LSC Use Only	UWUCC Use Only
Number ________	Number 91-15
Action ________	Action ________
Date ________	Date ________

CURRICULUM PROPOSAL COVER SHEET
University-Wide Undergraduate Curriculum Committee

I. TITLE/AUTHOR OF CHANGE
COURSE/PROGRAM TITLE CO 319W Software Engineering Concepts
DEPARTMENT Computer Science
CONTACT PERSON Dr. Charles J. Shubra

II. THIS COURSE IS BEING PROPOSED FOR:
X Course Approval Only (Writing Intensive)
____ Course Approval and Liberal Studies Approval
____ Liberal Studies and Approval only (course previously has been approved by the University Senate)

III. APPROVALS

John A Cross	Gary L Buterbaugh
Department Curriculum Committee	Department Chairperson
[illegible]	[illegible]
College Curriculum Committee	College Dean*
Director of Liberal Studies (where applicable)	Provost (where applicable)

*College Dean must consult with Provost before approving curriculum changes. Approval by College Dean indicates that the proposed change is consistent with long range planning documents, that all requests for resources made as part of the proposal can be met, and that the proposal has the support of the university administration.

IV. TIMETABLE

Date Submitted to LSC ________ to UWUCC ________	Semester/Year to be implemented ________	Date to be published in Catalog ________

Revised 5/88

[Attach remaining parts of proposal to this form.]

WRITING ACROSS THE CURRICULUM
REQUEST FOR APPROVAL TO USE W-DESIGNATION

LSC #________
Action ________

TYPE I. PROFESSOR COMMITMENT

( ) Professor________ Phone________
( ) Writing Workshop? (If not at IUP, where? when?________
( ) Proposal for one W-course (see instructions below)
( ) Agree to forward syllabus for subsequently offered W-courses?

TYPE II. DEPARTMENTAL COURSES

(X) Department Contact Person Charles Shubra Phone X7517
(X) Course Number/Title CO319W Software Engineering Concepts
(X) Statement concerning departmental responsibility. see pg 16
(X) Proposal for this W-course (see instructions below)

TYPE III. SPECIFIC COURSE AND SPECIFIC PROFESSOR(S)

( ) Professor(s)________ Phone________
( ) Course Number/Title ________
( ) Proposal for this W-course (see instructions below)

SIGNATURES:

Professor(s) [illegible]

Department Chairperson ________

College Dean ________

Director of Liberal Studies ________

COMPONENTS OF A "WRITING SUMMARY"

✓(I) pg 16 "Writing Summary" — one or two pages explaining how writing is used in the course. <u>First</u>, explain distinctive characteristics of the content or students which would help the Liberal Studies Committee understand the summary. <u>Second</u>, list and explain the types of writing activities; be especially careful to explain (1) what each writing activity is expected to accomplish as well as the (2) amount of writing, (3) frequency and number of assignments, and (4) whether there are opportunities for revision. If the activity is to be graded, indicate (5) evaluation standards and (6) percentage contribution to the student's final grade.

✓(II) A copy of the course syllabus.

✓(III) Samples of assignment sheets, instructions, or criteria concerning writing that are given to students.

Provide 12 copies to the Liberal Studies Committee.

PART II. 1

CATALOG DESCRIPTION

CO 319W - SOFTWARE ENGINEERING CONCEPTS 3C-0L-3SH

Prerequisites: CO 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.

PART II. 2 SUMMARY OF PROPOSED REVISIONS

In the past, the Computer Science Curriculum included a single course titled CO 320 Software Engineering which was taught by Dr. Howard Tompkins (retired) and by Dr. Shubra (still working). This course attempted to accomplish two goals:

1. Introduce and analyze the concepts and concepts encompassing the evolving discipline of software engineering.

2. Apply a selection of those concepts to the development of a large (realistic if not real) software based system.

As the body of knowledge encompassed in the discipline of software engineering has expanded and become more complex, Dr. Tompkins and Dr. Shubra agreed that a single course did not do justice to either goal. This conclusion has lead to a decision to replace the single existing course with two separate courses. You are now considering a course revision proposal for CO 319W Software Engineering Concepts. CO 319W contains the introduction and analysis of software engineering concepts present in the original CO 320 Software Engineering Course. The CO 319W course assignments will be of shorter duration with the goal of gaining a familiarity with specific isolated concepts. Students in CO 319W will read research material and be asked to do comparative analyses of the many tools, methodologies and approaches.

