

# CS 451 Software Engineering Survey

[ [Home](#) | [Class Presentations](#) | [Presentation Handouts](#) | [Final Study Guide](#) ]

## SYLLABUS

<b>Title:</b>	CS 451/551 Software Engineering Survey
<b>Instructor:</b>	Edwin C. Foudriat
<b>e-mail:</b>	<a href="mailto:foudr_e@cs.odu.edu">foudr_e@cs.odu.edu</a>
<b>Office Phone:</b>	(757) 683-6061 or CS Office (757)683-3915
<b>Home Phone:</b>	(757)898-4485
<b>Home Page:</b>	<a href="http://www.cs.odu.edu/~foudr_e/homepage.htm">http://www.cs.odu.edu/~foudr_e/homepage.htm</a>
<b>Lectures:</b>	Tuesday 10:00 am. - 12.45 pm.
<b>Office:</b>	ODU Ed 230-D
<b>Class Room:</b>	Comm Bldg Rm 201
<b>Office Hours:</b>	Tuesday 9:00 am -10:00 am & 2:00 pm - 3:30 pm. Thursday morning 9:00 am - 1:00 pm or by appointment
<b>Textbook:</b>	Software Engineering - A Practitioner's Approach; 5th Edition Roger S. Pressman; ISBN 0-07-365578-3
<b>Reference:</b>	Software Engineering, Ian Sommerville, 5th ed. 1995 ISBN 0-201-42765-6.

## Contents

<b>General Course Information</b>	<b>2</b>
Objectives:	2
Attendance:	2
Late/Missing Work:	2
Cheating:	2
Grading:	2
<b>Assignment Execution Information</b>	<b>3</b>
Problem Assignments:	3
Programming Assignments:	3
Assignment Submittal:	3
<b>Lecture Presentation Material</b>	<b>3</b>
Lecture Material:	3
Class General Information:	4
<b>Class Schedule</b>	<b>5</b>

# General Course Information

## Objectives:

Upon successful completion of this course, the student will have gained an increased appreciation of Software Engineering. The course will examine the software process from a number of aspects. We start by viewing Software Engineering from a management perspective. This provides an overview of its problem, process and product. With this understanding, we then examine various software engineering methods both conventional and object-oriented. We include analysis concepts, the design and development process, and software testing and metrics. With an understanding of these methods, the student will gain a much better appreciation of the problems which quality software development presents. We examine the superior features provided by object engineering, show how use of these methods prove to be extremely effective in software product development. These features include the concepts of modularity, information hiding, interfacing, etc., to support and enhance the software development process. By examining details of various aspects of Software Engineering, the student will better understand both the virtues and difficulties of the working in a real-world software development environment - where the goal is to provide quality product..

## Attendance:

Attendance at classes is not required, but students are responsible for all material covered and announcements made in class. Class notes, handouts and related information will be available by linking to the material in section Lecture Presentation Material. If a student has to miss class, be sure to get announcements, special information, etc., from another class member and to check with the General Course Information or General Course Information Handout file for the changing information.

## Late/Missing Work:

If a course assignment is not turned in on the due date or tests are missed, they will receive a grade of zero unless prior permission has been given by the instructor. Please note that the due dates for assignments are designed to allow more than enough time to complete the assignment. Occasional hardware failures are a fact of life in the computing field. Unless unusually severe or prolonged, such problems are not cause for a due date extension. Similarly, the 48 hours before an assignment due date there is invariably heavy competition among students for the available computing resources. Such competition is never an acceptable reason for requesting an extension. Budget your time accordingly!

## Cheating:

Everything turned in for grading in this course must be *your* work. The instructor reserves the right to question a student orally or in writing and to use his evaluation of the student's understanding of the assignment and of the submitted solution as evidence of cheating. Violations will be reported to the Honor Council for consideration for punitive action. However, much of effective Software Engineering involves effective communications. Therefore, it is entirely appropriate to seek and give assistance on both procedural (sending e-mail, access to resources, etc.) and conceptual matters (general problem approach, outside resources which may aid in understanding and solution, etc.). With regard to conceptual matters, I would encourage students to bring such interactions to the class room where they can be best shared by all. If there is any question on whether a particular behavior is appropriate, the student is encouraged to seek guidance from the instructor.

