Software Engineering

disciplinary Commons

Section 1: Background

Courses: Software Engineering I & Capstone Experience

Southeast Missouri State University - 2009-10

Level	4th year students (a few 3rd year)
Language	Neutral*
Typical number of Students enrolled	7 (has varied from 7 to 35)
Fraction of full-time student effort/year	One-fifth (both courses together)
Number of weeks	16 for each course
Contact hours per week	3 for each course
Lab hours per week	No formal lab; just one lab session in first course
Description of Labs	Work on projects using lab facilities

This particular course sequence in the Computer Science program fulfills the *real-world experience* aspect of the University's mission statement:

> "The University, through teaching and scholarship, challenges students to extend their intellectual capabilities, interests, and creative abilities; develop their talents; and acquire a lifelong enthusiasm for learning. Students benefit from a relevant, extensive, and thorough general education with a global perspective; professional and liberal arts and science curricula; co-curricular opportunities; and real-world experiences. By emphasizing student-centered and experiential learning, the University, in collaboration with other entities as appropriate, prepares individuals to participate responsibly in a diverse and technologically advanced world, and in this and other ways contributes to the development of the social, cultural, and economic life of the region, state, and nation. "

It also helps in providing additional evidence in achieving the program's two educational objectives:

1. Recognized by their peers and superiors for their technical skills in the computing field.
2. Recognized by their peers and superiors for their professional skills in the computing field (e.g. communication, teamwork, leadership).

Software Engineering

Section 2: Context

Audience
This portfolio is intended for instructors facilitating a two-course sequence in Software Engineering within a Computer Science or Computer Information Systems program.

Purpose
Since this portfolio grew out of the author's experience in teaching Software Engineering for the past ten years, instructors are who are beginning to teach such a course sequence could use it as another teaching source. The portfolio presented here contains all the teaching material for the first four weeks of the two semester course sequence. However, since complete syllabi for the two courses are given, the interested instructors could contact the author for additional information.

Teaching Philosophy
My return to academia after 14 years of experience in industry, where I developed large-scale applications and managed Information Technology (IT) resources, was a deliberate move. This move created an opportunity for teaching concepts and heuristics of system development in a practical way that blends theory and practice. This approach works very well in Southeast Missouri State University (SEMO) where computer science education is grounded in both science and practical experience. In line with the University's mission, the Software Engineering course sequence I teach incorporates real-world experience through the use of client sponsored system development projects. I take on the role of a project director in ensuring that the students' first system development experience is a successful one. Furthermore, my student-centered approach for delivering the concepts in system development (mainly, analysis, design, and implementation) involves assignments requiring team work. I offer this SE course sequence with content and assessments to suit ABET – Computing Accreditation Commission criteria that are not generally covered by most of the other courses in SEMO's CS curriculum. At the same time, I do the best I can in delivering the material in a seamless fashion – relating new concepts to the material the students have already seen in earlier courses.

Environment

The pre-requisites for the SE course are *Computer Science III* and *C and The Unix.* For a partial CS program pre-requisite structure, see Figure 1. Over the years, the class size has varied from 7 to 35, the composition (non-traditional vs traditional students) ratio has varied from 10% to 30%, and female-male ratio has varied from 0% to 15%. Most of the students come from the 25 counties SEMO serves. There are no note-worthy constraints in facilitating the course, except that the course involves a lot of planning and interaction with clients. Depending on the class size, the number of teams can vary from two to seven. Directing three teams in the Capstone Experience course would constitute a 3-credit hour teaching load in a semester system This SE sequence is part of a CS/CIS program, other than this, none of the contextual factors has any significant influence in the design and delivery of the courses.

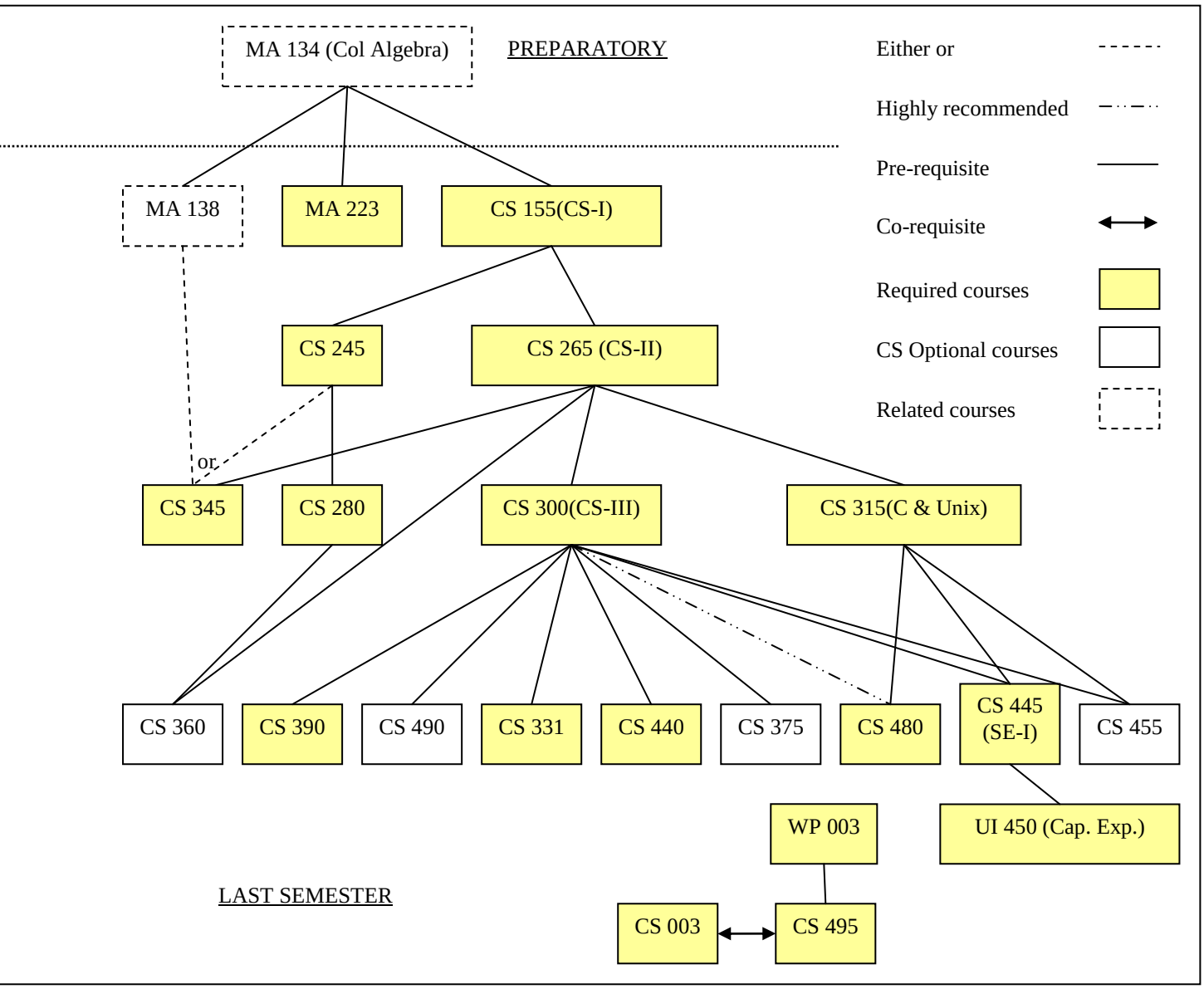

Figure 1. Partial Pre-requisite Structure of Computer Science Program

Software Engineering

Section 3: Course Content

Course Content – Summary

The course sequence has the following two courses: Software Engineering – I (CS445) and Capstone Experience (UI450). The focus in CS445 is on analysis and design aspects of system development and in UI450, on supporting management processes for system development and the application of all these processes for developing a prototype to meet a client's requirements. See the attached course sequence Content document.

