B. GENERAL INFORMATION TECHNOLOGY COURSES

GEIT 1311: Computer Organization
GEIT 1411: Computer Science I
GEIT 1412: Computer Science II
GEIT 2291: Professional Ethics
GEIT 3341: Database Design
GEIT 3351: Software Engineering I
GEIT 4351: Software Engineering II
GEIT 4361: Practical Training
ASSE 4311: Learning Assessment III

Course Title: GEIT 1311: Computer Organization

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course examines the functional components of computer systems. Topics discussed include processors, memory types and hierarchies, buses, I/O, interrupts, etc. with emphasis on how they affect program execution, parameter passing and inter-program communications between programs written in diverse languages

II. PMU Competencies and Learning Outcomes

Students in this course develop a conceptual understanding of the internal structures and mechanism in computer systems. In doing so, students gain an insight into the interaction between software and hardware that allows a more sophisticated understanding of the nature of software and the systems upon which that software executes.

The course provides opportunities for technical skill development as well as communication, collaboration and leadership skills through the maintenance of journals, detailing progress in group projects, and in-class presentations

This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects and establishes both conceptual reasoning skills and technical communication skills.

III. Detailed Course Description

GEIT 1311: Computer Organization examines the structures and components common to all commercial computer systems and the fundamental techniques used to effectively and efficiently handling processing tasks. This course reflects the current state of the field, as well as introducing the principles that are shaping the design of computing environments. Students in every specialty of computing need to appreciate the organizational paradigms that determine the capabilities, performance, and, ultimately, the success of computer systems.

Modern computer technology requires professionals of every computing specialty to understand both hardware and software. The interaction between hardware and software at a variety of levels also offers a framework for understanding the fundamentals of computing. The performance of future software systems will be dramatically affected by how well software designers understand the basic hardware techniques at work in a system. Thus, compiler writers, operating system designers, database programmers, and most other software engineers need a firm grounding in the principles presented in this course.

IV. Requirements Fulfilled

This course is required of all students pursuing a degree program within the College of Information Technology. It should be taken in the second semester of the freshman year.

V. Required Prerequisites

GEIT 1411: Computer Science I

VI. Learning Outcomes

In this course, students learn:

- To understand the internal functions of a computer system.
- To describe how a computer system connects to the outside world.
- To use a formal description language to describe machine structures.
- To use assembly languages to control a real and simulated machines.
- To describe and distinguish between CISC and RISC architectures.
- To learn to communicate the solutions of technical problems to other professionals.
- To develop improved collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to organization and structure of computer systems, to provide students with significant experience in low level (assembly language) control of a computer system, and to provide students with the opportunity to communicate their understanding to their peers in a classroom setting. With this in mind, the course grade involves an assessment of their performance on exams that focus on hardware structures, formal descriptions of systems and low level programming capabilities. Course grades are be based on

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of course-related material and classroom discussion and critique of the presentation.
- Two in-class exams to assess the student/s accumulative mastery of content covered prior to time of exam.
- Six programming assignments testing the student's understanding of the major concepts introduced during the course
- A comprehensive final exam to assess the student/s accumulative mastery of course material.

The final grade is based on 15% credit for the homework, 15% for the presentations and participation in classroom discussion, 20% on in-class exams, 30% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course Format

The course is primarily a lecture-based course in which the students are required to complete significant projects outside of class time. The course will include individual as well as group projects and provide opportunities for the presentation and defense of designed solutions. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

Classroom Hours (3 hours per week) **Class: 3**
Lab: 0

Web supplement: Course home page (the university's Web tool, WebCt or BLACKBOARD) should contain the following:

- Course syllabus.
- Course assignments.
- Keys to exams (after students have completed them).
- Model programmed solutions to programming assignments (once students have completed them)
- Course calendar (an active utility).
- Course e-mail (an active utility).
- Course discussion list (an active utility).
- Students course marks. (an active utility).

IX. Topics to be Covered

A. Computer abstractions and technology
B. Performance measurement and comparison
C. Data representations
D. Combinational logic circuits and Karnaugh maps
E. Sequential logic circuits
F. Introduction to assembly language
G. Computer arithmetic
H. Design of the datapath and control of a simple RISC processor
I. Pipelining
J. Memory system technology and architecture
K. Hierarchical memory systems and caching
L. Interfacing processors and peripheral devices
M. Parallel processors
N. Other advanced topics

X. Laboratory Exercises

There is no laboratory component in this course.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use including circuit and schematic design tools as well as assembler tools for the x86 architecture.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

David A. Patterson and John L. Hennessy, (2005). *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann,
ISBN 558604286

B. Alternative Textbooks

1. Miles J. Murdocca and Vincent P. Heuring, *Principles of Computer Architecture*, Prentice-Hall, 2000
   ISBN 0201436647
2. Richard C. Detmer, Introduction to 80x86 Assembly Language and Computer Architecture, Jones and Bartlett, 2001
   ISBN 0763717738

C. Supplemental Print Materials

As available from publisher.

D. Supplemental Online Materials

As available from publisher.

Course Title: GEIT 1411: Computer Science I

Semester Credit Hours: 4 (3,1)

I. Course Overview

Computer Science I is an introduction to programming and to the use of algorithms in designing programs. A software engineering approach to developing computer programs is stressed and object-oriented concepts are introduced. The course examines standard control structures, approaches to modularization, and the use of primitive and structured data types.

II. PMU Competencies and Learning Outcomes

Students of GEIT 1411 will develop both the conceptual basis and the practical skills in the design and implementation of programs to solve specific problems. These fundamental skills are necessary for continued success in Computer Engineering, Computer Science and information Technology. This course will make extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in handling Application Development Environments. The course will include individual as well as group projects and provide opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

GEIT 1411 is an introductory course in the design and implementation of solutions to specific problems and an introduction to computer programming languages and to the structures that such languages contain. In addition, the course provides experience in the management of application development environments. The course examines the organization of computer programs and the control structured used to manage the flow of instructions. Primitive data types are introduced along with techniques for manipulating those data types and their use in building aggregate data structures such as strings, and arrays. Techniques for improving readability and maintainability are taught including appropriate documentation techniques and program modularity, initially through functions, and later though object oriented techniques.

IV. Requirements Fulfilled

GEIT 1411 satisfies 4 hours of the requirements for degrees in Computer Engineering, Computer Science and information Technology. It is required of all students pursuing a degree program within the College of Information Technology. It should be taken in the first semester of the freshman year.

V. Required Prerequisites

None. This is first course in the Computer Engineering, Computer Science and Information Technology degree programs.

VI. Learning Outcomes

A. To develop understanding the nature and functionality of computer programming.
B. Gain experience and skill in using application development environments as profession tools in program development.
C. To understand and apply functional decomposition techniques to provide structure and readability to a computer program
D. To learn to communicate the solutions of technical problems to other professionals.
E. To develop improved collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind; to introduce students to the conceptual basis and practical issues associated with the development of computer programs, to provide students with significant experience in the development of computer programs within a profession development environment, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on exams that focus on the application of programming techniques to the solution of problems and the communication of designed solutions to those problems to an audience. Course grades are be based on

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class exams to assess students' accumulative mastery of content covered prior to time of exam.
- 8 programming assignments testing students understanding of the major concepts introduced during the course
- A comprehensive final exam to assess students' accumulative mastery of course material.

Students' final grades will be based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for weekly lab exercises, 20% on in-class exams, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal will contain daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal will be reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks will be included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he proceeds through the curriculum.

VIII. Course format

Instruction: This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

- Course syllabus.
- Course assignments.
- Keys to exams (after students have completed them).
- Model programmed solutions to programming assignments (once students have completed them)
- Course calendar (an active utility).
- Course e-mail (an active utility).
- Course discussion list (an active utility).
- Students course marks. (an active utility).