With the foundation provided in the proposed CO 319W course, a student in the proposed CO 320 course will be given an installation standards manual (a common industry practice) specifically identifying the techniques, practices, document formats, team organization and tools to be employed in completing a real (or at least realistic) software development task.

While the proposed CO 319W course will be taught in a traditional lecture format, the proposed CO 320 course will use a non-standard approach. The students will meet once a week for a lecture. The lectures will be used to impart details of the installation standards, use of local facilities and interesting lessons learned in the development of the projects. Recitation time (one two-hour session per week) will be used for walk-throughs (formal review meetings) and supervised project team work sessions. Since programming teams will have three to five students each; and, since two programming teams will be scheduled per recitation session, there will need to be one recitation section scheduled for each six to ten students enrolled in the course. The number of students per lecture section is not as critical because of the lack of demand for individual faculty attention. The lecture size should not exceed thirty students.

Since supervision of programmer teams, if done properly, consumes a great deal of a faculty member's energy; and, since the practical experience of developing software has tangible benefits for our faculty, I would suggest that every faculty member be assigned to at least one recitation session at least once every two years. The faculty member in charge of the lecture section will serve as a course coordinator and need not rotate except on an infrequent basis.

The original CO 320 course which attempted both an introduction to the concepts and practice of those concepts on large realistic team projects was scheduled as a traditional 3 credit hour course. All students attended all sessions. This did not permit the faculty member to provide the individualized attention to each project team. Further, students' understanding of the concepts they were to use in accomplishing the project were not developed to the degree needed. In fact, students were often required to apply techniques that had not yet been presented.

In summary, because of the growth of the amount of material present in software engineering and because of the less than optimal results of attempting to present complex material while using the material on a realistic project, a single lecture oriented course CO 320 is to be replaced with two courses: CO 319W which presents the material in a lecture format and CO 320 which uses a selection of the CO 319W material to solve large real(istic) projects in a recitation-lecture format.

A final comment, because the proposed CO 320 is meant to provide an experience with real projects which is a goal shared with the CO 493 Internship program, students will be allowed to count only the proposed CO 320 or CO 493 credit toward the major requirements (not both courses). Then the question arises as to the overlap, i.e., why have both courses, CO 320 and CO 493. The reason is that CO 493 is a competitive course taken by the top 1/3 to 1/2 of Computer Science majors. CO 320 (proposed) provides practical experience for those not eligible for CO 493. Further, CO 320 (proposed) is closely supervised by faculty, thus providing us an opportunity to work intensely with the students most needing the attention.

PART II. 3A. OLD COURSE SYLLABUS

The original CO 320 course as taught by Dr. Howard Tompkins varied with the project being worked on in the course. The students learned the lessons that were presented by the problems and challenges of each individual project. Because of this and the retirement of Dr. Tompkins, no original CO 320 syllabus has been located.

The syllabus for the proposed CO 319W course follows.

PART II. 3B NEW COURSE SYLLABUS

PART II. 3B. I. CATALOG DESCRIPTION

See part II. 1

PART II. 3B. II. 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.

PART II. 3B. III DETAILED COURSE OUTLINE

Indiana University of Pennsylvania
Computer Science Department
CO 319W - Software Engineering Concepts
Fall, 1989 - Dr. Shubra

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 computer scientist, or do they all do the same tasks, have the same education, shoulder the same responsibilities.
5. Formulate a discussion question from the readings. Answer the question and hand it in.
6. What directions are predicted in the articles for software development? Where do the authors agree/disagree? Which do you think will come to pass? How will the training of a programmer have to change.
7. What non-traditional program development techniques or alternatives to programs are being developed?