The set of course information given to the students include the aim, sequencing of topics, the assessments, and expected workload. There are eight learning outcomes for this course sequence. The course content also shows a mapping between the learning outcomes and candidate assessments from the two courses. The content concludes with a brief discussion on assessments.

Evolution from a single course to a two-course sequence

The first year I taught SE in SEMO, there was just one 4-credit hour course. The CS/CIS programs had quite a few outcomes, which were to be accomplished in this course. As a result, with the support from the advisory board, the department agreed to replace this four-credit-hour course with a two three-credit-hour course sequence, with core system development workflows in one course and support workflows in the other. I try to follow the Rational Unified Process (RUP). The topics in the first course are: SE concepts, life cycle models, development methodologies, Unified Modelling Language (UML), Requirements Specifications (Use Case Model), Use Case Analysis, Architecture styles, Sub-system design, Design Mechanisms, Object Design. The second course includes: Test Case Preparation, Project Management, and Communications Management during the first four weeks. The remaining time is allocated for working on a client sponsored project, where the students, working in teams, apply the heuristics for analyzing the requirements and preparing the design specifications and implementing a prototype

The topics are organized in line with the top-down system development approach. The first topic is the context of SE in the overall CS/CIS program. This is followed by a quick overview of system life cycle, the system development processes, and major deliverables that precede code. Even though the students had several courses involving them, they know little about requirements and design specifications. Further, they had very little exposure to the tools used for specifying the results of analysis design. So, the next topic is UML. Considerable amount of time is allocated to learning the heuristics and principles used in the core workflows: requirements, analysis and design. Students use Rational Rose (free from IBM for course use) for preparing the specifications.

Methodology

A running case study is used in the first course for applying the concepts dealing with use case model, use case analysis, architecture, sub-system design, and use case design. The students work in teams to produce the three intermediate products. This course sequence help integrate the knowledge elements from the previous courses. At times source are given in class to illustrate the value of designs involving sub-systems. Class diagrams are presented to show the under-the-hood interactions for handling persistence (in a relational database management system (RDBMS)) and distribution (in the Common Object Request Broker Architecture (CORBA)). Since the students are already exposed to user interface design in Computer Science-III, they expected to apply that knowledge in the capstone project. Client sponsored projects are assigned at the end of the first course. This provides time to learn any new tools required in their capstone project. Graded exams, homework, project documents are all reviewed in class to enhance understanding.

A well-structured course makes learning easier. Applying requirements, analysis and design concepts for a single project – especially client-sponsored – helps simulate real world experience. The team members take on different roles which help hone their professional skills. Ideally, even a CS/CIS program would benefit from a mandatory six-month SE apprenticeship in an industry. In its absence, the next best thing would be to work in teams on a client-sponsored project.

COURSE SYLLABUS: FALL SEMESTER 2***
CS445: SOFTWARE ENGINEERING-I

Facilitator: Ken Surendran **Office**:DH-021F **Tel**: 2208
Web: http://cstl-csm.semo.edu/surendran **e-mai**l: ksurendran@semo.edu
Office Hours: M, T, R: 10:00-11:0 and by appointment

CATALOG DESCRIPTION: CS445 Software Engineering-I. Provides an in depth understanding of the principles and techniques used in the analysis and design aspects of developing application systems. Apply techniques and tools to produce the system artifacts pertaining to analysis and design. Prerequisite: Junior standing and a minimum grade of 'C' in CS300 and CS315 (3).

AIM: To familiarize with the analysis and design concepts and techniques used in developing quality software products and apply them. In this course the participants will learn the concepts and techniques for analysis and design used under the object-oriented paradigm. The participants will, working in teams, produce the analysis and design documents for an in-house system requirement.

Text: *Object-Oriented Software Engineering* by Berd Bruegge and Allen H. Dutoit, 3rd edition. Pearson / Prentice Hall (2010) (Additional notes and handouts will be given to supplement the text)

Software: *Rational suite Development Studio or Visio*

Tentative Course Outline:

W-#	Topics
1	Course overview, Software Engineering concepts, SDLC
2	System Development Methods [Homework-1]
3	UML [Test-1]
4	Use Case Model [Homework-2]
5	Intro to Rational Rose Lab [Test-2]
6	Use Case Analysis – Three types of classes [Homework-3]
7	Use Case Analysis – Heuristics on identifying classes [Requirements Spec.]
8	Mid-term
9	Use Case Analysis – View of Participating Classes
10	System Design –Design Goals, Architecture Styles [Homework -4]
11	System Design – Sub-system design (System interfaces) [Test-3]
12	System design - Handling Distribution and Security [Use Case Analysis Report]
13	System Design Deployment and Handling Persistence [Homework-5]
14	Object Design - Patterns
15	Object design – Interface Specification
16	Course Review; Project Selection for UI450
17	Final Exam

Assessment and Grading:

Tests (Three – 5% each)	15%
Mid-term	10%
Homework (Five – 5% each)	25%
Group Assignment (10 + 10 + 10)	30%
Final Exam	20%

Letter grades assigned approximately on a 90, 80, 70, 60 scale.

Note: The above weekly schedule and the assessment plan are subject to changes. However they are intended to give you an idea of what the course is all about.

The three assignments are to be carried out in groups of three/four. All the teams will work on a single case study. Team members should take on various coordinating roles (other than analyst and designer, which roles everyone will play) such as activity coordinator, documentation coordinator, team communication manager, meeting organizer, etc. These roles can be rotated so that each person gets to play at least two additional roles.

All the teams will work on a single project. The expected deliverables for each phase will be given in stages later on. Since there are three phases, it is important to plan properly to meet the stated deadlines.

Assignment-1: Software Requirements Specification
Use Case Model, Supplementary Spec (QoS), Glossary
Assignment-2: Use Case Analysis
Sequence diagrams, View of Participating Classes, Analysis Mechanisms
Assignment-3: System Design Specification
System Architecture, Sub-system design, Object design, I/O & Database designs

Grades in group activities: For each phase of the group assignment, each team will be assigned a team grade; each student will receive a percent of that grade, proportionate to her/his contribution to that phase, based on peer observations and instructor evaluation. To enable this process, each member will state clearly her / his contributions in each of the three phases.