Classroom Hours (5 hours per week) **Class: 3**
Lab: 2

Web supplement: Course home page (the university's Web tool, WebCt or Blackboard) should contain the following:

IX. Topics to be Covered

A. Introduction
   1. Programming and Problem Solving
   2. Introduction to the programming language
   3. Testing and Debugging

B. Data manipulation
   1. Variables and assignment
   2. Input/Output
   3. Data Types, Operators and Expressions
   4. Flow Control

C. Procedural Abstraction
   1. Predefined functions
   2. Programmer-defined functions
   3. Local and global variables
   4. Function overloading
   5. Function parameters
   6. Call by reference vs. call by Value
   7. Function return types
   8. Pre and Post conditions

D. I/O Streams
   1. Stream Concepts
   2. File Streams
   3. Character Streams
   4. Stream formatting

E. Arrays
   1. Declaring and referencing arrays
   2. Arrays in functions
   3. Multidimensional arrays
   4. Strings

F. Abstract data types
   1. Aggregate data types
   2. Defining ADT operations
   3. Separate compilation

G. Object Oriented Analysis, Design and Programming
   1. Class Concepts
   2. Public and private methods
   3. Public and private data members
   4. Inheritance

X. Laboratory Exercises

This course requires a weekly 2-hour lab component. Topics to be covered in the laboratory sessions should include:

- Designing the first program – introducing the application development environment and the design, implementation testing and debugging of a simple program contained with a single file.
- Variables and assignments – introducing all of the commonly used primitive data types, assigning values to variables, distinguishing between constants and variables, performing computations on variables using language defined operators.

- Flow control I - exercise in the use of simple if statements for and while loops. The exercise should also include limited I/O and the simplest formatting commands.
- Functions I – introducing functions using call by value and returning a primitive data type or void. This laboratory should introduce the concept of compilation from multiple files, at least one for the function definitions and one for the main program.
- I/O I – introducing basic stream I/O and formatting commands to improve the quality of output.
- Flow Control II - introducing do/while, case and nested loops.
- Functions II – introducing call by reference functions and functions.
- Arrays I – introducing arrays of primitive data types and their processing including using loops to populate the array, to locate an element within an array and to produce summary statistics from an array.
- Arrays II – introducing the populating and process of multidimensional arrays using nested loops. This laboratory should also include passing one-dimensional and multidimensional arrays to functions.
- I/O II – introducing file and character streams and the storage of data in sequential and random access files.
- String manipulation – introducing the String as an array, as a class and predefined string functionality.
- Object Oriented Programming I - introducing the concept of a class, its member data, member functions and the instantiation and manipulation of objects of that class in an application.
- Object Oriented Programming II – introducing operator method overloading, operator overloading and inheritance.
- Two laboratory sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories across more than one session .

XI. Technology component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component which would be best implemented in university provided laboratory space.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student will select one problem, which the student thinks best reflects the way the mathematical topic will be used in a technical context. A detailed solution to the problem will be included in the student's reflective notebook.

XIII. Textbooks and Teaching aids

A. Required Textbook

Savitch, W. (1999). *Problem Solving with C++: The Object of Programming*.
Addison-Wesley
ISBN 0-201-35749-6

B. Alternative Textbooks

Dale, N., Weems, C. & Headington, M (2000). *Programming and Problem Solving with C++*.
Jones & Bartlett, Sudbury Massachusetts
ISBN 0-7637-1063-6

Hughes, D. (2004) *Fundamentals of Computer Science using Java*
Jones & Bartlett, Sudbury Massachusetts
ISBN 0-7637-1761-4

Deitel H.M. & Deitel, P.J. (2003). *Java: How to Program 5th Ed.*
Prentice Hall, Upper Saddle River New Jersey
ISBN: 0-13-101621-0

C. Supplemental Print Materials

As available from publisher

D. Supplemental Online Materials

As available from publisher

Course Title: GEIT 1412: Computer Science II

Semester Credit Hours: 4 (3,1)

I. Course Overview

This course is a continuation and extension to GEIT 1411 Computer Science I. It introduces the student to a systematic study of basic data structures such as queues, stacks and binary trees including searching and sorting algorithms and their associated computational costs. A software engineering approach to developing computer programs is stressed and object-oriented concepts are emphasized. Reusability of code, effective software development methodologies and good programming practices are significant components of the course.

II. PMU Competencies and Learning Outcomes

Students of GEIT 1412 extend and develop skills introduced in GEIT 1411, Computer Science I. These skills are necessary for professional success in computer engineering, computer science and information technology. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. The course includes individual as well as group projects and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

As a continuation of GEIT 1411: Computer Science I, GEIT 1412: Computer Science II presents students with the concepts, implementation and characteristics of basic, well-known data structures such as linked lists, queues stacks and trees, and their use in solving programming problems. An object-oriented approach to development is stressed to encourage code reuse, encapsulation and to illustrate concepts such as inheritance, polymorphism and template classes. Students are exposed to a variety of techniques for the implementation of each data structure including static and dynamic data structures and recursive solutions. One important lasting effect of this course is to enhance and develop the ability to specify, design, implement and test solutions to programming problems utilizing the data structures and proven algorithms presented in this course.

IV. Requirements Fulfilled

This course is required for all students regardless of major in Information Technology in the College of Information Technology. It should be taken in the second semester of the freshman year.

V. Required Prerequisites

GEIT 1411: Computer Science I

Student also are expected to have successfully completed the first mathematics course in their degree program.

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of, and ability to utilize recursion, linked lists, stacks, queues and trees.
- To be able to discuss the advantages and disadvantages of different implementations of each of the data structures.
- To understand and apply techniques of algorithm analysis and express performance characteristics of algorithms.
- To utilize advanced object oriented concepts in the systematic design of solutions to programming problems.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the development of computer programs utilizing basic well known data structures, to provide students with significant experience in the development of computer programs from an object-oriented perspective within a profession development environment, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on exams that focus on the application of programming techniques to the solution of problems and the communication of designed solutions to those problems to an audience. Course grades are be based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class exams to assess the student's accumulative mastery of content covered prior to time of exam.
- Eight programming assignments testing the student's understanding of the major concepts introduced during the course
- A comprehensive final exam to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 20% for weekly lab exercises, 20% on in-class exams, 30% on programming assignments and 10% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he or she proceeds through the curriculum.

VIII. Course Format

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

Classroom Hours (5 hours per week) **Class: 3**
Lab: 2

Web supplement: Course home page (the university's Web tool, WebCT or BLACKBOARD) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

IX. Topics to be Covered

A. Object-oriented design
   1. Goals and principles
   2. Class design issues
   3. Inheritance and polymorphism
   4. Templates
   5. Exceptions
   6. Recursion

B. Stacks
   1. Designing a stack class
   2. Implementation issues
   3. Application examples
      a. Function calls
      b. Reverse Polish Notation

C. Queues
   1. Introduction to the queue ADT
   2. Array-based queue implementation

D. Templates and standard containers
   1. Reusability and genericity
   2. Function overloading
   3. Class genericity
   4. The vector container
   5. Other standard containers (dequ, stack and queue)

E. Algorithm analysis
   1. Recursion
   2. Algorithm efficiency
   3. Analysis of algorithms
   4. Proving algorithm correctness

F. Lists
   1. Introduction to the linked list ADT
   2. Array-based implementation of the linked list
   3. Pointers in C++
   4. Run-time allocation and deallocation
   5. Pointer-based implementation of the linked list
   6. Standard list template