## Grading:

Your grade will be based upon your performance as weighted in the the following areas:

Last Modified 1/17/02

Midterm	30%
Project Assignments	35%
Final Exam	35%

## Assignment Execution Information

### Problem Assignments:

Most problem assignments from the text will be in the form of written discussion related to the question(s). Since the discussion may include graphical and table information where appropriate, the material can best be prepared using a modern word processor. Since there exists a wide variety of systems which may be available to students some of which may not be available to the instructor, submission of written discussion assignments can cause problems. The preferred method is to translate the submitted product to Adobe Reader 3.0 format (\*.pdf). As an alternative students may use Rich Text Format (\*.rtf in a Window's file designation format). Most modern word processors can deal with files in \*.rtf format. However, in some translations, it tends to muddle tables and graphs. Check you translated product before submittal. If problems arise, the instructor and student will have to deal with submittals on an individual basis.

### Programming Assignments:

Some assignments may require some code development and programming. Programming should be done in C++. It is acceptable to develop code for the assignments using any available compiler. Those suggested include Borland C++, Microsoft Visual C++, or Unix GNU compilers. I will not intend to attempt to execute the program, but will code read it to determine if its design meets the intent of the problem. I may compile it determine your programming skills.

The material submitted with the assignment should include all code developed by the student including header and source code file(s). The student should design the header and source files using acceptable software engineering standards, input, test files and output where applicable.

### Assignment Submittal:

Assignment should be submitted via e-mail. The material submitted should be attached in appropriate file format. Assignment timing will be judged based upon the submitted e-mail time.

## Lecture Presentation Material

### Lecture Material:

The lecture presentation material is available in two formats. First the material can be viewed by using a web browser such a Netscape(Version 3.0 or later) or Microsoft Internet Explorer. The material is formatted as Adobe (x.pdf) files. They can be read using Adobe Acrobat Reader (Version 3.0 or later) which is available free from the [adobe.com](http://adobe.com) web site. AdobeAcrobat Reader can be set up as a helper application for your web browser. The student can copy and print any of the handout material.

To assist students in taking notes in class, another format for the lecture material is available. It is a handout format where slides are printed in reduced size. There is sufficient room on each sheet to place notes as you see fit. To use, you should copy and print the slide or handout material. Printing can be done directly from Adobe Reader 3.0 to any printer which you have online.

The basic material outline is shown below and can be accessed via browser:

[General Course Information](#)

[Course Introduction](#)

[Chapter 1 - 2 Software Eng. Intro.](#)

[Chapter 3 - 4 Management Concepts](#)

[Course Intro Handout](#)

[Chapter 1 - 2 Handout](#)

[Chapter 3 - 4 Handout](#)

Last Modified 2/11/02

[Chapter 5 - 9 Project Planning & Scheduling](#)  
[Chapter 10 Requirements & Specifications](#)  
[Chapter 11 Req. & Spec Analysis](#)  
[Chapter 12 Req. & Spec Analysis \(cont'd\)](#)

[Chapter 5 - 9 Handouts](#)  
[Chapter 10 Handouts](#)  
[Chapter 11 Handouts](#)  
[Chapter 12 Handouts](#)

[Chapter 15 Interface Design](#)  
[Chapter 13&14 Design](#)  
[Chapter 20-22 Object-oriented Dev](#)  
[Chapter 17-19 Testing](#)

[Chapter 15 Handouts](#)  
[Chapter 13&14 Handouts](#)  
[Chapter 20-22 Handouts](#)  
[Chapter 17-19 Handouts](#)

The material being presented and that available in the course is subject to change during the semester. I will try to call your attention to changes as they occur. However, each slide, each handout file, and each page of this document will have a "Last modified" indication on it so that you can tell if the file which you have is the latest version.

**Class General Information:**

During some class meetings, usually at the beginning of the class period, general class information may be presented. This material will be available for review by accessing the General Course Information file. The material in this file will also be dated and the file will have an index as first page so that the student can more easily find information related to a particular class period.