3 The Software Life Cycle - A Model of Software Development

a. Zelkowitz, M.V., "Perspectives on Software Engineering", Computing Surveys, Vol. 10, No. 2, June, 1978, pp. 197-216.

b. Boehm, B.W., "A Spiral Model of Software Development and Enhancement", COMPUTER, Vol. 21, No. 4, May, 1988, pp. 61-72.

Discussion Questions for Topic 3 (Life Cycle):

1. Define the stages or steps in the software lifeycle. Is it a good model of the life of software? Identify flaws that could exist with this model. What percent of the total cost of the system is allocated to each phase?
2. For each stage in the life cycle, identify the tasks to be accomplished and the outputs (products generated).
3. Some authors make the claim that software development is largely an information collection and organization activity. Do you agree or not? Why?
4. There are two broad categories of activities that fall under the step called maintenance. What are they? How do they differ in terms of what has to be done and who should pay?
5. Two terms (requirements/specifications) appear to get confused in the literature. Are the two terms different or not? Provide a definition for each and note your sources.

4 Requirements and Specifications

a. Yeh, R.T. and Zave, P., "Specifying Software Requirements", Proceedings of the IEEE, Vol. 68, No. 9, Sept. 1980, pp. 1077-1085.
b. Brooks, Chapters 6, 7, 8, 9, 10.
c. Bell, T.E., Thayer, T.A., "Software Requirements: Are they Really a Problem, Proc of 2nd Conference on Software Engineering., pp. 61-68.
d. Morton, R. and Freburger, K., "Toward Methodology for Functional Specifications", COMPSAC 80, IEEE Computer Science Press, Oct. 1980, pp. 201-206.

Discussion Questions for Topic 4 (Requirements Analysis)

1. What is the role of the requirements specification document (RSD)? Its attributes? Major problems and obstacles in writing the RSD?
2. What is meant by a black box view vs. a white box view? How do you think functional specifications differ from operational specifications?
3. What is the impact of the choice of language for use in the RSD? What are the available choices?
4. What is synergism? What is a system? What is the total systems approach?

5. Characterize the major activities of analysis. How does analysis differ from synthesis?
6. What is prototyping and how does it apply to requirements?
7. Prepare a 2-5 page typed critique of a given requirements document. You should address the sections present by comparing them with those suggested in one of the articles. Next critique what does appear in the document by first counting the number of errors found (use Bell and Thayer figure 2 for error categories) by category and, for each category, give two examples of errors in the document.

5 Design Issues

a. Peters, L.T., "Software Representation and Composition Techniques", Proceedings of the IEEE, Vol. 68, No. 9, Sept., 1980, pp. 1085-1093.
b. Jones Capers, "A Survey of Programming Design and Specification Techniques", Proceedings of Specifications of Reliable Software, IEEE NR79CH140190C.
c. McClure, C. L., "Top-Down, Bottom-Up, and Structured Programming", IEEE Transactions on Software Engineering, Vol. SE-1, No. 4, Dec. 1975, pp. 397-403.
d. Brooks, Chapters 4, 5, 11.

Discussion Questions for Topic 5 (Design Issues)

1. Critics of Top Down Design claim that it is impossible to do unless you already know how to solve the problem completely. Do you agree or not and why?
2. Describe Top Down vs. Bottom Up Development. What are the characteristics of each. Describe an environment where each has a clear advantage over the other.
3. What work units exist in the design phase and what percent of the total design time is spent in each task.
4. What is meant by functional analysis vs. data analysis? Produce an example of each.
5. What are the major types of design errors?

6. What are the issues related to the portrayal of software design? What software design techniques have you used? Choose one of the techniques you have used and describe how it addresses the issues

of portrayal of software design.