G. Other linked structures
   1. Variations of singly linked lists
   2. Doubly linked lists
   3. Hash tables

H. Binary Trees
   1. Introduction to the binary tree ADT
   2. Binary search trees
   3. Binary trees as recursive data structures
   4. Huffman codes

I. Sorting
   1. O(N2) sorting schemes
   2. Heaps and the heapsort
   3. Quicksort
   4. Mergesort

X. Laboratory Exercises

This course requires a weekly two-hour lab component. Topics to be covered in the laboratory sessions should include:

- Displaying integers in binary – exercises in class development
- Enumerated types – exercises in extending primitive data types to an ADT and array-based implementation of a data structure,
- The stack – exercises in the implementation of a stack class and its use in programming problems
- Queues – exercises in developing a container class. A first look at the development of a template class.
- Vectors – an investigation of the Vector container class from the standard template library and its use in programming problems
- Recursion – introduction of the practical aspects of recursion and consideration of algorithm complexity and comparisons of algorithms.
- Pointers – introduction of pointers and dynamic allocation and deallocation of space.
- Linked lists – an instruction to linked lists and the construction of a pointer-based list class.
- Binary search tree – Design and implementation of a binary search tree class including a variety of traversal methods
- Sorting I – exercises in the use of sorting algorithms including selection sort, bubblesort, and insertion sort
- Sorting II – Additional sorting algorithms including heapsort, shellsort and quicksort
- Inheritance – exercise in inheritance illustrating how classes can be derived from other class, multiple inheritance and abstract classes.
- Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories across more than one session.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use. The course has a laboratory component that would be best implemented in university-provided laboratory space.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selects one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Nyhoff, L., *C++ An Introduction to Data Structures*, (1999) Prentice-Hall, New Jersey
ISBN 0-02-388725-7

B. Alternative Textbooks

Goodrich, M.T., Tamassia, R., & Mount D.M., *Data Structures and Algorithms in C++*, (2004) John Wiley & sons, New Jersey.
ISBN 00-471-20208-8

C. Supplemental Print Materials

Nyhoff, L. *C++ An Introduction to Data Structures* Lab Manual, Prentice-Hall, New Jersey
ISBN 0-13-3084691-0

D. Supplemental Online Materials

As available from publisher.

Course Title: GEIT 2291: Professional Ethics

Semester Credit Hours: 2 (2,0)

I. Course Overview

This course is designed to educate students on the impact ethical issues have on the use of information technology in the modern business world. It examines the ethical codes of the professional societies and the philosophical bases of ethical decision-making. Students acquire the foundation that helps them make appropriate decisions when faced with ethical dilemmas.

II. PMU Competencies and Learning Outcomes

This course introduces students to the importance of ethics and professional conduct in their profession. Students develop both the conceptual basis and the practical skills in solving various ethics-related problems arising in various aspects of a modern organization. Students learn to distinguish between ethical and unethical conduct as well as take corrective actions to prevent or solve unethical situations. More importantly, discussion of cases in ethics and professional conduct involves students in the critical thinking process using ethical principles and IT concepts. This course makes an extensive use of the PMU technology infrastructure to provide communication between faculty and students. A significant part of this course involves on-line discussion of cases in ethics and professional conduct.

III. Detailed Course Description

It is critical that students of information technology are introduced to the impact that ethical issues have on the use of information technology in the modern business world. Through an examination of ethical codes of various professional societies, the philosophical bases of ethical decision-making, and several contemporary case studies, students develop the foundation that prepares them to make appropriate decisions when faced with ethical dilemmas. They learn the five rules of ethical behavior, examine how personal values influence professional behavior, and follow a ten-step process for solving ethics-related business problems. In short, this course helps students recognize and think through ethical issues when they arise, correct unethical practices that may have been previously unnoticed or ignored, and communicate the need for applying ethical principles at all organizational levels.

A significant portion of this course is devoted to the discussion of cases in ethics. Two types of case studies are used .The “What would you do?” scenarios present real-to-life dilemmas that can be used as a basis for student exercises to involve student in the critical thinking process using principles presented in class. Real-world cases reinforce important ethical principles and IT concepts and illustrate how real companies have addressed ethical issues associated with IT.

IV. Requirements Fulfilled

This course is required for all students regardless of major in Information Technology in the College of Information Technology. It should be taken in the first semester of the sophomore year.

V. Required Prerequisites

This course does not have a prerequisite.

VI. Learning Outcomes

In this course, students learn:

- To understand the importance of ethics and professional conduct in the workplace.
- To learn how to recognize and prevent unethical behavior as well as promote ethical behavior in the workplace.
- To become familiar with the code of ethics and professional conduct of various professional organizations such as IEEE, ACM, and AITP.
- To learn how codes of ethics, professional organizations, certification, and licensing affect the ethical behavior of IT professional.
- To understand the key trade-offs and ethical issues associated with the safeguarding of data and information systems, software re-engineering, and software quality.
- To instill in students the notion of accountability in their profession as well as to society at large.

VII. Assessment Strategy

The student's performance in this course is assessed on the basis of:

- One mid-term and one final examination.
- Four out-of-class assignments in the form of mini-cases on professional ethics and conduct.
- Four mini cases to be discussed in an online, discussion group format.

Relative weights assigned to these items in determining student's final grade are suggested as follows:

- Each of the two examinations accounts for 30% of the final grade. Combined, the two examinations account for 60% of the grade.
- Four mini-cases submitted in written form account for 20% of the grade.
- Four On-line cases account for 20% of the grade.

The two essay-based examinations are used to assess an understanding of ethical theories and concepts covered in class. Cases based on "What would you do?" scenarios are used to assess student's understanding of ethical principles and process for solving ethics-related problems. Real-world cases are used to assess student's understanding of how real companies have addressed ethical issues associated with IT.

VIII. Course Format

This course utilizes a mix of in-class lectures and student- as well as instructor-led discussions designed to help students develop a deep understanding of the role of ethics and professional conduct in their chosen discipline. While class meetings are utilized to introduce students to the theory of ethical reasoning and process for resolving ethical conflicts, contemporary case studies and "What would you do?" scenarios are used to impart an understanding of guiding principles for ethical and professional conduct. These cases are drawn from the required textbook and supplementary printed material.

Once every two weeks, the instructor assigns one real-world case for students to analyze and respond with a written report. Randomly selected students are asked to make oral presentations of their response.

Once every two weeks and alternating with written case analyses, the instructor posts a "What would you do?" scenario for on-line discussion. An online discussion group is set up to discuss these scenarios outside of the classroom. Students are required to actively participate in this online forum. In addition to contributing to the discussion, the instructor monitors discussion activity for grading purposes. Further, the instructor chooses students at random to present an oral summary of the discussion to the class. This motivates all students to participate in on-line case discussions.

In addition, the instructor should consider creating a Web site for this course using Web technologies such as WebCT or BLACKBOARD. At minimum, the site should include:

- Course syllabus
- Lecture material (for example, PowerPoint slides, lecture notes, etc.). These should be placed on the site ahead of class meeting so that students may use the material to prepare for the lecture
- Summary of cases in ethics that are discussed in class
- Out-of-class assignments in the form of mini cases in Ethics
- Keys to exams (after students have completed them)
- Suggested solution to mini cases (after graded assignments have been returned)
- Mechanism for students to digitally submit their assignments
- Course calendar
- Mechanism to communicate electronically (for example, e-mail)
- Discussion groups. Students participate in one on-line case discussion every two weeks.
- The student's course performance measures

Classroom Hours (2 hours per week) **Class: 2**
Lab: 0

IX. Topics to be Covered

A. Overview of ethics
   1. Why ethics
   2. Ethics in the business world
   3. Forces that shape ethical behavior
   4. Introduction to ethical reasoning
   5. Solving ethical problems - The Ten Step method

B. Ethics for IT professionals and IT users
   1. Ethical behavior of IT professionals
   2. Code of ethics and professional conduct for Association for Computing Machinery, Association of Information Technology Professionals, Software Engineering (IEEE-CS/ACM), and IEEE.