Student Expectations: Participants can expect to spend about 9 hours per week outside class for learning the material and working on the assignments. Since there are three assignments which require working in groups, make sure to choose team members who have common free time for meetings outside the class.

Participants will read the appropriate chapters before class and be prepared to contribute to the in-class learning. Rational Rose is available in the CS labs. The participants should develop the necessary skills in using it.

COURSE SYLLABUS: SPRING SEMESTER 2***
UI450: Capstone Experience

Facilitator: Ken Surendran **Office**:DH-021F **Tel**: 2208
Web: http://cstl-csm.semo.edu/surendran **e-mai**l: ksurendran@semo.edu
Office Hours: M, W, F: 10:00-11:00 and by appointment

CATALOG DESCRIPTION: UI450: Capstone Experience. Apply the principles, techniques, processes and tools for developing a product such as quality software with all documentation for a client. Work in a team to complete a client sponsored project from inception to handover. Prerequisites: A minimum grade of 'C' in CS445, Senior standing in CS Major, and passage of the 75 hr Writing Test. (3).

AIM: To apply the concepts and techniques learned in CS445 for developing quality software products and to examine the supportive workflows such as project management, quality management, communication management, and configuration management. The participants, working in groups on a client-sponsored project, will apply the techniques and standard object-oriented tools to develop a software product together with all the documentation and other system artifacts. They will interact with the client to understand the application domain. In addition, they will be exposed to recent developments in the field of software engineering through literature survey and class presentations. The course also meets some of the nine university study objectives.

Text: *Object-Oriented Software Engineering* by Berd Bruegge and Allen H. Dutoit, Prentice Hall 3rd edition (2010)
Software: *Rational suite Enterprise* (temporary licenses provided by IBM / Rational *Microsoft Project* (available in labs)

Tentative Course Outline:

Week #	**Topics ***
1	Course overview, Review methodologies
2	Testing - Use Case Presentation
3	Project Management; [Exam –1]
4	Communications Management [UCM presentation]
5	Quality & Configuration Management [Requirements Spec]
6	Work on Project. [Exam –2]
7	Work on Project
8	Work on Project [UI presentation & Analysis Spec]
	Spring Break
9	Research Presentations
10	Work on Projects [Baseline system & final design]
11	Work on Project; Weekly review
12	Work on Project; Weekly review
13	Work on Project; weekly review
14	[Mock Project demo to class}
15	[Project presentation and demo of final system to client, faculty, and public
16	[Project documentation along with system binary].

Assessment and Grading:
Exams (Two - both closed book: 10 + 10) 20%
Project presentations in class (UC; UCM, UI) 15%
Research Presentation (team) 10%
Project Reports in three parts: (10+10+10)
UCM, Analysis, Design (Intermediary deliverables)30%
Project report and Presentation (15 +10) 25%
Letter grades assigned approximately on a 90, 80, 70, 60 scale.
Project grade: For each phase of the project, each team will be assigned a team grade; each student will receive a percent of that grade, proportionate to her/his contribution to that phase, based on peer and instructor evaluation.

Team project Deliverables:

Inception Phase:
Elaboration Phase:
Elaboration & Construction Phases:
Construction & Transition Phases:
Details of the contents for each phase will be provided in a separate document.

Research Presentation:

One of the purposes of this assignment is to let you acquire a better understanding of the topics that were not considered in detail during the lecture sessions. A list of topics is provided below from which each team will choose one. It will be a 30-minute presentation per team (two topics per session), each member contributing one or two aspects. Sample topics are:

1. CMM– quality measures for software building organizations
2. Aspect Oriented Programming
3. Agile methodology - eXtreme Programming (Highlights)
4. CoCoMo – Cost Models
5. Service Oriented Architecture
6. Process-Oriented Analysis and Design

You are encouraged to consult me (e-mail or in person) on any questions you have on this assignment as you go along. Other team members and I will evaluate the presentations.

Student Expectations:

Participants can expect to spend about 9 hours per week outside class learning the material and working on the team project. They will read the appropriate chapters before class and be prepared to participate in the discussions. They will prepare and present research topic assigned. They will work on the team project (sponsored by external clients) and in that they will use Rational Rose, which is available on all the project machines, for documentation. Teamwork is critical for this course.
The learning objectives of the above two course are that the student will be able to

A. Discuss the challenges in software engineering and examine the paradigms and methodologies used for developing quality software products.

Assessment: Homework 1 and Test 1 in CS445

B. Prepare requirements specification and analysis models, applying systems analysis techniques

Assessment: Group Assignment 1&2, Mid-term in CS445

C. Prepare architectural model, and detailed design models, which include user interface, database and system designs.

Assessment: Group Assignment 3, Final Exam in CS445

D. Discuss the various testing concepts for establishing quality assurance

Assessment: Exam-1 in UI450

E. Discuss project management and communications management issues in software development.

Assessment: Exam-2 in UI450

F. Apply the knowledge from their major discipline to develop, working in teams, a software product for a client together with all the documentation and other system artifacts.

Assessment: Group project leading to prototype development, presentation & demo

G. Orally present the intermediate system artifacts (generated during analysis and design) for review and evaluation.

Assessment: Class presentations (three)

H. Create analysis and design documents pertaining to the system they are developing.

Assessment: Project report

Software Engineering

Section 4: Instructional Design

Summary

Although the focus in CS445 – the first of the two-course sequence –is on analysis and design aspects of system development, the course starts with an introduction to software engineering that addresses the following preliminary topics: inherent difficulties in building software products, different life cycle models of software development, processes used for building software, the analysis and design models – the artifacts produced before implementation, and tools used for documenting the results of analysis and design. These topics are discussed during the first four weeks. This document discusses the design and delivery of these preliminary topics that lay the foundation for the main intent of the course. It presents the teaching mechanisms used for each of the above topics. The artifacts used include sample presentation slides, homework assignments, tests, and a case study for requirements specification. The artifacts pertaining to the lab are discussed but not included here. All of these materials pertain to the first four weeks to serve as samples.

Building Software Product:

The students in this course have written programs for solving simple problems in their earlier courses. However, they have not developed any software intended for use by others. Hence the discussion starts with examining the characteristics of software products and identifying the difficulties in developing them. We discuss the IEEE Software article *No Silver Bullet: Software Engineering Reloaded* by Steven Fraser and Dennis Mancl in class. The first homework has a question on the *No Silver Bullet: Essence and Accidents of Software Engineering* article by Professor Fred Brooks. Through these, the students recognize the progress made for improving the software development practice. These include the need for using a development process and for learning the characteristics of various life cycle models and methodologies. They also recognize the tasks in the pre-implementation activities – i.e., requirements gathering, analysis and design – and the value of different *views* (intermediary models) of the intended software product required by various stakeholders in the project. The week-1 and week-2 slides form the basis for class discussion. The students are evaluated on their understanding of these preliminary topics through a test and two homework assessments (appended Test-1, Homework-1 and Homework-2).