7. What is prototyping and how does it apply to design?

6 Design Methodologies

a. Ross, D. T., Schoman, K. E., "Structured Analysis for Requirements Definitions", IEEE Trans on Soft. Eng., Vol. SE-3, No. 1, Jan. 77, pp. 41-48.

b. Teichroew, D.L, Hershey, E.A., "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems", IEEE Trans. on Software Engineering, Vol. SE-3, No. 1, Jan. 77, pp. 41-48.

c. Stay, J.F., "Hipo and Interactive Program Design", IBM SYS J, 1976.

d. Stevens, W.P., Myers, G.J. Constantine, L.L., "Structured Design", Vol. 13, No. 2, 1974, pp. 115-139.

e. Jackson, M., "The Jackson Design Methodology", Infotech State of the Art Report, Structured Programming, pp. 219-234. At library reserve desk only.

Discussion Questions for Topic 6 (Design Methodologies):

1. Define a methodology. How well do each of the above methodologies - Structured Design, Jackson-Warnier, SADT, PSL/PSA - meet the definition?

2. Methodologies can be studied, understood and classified using the following characteristics: applicable program domain, form of the specifications required as input, the range of the software life cycle addressed by the methodology, the concept of a system which forms the basis of the methodology, thought disciplining aspects (aids to question identification and decision making), developmental notation, confidence gaining measures (testing, reviews, proofs), degree of automation of the methodology, limitation on application of the methodology. As you study the assigned methodologies, maintain an individual page of each methodology which includes the above characteristics and describes the methodology using these characteristics. (To be collected and graded at the end of class discussion on this topic.)

3. Several problem statements are attached. Working in groups of two, select a problem and apply both

of Jackson's Structured Design methodologies to the problem. Take your development to the pseudocode state, i.e. program the resulting solution without getting caught in all the messy syntax. I want to see the methodology (and needed documentation) down to the implementation phase. Record your problems with applying the methodology and suggestions for improving the methodology. Be prepared to present your results to the class.

7 Implementation Techniques and Issues

a. Wirth, N., "On the Composition of Well-Structured Programs", Computing Surveys, Vol. 16, No. 4, Dec. 74, pp. 247-259.
b. Baker, F.T., Structured Programming in a Production Programming Environment", IEEE Transactions of Software Engineering, Vol. Se-1, No. 2, June, 1975, pp. 241-252.
c. Connell, C., "The Art of Programming", DEC Professional, Vol. 6, No. 12, Dec. 1987, pp. 32-36.
d. Fagan, M. E., "Advances in Software Inspections" Transactions on Software Engineering, Vol. SE-12, No. 7, July, 1986, pp. 744.751.

8 Development Tools:

a. Gibson, M. L., "A Guide to Selecting CASE Tools", Datamation, Vol. 34, No. 13, July, 1988, pp. 65-66.
b. Hawley, S. A., "CASE for Sale", DEC Professional, Vol. 6, No. 12, DEc. 1987, pp. 52-54.
c. Voelcker, J., "Automating Software: Proceed with Caution", IEEE Spectrum, Vol. 25, No. 7, July, 1988, pp. 35-37.

Discussion Questions for Topic 8 (Development Tools):

1. From your experience with software development, create a list of the tasks or jobs that you perform. For each of these, identify the inputs and outputs. For each task, estimate the percentage of time spent. The tasks should be much more detailed than the phases of the software life cycle. You can restrict yourself to the design, implementation, and testing phases.
2. List any and all tools (software) that you have used anywhere in the software life cycle. For each tool, briefly describe its function and comment on its most useful features. If you could

improve it, what features would you add.

3. Compare and contrast a Dictionary/Directory with the Cobol copy feature.

9 Software Quality

a. Miller, E.F., "Program Testing", <u>COMPUTER</u>, April, 1978, pp. 10-12.
b. Fairley, R. E., "Tutorial: Static Analysis and Dynamic Testing of Computer Software", <u>COMPUTER</u>, April, 1978, pp. 14-23.
c. DeMillo, R.A., Lipton, R.J. and Sayward, F.G., "Hints on Test Data Selection: Help for the Practicing Programmer", <u>COMPUTER</u>, April, '978, pp. 34-41.
d. Darringer, J.A., "Applications of Symbolic Execution to Program Testing", <u>COMPUTER</u>, April 1978, pp. 51-60.
e. Brooks, Chapter 13.