C. Cybercrime
   1. Types of attacks
   2. Risk assessment
   3. Prevention, detection, and response

D. Information and personal privacy
   1. Invasion of privacy
   2. Electronically sharing/selling personal information
   3. Consumer profiling
   4. Spamming

E. Intellectual Property
   1. What is intellectual property?
   2. Key intellectual property issues (copyright infringement, software piracy, etc.)

F. Software Development
   1. Importance of software quality
   2. Software development process

G. Employer/Employee Issues
   1. Use of non-traditional workers (contract, off-shoring)
   2. Ethical considerations in interacting with supervisors, subordinates, and peers
   3. Ethics and performance appraisal

X. Laboratory Exercises

This course does not require a separate lab.

XI. Technology Component

- In class, the instructor makes use of state-of-the art multimedia projection equipment and software. These are used to project slides and Web-based content relevant to the concepts of and issues in professional ethics.
- Outside class, the instructor uses Web-based course management software (for example, WebCT, BLACKBOARD) to interact with students as described under course format section.
- All case analyses are submitted and examinations are taken online. Further, an online discussion group is set up to discuss some of the case studies assigned.

XII. Special Projects/Activities

There are no special projects or activities assigned in this course.

XIII. Textbooks and Teaching Aids

A. Required Textbook

George Reynolds; *Ethics in Information Technology*; Course Technology; 2003
ISBN: 0619062770

B. Alternative Textbooks

None.

C. Supplemental Print Materials

Robert B. Maddux and Dorothy Maddux; *Ethics in Business: A Guide for Managers*; Crisp Publications; 2003
ISBN: 0931961696.

D. Supplemental Online Materials

As available from publisher.

Course Title: GEIT 3341: Database Design

Semester Credit Hours: 3 (3,0)

I. Course Overview

The objective of this course is to give students an understanding of key issues related to database design and implementation to support the automation of key business processes in organizations. The course is designed so as to cover topics that are relevant from a database design and implementation perspective; particularly one that involves the provision of online access to data resources to a variety of physically distributed organizational users. It includes a mix of lectures (some of which are conducted in the laboratory) and discussions on contemporary articles from industry publications.

II. PMU Competencies and Learning Outcomes

This course helps students develop the ability to become conversant with applied database design issues and understand the related terms and issues that are important for database designers around the world. Additionally, the course provides the students with the communication, leadership and teamwork skills necessary to effectively work as professionals in teams, or in charge of teams, responsible for database design projects. Finally, the course imparts on the students an understanding of database design as more than simply the design of data repositories, but also as the design of database management resources that support the core and mission-critical business processes of an organization.

III. Detailed Course Description

The course begins with a discussion of ethical issues, legal issues, and aspects conducive to effective teamwork, in the context of database design. It then proceeds with a review of concepts and methods that are applicable to most database design projects. This covers concepts such as those of records and fields to diagramming tools such as entity-relationship diagrams. Next the course covers a central topic in database design, namely normalization. The 1st, 2nd and 3rd normal forms are covered in detail. Higher-level normal forms are also covered, although more briefly. The course concludes with a discussion of advanced issues in connection with database design, geared at Web-based access to single and multiple databases. The emphasis in this course is on database design using CASE tools, which allow designers to minimize the amount of low-level programming they have to perform in order to develop functional database systems.

IV. Requirements Fulfilled

This course is required of all students majoring in computer engineering, computer science or information technology in the College of Information Technology.

V. Required Prerequisites

- GEIT 1411: Computer Science I
- GEIT 1412: Computer Science II.

VI. Learning Outcomes

In this course, students learn:

- To become conversant with database design issues and understand the related terms and issues relevant to database designers around the world.
- To acquire the communication, leadership and teamwork skills necessary for effectively work as professionals in teams, or in charge of teams, responsible for database design and implementation projects.
- To understand the role of database design as more than simply the design of data repositories, but also as the design of database management resources that support the core and mission-critical business processes of an organization.

VII. Assessment Strategy

Students are assessed based on: their performance in two exams (midterm and final); their class participation, which includes the discussion of recent articles taken from online industry publications; and the quality of a final team project and related oral presentation. The relative weights of each of these items on the final grade are as follows:

- The midterm and final exams each account for 25% of the grade. Combined, they account for 50% of the grade.
- Class participation accounts for 10% of the grade, and is evaluated based on the ability of students to add to the material already provided by the instructor to them.
- The final team project accounts for 40% of the grade. It is evaluated based on a project document, oral presentation, and client perceptions of the team project. The project must be conducted in collaboration with a client organization (for example, a department at a large company or non-profit organization). A letter from the main contact person at the client organization, discussing and evaluating the project and its outcomes, must be provided to the instructor. The letter should contain the contact information of the person writing so the instructor can call him/her up and inquire about the project.

The exams encourage the students to review all of the concepts and methods discussed in class, which are primarily based on textbook material. This is complemented by the class discussions on recent articles taken from online industry publications, which allow the students to become conversant with the industry-specific lingo related to database design issues. The final project provides an experience where concepts, methods, and industry-relevant issues are all brought together in a very applied manner to solve a real problem faced by a real organization. While this project is not as extensive as a program capstone project, it gives the students the necessary exposure to industry-relevant issues to prepare them for the future challenge of conducting a final program capstone project, and subsequently pursuing a successful career as IT professionals.

VIII. Course Format

Four of the course's class meetings are used for laboratory demonstrations and activities geared at helping the students learn the several steps involved in designing and implementing a database system. The other class meetings are split into two main components: lectures, and class discussions. The lectures cover several topics outlined later in this syllabus. The class discussions are based on recent articles taken from online industry publications such as the *Searchers* and *CIO* magazines, which are freely available from the Web. The instructor provides the links to the articles, which are then downloaded by the students and read prior to class. In class, the students discuss the articles in small teams for about 20 minutes, developing three provocative questions per team. This is followed by a discussion involving the whole class, where each team asks one of the questions they developed, and other teams answer them, until all teams asked at least one of their questions. This discussion format is likely to lead to lively debate on topics that are directly addressed by the article, as well as on topics that are indirectly related to the article.

Classroom Hours (3 hours per week) **Class/lab: 3**

IX. Topics to be Covered

A. Ethical issues, legal issues, and effective teamwork
   1. Ethical and legal issues in database design
   2. Typical database design team composition
   3. Conflict resolution in database design teams
   4. Effective teamwork in database design teams

B. Basic concepts and methods
   1. Structured vs. "flat" databases
   2. Records and fields
   3. Key and foreign key fields
   4. Atomic and compound keys
   5. Partial and full dependencies
   6. Database entities and relationships
   7. Entity-relationship diagrams

C. Normalization
   1. Cardinality and modality
   2. Normalization
   3. 1st Normal form
   4. 2nd Normal form
   5. 3rd Normal form
   6. Higher-level normal forms

D. Advanced issues
   1. Database development environments
   2. Web-based database manipulation through SQL
   3. Providing Web-based access to a single database
   4. Web-based integration and access to dispersed databases
   5. Developing client/server Web-based database systems

X. Laboratory Exercises

This course has four laboratory sessions, which are scheduled using time from standard class meetings. In the laboratory sessions, students learn the several steps involved in the design and implementation, using a CASE tool, of several related databases, and the provision of integrated access to these databases through Web-based interfaces. The database servers that is host the related databases are set up separately from a Web server, whereby the databases are accessed by users by means of Web browsers (simulating what usually happens in organizational settings). Access to the database servers is controlled by the creation of user accounts with differential access to various components of the databases. That access is based on an authentication and access control module, which is also designed and implemented using a CASE tool.