*Note: Much of the above material will not change in either content or format, although links will be established for presentation material as it is completed and placed in the course file(s). The material starting on page 5 will change as problem assignments are made, completed, graded and discussed.*

# Class Schedule

CS 451/551 Software Engineering Survey Class Schedule			
Date(s)*	Subject-Lecture	Text Section(s)	Important Features
Jan. 15	Course Info. & Intro. Software Eng. Intro.	Chapter 1 & 2	Course Req.; Syllabus; Activity SE Product & Process
Jan. 22	Software Eng. Intro. (con't) Management Concepts	Chapter 2, 3-5	SE Process & Problem; Management Concepts, SE Metrics Discuss prob. 1.8
Jan. 22	Problem Assignment 1 - problems 2.2; 2.9; 3.11; 4.3; 4.9. CS 551 students 4.10		
Jan. 22	Last day to register, add or drop course. Last day to audit or pass/fail or other options		
Jan. 29	Management Planning & Scheduling	Chapters 4 - 9	SE Metrics, Planning, Risk An. & Mitigation, Scheduling, SQA & Conf. Management
Feb. 5	Last day to withdraw and receive half-tuition refund or credit		
Feb. 5	SE Management Activities (Completed)	Chapters 1 - 9	Management Activities Summary
Feb. 12	Problem development	Chapter 10	Systems Engineering
Feb. 12	Problem assignment 1 due. Problem assignment 2 - problem 5.4 (Include a schedule over a year period -make reasonable assumptions as to personnel and time needed)		
Feb. 19	Requirements	Chapter 10 - 11 & Chapters 4 - 5 of Sommerville	Requirements scenarios.
Feb. 26	Req. & Specifications	Chapter 11-12 & Chapter 7 - 8 of Sommerville	Definition quality, functional & non- funct. requirements. formal and algebraic specifications
Feb. 26	Problem assignment 2 due. Problem assignment 3 - 9.2, 10.8**, 11.4, 11.12		
Mar. 5	Midterm Review - Covers material thru Feb. 26		
Mar. 11 - 16	Spring holiday - no classes		
March 14	Midterm due if you need an evaluation before withdrawal. Otherwise March 18		
March 19	Graphical Interfaces	Chapter 15	Users & interfacing with them
Mar. 19	Last day for withdrawing from class		
Mar. 26	Design	Chapter 13 & 14 & 16	Design Principles, Architectural design 7 Briefly interface design
Mar. 26	Problem assignment 3 due. See revised problem 10.8**.		
Apr. 2	Object-oriented concepts	Chapter 20 - 22	Object-oriented principles
Apr. 2	Problem assignment 4 - 14.5; 20.4; 20.5 (use Fig . 21.7); 22.2; 17.11; 18.5		
Apr. 9	Object-oriented design & analysis	Chapter 20 - 22	Features of OOD which support SE
Apr. 16	Software Testing	Chapter 17 - 18	Testing methods, component, interface, integration; strategies
Apr. 23	Technical Software Metrics & Reliability	Chapters 18 - 19	Metrics, specs., statistical measures, prog. for reliability, fault tolerance & recovery
April 29	Problem 4 Assignment due		
Apr. 30	Final Review - Problem 4 discussed. Take home final available		
May 7	Final Exam Period - take home final due		
* Dates for class presentation of a chapter's material are approximate. You should adjust your reading and study of the assigned material to correspond to that actually presented & planned during a class period.			
** See problem restatement and submittal on page 8			

Last Modified 4/27/02

## Problem Assignment 1

2.2 Is there ever a case when the generic phases of the software engineering process don't apply? If so describe it.

Ans: Generic Phases

1. What is the problem? - Problem definitions (Requirements & specifications)
2. What entity(s) is needed to solve the problem? (Preliminary design and analysis)
3. How will the entity and the solution be realized? (Detailed design)
4. How will the entity be constructed? (Detailed design, development & coding)
5. What approach will be used to uncover errors made in the design and construction? (Test and evaluation)
6. How will the entity be supported over the long term when corrections adaptations and enhancements exist or are requested by the user? (Maintenance & upgrade)

Each of the phases could be minimal depending upon the nature of the problem and the skill and experience of the software engineering team involved. Some of the design and development phases may be minimal or shortened. For example, if the solution is to buy, then design and development would be the responsibility of the contractor. However, you may have activity in the stage to make sure the contractor is doing it right. Obviously, some of the latter phases would not exist if the project were cancelled. This could be especially applicable to step 6 if after step 5, the software was recalled and cancelled. Also, the first phases, 1 & possibly 2 could be omitted if they existed from another program. However, here great care should be taken to understand that the material from 1 & 2 are completely applicable which negates the idea that the steps don't apply.