Modeling Language

Since UML offers the standard for the models (diagrams) used in expressing the results of the pre-implementation activities, the students need to get some exposure to both UML and to an UML-compliant tool such as Rational Software Architect (Rational Rose) before we discuss requirements specification. Even though UML 2.0 has some 13 diagrams, initially we focus on just four: Use Case, Sequence, Class, State diagrams. (The Pareto rule holds here: 80% of

analysis and design requires just 20% of the model diagrams.) I use an example from a familiar application domain for learning the symbols used in these diagrams. The students work in pairs on six pre-packaged lab exercises for developing their skills is using Rational Rose for drawing those four diagrams. The students can also use Rational Rose on their home machines (they access the university license server).

While discussing the Use Case topic, we emphasize the services provided by the system to the users of the system. While discussing the class diagram, we show them – without going into a lot of details - how the class is translated into code. While discussing sequence diagrams, we tell them how the messages translate into responsibilities of the classes. These discussions help students in relating the new material they are learning to what they already know. (Since we use a top-down approach in this course, these discussions help our students.) We discuss UML and Use Case Model (UCM) during the next two weeks, See week-3 and week-4 slides used in the discussions. We use assignment-1, Homework-3 and Test-2 to evaluate students' learning.

Requirements Specification

We use a familiar application case study – course registration system – for introducing use case diagram. We draw the use case diagram on white board with input from students. We discuss the heuristics for identifying actors and uses cases. The students work in groups on additional cases to identify actors and use cases. Some of their use case diagrams are discussed in class. At this stage, we introduce the <<include>> and <<extend>> stereotypes for refining use cases. Next, we discuss scenarios and use case descriptions. We give students a complete use case model (UCM) – use case diagram and use case descriptions for each use case with its *happy path* and alternative flows. In the sample requirements specification, we include, in addition to UCM, a glossary of terms in the application domain and the non-functional (Quality of Service) requirements of the system under development (based on the information in the case study).

The students work in groups on a case study for to prepare the requirements specification in-line with the standards used in the sample specification (See appended Assignment that has three parts.) We also evaluate the students for their understanding on UML and on requirements specification through a test (see test-2 and homework-3). The students work on the same case study for their analysis and design specifications assignments. See the case study in the following that has three parts (Requirements – UCM, Use Case Analysis and Design. The instructor serves as the client. The students working in groups take on various roles. Guidelines are provided for organizing the various specifications.). Depending on the group size, the number of use cases for the last two phases are adjusted to balance the workload.

Facilitation

Session objectives and topics are discussed first. Handouts (Microsoft PowerPoint mainly, and others as needed) are used in class. Only selected slides are discussed. (Since the students rent books, detailed PowerPoint handouts are provided.) Activities for the first four weeks are briefly discussed.

Week-1: course objective, difference between hardware and software; difficulties in building software, history of SE project success, software engineering artefacts – all through class discussions. Assign Homework -1

Week-2: SDLC, need for A&D, comparing different life cycle models, two methodologies and comparisons (Rational Unified Process (RUP) and Extreme Programming (XP)) – using diagrams and class discussions; identify workflows;
assign Homework-2

Week-3: Intro to UML. Class discussions using ppt; topics: brief history of UML, five views; emphasize just four diagrams - use case, sequence, class, state. Lab session: six exercises to learn to draw the four diagrams (in Week 3 or 4). Conduct Test-1; Assign Homework -3

Week-4: Use Case Model. Provide a sample UCM for a given case study. Discuss use case diagram – include and extend stereotypes, use case descriptions, flows and alternate flows, additional cases for students to discuss their use cases in class; emphasize heuristics for finding actors and use cases, importance of glossary, need for gathering non-functional requirements. Discuss Assignment-1

CS445: SOFTWARE ENGINEERING I

Homework-1: (due on 9/01) Counts 5% towards the final grade
Your response to all the following questions should be in full sentences and word processed.
There are several URLs on *No Silver Bullet* article by professor Fred Brooks; e.g.,
http://www.virtualschool.edu/mon/SoftwareEngineering/BrooksNoSilverBullet.html

1. Briefly summarize – in your own words - Professor Fred Brooks' main observations regarding software development (i.e., essential and accidental difficulties) and his message for improving the situation in Software Engineering found in No Silver Bullet articles (about 150 words); (15 Points)

2. Solve Problem-# 2 under Chapter1 in our text (5 Points)
3. Describe in your own words what is represented in Figure 1.1 in text (5 Points)
4. Solve Problem-#10 under Chapter 1 in our text (5 Points)

Hand in your solution document on 9/01. Late submissions may not be graded.

Homework-2: (due on 9/17) Counts 5% towards the final grade
Your response to all the following questions should be in full sentences and word processed.

1. Solve Problem-# 2 under Chapter15 in our text (7 Points)

2. Solve Problem-# 6 under Chapter 1 in our text; provide an explanation when the choice is not clear. (5 Points)

3. Solve Problem-# 7 under Chapter 1 in our text; provide an explanation when the choice is not clear. (3 Points)

4. Solve problem-#1 in Chapter 16 (pp689) (4 Points)

5. State any five characteristics of XP. (Section 16.4 has some info on this.) (5 Point)

6. State very briefly the main differences between the light weight agile method like XP (Beck) and plan-driven like the heavy duty UP (Royce) methodologies. The following article (an interesting debate between two veterans in the field) may give some pointers. Look at other resources you may be able to find on –line. Do you think they can co-exist? (6 Points)

   Use the following link and click on the PDF under this article

   http://www2.computer.org/portal/web/csdl/doi/10.1109/MC.2003.1204374

CS445: SOFTWARE ENGINEERING I

Homework-3: (due 10/01) Counts 5% towards the final grade

1. What do you understand by the term *system*? Given an example system other than computer or information system. This system should be such it interacts with at least one other system. For that system, indicate what are its *input, output and interface* (8 Points)

2. Assuming that you are asked to develop a system for managing the activities in a small library. (8 Points)
   a. List two possible actors in this application.
   b. Identify two possible Use Cases and
   c. Describe one of the use cases by stating the step-by-step dialogue between the user and the system.

3. Solve Problem # 4 under Chapter 2 in our text. Make sure to use <<extend>> relationship. (7 Points)

4. Solve Problem # 6 under Chapter 2 in our text. Make sure to indicate the multiplicities at both the ends of the relationships. (7 Points)

CS445: Software Engineering-1: Team Assignments

Both the teams will work on the same system development project that is described below. The assignments are limited to analysis and design (no implementation) and will be carried out in three stages. There are deliverables for each stage. Detailed information on the deliverables will be given at the beginning of each stage. The deliverables for stage 1 are listed in the next section. The three stages are:

1. Requirements Specification (Use Case Model). This is due on 10/08 and it counts 10% towards your final grade.

2. Use Case Analysis with interaction, package and class diagrams. This is due on 11/10 and it also counts 10% towards the final grade.

3. Design Specification with class and object diagrams, user interface and database designs. This is due on 12/10 and it counts 10% towards the final grade.

This project is about developing a system for managing PC support unit in SEMO. An outline of the project and the broad requirements are stated in the following pages. As the project sponsor, I will be providing additional information / clarifications concerning the requirements by way of answering your questions. (Remember to ask questions concerning the case study in class as and when needed.)

