UNIVERSITY OF NEBRASKA AT OMAHA
COURSE SYLLABUS/DESCRIPTION

Department and Course Number	CSCI 4980
Course Title	Topics in Computer Science
Course Coordinator	Stanley Wileman
Total Credits	1–3 (variable credit)
Date of Last Revision	June 13, 2003

1.0 COURSE DESCRIPTION

1.1 Overview of content and purpose of the course
This course is used to provide a mechanism for offering instruction in subject areas that are not covered in other regularly-scheduled courses. Traditionally, it is used to focus on areas of faculty research interest, or on evolving subject areas in computer science.

1.2 For whom course is intended
This course is intended primarily for juniors and seniors in computer science.

1.3 Prerequisites of the course (courses)
Junior or senior standing is required. It is also usually the case that permission of the faculty teaching the course is required. Topics offered under the CSCI 4980 designation must be approved by the undergraduate curriculum committee. A sample course proposal is included as an appendix to this syllabus.

1.4 Prerequisites of the course (topics)
The specific topics appropriate as prerequisites for this course depend on the particular topic to be covered. The prerequisites will be identified in the faculty proposal for the topics course.

1.5 Unusual circumstances of the course
None

2.0 OBJECTIVES

The objectives for each offering of CSCI 4980 will be different, but will be specified in the faculty proposal for the course.

3.0 CONTENT AND ORGANIZATION

The topics covered in the course and their organization depend entirely on the subject area. They will be specified in the faculty proposal for the course.

4.0 TEACHING METHODOLOGY

4.1 Methods to be used
Topics courses may be presented as traditional lecture courses, or may be presented using faculty-led discussions where each student is expected to actively participate.

4.2 Student role in the course
Student expectations are dependent on the teaching methods used. In all cases, however, identification of appropriate resources (journal articles, textbooks, web resources, and so forth) that cover the subject will be provided, and students will be expected to become familiar with the resources. If discussions are used as a teaching method, students are expected to actively participate.

5.0 EVALUATION

5.1 Types of student projects used for evaluation

The student work used for evaluation will be dependent on the faculty teaching the course. Traditional quizzes and examinations, written reports, oral presentations, written homework problems, and programming assignments may be used, as well as participation in discussions.

5.2 Basis for determining the final grade

The final grade is determined using the criteria established by the supervising faculty and documented in the the course proposal.

5.3 Grading scale

The grading scale for each independent study is established by the supervising faculty and documented in the course proposal.

6.0 RESOURCE MATERIAL

Resource materials are expected to include textbooks and research papers in scholarly journals. Material from the web and other unpublished works may be used as the basis for the study with the approval of the supervising faculty.

7.0 Estimate Computer Science Accreditation Board (CSAB) Category Content (class time in hours):

The specific CSAB categories in which the subject matter can be categorized depends on the specific topic. Since CSCI 4980 is used for junior/senior level topics, it is expected that the majority of the material will be in the advanced category. An estimate is that the equivalent of 10 hours of lecture on core topics and 35 hours of lecture on advanced topics will be presented. The specific CSAB categories covered, and the number of hours of instruction in each category, will be specified in the faculty course proposal.

8.0 Oral and Written Communication

As expected, it is not possible to indicate what oral and written communication will be required in each offering of this course. Such will, however, be specified in the faculty course proposal.

9.0 Social and Ethical Issues

These issues may be covered in some topics courses.

10.0 Theoretical content

Some topics courses may cover theoretical material.

11.0 Problem analysis

The extent to which a specific topics course includes problem analysis is dependent on the specific course offering, and will be specified in the faculty course proposal.

12.0 Solution design

The extent to which a specific topics course includes solution design is dependent on the specific course offering, and will be specified in the faculty course proposal.

CHANGE HISTORY

Date	*Change*	*By whom*	*Comments*
6/12/2003	Added ABET-specific sections and copy of sample faculty course proposal.	Wileman	

CSCI 4980 — Topics in Computer Science
Course Proposal

Subject :	Apprenticeship in Software Architecture
Coordinator:	Victor Winter
Total Credits:	3

Course Description

Purpose

The Apprenticeship in Software Architecture and Design is meant to give students real, industry-quality experience on software development projects to provide a frame of reference students can relate to when taking upper-level Computer Science courses.

Upper-level Computer Science courses are heavily based in theory, not implementation, as they should be to provide students with a technology-agnostic education. However, students require experience in the various aspects of implementation to truly appreciate and understand what the theory behind it teaches them.

Content

Students will work on real-world projects, provided either in-house by the Computer Science department or, preferably, industry.

The specific content of the course will vary depending on the project being implemented, but will generally focus on how to recognize, analyze, and implement best-practices such as design patterns, coding standards, and decoupled architectures.

Target Student Base

Any Computer Science student with a firm understanding of basic programming principles (syntax and data structures) and wants to build upon this basis in the context of a larger, realistic development environment will be able to succeed in the class.

Course Prerequisites

- CSCI 1620
- CSCI 3320

Prerequisites can be waived with the instructor’s approval.

Topic Prerequisites

Students will be expected to be proficient in imperative programming languages.

