. Suggested Textbook

Schach, Stephen, Classical Object-Oriented Software Engineering with UML and C++, Fourth Edition, McGraw Hill, 1999.

VI. Bibliography

1. Braude, Eric J., Software Engineering, An Object-Oriented Perspective, Wiley, 2001.
2. Fowler, Martin and Scott, Kendall, UML Distilled, 2nd Edition, Addison-Wesley, 2000.
3. Pressman, Roger S., Software Engineering, A Practitioner's Approach, 5th Edition, McGraw Hill, 2001.
4. Sommerville, Ian, Software Engineering, 6th Edition, Addison-Wesley, 2001.

Old Syllabus of Records

CATALOG DESCRIPTION Appendix B

COSC 319 3c-0l-3sh

Prerequisites: COSC 315 or permission of the instructor

Software engineering concepts include the collection of tools, procedures, methodologies and accumulated knowledge about the development and maintenance of software based systems. This course is strongly suggested for any student planning to take an internship in Computer Science. After an overview of the phases of the software lifecycle, current methodologies, tools and techniques being applied to each phase will be discussed in depth with localized exercises given to reinforce learning of concepts.

COURSE GOALS AND OBJECTIVES

This course will serve to broaden the student's understanding of the issues and latest developments in the area of software development and maintenance. To reach this goal, the following objectives need to be met:

1. Define the current state of software development and maintenance characterized as "the software crisis".

2. Understand the multidimensional aspect of software engineering which is the current best attempt at solving the software crisis.

3. Become familiar with popular models of the software development and maintenance process.

4. Using the waterfall model, study the inputs, outputs, and processes present in each phase.

5. Study the core concepts present in several popular methodologies and be able to identify strengths and weaknesses of each.

6. Study existing CASE tools to be able to identify opportunities to automate tasks through the use of such tools.

7. Consider the issues and techniques present in confidence gaining measures residing in each phase of the software lifecycle.

8. Briefly investigate problems present in project management.

COURSE OUTLINE Fall, 1989

This course will serve to broaden the student's understanding of the issues and latest developments in the critical area of software design and development. The course will be conducted as a seminar, collections of papers will be read and actively discussed in class. The teacher will lead most of the discussions, but the students will be expected to participate in the discussions. Questions have been formulated which will serve as the basis of the discussion. Before coming to class, students should read (perhaps reread ...) and summarize the assigned article(s). This summary should be a page of notes that capture the important points of the article. Students should also attempt to answer the discussion questions. The answers do not have to be written, but a valid attempt should be made to structure an answer to each question.

A student's grade will be determined by his performance in the discussions, projects, quizzes and exams.

The following subjects will be addressed:

Topic	Class Hours	Subject
1	.5	Course Introduction and Administration
2	3.0	The Software Crisis and Software Engineering
3	1.5	The Software Life Cycle - A Model of Software Development
4	1.5	Requirements Analysis
5	3.0	Design Issues
6	6.0	Design Methodologies
7	3.0	Implementation Techniques
8	3.0	Development Tools
9	6.0	Software Quality
10	6.0	Generic Code and Automatic Code Generation

11	3.0	Programming Environments
12	3.0	Management of Software Development
13	3.0	Maintenance

What follows is an overview of the topics, including a reading list for each topic.

Topic: Readings:

1 Course Introduction and Administration

a. Syllabus and Course Introduction

2 Software Crisis and Software Engineering

a. Zelkowitz, M.V., "Perspectives on Software Engineering", Computing Surveys, Vol. 10, No. 2, June, 1978, pp. 197-216.
b. Brooks, F. P., "No Silver Bullet: Essence and Accidents of Software Engineering", COMPUTER, Vol. 19, No. 4, pp. 10-19.
c. Goldberg, R., "Software Engineering: An Emerging Discipline", IBM Systems Journal, Vol. 25, No. 314, 1986, pp. 334-353.
d. Brooks, Frederick P., The Mythical Man-Month, Addison-Wesley Publishing Company, Reading, Mass. (1980), Chapters 1, 2 & 3.

Discussion Questions for Topic 2 (Software Engineering)

1. What is software? List the five major problems with software in order of importance. What does the future of the software problem look like?
2. Define Software Engineering. What is it composed of? What is it about? Why is it less successful than other Engineering Disciplines?
3. From your experience, have you encountered the "software crisis"? What form of software engineering have you seen practiced? (Exclude course experience at IUP.)
4. How does a Software Engineer differ from: a programmer, an analyst, a coder, a