Case study for a three-stage group assignment:

PC Support in SEMO (A Case Study for Group Project)

Background:

In SEMO, the PC Support (PCS) is a unit within the Information Technology Services (ITS) that is responsible for providing all the PC related services. PCs are used in open-labs, closed labs, and also in the offices for use by the faculty and staff members. In all, some 4000 PCs are used in SEMO, distributed across the campus. (There are also some 3000 PCs belonging to students that are connected to the SEMO network from residence halls.) PCS is responsible for installing these PCs in SEMO (excluding those belonging to students), maintaining them (including updates and upgrades), keeping track of the individual hardware units (CPU box, monitor, key-board & mouse, and printers) and all software installed in all the lab machines. This would mean PCS has to keep track of the inventory (hardware and software but not consumables) related to PCs in SEMO. They also need to know where the PCs are located in order to provide timely support.

Users report problems related to PCs to Help Desk within ITS. If the folks in Help Desk are unable to resolve the problem, they log the problem into the system for the attention of the people in the technical / network units. This would mean they need a full fledged work order management system that has an interface to the existing help desk system (HDS). The HDS will create work orders for unresolved problems. It is also possible to create other work orders not going through the HDS (more on work order system later.).

PCS organize training sessions for faculty and staff members in the use of various applications. This would mean they need to keep track of the training information concerning every faculty and staff. PCS is also responsible for managing the use of open-labs. For this, they employ

students on a part-time basis to look after the open-labs. This would mean that they need to keep information on part-me students, to schedule their lab assignments, and to pay the students based on the actual hours they worked in the lab.

Organization:
Adam Spencer, a CS graduate from SEMO, directs the PCS. However, it is Clare Roberts, Manager User Services, who manages the day-to-day activities of PCS since Adam has several other responsibilities pertaining to ITS operations. Jessica Simon supervises the Help Desk unit where quite a few part-time students work. Rose Hogan (Network. Services), and Keith Joyce (Tech Services) are the other two major players in PCS. Currently, Clare also looks after the user training. There are also three application programmers who develop PC based applications for the user community. Another nine more people work in PCS. The workload is, however, rather heavy since PCS serve a user community close to 10,000.

PC Register
PC Asset Register System (ARS) will keep an inventory of PCs in SEMO. The register will keep comprehensive information such as serial numbers of the attached hardware units, major software products along with their versions, supplier and warranty expiry dates. When an attached unit for a PC is changed, its registration details should also be changed. The facility will include: adding new PCs to the system, recording location changes and software changes, and retiring PCs. In addition, ARS should keep track of statuses of PCs (working / not working / requiring services by PCS). A unique asset number (tag number) has to be used for each PC. (The same number will be used in Help Desk System and Work Order System – described in the next section - so that all the maintenance history of each PC can be easily gathered.). ARS should print the maintenance history of a PC upon request. (The WOS will be used to gather the maintenance details.) Also useful to have is a warranty expiry report that lists the PCs whose warranties expire within the next 3 months. It may be a good idea to maintain a list (a table) of all buildings and rooms on campus so that when one enters the location for a PC, the location code can be validated using this table. Also we will be able to get a list of PCs in a specific room or lab.

Work Order:
Keith reviews the list of work orders sent from the Helpdesk System (HDS) and selects a few at a time for resolution. Keith will review the problem details, mark its priority and enter the tasks on the work order forms provided by WOS, and assign these tasks to individuals in both TS and NG. These technicians periodically check their work assignments and plan their activities. After attending to the problems, they record on WOS the results and the actions they took in resolving the problems. If the problem is resolved, the work order will be closed (and the corresponding problem on HDS also will be closed). Keith needs a reporting facility that shows, for a given period, the number of work orders received, resolved, and still open. (An interface with HDS is implied here.)

Training Information Management
The Training Information Management System (TIMS) is an extension that keeps track of the training programs faculty and staff undergo. The scope of the system is rather limited. As and when a training course is concluded, the course organizer will send to Adam the course title, duration, and the list of people attended. Adam's secretary enters this information. Adam wants to periodically (once a semester) prepare the training profile for each person by department and send them to the chair /unit heads for their use

Student Worker Management

In the Student Worker Management (SWMS) sub-system, Clare will have a facility to register the student workers. She will give them the access rights to enter the hours they work and also to see their schedules. Clare also will allocate work to the various students on SWMS showing their Labs (listed in ARS) and timings. Since the pay rates vary with the time of day, there is a need to show when a student worker actually started work and when she/he left the lab. The students can log their check-in and checkout time on the SWMS. Once in two weeks, Clare will run the pay calculation facility (on SWMS) and produce a note to the pay office listing the students' details (including the Student Id) and the amount they need to be paid. She will also indicate the respective cost centres (user departments). Clare needs facilities to produce summary of activities for the student workers and also summaries by Lab whenever needed (usually every semester).

Note: All users of the system should first be asked to log on to the system before they can be allowed to access any of the facilities. For any user, the system access authorization should be tailored to the task he / she carry out. Each group of users sees only the facilities they need to use.

Stage-1: Use Case Model: Assignment due date: 10/08/2009

Deliverables:

The deliverables for the first stage include the following:

1. Introduction (Context and Scope of your project))
2. Use Case Model (use case diagrams, use case descriptions in the standard format)
3. Glossary of terms
4. Supplementary Specification.
5. Tentative plan of work for the remaining (Design and Implementation) phases.

See the sample UCM we discussed in class. Make sure you use Rational Rose for preparing your Use Case diagram. Your group should submit a hard copy of the document and also a softcopy (on a returnable flash drive) containing the doc files and model files.

As your group discusses the case, write down all the questions you have to get clarifications in class.

Stage-2: Use Case Analysis: Assignment due date: 11/12/2009

Use Case Analysis

The deliverables for the second stage - Use Case Analysis - include the following:

0. Cover page (project title; authors) and Contents page
1. Introduction consisting of: a final (revised) use case diagram and a summary of the contents of this analysis document
2. Interaction diagrams (sequence **or** communication/collaboration) for the main flow and one or two alternate flows for each of the use cases. These are based on the UC descriptions (revised in some cases) from the previous stage.
3. View of participating classes (VOPC) for each use case realization.
4. For large classes, show their attributes and responsibilities (custom methods).
5. Provide a table showing the mapping of class versus analysis mechanisms that captures the quality of service (non-functional) requirements. Focus mainly on distribution for appropriate control classes, security and or persistence for entity classes.
6. Contribution List (each member should analyze one or two use cases

Note:

1. Explain what is presented in each section, making reference to the respective model diagrams, which may be placed in an appendix. This being a technical report, you need to make reference to the model diagrams in the main body of your report.

2. Provide a softcopy of all the model diagrams pertaining to the analysis phase; use Rational Rose.

3. You need to first create a use case realization diagram for each use case. (See the model diagrams under Logical View for the sample analysis model (use case realization) available on my web page.