Discussion Questions for Topic 9:

1. Given a program and documentation, construct a test plan, complete with data files and expected results.
2. Given a flow chart, identify the number of paths through the program.

10 Generic Code and Automatic Code Generation

a. Logrippo, L., Skuce, P.H., "File Structures, Program Structures, and Attributed Grammars", <u>IEEE Transactions on Software Engineering</u>, Vol. Se-9, No. 3, May, 1983, pp. 260-267.

Discussion Questions for Topic 10 (Generic Code and Automatic Code Generation):

1. Handout on generic code generation.

11 Programming Environments

a. Dant, S. A., Ellison, R. J., Feiler, P. H., Habermann, A. N., "Software Development Environments", <u>COMPUTER</u>, Vol. 20, No. 11, Nov. 1987, pp. 18-28.
b. Miller, D. B., "EXCELERATOR in the Fast Lane", <u>DEC Professional</u>, Vol. 7, No. 7, July, 1988, pp. 72-80.
c. Buxton, J. N. and Druffel, L.E., "Requirements for an ADA Programming Support Environment: Rationale for Stoneman", <u>Proc COMPSAC 80</u>, Oct. 1980, pp. 66-

72.

d. Wasserman, A.I., "Automated Development Environments", COMPUTER, Vol. 14, No. 4, April 1981, pp. 7-10.

e. Kernighan, B.W., Mashey, J.R., "The Unix Programming Environment", COMPUTER, Vol. 14, No. 4, April, 1981, pp. 12-24.

Discussion Questions for Topic 11:

1. How does a programming environment differ from a collection of tools?
2. Construct a detailed description of the programming environment at IUP.

12 Management of Software Development

a. Phan, D., Vogel, D., Nunamaker, J., "The Search for Perfect Project Management", Computer World, Sept. 26, 1988, pp. 95-100.

13 Maintenance

a. Schneidewind, N. F., "The State of Software Maintenance", IEEE Transaction on Software Engineering, Vol. SE-13, No. 3, March, 1987, pp. 303-310.

PART II.3B. IV. EVALUATION METHODS

Your grade 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.

Totals close to, but below, these points are then evaluated and perhaps included in the higher bracket.

Points are allocated as follows:

2 exams	200 points (100 points each)
1 final	100 points
Document critique	50 points
Homework	50 points
	400 points

PART II.3B. V. REQUIRED TEXTBOOK(S) SUPPLEMENTAL BOOKS AND READINGS

See PART II.3B. III. Detailed Course Outline

PART II.3B. VI. SPECIAL RESOURCE REQUIREMENTS

Based on current course demand, a single IBM (compatible) 386 with 4 MEG RAM, an 80 MEG hard disk, color monitor and high quality laser printer needed to run the Excelerator software package is sufficient. Such a machine is available. Should the load increase, a second system would be needed.

Because of the writing of outlines and the critique, the students need access to a computer-based word processor, such as those provided by the ISCC.

The students need access to the VAX set software which is currently available.

See PART II.3B. VII. Bibliography

See PART II. 3B. III. Detailed Course Outline.

Part II.4 JUSTIFICATION/RATIONALE FOR THE REVISION

Because of the ongoing advances in the field of software engineering, the complexity and the amount of material to be mastered has increased greatly. Further, because of the need to apply the techniques to a project of sufficient size to demonstrate the utility of the software engineering techniques working in unison, it became impossible to teach techniques and apply them at the same time.

Where once a single 3-credit hour course in software engineering was the norm, now the two course sequence is recognized by educators as the preferred method of presentation.

STATEMENT CONCERNING DEPARTMENTAL RESPONSIBILITY

The departmental chairperson will review the course syllabus and the writing assignments to ensure they fulfill the requirements. Further, the department chairperson will only assign faculty to teach the course who have completed training in the teaching of writing intensive courses.