2.9 As you move outward along the process flow path of the spiral model, what can you say about the software that is being developed or maintained?

First, at each level of the spiral the actual software which represents the solution is considerably more complex. In some cases, the actual software in the concept development phase does not exist. However, there may be significant peripheral software in this phase, software for modeling and analysis which may be highly complex. Second, the bulk of software in the latter phases should be more reliable, better documented and more thoroughly tested. This may even to the point where the software is versioned so that new software is designed and developed to specifically enhance or apply to a new version and it does not "replace" the software already in existence. Third, in the latter spiral phases, some of the initial software may no longer exist, may have been abandoned, or may no longer be applicable to the product because it has not been maintained.

3.11 You have been asked to develop a small application that analyzes each course offered by a university and reports the average grade obtained in the course (for a given term). Write a statement of the scope that bounds the problems

One method for preliminarily scoping a problem is to state its three fundamental aspects - Problem Description, Approach and Justification. In this case, since justification is trivial - i.e., you have been asked, it will be omitted. Second, parts of the problem definition is already given. Hence, the question can be answered by an expansion of the Description and a detailed description of the Approach.

Description Enhancement: Some grades such as "W" and "I" do not figure in the average. However, it will be assumed that the "W" count will be supplied as an enhancement. In addition, some courses can be taken as Pass/Fail. The user should be requested to provide a decision as to how P/F grades are averaged. (It will be assumed in the initial design that they will not be counted in the average.)

Approach:

Start by determining the student-course information which is available from the registrar (organization which has instituted the request). Its format and method of access should be made available by the requesters. A couple of possibilities are the machine readable grade sheets submitted by the instructor or the sheets have been processed into an electronic database. Assuming it is a database format, develop an query which will access course's grade sheet, extracting from it each student's grade. Note the query does not required either student's name or ID so the query software can be developed and used with limited database access permissions. The semester's catalog (also assumed available

Last Modified 2/20/02

electronically) will be scanned and each course section, recorded by appropriate ID as needed to interrogate the registrar's database. The query result will then be scanned, passed to function with will use the grade in a switch based upon grade type. This will allow maximum flexibility to enhance the software to handle additional grade types. The "W" grade will be ignored, the grades "A" - "F" along with "+" and "-" will be weighted according to the universities GPA weighting ,summed, counted as a grade count and at the end averaged. The pass/fail grades, "P" and "FF", will be attendance counted. Other grade types will be handled as required by the requesters. The results of each query with appropriate course ID will be stored in a database with restricted availability to the organization requesting the information. They should be able to alter the database availability as needed.

4.3 Describe the difference between process and project metrics in your own word.

Process metrics are designed to measure the quality of the process, i.e., how well the elements of the process are able to produce a viable product. The project metrics are designed to measure the character and quality of the product. For example, KLOC, a basic metric, may be used in measures for both metrics. As a process metric, the metric:

$$\text{Process\_code\_efficiency} = \text{KLOC}/(\text{process\_man\_hours})$$

This process may be compared to other processes to determine whether this process organization or group was more successful than previous ones.

As a project weighting metric, the metric:

$$\text{Expected defects} = \text{KLOC} * (\text{error\_found}) * \text{F}(\text{prior\_defect\_analysis})$$

where prior\_defect\_analysis is:

$$\text{prior\_defect\_analysis} = \text{Average}(\text{defects}/\text{KLOC} * \text{error\_found}) \text{ over previous projects}$$

where defects are those problems found after release and error\_found are those problems found and fixed before release.

One could provide a large number of examples to emphasis the difference.

4.9 Team A found 341 error during the software engineering process prior to release. Team B found 184 errors. What additional measures would have to be made for projects A and B to determine which of the teams eliminated errors more efficiently? What metrics would you propose to help in making the determination? What historical data might be useful?

The first and foremost measure LOC. A close second would be time, man\_hours spent in testing. A second measure might be the complexity of the code - objects, interfaces, etc. - suitable function points (FP) or similar. Alternatively or in addition, one could use some form of code weighting. Further, one could use error types or identify sources or causes of errors which some kind of knowledge as to the difficulty of finding different types of errors.

In some respects, we have answered the last question in that a weighting based upon complexity or error types would require some form of past history.