4. Since there are several interaction diagrams, print hard copies for, in all, **four** sequence or collaboration/communication diagrams from different use cases (other than login or report generation)

Stage-III: Design Specification: Assignment due date: 12/10/2009

Design Specification

The deliverables for the final stage - Design - include the following:

0 Cover page and Contents page: project title; and authors on cover page.
1 Introduction: Summary of contents and major design decisions.
2 System Design: Choose an Architectural style of the system that shows the various sub-systems and packages (how you have decomposed your system)
3 Sub-system design:
    a. Provide one sample subsystem design using sequence and class diagrams for realizing an operation in a system interface.
    b. Suggest design/implementation mechanisms (off-the-shelf components) for handling persistence, security and distribution
4 Class (object) design:
    a. Provide a state diagram for any one dynamic object (consider the work-order object that goes through various stages from *creation* to *closed*.)
    b. Present the final class diagrams for the system and discuss any patterns used in the above
5 Database design: provide a normalized ERD for the entity classes
6 User interface design: include sample forms and reports (no need to code these)
7 Contribution list

Present all the above in the form of a report. Have a section for each of the eight items. Provide a brief narrative under each section and, in that, make reference to the diagrams that are placed in that section.

Provide a softcopy of all the model diagrams (using Rational Rose) pertaining to this deliverable. See the model diagrams under Logical View (see process model, the three layers, user interfaces), and Component view for the course registration system on zeppo (S – drive) under CS445 Rose Model.

Note:
This work should reflect contributions from every member. Attach a summary sheet that indicates each member's contribution to the project.

Name (PRINT):_____

CS445: Software Engineering - I: Test-1: September 10, 2009

Total Points: 30 Counts 5% towards the final grade Time: 30 Minutes

Q1: For each of the words / phrases on the left, enter the corresponding letter that answers the question on the right. (7 Points)

1.	Requirements Analysis	( )	(a)	Is the problem solved?
2.	System Design	( )	(b)	Are enhancements needed?
3.	Program Design	( )	(c)	Can the customer use the solution?
4.	Program Implementation	( )	(d)	What is the solution?
5.	Testing	( )	(e)	How is the solution constructed?
6.	Delivery	( )	(f)	What is the problem?
7.	Maintenance	( )	(g)	What are the mechanisms that best implement the solution?

Q2. Fill in the blanks using the list of software engineering concepts given below: activities, document, equipment, model, resources, tasks, time. (5 Points)

A system development project requires several _____, in each of which there are several _____ to be performed. work products are produced by applying _____ when carrying out these _____. The main _____ are: participants, _____, and _____. The main work products are: system (reality), _____ (computer simulation), and _____. We use the term _____ to refer to any abstraction of the system.

Q3. What is a system development life cycle? (2 Points)

Explanation of SDLC:

Q4. With respect to an ATM system, identify which of these statements are functional requirements, and which are nonfunctional requirements. Where the choice is not clear cut provide a reason for your choice *(4 Points)*

(i) A customer should be able to withdraw cash from his/her account (when there is enough balance in the account) using the ATM system

(ii) The ATM system must always be available

(iii) The ATM system must be written in Java.

(iv) The ATM system must be easy to use

Q5. For each of the words / phrases on the left, enter the corresponding letter that provides the matching explanation / meaning on the right. (4 Points)

1.	Inception in UP (Note: UP stands for Unified Process)	( )	(a)	Goes through all the development process workflows
2.	Elaboration	( )	(b)	Getting the product ready for customer use.
3.	Iteration in UP	( )	(c)	Software architecture is designed
4.	Transition in UP	( )	(d)	A life cycle phase in which the scope of the system is defined

Q6. Briefly describe how testing can be initiated (preparation of various test plans) well before implementation activities. Explain why this is desirable.
(Hint: Recall V-model) *(4 points)*

Q7. What is done in each of the four phases (shown in four quadrants) of the spiral model (4 Points)

1.

2.

3.

4.

Q8. (For two bonus points)

(i) State one weakness of the waterfall model

(ii) State one main advantage of an agile methodology

Name (PRINT):_____

CS445: Software Engineering-I: Test-2: 09/24/2009

Total Points: 30 Counts 5% towards the final gradeTime: 30 Minutes

Q.1. For each of the following indicate if it is True (√) or False: (X) (4 Points).

1. _____ UML is a standard programming language.

2. _____ A scenario is an instance of a use case

3. _____ Actors are always part of the system

4. _____ Two major functions of project management are: planning and control

Q.2. For the following, circle the most appropriate answer: (5 Points)

1. Which of the following (is) are UML dynamic (behavioral view) models?
   A) Interaction diagrams B) Activity Diagram
   C) Both A and B D) None of the above

2. In a use-case diagram, an actor represents:
   A) a human user B) an external hardware
   C) an external system D) any of the above

3. The symbol for a class is a:
   A) 3 dimensional box B) Rectangle with 1 or 3 sections
   C) folder D) Diamond

4. Concerning WBS, which of the following statements is incorrect?
   A) is an outcome oriented analysis of work involved in a project
   B) it serves as a foundation document in project management
   C) it defines the total scope of the project
   D) none of the above (all three are correct)

5. A stereotype is shown on a diagram as text surrounded by:
   A) { } B) [ ] C) << >> D) < >

Bonus question (for 2 points):
Q3. State briefly the significance of placing the user view model at the center, overlapping the other four model views.

Q.4. Use case Diagram *(8 Points)*

Identify and list all the actors and all the use-cases in the system requirements presented below. No need to draw the use case diagram.

ABC is a software company whose customers are organizations that usually outsource their software requirements. ABC has several software products which are used by its customers. Whenever new requirements or changes to existing systems come up, these customers request ABC to make the necessary changes to their system or provide additional systems for a negotiated fee. ABC is looking for a web-based system to manage such change requests received from its customers. The facilities required in this new Change Management System are described below.

Using the system, a customer will be able to request for system enhancements by filling in a form on-line and attaching files to provide the details of their requests. A Clerk in ABC, upon receiving a request, records the request into the system. During this process, the system gets a serial number from an existing Project Management System (not part of the intended system) used in ABC and assigns an Id number (change request number) to the request. An Assessor in ABC then examines the registered request and records, using the system, his / her assessment (work to be done and a fee) for implementing the requested changes. A Contract Manager in ABC verifies the request and the assessment and makes a formal proposal (using the system) for fulfilling the request. (Sometimes, the Contract Manger might ask the Assessor to revise a few things in the assessment before sending the proposal to the customer.) The system lets the customer and the Contract Manager negotiate the contract conditions (fees, scope of work etc.). Finally the customer notifies the Contract Manager the acceptance or rejection of the proposal. If it is rejected, the request is closed without any further action and simple archived.

The system lets the Contract Manager assign a Technical Coordinator in ABC to work out the implementation details for customer-approved projects. The Technical Coordinator makes a thorough analysis and records a solution plan for the request into the system. The Contract Manger checks this solution plan and, after review, sends the solution (through the system) to the customer for approval. Once the solution is accepted by the customer, the Technical Coordinator assigns a team to implement the solution. When the solution is implemented and the customer has accepted the changed system, the request is considered closed and archived.