XI. Technology Component

A. In class, the instructor makes use of state-of-the art multimedia projection equipment and software. These are used to project slides and Web-based content, as well as play freely available Web-based video clips from Web sites covering topics relevant to the class (for example, CNN.com Technology).

B. Outside class, the instructor uses Web-based course management software to interact with students, provide feedback on their performance, make available links to online articles, as well as receive documents (for example, draft versions of project reports) and provide feedback on them.

C. Outside class, in the laboratory setting, the instructor makes use of industry-strength commercial Web and database server software, as well as CASE tools, to create a simulated database design and operation environment.

XII. Special Projects/Activities

The team project consists of meeting with members of a client organization (for example, a department at a large company or non-profit organization), gathering relevant information from them, and developing a prototype database system, as well as a document containing the following elements:

- A set of organizational problems that could potentially be solved through the implementation of a database technology. For example, a team may study a manufacturing organization that uses the Web to interact with suppliers of raw materials, and find out that the deployment of a particular database technology could solve key problems facing the organization.
- A detailed description of a technology solution to the problems above. This description should include hardware and software details, as well as details in connection with how the technology is integrated with existing technologies in the client organization.
- A detailed description of the costs and potential benefits, from an organizational perspective, associated with the technology solution.

The prototype database system should incorporate about 50% of the functionality of the technology solution proposed by the student teams in their reports. It should be developed so that its interfaces are Web-based.

Oral presentation. Teams summarize and explain the information contained in their project document in an oral presentation in class at the end of the semester. This oral presentation should also incorporate a demonstration of the prototype database system developed by the students.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Thomas M. Connolly and Carolyn E. Begg; *Database Systems: A Practical Approach to Design, Implementation and Management*; 4th edition (May 24, 2004) Pearson Addison Wesley
ISBN: 0321210255.

B. Alternative Textbooks
Peter Rob and Carlos Coronel; *Database Systems: Design, Implementation, and Management*; 5 edition (December 18, 2001), Course Technology
ISBN: 061906269X.

C. Supplemental Print Materials

David Lane and Hugh E. Williams; *Web Database Applications with PHP & MySQL*, 1st edition (March 2002) O'Reilly & Associates
ISBN: 0596000413.

D. Supplemental Online Materials

Recent articles taken from online industry publications such as the Searchers and CIO magazines. The instructor provides the links to the articles, which are freely available from the Web.

Course Title: GEIT 3351: Software Engineering I

Semester Credit Hours: 3 (3,0)

I. Course Overview

The course is designed to provide an introduction to the theory and practice of software development and maintenance. The focus is on the full software development life cycle, including coverage of tools, techniques, principles, and guidelines for software requirements, specification, design and implementation. Particular emphasis is placed on the principles and methods used to develop and validate software requirements. Students are guided toward a better understanding of the various tasks and specialties that contribute to the development of a software product.

II. PMU Competencies and Learning Outcomes

This course helps students develop the ability to become conversant with software engineering topics and acquire a strong understanding of the software development life cycle. In the process, students gain an appreciation for the issues that are important for software engineering practitioners around the world. Software development is largely a collaborative effort, often times spanning multiple countries. It also requires logical, analytical and creative thinking skills to design robust software as well as strong technical writing skills to generate proper design documents. Therefore, the course provides opportunities to help strengthen these skills in students. Additionally, the course provides the students with the communication, leadership, and teamwork skills necessary to effectively work as professionals in teams responsible for software engineering projects.

III. Detailed Course Description

The course begins with a discussion of ethical issues, legal issues, and aspects conducive to effective teamwork, in the context of software engineering projects, with a primary focus on design and implementation issues. It then proceeds with an overview of software engineering process and looks at the software life cycle from an object-oriented perspective. Several life cycle models are discussed, including the Waterfall model, prototyping, spiral model, and RAD. Students are introduced to object-oriented modeling and requirements analysis leading to the preparation of requirements specifications. Next, the course introduces students to the software design process, covering topics in process architecture, UI and class design, and culminating into design specifications. The course concludes with a discussion of an implementation plan. Students are introduced to various implementation approaches, concepts of testing and developing a test plan. A brief introduction to project management topics is given to illustrate the final phase of the software development life cycle. GEIT 4351, Software Engineering II, picks up from here and hoes into the details project management and actual implementation.

IV. Requirements Fulfilled

This course is required of all students pursuing degrees in computer engineering, computer science, and information technology.

V. Required Prerequisites

- GEIT 1411: Computer Science I
- GEIT 1412: Computer Science II.

VI. Learning Outcomes

In this course, students learn:

- To become conversant with software engineering topics and understand the related terms and issues relevant to software engineers around the world.
- To be able to discuss and demonstrate approaches, techniques, or methods for creating a feasibility report; eliciting and writing requirements specifications; verifying and validating requirements; developing a rapid prototype; and managing, planning and scheduling large projects.
- To understand and use the appropriate life cycle and process model for the development of software product.
- To understand the role of software engineering as a strategic tool.
- To be able to effectively communicate verbally and in writing the deliverables of software development project.
- To acquire the communication, leadership and teamwork skills necessary for effectively work as professionals in teams, or in charge of teams, responsible for software engineering projects.

VII. Assessment Strategy

The student's performance in this course may be assessed on the basis of:

- One midterm and one final examination.
- One project completed outside of class.
- Class participation.

Relative weights assigned to these items in determining student's final grade are suggested as follows:

- Each exam accounts for 25% of the grade. Combined, the two examinations account for 50% of the grade.
- The team project accounts for 40% of the grade.
- Class participation accounts for 10% of the final grade.

The examinations are designed to assess the mastery the concepts and methods discussed in class, which are primarily based on textbook material. This is complemented by the class discussions on recent articles taken from online industry publications, which allow the students to become conversant with the industry-specific issues related to software engineering. The final team project provides an experience where concepts, methods, and industry-relevant issues are all brought together in a very applied manner to design a software product. Using one of the software design methodology (for example Waterfall life cycle model), students develop all life cycle deliverables, requirement documents, specification and design documents. This set of documents comes at the starting point for the project in GEIT 4351: Software Engineering II, where students design, prototype, test and implement the system.

VIII. Course Format

This course utilizes a mix of lectures and class-discussions to help students learn the various tasks involved in the software development process. The lectures cover several topics outlined later in this syllabus. The class discussions are based on recent articles taken from online industry publications, which are freely available from the Web. The instructor provides the links to the articles, which are then downloaded by the students and read prior to the class.

In addition, the instructor should consider creating a Web site for this course using Web technologies such as WebCT or BLACKBOARD. At minimum, the site should include:

- Course syllabus.
- Lecture material (for example PowerPoint slides, lecture notes, etc.). These should be placed on the site ahead of class meeting so that students may use the material to prepare for the lecture.
- Keys to exams (after students have completed them).
- Course calendar.
- Mechanism to communicate electronically (for example e-mail)
- Discussion groups.
- Project-related Resources
- Students course performance measures.