4.10 Present an argument against lines of code as a measure for software productivity. Will your case hold up when dozens or hundreds of projects is considered?.

One of the major arguments against lines of code is that the lines depend upon the language used. It is well known that earlier languages like Fortran and Cobol do not have the sophistication of later languages like C++, Java, Ada, etc. A second major argument is that some algorithms take significantly more code than others but may not reflect the difficulty of coding. A third situation depends upon whether the coding utilizes libraries and/or complex interfaces like component object models, etc. In this case the number of lines decreases significantly but it is much more difficult to write a correct line of code. Finally, lines of code is may be used to relate to errors. The sophistication of the compiler in detecting errors and alerting the programer of potential errors is important parameter when errors are related to lines of code.

In all of the situations noted above, the arguments would be valid in a large number of projects.

## CS 451/551

### Take Home Midterm Examination

March 5, 2002

Answers must be submitted by:

1. If you are considering dropping the course by the drop date March 19, 2002, then you need to submit your exam by end of day March 14, 2002. I will look at the exam. While I will not give you an exact grade and grade range by March 19, 2002, we will be able to discuss your situation so the you can make a drop or stay decision.
2. Otherwise the exam will be due by the end of the day March 18, 2002.

You have been asked to design and build an automated cookbook to be incorporated with a microwave oven. You need to complete the information necessary to initiate the project. The following information is needed for this analysis

1. A project description. The project description should discuss the automated cookbook indicating its functionality, its interface and control of the microwave oven and its interface and control with the chef. The oven interface should allow a single recipe being prepared to control heat and time settings; the chef may need to interact with the dish being prepared. The chef interface should allow for inserting and editing of recipes and a file system for creating, storing and finding them in the cookbook memory.
2. Requirement and specifications. The software for the project should be scoped with the major software functions (or objects) defined and decomposed into simpler functions (or derived objects). To enable scoping, the analysis should be done using either viewpoint, service operation and service actions or by decomposition with flow and control graphs. Using this analysis, an outline specification in the format of function or object points so that code can be designed and tested should be written.
3. Process analysis. Based on the software requirements and specifications, the project process needs should be estimated. The basis for this estimate should be LOC. Assume that 1000 LOC per person-month and \$4000 per person-month bases for the estimate. Also assume that you are assigned to do the job. Show an appropriate division of your time to accomplish the task including the fact that during portions of the task only part of your time may be needed..

Assume that the microwave oven can be attached to a laptop or pc through a COM port and that an electronic switch can be installed into the oven to enable the controls that you select. The computer has a version of Windows installed so a general file system and editor are available.

You should provide your solution in the format of an attached e-mail. I suggest using \*.rtf format using Windows. If possible embed any charts and figure within the document instead of linking them.

*The work you submit must be your own.*

A downloadable copy of the midterm can be found in:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdhtmlfolder/midterm/midtermexam.rtf](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdhtmlfolder/midterm/midtermexam.rtf)

A copy of the midterm exam solution can be found in:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdhtmlfolder/midterm/midtermsolution.rtf](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdhtmlfolder/midterm/midtermsolution.rtf)

A copy of the midterm review material can be found as:

[Slides](#)

[Handout](#)

### **Problem assignment 3 revision for problem 10.8**

Problem 10.8 Your instructor will distribute a high-level description of a computer-based system or product.

- a. Develop a set of questions that you should ask as the interface design engineer.
- b. In class we will compare your questions and see how we can answer some of them

The product is Figure 15.2 found on page 412. Your questions should be directed toward potential purchasers of the SafeHome system; their evaluation of the user interface, its good and bad features.

9.2 Discuss the reasons for baselines in your own words.

The baseline for a software system forms the foundation for its maintenance and upgrade. The baseline should be a proven system with most of the features of the original specifications working properly. It should be well documented with operational manual (and/or other forms of documentation) as needed. After baselining, any changes or upgrades to be software engineered, i.e., go through the project steps; prior, changes can be made with much less formality as long as the specifications and documentation are changed to reflect the changes. Also as you upgrade, you may be able to measure how useful the upgrades were.