<u>List of Actors</u>:

List of all Use-cases (give suitable names)

Q.5 In dealing with use cases, explain the uses of *<<extend>>* and *<<include>>* relationships. Give suitable examples to illustrate your answer *(4 Points)*

(i)<< extend>>

(ii) *<<include>>*

Q.6. Draw the class diagram for the following descriptions; make sure to use appropriate symbols and multiplicities for the relationships. Also, indicate the name of the relationship where asked for. (9 Points)

a) Part-time and Full-time students are two types of Students

Class diagram:

Name of the relationship:

b) A Paragraph is composed of a number of Sentences, which in turn are composed of a number of Words

Class Diagram:

Name of the relationship:

c) A Mouse (in the sense of a computer peripheral) is made up of two to four Buttons and a Wheel (used for scrolling)

Class Diagram:

Name of the relationship:

Software Engineering

disciplinary Commons

Section 5: Course Assessment

Both the courses were assessed for the achievement of stated learning outcomes. As SEMO now has an ABET accredited Computer Science program, the school follows the standard peer review process where a colleague reviews the entire course portfolios. In this, students' sample work and the instructor's evaluation are cross-checked. Two sample student assignments are attached. My own end- of- course reviews for the two courses (cs445Review and UI450Review) are attached**.** In these reviews, I also discuss students' feedback. In particular the students are asked to indicate, on a five-point scale, the extent to which they learned the various learning outcomes.

Another Commons member did visit SEMO to observe my class performance and to meet with my students. However, it was towards the end of semester and this did not represent a typical class session. Hence, there was no formal review from the observer.

Homeework-1: CS445 assignment

Q1. In this article Professor Fred brooks says that software disappoints because it is so different from tangible goods, that the solutions that make tangible goods meet our expectations cannot possibly apply. He talks about some essential difficulties and says that we cannot expect to see two fold rise in software development every two years. He says the main essential difficulties are of complexity, conformity, changeability, and invisibility. He says that programs will always be complex as they are made from client needs and specifications and require a lot of coding and have a large number of bits and states. They will always be faced with conformity as it is developed by different people. Changeability is another problem as due to advancement of technology and ever changing needs, a software must be ever changing or it becomes robust. Also the problem of invisibility is there as we cannot visualize the software in our mind as they are very complex and thus do not permit the mind to use some of its most powerful conceptual tools.

There are also accidental difficulties. All a high-level language can do is to furnish al the constructs that the programmer imagines in the abstract program. To be sure, the level of our thinking about data structures, data types, and operations is steadily rising, but at an ever-decreasing rate. At some point the elaboration of a high-level language creates a tool mastery burden that increases, not reduces, the intellectual task of the user who rarely uses the esoteric constructs.

He says that there are certain ways to improve the situation. Time-sharing has brought a major improvement in the productivity of programmers and in the quality of their products, although not so large as that brought by high-level languages. Integrated Program Development Environments attack the accidental difficulties that result from using individual programs together, by providing integrated libraries, unified file formats, and pipes and

filters. As a result, conceptual structures that in principle could always call, feed, and use one another can indeed easily do so in practice. Two separate ideas: Abstract data types and Hierarchical types remove yet another accidental difficulty from the process, allowing the designer to express the essence of the design without having to express large amounts of syntactic material that add no information content. Also he says that one of the most promising of the current technological efforts, and one that attacks the essence, not the accidents, of the software problem, is the development of approaches and tools for rapid prototyping of systems as prototyping is part of the iterative specification of requirements. (Grading: Above Average)

Q2. Two advantages of using a programming language as a sole notation throughout the development process are

a. One person cannot understand the whole project. Therefore if it is written in one language many people can understand different parts of the program and can then combine their knowledge to share it with others.
b. As client needs are constantly changing it is easier to make changes if only one notation (the language) is used which is understood by everyone involved.
   Two disadvantages are:
   a. Developers do not anticipate seldom occurring situations.
   b. Developers do not anticipate the user actively misusing the system.
   (Grading: Average)

Q3. Figure 1.1 of the text shows a diagrammatic representation of the process of development of a project. As seen from bottom upwards, a document, model and system are required to make a work product which comes in the first phase. Also comprising the first phase are the resources being used which comprise of people, time and equipment. A task is given to a team which creates a work product and whose development requires the use of resources. The task comes under an activity and there are many activities assigned to different teams. All the activities together form the main project. (Grading: Good)

Q4. When developing an aircraft or a bridge you have all the specifications beforehand i.e. how much material would be needed, how much labor would be needed, what are the dimensions of physical resources which are going to be used. Also where physical resources have to be used only physical work is mainly used i.e. the labor force and that is person to person controlled. Whereas in the software development you have to take care of every person in the world, how everyone thinks differently from others. Due to these ever changing needs we need to take into consideration all the special cases related to the mind. Also technological advancements in computer related fields are faster thereby the ever changing demands. Construction is not as major changing because you have to manufacture and assemble which remains same almost throughout decades.

(Grading: Rather Poor)

Homework-2

15-2: The IEEE standard model is partitioned into three parts, the development, management, and integral processes. The processes listed are shown to have relationships with each other and no particular order/direction is given (although a loose idea can be derived by analyzing the model). The specific processes that have been omitted from the waterfall model are Maintenance, Retirement, Project Monitoring and Control, S/W Quality Management, Configuration Management, Documentation Development, and Training. (Good)

1-6: The only functional requirement is: The TicketDistributor must enable a traveler to buy weekly passes. The other requirements – that it must be written in Java, that it must be easy to use, that it always be available, and what it should do when it fails – are nonfunctional requirements. (Good)

1-7: The only of these decisions that would be made during requirement elicitation would be that the TicketDistributor provides the traveler with online help. Defining particular subsystems and specific hardware would occur during the system design phase. (Above Average)

16-1: The Software Life Cycle defines the phases and parts of a project, moving from its conception towards its eventual death/replacement. A methodology defines the tools and methods that are used to make a project move forward and when their application is appropriate.

XP stands for eXtreme Programming and is a time efficient methodology. It centers on making use of skilled programmers to complete actual code for a project. This is accomplished with minimal project design time, where details of the implementation are handled on-the-fly. It should be noted that novice programmers that attempt this type of software development will likely hit snags and possibly dead ends, as their foresight into what should work and what may go wrong is less developed. Also, development in XP focuses on completing rather small chunks at a time, such that timelines for tasks are over the course of a few days – any task longer than this should be broken down into smaller chunks. One drawback of XP is that the created code is rarely as reusable as code that was written to implement a well-designed and developed module. One benefit of XP is that the speed at which the system is implemented is much quicker, which allows the product – albeit maybe only an alpha/incomplete version – to be in the hands of the user much quicker. This assists in testing and ensures that the project moves towards what the user wants, so long as the user provides good feedback and the programmers take this into consideration.