Classroom Hours (3 hours per week) **Class: 3**

IX. Topics to be Covered

A. Ethical issues, legal issues, and effective teamwork
   1. Ethical and legal issues in software engineering (SE)
   2. Typical SE team composition
   3. Conflict resolution in SE teams
   4. Effective teamwork in SE teams

B. Introduction to software engineering
   1. History of software engineering techniques
   2. Elements of a software development paradigm
   3. The role of the project

C. Object-oriented paradigm overview
   1. Object-oriented conceptualization
   2. The software life cycle
   3. Object-oriented modeling

D. Object-oriented analysis
   1. Process of requirements analysis
   2. Requirements specifications
   3. Evaluating, refining and verifying requirements specifications

E. Software design
   1. Product design overview and objectives
   2. Process architecture
   3. UI design

F. Class design
   1. Class design phase
   2. Class design process
   3. Class design verification

G. Implementation
   1. Implementation approaches
   2. Implementation plan
   3. Documentation

H. Testing
   1. Principles of OO-testing
   2. Testing techniques and strategies

I. A brief introduction to project management
   1. Configuration management
   2. Project planning and management
   3. Risk management

X. Laboratory Exercises

There is no laboratory component in this course.

XI. Technology Component

A. In class, the instructor makes use of state-of-the art multimedia projection equipment and software. These are used to project slides and Web-based content relevant to Web server administration.

B. Outside class, the instructor uses Web-based course management software (for example WebCT, BLACKBOARD) to interact with students as described under course format section.

XII. Special Projects/Activities

Software Engineering is all about developing quality, robust, and secure software. A semester project is intended to give students a feeling of how software is really being developed. In addition, the ability to work with other software developers is essential and critical in a software development project. Therefore, students are required to work effectively in teams throughout the semester and prepare the following deliverables for a proposed system in two phases:

- In Phase one (about eight weeks), students prepare:
  - Requirements Specifications,
  - Design specifications
  - Implementation plan
  - A prototype of the software
- In Phase two (next eight weeks), the complete sets of documentation are redistributed amongst the teams. So each team gets the requirements, design and software written by another team. Each team is now asked to evaluate the set of documents, test the software, and write an evaluation report.
- At the end of the project, each team gives a presentation of its findings.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Evelyn Stiller and Cathie LeBlanc; *Project-Based Software Engineering: An Object-Oriented Approach*; Addison Wesley; 2002
ISBN: 0-201-74225-X

B. Alternative Textbooks

1. David Gustafson; *Schaum's Outline of Software Engineering*; McGraw-Hill, 1st edition
   ISBN: 0071377948
2. Carlo Ghezzi; Fundamental *Principles of Software Engineering*; Publisher: Prentice-Hall, 2nd edition; 2002
   ISBN: 0113056996.

C. Supplemental Print Materials

As available from publisher.

D. Supplemental Online Materials

Recent articles taken from online industry publications such as the Methods and Tools, and CTO magazines. The instructor provides the links to the articles, which are freely available from the Web.

Course Title: GEIT 4351: Software Engineering II

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is a continuation and extension of GEIT 3351: Software Engineering I. The objective of this course is to give students an understanding of key issues involved in the design and implementation of computer software to automate business processes in organizations. The course is designed so as to cover topics that are relevant from a software engineering perspective, with a focus on software design and implementation, and software development project management. It is very applied, and one of its main components is a team project focusing on software design and implementation.

II. PMU Competencies and Learning Outcomes

This course helps students develop the ability to become conversant with software engineering topics and understand the related terms and issues that are important for software engineering practitioners around the world. Additionally, the course provides the students with the communication, leadership, and teamwork skills necessary to effectively work as professionals in teams, or in charge of teams, responsible for software engineering projects. Finally, the course looks at software engineering as more than programming, that is, as a complex collaborative task whose goal is to improve the quality and productivity of business processes, and the overall competitiveness of the organization housing the business processes, through the design and implementation of software that meets high standards of quality.

III. Detailed Course Description

The course begins with a discussion of ethical issues, legal issues, and aspects conducive to effective teamwork, in the context of software engineering projects, with a focus on design and implementation issues. It then proceeds with a review of project management topics that are relevant in the context of software engineering projects. This covers several fundamental project management issues such as task decomposition and related budgeting, project charting, project scope management, time and cost management, human resources management, communication management, and project risk management. The course concludes with a discussion of advanced issues in connection with software engineering projects, such as emerging tools for software engineering, the relationship between software engineering and business process redesign, contemporary and best practices in software engineering, and the relationship between those practices and software engineering certification based on the Capability Maturity Model. This course emphasizes both software design and implementation issues and software engineering project management issues. This dual emphasis orientation is aimed at providing students with a realistic view of issues related to real

software engineering projects, which are more often than not complex collaborative projects with a clear expectation of organizational impacts in terms of quality, productivity and/or competitiveness enhancements.

IV. Requirements Fulfilled

This course is required of all students pursuing degrees in computer engineering, computer science, and information technology.

V. Required Prerequisites

- GEIT 1411: Computer Science I
- GEIT 1412: Computer Science II
- GEIT 3351: Software Engineering I.

VI. Learning Outcomes

In this course, students learn:

- To become conversant with software engineering topics and understand the related terms and issues relevant to software engineers around the world.
- To acquire the communication, leadership and teamwork skills necessary for effectively work as professionals in teams, or in charge of teams, responsible for software engineering projects.
- To understand the role of software engineering as a key competitiveness-enhancing tool for organizations, both large and small.

VII. Assessment Strategy

Students are assessed based on: (a) their performance in two exams (midterm and final); (b) their class participation, which includes active participation in speakers' presentations, and the discussion of recent articles taken from online industry publications; and (c) the quality of a final team project and related oral presentation. The relative weights of each of these items on the final grade are as follows:

- The midterm and final exams each account for 25% of the grade. Combined, they account for 50% of the grade.
- Class participation accounts for 10% of the grade, and is evaluated based on the student's active participation in speakers' presentations, and the ability of students to add to the material already provided by the instructor to them.
- The final team project accounts for 40% of the grade. It is evaluated based on a project document, oral presentation, and client perceptions of the team project. The project must be conducted in collaboration with a client organization (for example, a department at a large

company or non-profit organization). A letter from the main contact person at the client organization, discussing and evaluating the project and its outcomes, must be provided to the instructor. The letter should contain the contact information of the person writing so the instructor can call him/her up and inquire about the project.

The exams encourage the students to review all of the concepts and methods discussed in class, which are primarily based on textbook material. This is complemented by the class discussions on recent articles taken from online industry publications, which allow the students to become conversant with the industry-specific lingo related to software engineering issues. The final project provides an experience where concepts, methods, and industry-relevant issues are all brought together in a very applied manner to solve a real problem faced by a real organization. While this project is not as extensive as a program capstone project, it gives the students the necessary exposure to industry-relevant issues to prepare them for the future challenge of conducting a final program capstone project, and subsequently pursuing a successful career as IT professionals.

VIII. Course Format

In four of the class meetings, invited speakers with practical experience in software engineering-related issues, and software development team members and/or managers, give presentations on topics related to software engineering and/or project management. These presentations last approximately 45 minutes, followed by 15 minutes for questions and answers. The remainder of the time in class is used for project work, lectures, and/or class discussions. The remaining class meetings are split into two main components: lectures, and class discussions. The lectures cover several topics outlined later in this syllabus. The class discussions are based on recent articles taken from online industry publications such as the *Methods and Tools*, and *CTO* magazines, which are freely available from the Web. The instructor provides the links to the articles, which are then downloaded by the students and read prior to the class. In class, the students discuss the articles in small teams for about 20 minutes, developing 3 provocative questions per team. This is followed by a discussion involving the whole class, where each team asks one of the questions they developed, and other teams answer them, until all teams asked at least one of their questions. This discussion format is likely to lead to lively debate on topics that are directly addressed by the article, as well as on topics that are indirectly related to the article.