I. WRITING SUMMARY FOR CO 319W

Software Engineering is a new field of study with most meaningful advances occurring in the last fifteen years and many more to come in the future. Because it is an evolving discipline, up-to-date textbooks are difficult to locate and much of the reading is present only in technical publications. No single theory or solution has proven to be the best in all situations; therefore, the students are exposed to several possible techniques for dealing with software development problems. The students need to undertake critical analysis of the various techniques to be able to identify strengths and weaknesses of each.

Software Engineering is founded on a recognizable pattern of phases that every software development project progresses through. These phases are called the "Software Life Cycle" and the major activity in each of the phases is technical writing. In fact, a written document is an artifact of each of the phases. A technical review called a "Walkthru" of the written artifact is used between phases as a quality assurance mechanism.

TYPES OF ACTIVITIES

A. Written answers to discussion questions.

1. These questions lead the students to critical aspects of the topic being presented. They help the student to understand the concepts, to identify areas needing additional advances and to critically analyze proposed problem solutions. This also helps assure that students have read and thought about the key concepts presented in a topic.

2. Question responses will be approximately 100 - 200 words each.

3. Students will be assigned 2-3 of these questions per topic.

4. The written answers will be collected and submission recorded on a check-off basis. The answers will be discussed in a topic review with students offering their answers.

B. Outlines of Technical Articles

1. As articles from the technical literature are assigned as readings, the students will be required to submit a one to two page outline of the key concepts presented in the article. These outlines assure that the students have read the articles and considered their content. Students are encouraged to consider topic discussion questions, and the view of each author on those questions as they read the articles.

2. One to two pages per outline is the volume of writing.

3. The frequency and number of assignments depends on the availability of a course text. Currently, the course is taught from a collection of technical readings. Therefore, the frequency of outline assignments is one outline for each hour of lecture. Currently, 30 to 35 outlines are required. If a suitable textbook is identified (and several possibilities are in production) technical articles will be used only to supplement the chapters. Thus, only one outline per chapter may be needed with a frequency of one per week vs. one per lecture.

4. The outlines are collected and given back to the students during each examination. The examinations thus take the form of an open notes examination. Student are encouraged to bring a copy of their outlines to topic discussions so that they might contribute to discussions and also note errors or omissions in their outlines. Only the original outlines are used for examinations. The examinations consist of mostly short essay questions whose answers are based on outlines and class discussion of the articles. Examination questions are graded. The original outlines are not graded. To assure that students are making the optimum use of the outlining technique to capture critical concepts in the technical articles, the first student outline will be reviewed by the instructor with suggestions for revision made on the document. Students will then resubmit a revised outline for this initial outline.

   The outline for the second technical article will be collected after the lecture which covers that article. Students will thus be allowed to record omissions on the second article outline before it is submitted to the professor. The professor's lecture thus serves as a review of the student's outline with the modified outline serving as the revision.

C. PREPARATION OF A REQUIREMENTS DOCUMENT FROM THE VERBAL STATEMENT OF A PROBLEM

1. Practicing software engineers are frequently required to take the verbal description of a problem and produce a formal document (Requirements Document) which provides a detailed description of the

problem requirements. As an introduction to this process and its many pitfalls, students will be required to listen to a problem description and produce a written requirement.

2. The document will consist of one to two pages.

3. This assignment will be given once during the semester.

4. The initial student documents will be collected and reviewed. Students will then resubmit their requirements document based on the teacher's comments. The revised requirements document will then be graded.

D. CRITIQUE OF A SOFTWARE LIFE CYCLE DOCUMENT:

1. A "walk-thru" is a procedure used to assure the quality of the artifact produced during each phase of the software lifecycle. To prepare to participate in a "walk-thru", reviewers are given materials (most often technical documents dealing with the project) to critically analyze. My students are given a completed Requirements Document which describes the problem to be solved and a Specifications Document which contains the proposed solution to the problem. They are asked to prepare a written critique of one of the documents, identifying specific problems.