XP and UP have their own strengths and weaknesses. XP provides quick response and delivery. If the product is less than what is desired, improvements and additions can be added quickly. The drawbacks are that novice programmers will likely not foresee critical issues, which can cause great time loss in reimplementation or workarounds. It is likely that the pieces of the system will be somewhat rough and only fit with the other pieces of that system

(meaning reuse of code can require about the same amount of time to rework pieces as it would take to rewrite the code for the particular situation). UP, on the other hand, focuses much more on planning and design so that implementation goes smooth with few to no unforeseen issues. By working with the user during this design process, approval for the system can be gained, but it is possible that the user will not be aware of desired features and reworking of the system may need to occur. The pieces of the system can be well-planned so that they can be reused. This also allows tasks to be divided across many people, including novice programmers that need only ensure that their implementation match the design.

It would be difficult to efficiently implement both XP and UP. Unless a system can clearly be split into distinct parts early on where one part can be better implemented with one methodology and the other part by the other; or, if a system were to be developed using both methodologies by two distinct groups, with only occasional meetings between the groups, which could lead to better solutions between the two that could then be combined into a single working product. In some special cases such as these, use of both methodologies may be appropriate. However, trying to implement both would require the use more members, as having developers switch back and forth between the two ways of going about the system development could cause loss of focus that would be present if a single plan were being followed. The additional members would increase the cost of the project.

(Good)

Course Assessment Report

CS445 Software Engineering I (3)
Fall 2009
Ken Surendran

Catalog Description:

Provides an in depth understanding of the principles and techniques used in software engineering. Apply techniques and tools to produce the system artifacts pertaining to requirements, analysis and design. Prerequisite: Senior standing, permission of the department, and a minimum grade of 'C' in CS300 and CS315.

Grades Distribution:

A	B	C	D	X or F	W	I	Total
7	0	0	0	0	0	0	7

Modifications Made to the Course:

1. Allocated more sessions on object design (removed detailed discussions on a sub-system related topic - Security).
2. To provide greater focus on design, two of the workflows (project management and quality assurance) will be addressed in the capstone Experience course (UI450 that is offered in the following semester).

3. Adjusted the scope of the assignment to suit the team sizes

4. Two more homework were added to emphasize design.

Course Outcome Assessment:

1. Discuss the challenges in software engineering and examine the paradigms and methodologies used for developing quality software products.

Assessment Item 1: Homework-1
Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments: 2, 0, 1, 0

Assessment Item 2: Test 1 (Q 6 & 7)
EAMU: 1, 1, 1, 0

2. Prepare requirements specification and analysis models, applying system analysis techniques

Assessment Item 1: Assignments-1&2
Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments: 2, 0, 0, 0
Assessment Item 2: Mid-term Exam (Questions 4, 5, 6)
EAMU: 1, 3, 2, 0

Assessment Item 3: Final Exam (Part-I)
EAMU: 2, 1, 0, 0

3. Prepare architectural model, and detailed design models, which include user interface, database and system designs.

Assessment Item 1: Assignment-3
Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments: 2, 0, 0, 0

Assessment Item 2: Final Exam (Part-II & III)
EAMU: 2, 1, 0, 0

4. Discuss project management and communications management issues in software development (Covered in UI450).

Assessment Item 1:
Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments: 0, 0, 0, 0

Assessment Item 2:
EAMU: (0, 0, 0, 0)

5. Discuss the various testing concepts for establishing quality assurance (covered in UI450)

Assessment Item 1:
Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments: 0, 0, 0, 0

Assessment Item 2:
EAMU: (0, 0, 0, 0)

Student Feedback:

It is clear that the students have developed skills in preparing use-case diagrams, use case descriptions, sequence diagrams, (based on these) the class diagrams, and state diagrams. Their level of understanding in system architecture appears above average. Compared to previous group they seem to have a better understanding on system architecture and design patterns.

Reflection:
In this course, the students see the value of earlier courses – such as programming, database. There was just one student from CIS; only one student had not taken Database yet. It would be useful to allocate some more time for design principles.

Proposed Actions for Course Improvement:

1. Allocate one more sessions on design principles and reduce one for analysis

2. Adding homework has helped; so retain the number of homework to five. Introduce a few quizzes, in addition to tests

3. Let the students reverse engineer the programs they had coded in earlier courses as they learn design principles.

Course Assessment Report

UI450 Capstone Experience (3)
Spring 2010
Ken Surendran

Catalog Description:

Generic UI450: Group-based solution of open-ended problems based on "real world" scenarios requiring interaction among students with diverse training.
Specific to UI450-02: Apply the principles, techniques, processes and tools for developing a product such as quality software with all documentation for a client. Work in a team to complete a client sponsored project from inception to handover. Prerequisites: A minimum grade of 'C' in CS445, Senior standing in CS Major, and passage of the 75 hr Writing Test.

Grades Distribution:

A	B	C	D	X or F	W	I	Total
10	0	0	0	0	0	0	10

Modifications Made to the Course:

5. Due to administrative reasons, the UI450 had three students from MIS and they were given a separate project.

6. The project team size was three or four (instead of four or five)

7. Projects for the CS students were all client sponsored while MIS had a SEMO sponsor.

Course Outcome Assessment:

6. Discuss project management and communications management issues in software development.
   Assessment Item 1: Exam -2
   Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments – 1, 1, 1, 1

7. Discuss the various testing concepts for establishing quality assurance
   Assessment Item 1: Exam -1
   Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments – 1, 1, 2, 1

8. Apply the knowledge from their major discipline to develop, working in teams, a software product for a client together with all the documentation and other system artifacts.
   Assessment Item 1: Software and Final Project report
   Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments – 2, 1, 1, 0

9. Orally present the intermediate system artifacts (generated during analysis and design) for review and evaluation.

   Assessment Item 1: Project Presentation and Demo
   Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments – 2, 1, 0, 0
   Assessment Item 1: Project Presentation – Intermediary artifacts
   sample assignments – 5, 4, 0, 0 artifacts: 3, 0, 0, 0.

10. Create analysis and design documents pertaining to the system they are developing

    Assessment Item 1: Project Report
    Exemplary, Adequate, Minimal, Unsatisfactory (EAMU): number of each from sample assignments – 1, 1, 1, 0

Student Feedback:
The students seem to have had a positive experience in this course since they were able to complete the projects to their respective client's satisfaction. In terms of learning outcomes, their ratings on a five-point scale are:
CO-1: 4.2; CO-2: 3.2: CO-3: 4.3; CO-4: 3.5 and CO-5: 4.3.
This seems to be their first exposure to testing concepts, in particular the idea of test cases and also preparing test cases for coding.

Reflection:

The external and internal evaluators rated all the projects favorably. The clients expressed satisfaction. There were three 3-member teams. Smaller teams may have difficulties in meeting the schedules during the early part of the project since two many things are going on in parallel. The minimum team size could be kept to four (as it used to be in the past). Peer assessment seems to work better when the class size is 20+.

Proposed Actions for Course Improvement:

1. Need to discuss more on testing.
2. Include an assignment on writing test cases for problems they did in earlier courses.
3. Stress the importance of intermediary artifacts and review process.
4. Keep the team size to four to balance the workload.