Classroom Hours (3 hours per week) **Class: 3**
Lab: 0

IX. Topics to be Covered

A. Ethical issues, legal issues, and effective teamwork
   1. Ethical and legal issues in software engineering (SE)
   2. Typical SE team composition
   3. Conflict resolution in SE teams
   4. Effective teamwork in SE teams

B. Project management issues
   1. Task decomposition and budgeting
   2. Project charting
   3. Project scope management
   4. Project time management
   5. Project cost management
   6. Project human resources management
   7. Project communication management
   8. Project risk management

C. Advanced issues
   1. Emerging tools for SE
   2. SE as a follow-up to business process redesign
   3. Contemporary best practices in SE
   4. SE process certification and the Capability Maturity Model

X. Laboratory Exercises

There is no laboratory component in this course.

XI. Technology Component

A. In class, the instructor makes use of state-of-the art multimedia projection equipment and software. These are used to project slides and Web-based content, as well as play freely available Web-based video clips from Web sites covering topics relevant to the class (for example, CNN.com Technology).

B. Outside class, the instructor uses Web-based course management software to interact with students, provide feedback on their performance, make available links to online articles, as well as receive documents (for example, draft versions of project reports) and provide feedback on them.

XII. Special Projects/Activities

The team project consists of meeting with members of a client organization (for example, a department at a large company or non-profit organization), gathering relevant information from them, and developing a prototype software system. It is highly recommended that this project be a continuation of the work previously conducted in GEIT 3351: Software Engineering I. The team project also involves the preparation of a document containing the following elements:

- A set of organizational problems that could potentially be solved through the implementation of a software system. For example, a team may study a manufacturing organization that uses the Web to interact with suppliers of raw materials, and find out that the deployment of a

particular software system for online inventory tracking by the suppliers could solve key problems facing the organization – for example, it could contribute to the reduction of inventory levels and thus allow for the adoption of "leaner" manufacturing approaches such as the "just-in-time" approach.

- A detailed specification of a software system solution to the problems above. This specification should include hardware and software details, as well as details in connection with how the system is integrated with existing technologies and systems in the client organization.
- A detailed description of the costs and potential benefits, from an organizational perspective, associated with the software system solution.
- A detailed plan for the implementation of the software system, including a project task decomposition and related budgeting, a project timeline and related chart, and project risk management plan (with contingency options, in case the project is not successfully implemented as planned).

The prototype software system should incorporate about 50% of the functionality of the technology solution proposed by the student teams in their reports.

Oral presentation. Teams summarize and explain the information contained in their project document in an oral presentation in class at the end of the semester. This oral presentation should also incorporate a demonstration of the prototype software system developed by the students.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Robert K. Wysocki and Rudd McGary; *Effective Project Management: Traditional, Adaptive, Extreme*; John Wiley & Sons; 3rd edition (July 25, 2003);
ISBN: 0471432210.

B. Alternative Textbooks

Project Management Institute, *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*; Project Management Institute; (January 2001);
ISBN: 1880410230.

C. Supplemental Print Materials

Capability Maturity Model Integration – FAQ Web site; by Software Engineering Institute; Software Engineering Institute; (Use most recent FAQ document); available for free download from: http://www.sei.cmu.edu/cmmi/adoption/ques-ans.html

D. Supplemental Online Materials

Recent articles taken from online industry publications such as the Methods and Tools, and CTO magazines. The instructor provides the links to the articles, which are freely available from the Web.

Course Title: GEIT 4361: Practical Training

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course provides opportunities for students to apply the academic concepts, skills and techniques learned in their coursework to a professional work-oriented setting. The course offers the potential for a one-semester internship with a regional employer or a directed study course providing practical learning experiences that benefit the community.

II. PMU Competencies and Learning Outcomes

GEIT 4361: Practical Training is primarily practical in nature with no lecture content. Students' experience will be based on the individual placement of the student into an internship or directed practical training. This course requires students to apply all the PMU competencies (including professional competence, critical thinking, communication, leadership, and teamwork) to an organization. The internship is designed to sharpen the student's analytical and problem solving skills through research and guidance from the instructor and the business to which the student is assigned. This course allows for integrating different subject areas to deal with a problem or a situation that is not one-dimensional.

III. Detailed Course Description

The role of the practical training is to provide students with an appreciation of the types of work involved with their major before they actually enter the job market. The practical training course also provides students with first-hand experience and supplements the theories they have learned in the classroom. It allows them to draw upon various concepts to solve complex, real world problems. It provides the business or consumer with an opportunity to have students with fresh ideas work on an issue or a problem currently facing them. Internships may also provide employers with a risk-free chance to try potential employees before actually hiring them.

IV. Requirements Fulfilled

GEIT 4361: Practical Training satisfies three hours of the requirements for degrees in Information Technology, Computer Science, and Computer Engineering. It is an elective course available to all students pursuing a degree programs within the College of Information Technology. This course should be taken in the senior year.

V. Required Prerequisites

Senior year standing and the consent of the instructor.

VI. Learning Outcomes

A. To learn effective professional behaviors in a business/commercial setting.

B. To learn to apply academic concepts to the solution of real-world problems.

C. To enhance professional communication and collaboration skills.

VII. Assessment Strategy

Internship

A faculty member supervises the student and monitors his or her progress. The employer provides written feedback on the student's performance and professional competencies and skills emphasized by the PMU. This feedback enables the instructor to assess student work and assign a grade for the course.

Directed Study

If it not possible for the student to participate in an internship, the student may be allowed to take a directed study course. Such a course will be designed to provide practical learning experience under the supervision of a faculty member and targeted to designing solutions that meet the real world technology needs of the local community.

To facilitate directed studies opportunities, is recommended that the PMU establish a Community Technology Resource Center to encourage members of the community to seek professional advice for small technology projects that the students can complete under the supervision of the faculty.

VIII. Course Format

Internship

The student works specified hours at an employer's location. Total hours devoted to this activity should be 10 hours a week for one semester. The workload includes reports and presentations.

Students are expected to file formal reports and make presentations to the supervising faculty each month.

Directed Study

Directed study students are expected to develop a schedule equivalent to 150 total hours in managing their project. The schedule is to be validated and approved by the supervising faculty.

Students are expected to file formal reports of their progress and make presentations to the supervising faculty each month.

Internship Hours (10 hours per week)
Directed Study (10 hours per week)

IX. Topics to be Covered

No new topics are addressed. Rather, the course involves application of course material learned in the College of Information Technology core curriculum and the student's major field of specialization.

X. Laboratory Exercises

This course does not require a separate laboratory. However, Internship students should have access to the technology infrastructure at the business location and directed study students should have access to the university technology infrastructure.

XI. Technology Component

Technology is used as needed and depends on the level of technology present at the business location.

XII. Special Projects / Activities

Students are required to keep a "reflective notebook" in which, after each session with the employer, client or supervising faculty, they enter their own assessments of what they learned, and what questions remain. Students should use their notebooks as tools for identifying problems and evaluating potential solutions

XIII. Textbooks and Teaching Aids

A. Required Textbook

None.

B. Alternative Textbooks

None.

C. Supplemental Print Materials

None.

D. Supplemental Online Materials
None

Course Title: ASSE 4311: Learning Assessment III

Semester Credit Hours: 3 (3,0)

I. Course Overview

This is the capstone course required of all students pursuing an undergraduate degree program within the College of Information Technology. The objective of this course is to bring together in an applied manner the knowledge and skills obtained by the students throughout their undergraduate program. The course is designed so as to cover topics that are relevant from an integrated IT systems design and implementation perspective. The term "integrated IT systems design and implementation" refers to complex collaborative efforts that bring together knowledge skills in the related areas of computer science, computer engineering, and information technology (as covered by the three undergraduate programs offered by the College of Information Technology). The course is very applied. One of its main components is a team project focusing on integrated IT systems design and implementation. The course also includes a mix of speakers' presentations, project work, and discussions on contemporary articles from industry publications.