2. This critique consists of five to ten pages of writing.

3. There is only one of these assignments given approximately one-third of the way through the semester.

4. Students are not given an opportunity for revision in this assignment.

5. The assignment is graded on the basis of 50 points (1/2 of an examination). Evaluation consists of the identification and explanation of the number and types of errors located, as well as the organization of their critique. This constitutes 13% of their grade.

II. A copy of the Course syllabus is included in Part II.

III. Samples of Assignments

A. The discussion questions appear after each topic in the course syllabus.

B. Outlines for technical articles.

IUP Computer Science Department Dr. Shubra
CO 319W Fall, 1990

TECHNICAL ARTICLE OUTLINES

You are to prepare a one to two page typed outline for each technical reading assigned in class. You should attempt to identify the key components of each article, i.e., what points are the authors attempting to make. Further, you should read the discussion questions at the end of each topic in the outline and consider the authors's view on these questions.

I will collect your outline prior to class discussion of the topic. These outlines will then be given back to you to use during your examinations, so prepare them carefully. Further, you should bring a copy of each outline to class so that you may use them to contribute to class discussions and may take notes on points raised during discussions.

The first outline will be collected and reviewed by your instructor. You will then be given a chance to modify this first outline based on the instructor's comments.

The second article outline will not be collected until after the instructor has presented his lecture on that article. During the lecture, you will permitted to revise your outlines by taking notes on your outline. This revised outline will be collected at the end of the lecture.

All other outlines will be collected prior to the beginning of the lecture.

IUP Computer Science Department
CO 319W

Dr. Shubra
Fall, 1990

SPECIFICATIONS CRITIQUE
(50 POINTS)

You are to prepare a written critique of the Software Specification document on reserve at the library. A copy of the document, less the appendices can be accessed on the IUP mainframe using fid SYSTEM-SPECIFICATIONS-DOC.11512. Your application knowledge is contained in the Software Requirements document also on reserve at the library. A copy of the Software Requirement document is available on the Honeywell mainframe using fid SYSTEM-REQUIREMENTS-DOC.11512.

You should use the Bell and Thayer article (Bell, T.E. and Thayer, T.A., "Software Requirements: Are They Really a problem", Proceedings of 2nd Conference on Software Engineering, pp. 61-68) as an indication of the types of errors/omissions to search for.

Your critique is to be a typed, double spaced document (5 - 10 pages) with the following organization:

1. Identification block consisting of the following in the upper right hand corner:

   Student Name
   CO 319W
   Specification Critique
   Date due

2. A brief introduction to the critique.

3. An enumerated list of critique items. Each of these items must contain the page and line number (from the Honeywell file) where the error is located followed by a quoted portion of the document that contains the error. This is followed by a one to two sentence description of what you view the error to be. This enumerated list is to be ordered in ascending sequence by page number, line number.

Evaluation

You will be graded on

1) the completeness of your critique,
2) the explanation of the errors found,
3) the organization, spelling and syntax used.

PART III. LETTERS OF SUPPORT

A copy of the revision of CO 319W and the new course proposal for CO 320 have been sent to Mr. Kenneth Shildt, Chairperson of Finance and MIS Department for their comments.

IUP Computer Science Department

April 11, 1991

SUBJECT: Curriculum Proposals for CO 319W and CO 320

TO: University Wide Curriculum Committee
University Wide Liberal Studies Committee
Curriculum Committee of College of Natural Sciences and Mathematics
Computer Science Department Curriculum Committee
Computer Science Faculty

FROM: Dr. Charles J. Shubra

The curriculum of the Computer Science Department officially includes CO 320 Software Engineering as an approved course. I wish to split this single course into a two course sequence consisting of CO 319W Software Engineering Concepts and CO 320 Software Engineering Practice. The rationale for the change is contained in the proposals. Since the lecture content of the proposed CO 319W is essentially an expansion of the lecture content of the original CO 320 course, it was decided in a phone call to Dr. Juliette to submit CO 319W Software Engineering Concepts as a revision of the original CO 320 Software Engineering course and treat the proposed CO 320 Software Engineering Practice as a new course proposal.

cam