Last Modified 4/12/02

10.8 Your instructor will distribute a high-level description of a computer-based system or product.

A. Develop a set of question that you should ask as a systems engineer.

B. Propose at least two different allocations for the system base on answers to your questions.

C. In class, compare you allocation to those of fellow students.

*We will delay this question until after considering of the material in chapter 15 and will direct the question to an interface. See linkspage or software engineering syllabus for update.*

11.4 Through this chapter we refer to the “customer”. Describe the “customer” for information system developers, for builders of computer-based products, for system builders. Be careful here, that may be more to this problem than you first imagine.

Customer for information system developers - Information system developers direct their product to gather and solve organization problems. Typical would be a business which would collect information on its product, sales, costs, etc.

Customer for builders of computer-based products - These customers would develop computer software for sale to users. Typical would be a company developing a line of software to sell to the general public. However, computer-based product developer span a wide range of activity since they may actually be engaged in development of products used by information system customers.

Customers for system builders - The system here implies the basic software upon which user software interfaces. For example, operating system software, library software such as Standard Template Library. Hence, customers for system builders would be those who would use this basic software at the under pinning for their software.

When one begins to examine who the “customer” is for each condition, it appears that in every case is has a similar identity. If you are developing software to serve any one of the “customer” above you can be developing software they all can use

11.12 Try to identify software components of SafeHome that might be *reusable* in other products or systems. Attempt to categorize these components.

At this stage in the product development, it may be more feasible to look for components from other systems which may be applicable to this development. Also, we don't have any project history to go on, so answering the question positively can be a problem. As a second option, we could take the question at face value. I would consider *reuseability* extended to follow-on projects.

However, there is one important consideration which is related to *reuseability*; i.e., how can we design the project components (objects) so that some of them have code that can easily be adapted for to other components in the project.

List of software components:

1. Intrusion control object

A. Door sensor object (many employed)

B. Window sensor object (many employed)

C. Motion detection sensor object (many employed)

There can be considerable reuseability here. The action of all devices is to return a boolean state (*safe/unsafe*) when interrogated. Upon testing or receiving an *unsafe* state signal, the operation would be to 1) have the sensor tested for failure to reduce false alarms and 2) pass the results (*good/bad*) to the intrusion control object for action. There is considerable design commonality here for the other objects.

2. Gas control object.

A. Smoke sensor object (many employed)

B. CO sensor object (many employed)

C. Fire sensor object (many employed)

The same rational as in 1 above can be used here.

There may be considerable commonality between 1 & 2. Both exercise control over sensors. When receiving an *unsafe* alarm, 1) set a message to the sensor object requesting a self-test; 2) gather information on sensor type and location; and 3) based upon *unsafe* and the result of 1) and 2), construct and deliver a message to the alarm object.

We could continue this process for the other SafeHome components. However, you get the idea that we are now starting design (preliminary design).

I would make one additional observation. If you were connected with a company making safety products, you need to consider during the design and development how you could make these components as library objects. The basic library code would be developed to be common with an interface tailored to the particular object of that class.

A copy of the assignment can be found at the link:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdfhtmfolder/probassigns/assignmt3.rtf](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdfhtmfolder/probassigns/assignmt3.rtf)

### Final review material

The final review slides and handout can be accessed. The final review material has added to the midterm review material. It is available as separate slides and handouts.

[Slides](#)

[Handout](#)

Final Exam:

A copy of the final exam can be found thru the link:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdfhtmfolder/final/finalexam.rtf](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdfhtmfolder/final/finalexam.rtf)

## CS 451/551 Take Home Final Examination April 30,2002

**Answers must be submitted by May 7, 2002. Submission should be either in \*.rtf or \*.doc format. I will return graded exams. If you wish that your course grade be included in the returned exam, you must tell me. This can be done either by request when the exam is e-mailed or by separate e-mail request. The reason for this requirement is that e-mail is not secure so you need to anticipate the file and be responsible for it's processing.**

*The work you submit must be your own.*

Problem 1. (20)

- a. Develop the data architecture and entity/relationship diagram for software that replaces a Rolodex.
- b. Define the steps and develop the state transition diagram for a user's search action.