II. PMU Competencies and Learning Outcomes

This course helps students develop the ability to become conversant with integrated IT systems design and implementation topics. Additionally, the course provides the students with the communication, leadership and teamwork skills necessary to effectively work as professionals in teams, or in charge of teams, responsible for integrated IT systems design and implementation projects. Finally, the course looks at IT from a strategic and integrated perspective, that is, a holistic or whole-organization perspective, rather than viewing IT as a set of tools used in a localized manner to improve certain parts of an organization (for example, a specific department, area, or process).

III. Detailed Course Description

The course comprises a set of speakers' presentations and class discussions on contemporary articles from industry publications addressing integrated IT systems design and implementation. Particular attention is paid to the complex nature of integrated IT systems design and implementation projects. Such projects usually involve a high level of heterogeneity in terms of technologies used, cultural background of those participating in the project, and types of expertise involved. This course emphasizes a holistic or whole-organization perspective, rather than the more common perspective of IT as a set of tools to be used in a localized manner to improve parts of an organization that do not usually interact

with each other (for example, a specific department, area, or process). The goal with this emphasis is to provide students with a realistic view of what the job of a chief technology officer (CTO) entails. CTOs are usually responsible for managing the IT resources of an entire organization, and ensuring that those resources are used in a synergistic way to benefit the organization as a whole.

IV. Requirements Fulfilled

This course is required of all students pursuing degrees in computer engineering, computer science, and information technology.

V. Required Prerequisites

- GEIT 1411: Computer Science I
- GEIT 1412: Computer Science II
- GEIT 1311: Computer Organization I
- GEIT2291: Professional Ethics
- GEIT 3341: Database Design
- GEIT 3351: Software Engineering I
- GEIT 4351: Software Engineering II.

VI. Learning Outcomes

In this course, students learn:

- To become conversant with integrated IT systems design and implementation topics and understand the related terms and issues relevant to chief technology officers around the world.
- To acquire the communication, leadership and teamwork skills necessary for effectively work as professionals in teams, or in charge of teams, responsible for with integrated IT systems design and implementation projects.
- To understand the role of with integrated IT systems design and implementation projects as important competitiveness-enhancing endeavors for organizations, both large and small.

VII. Assessment Strategy

Students are assessed based on: (a) their class participation, which includes active participation in speakers' presentations, and the discussion of recent articles taken from online industry publications; and (b) the quality of a final team project and related oral presentation. The relative weights of each of these items on the final grade are as follows:

- Class participation accounts for 30% of the grade, and is evaluated based on the student's active participation in speakers' presentations, and the ability of students to add to the material already provided by the instructor to them.

- The final team project accounts for 70% of the grade. It is evaluated based on a project document, oral presentation, and client perceptions of the team project. The project must be conducted in collaboration with a client organization (for example, a department at a large company or non-profit organization). A letter from the main contact person at the client organization, discussing and evaluating the project and its outcomes, must be provided to the instructor. The letter should contain the contact information of the person writing so the instructor can call him/her up and inquire about the project.

The speakers' presentations and the discussion of recent articles taken from online industry publications allow the students to become conversant with the industry-specific terminology related to integrated IT systems design and implementation issues. The final project provides an experience where concepts, methods, and industry-relevant issues are all brought together in a very applied manner to solve a real problem faced by a real organization. It gives the students the necessary exposure to industry-relevant issues to prepare them for the future challenge of pursuing a successful career as IT professionals.

VIII. Course Format

In 50% of the class meetings, invited speakers with practical industry experience (ideally senior managers) give presentations on topics related to integrated IT systems design and implementation. These presentations last approximately 45 minutes, followed by 15 minutes for questions and answers. The remainder of the time in class is used for project work and/or class discussions. The remaining class meetings are used for project work and/or class discussions. The latter are based on recent articles taken from online industry publications such as the *CTO* and *CIO* magazines, which are freely available from the Web. The instructor provides the links to the articles, which are then downloaded by the students and read prior to the class. In class, the students discuss the articles in small teams for about 20 minutes, developing three provocative questions per team. This is followed by a discussion involving the whole class, where each team asks one of the questions they developed, and other teams answer them, until all teams asked at least one of their questions. This discussion format is likely to lead to lively debate on topics that are directly addressed by the article, as well as on topics that are indirectly related to the article.

Classroom Hours (3 hours per week) **Class: 3**
Lab: 0

IX. Topics to be Covered

A. Ethical issues, legal issues, and effective teamwork
   1. Ethical and legal issues in integrated IT design and implementation
   2. Typical integrated IT design and implementation teams
   3. Conflict resolution in integrated IT design and implementation teams
   4. Effective teamwork in integrated IT design and implementation teams

B. Integrated IT design and implementation
   1. Classic cases of integrated IT design and implementation
   2. Contemporary cases of integrated IT design and implementation
   3. Integrated IT design and implementation project management
   4. Best practices in integrated IT design and implementation
   5. Classic cases of IT use as a strategic weapon
   6. Developing an organization-wide IT strategy

X. Laboratory Exercises

There is no laboratory component in this course.

XI. Technology Component

A. In class, the instructor makes use of state-of-the art multimedia projection equipment and software. These are used to project slides and Web-based content, as well as play freely available Web-based video clips from Web sites covering topics relevant to the class (for example, CNN.com Technology).

B. Outside class, the instructor uses Web-based course management software to interact with students, provide feedback on their performance, make available links to online articles, as well as receive documents (for example, draft versions of project reports) and provide feedback on them.

XII. Special Projects/Activities

The team project consists of meeting with members of a client organization (for example, a department at a large company or non-profit organization), gathering relevant information from them, and conducting an integrated IT systems design and implementation project. The team project also involves the development of a prototype integrated IT solution, and the preparation of a document containing the following elements:

- A set of organizational problems that could potentially be solved through an integrated IT systems design and implementation project. This project must bring together topics that are unique to each of the sister areas of computer science, computer engineering, and IT applications (covered by the three undergraduate programs offered by the College of IT). It is strongly recommended that student teams include individuals representing each of the three undergraduate majors of the College of IT.

- A detailed specification of an integrated IT solution to the problems above. This specification should include hardware and software details, as well as details in connection with how the solution is integrated with existing technologies and systems in the client organization.
- A detailed description of the costs and potential benefits, from an organizational perspective, associated with the integrated IT solution.
- A detailed plan for the implementation of the integrated IT solution, including a project task decomposition and related budgeting, a project timeline and related chart, and project risk management plan (with contingency options, in case the project is not successfully implemented as planned).

The integrated IT solution actually implemented (in a prototyped manner) should incorporate about 60-80% of the functionality of the solution proposed by the student teams in their reports.

Oral presentation. Teams summarize and explain the information contained in their project document in an oral presentation in class at the end of the semester. This oral presentation should also incorporate a demonstration of the integrated IT solution actually implemented by the students.

XIII. Textbooks and Teaching Aids

A. Required Textbook

Tom Demarco and Timothy Lister, *Peopleware: Productive Projects and Teams*; Dorset House; 2nd edition (February 1, 1999) ISBN: 0932633439.

B. Alternative Textbooks

John Ward and Joe Peppard, *Strategic Planning for Information Systems*; John Wiley & Sons; 3rd edition (June 15, 2002) ISBN: 0470841478.

C. Supplemental Print Materials

Roger Beach, *Adopting Internet technology in manufacturing: a strategic perspective*, Production Planning & Control, Jan 2004, Vol. 15 Issue 1, p80, 10p.

D. Supplemental Online Materials

Recent articles taken from online industry publications such as the CTO and CIO magazines. The instructor provides the links to the articles, which are freely available from the Web.