It is expected that if students know C++, they will be able to quickly pick up other imperative languages such as Java or C# without any class time being spent on syntax. Other language-specific issues, such as basic compiling and program execution, will be covered briefly in class.
Students will also be expected to have some minimal experience in relational databases. They should be able to quickly pick up the syntax of SQL and any various APIs (e.g. JDBC, ODBC) used to access data from the programming language.

Objectives

Develop instincts for solving problems cleanly and elegantly
Ability to understand and appreciate industry best practices
Work effectively in software development groups, both in-person and electronically

Content

The specific content will depend on the projects chosen for the class. Given a project, the class will focus around implementing cutting-edge solutions using the best, most effective means.

Each project will have a series of milestones, roughly following this schedule:

Milestone	Contact Hours	Weeks	Description
1. Prototype	9	3	The first iteration of the system. Mostly exploring of different architectures to get a feel for what works and what doesn't.
2. Alpha	12	4	The beginnings of the real code base. May or may not be based on a prototype, but does definitely reflect the lessons learned in prototyping.
3. Beta	15	5	A near working version of the system. All features are implemented.
4. Release	6	3	Cleaning up, documentation, install techniques, delivery of the system to the client.

Teaching Methodology

Methods

Class time will be in-formal. Instead of lectures, students will actively participate in discussions, guiding both the topics and content discussed.

The instructor will focus on issues currently facing the project teams. Help will be given to pinpoint an issue, offer solutions, and go into discussions concerning the benefits and implications of each solution.

This process would likely in involve in-class code reviews, architectural discussions, prototyping, and the like.

Student Role

Students will be expected to work in small groups of three to five working a real-world project with an expected deliverable of a working system at the end of the semester.

This may require a significant time being spent outside of class. Students will be expected to work on their own and also meet with their team at least once per week outside of class.

Contact Hours

In-class contact hours are 3 per week.
However, an extra 3-6 hours per week will be expected of students to work on their project and meet with the project Teaching Assistant to discuss various implementation details that may not be discussed in class.

Evaluation

Students will be heavily evaluated on the results on their project.

The coordinator and TAs will continually monitor projects and if it is deemed the project was just not able to be completed within a semester's timeframe; students will not be harshly graded. However, projects will be chosen with the semester timeframe in mind, so students will be expected to have them completed.

Students will also be graded on their class and group participation. The coordinator and TAs will interact with the project teams and note the participation level of each team member. If concern arises that a team member is not contributing as well as he or she should, they will be notified and urged to participate more or risk their grade being affected.

Components

Component	Grading
Project	60%
Participation	40%

Grading scale

Points	Grade
97-100%	A+
93-96%	A
90-92%	A-
87-89%	B+
83-86%	B
80-82%	B-
77-79%	C+
73-76%	C
70-72%	C-
67-69%	D+
63-66%	D
60-62%	D-
0-59%	F

Resource Material

The course has no official text books.

Students will be encouraged to reference material from both online and print sources, but what the sources actually are depends on the student's project.

The coordinator and TAs will refer students to various readily-available resources as needed.

Computing Accreditation Commission Categories

CAC Category	Core	Advanced
Hardware and software		
Networking and telecommunications		
Modern programming language		9
Analysis and design		21
Data management	3	
Role of IS in organizations		

Oral and Written Communications

Students will not be required to submit formal reports.

However, they will be graded on keeping the architectural documentation, including informal requirements and specifications, for their system up to date.

Social and Ethical Issues

Students will potentially be working on product for industry. In this case, it would be desirable for students to be paid for working on their projects more than the 5-10 hours per week already expected of them.

This would benefit the students in letting them spend more time working on the project and hence learning from their increased involvement.

Also, industry would benefit in that the projects would have considerably better implementations due to the increased time students could spend on them.

However, how this money gets to students is undecided. Potentially, students could be put directly on the company's payroll after they have proved they can provide valuable contributions, both quality-wise and quantity-wise, to the project.

Another possibility would be to have companies contract the Computer Science department for the work, then after taking some small overhead, the department would pay students whatever hourly rate had been decided with the company.

Having students work for pay is not essential to the class, but it would greatly increase the amount of time students could spend on projects and hence benefit all parties involved.

Theoretical Content

This class will not directly teach theoretical content, as it is intended as a preparation course for the advanced-level Computer Science courses that deal solely with theoretical content.

Problem Analysis

Students learn how to be productive when it comes to solving problems encountered in software development problems. This will be done by teaching them not to just solve problems, but how to recognize the best solutions that result in the most productivity saved, most stable system, most modular system, etc.

Solution Design

Students learn how to approach system design from an implementation perspective.
Although valuable, long requirements documents and the like will not be a major factor in the class, and they are already covered in Software Engineering. Instead, students will focus on designing best architectures and systems strictly from the source code point of view.

How these architectures and systems correlate with the project requirements and the like will naturally be emphasized and care will be taken to implement them. But when putting together the actual system, students will concentrate on the aspects that distinguish a real, robust, flexible system from those that students may typically work on in academic environments.

Knowing these distinguishing features is the core experience students in the class should develop to enable them to go on to more pure theoretical classes later in the Computer Science program and have a better understanding and appreciation for the material covered.