Problem 2. (15)

The following code describes a Safe-Sort function in C++. Develop a white-box test plan for Path and Condition Testing the code. You can assume in your test plan that your language system has debug capability.

```
void Safe_Sort(int* X, int N, bool& Err)
{
    // Allocate memory for array copy
    int* Copy = new int[N];
    // Make a copy of the array being sorted
    for (int n = 0; n < N; n++)
        Copy[n] = X[n];
    try {
        // Code here to sort the array X
        Err = false;
        // Now test the array
        for (int i = 0; i < N-1; i++)
```

Last Modified 4/27/02

```

        if (X[i] > X[i+1])
            throw Sort_error;
    } // end try block
catch (Sort_error)
// restore state and indicate to calling procedure
// that sort error has occurred
{
    for (int i = 0; i < N; i++)
        X[i] = Copy[i];
    Err = true;
}
// No exception in C++ caught. return dynamic memory
delete[] Copy;
} // end Safe_Sort

```

Problem 3. (15)

Suggest appropriate reliability metrics for the following software systems and suggest appropriate values for them. Give reasons for your choice and values.

- a. a system which monitors patients in a hospital intensive care ward.
- b. an automated vending machine control system
- c. a word processor
- d. a system to control braking in a car.

Problem 4. (20)

Consider the control panel and sensor objects as discussed as part of the *SafeHome* security system in the text. Model the two objects using the Card/Responsibility/Collaboration as noted in Figure 21.3. The consider the sensor object as a base object where actual sensors can inherit its features. Collaboration should use either the client/server or peer-to-peer model. Provide reasons for your choice. (See Sec. 22.2)

Problem 5. (15)

You have been given the task of *selling* formal specification techniques to a software development organization. Outline how you would go about explaining the advantages of formal specification and countering the reservations of the practicing software engineers.

Problem 6. (15)

A. In section 7.2.1, we present an example of the “communication overhead” that can occur when multiple people work on a software project. Develop a counter example that illustrates how engineers who are well-versed in good software engineering practices and use formal technical reviews can increase the production rate of the team. (See problem 7.4 for additional hints)

B. Apply the ideas of good communication to Component Object Models. How can a good software engineer improve the ability of others to understand and more easily use the services in a COM library.

A copy of the final exam solution can be found in:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdfhtmfolder/final/finalexamsolution.doc](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdfhtmfolder/final/finalexamsolution.doc)

Last Modified 4/27/02

Problem 14.5. Present two or three examples of applications for each of the architectural styles noted in Section 14.3.1.

Data centered architectures - the software activity for this architecture is centered about using, updating, adding, deleting or other modifying information in a database. Examples here would include:

1. A Student Record Application - information for a student transcript system would form the basis for such a system. However, one could augment the information relating to a student record with department requirement's information to provide an Advisor Assistant.
2. A Bank Account Application - information related to account(s) that a client has with a bank. The information could be provided online (with necessary security) so that the client could do banking online.
3. A Recipe File System Application - to reduce home paperwork, an application can store recipes.

Data flow architectures - the software activity is related to processing input data to extract useful information. The word *filter* fits this category.

1. A SETI (Search for Extra-Terrestrial Intelligence) Application - SETI is government program where large radio antennas scan the skies, recording radio wave information from the heavens. It is assumed that if other intelligence exists in the universe, that group will eventually begin transmitting radio signals with embedded intelligence. The received signals are digitized and sent to processors to examine each wave form.
2. A Simulator Application for Studying a System - In general simulators generate input which is provided to a computer model of the system under study. The simulator captures information concerning the system performance and outputs the information in format that the persons studying the system can ascertain its performance.
3. A Income Tax Processing Application - the user inputs the information related to the his income. The application then figures the tax and outputs the information in a format so that it can be directly submitted to the tax collection agency.

Call and return architectures - the software is related to request and response activity; i.e., your author illustrates it which the remote procedure call.

1. Client/Server Applications - this is a very general format for call & return; as a matter of fact it would be my opinion that Pressman should have included Client/Server as an architecture as opposed to call & return. However, one format for client/server applications is precisely one that call a server to request an interface action; upon receiving the request the server performs the action, returning a result.
2. SQL Applications - Many SQL scripts request information from a database through the activity where databases are searched for the requested information based upon keys; the data is found, formatted properly and returned.

Object-oriented architectures - the software is structured based upon objects which embed data and processing within an object structure. The major justification for this architecture is that it enables effective software design, development, coding and testing (debugging). Almost any complex software system is an applicable candidate for object-oriented design.

1. COM(DCOM) architecture - as we studied in the segment on object-oriented design and analysis, Component Object Model is an extremely powerful mechanism for developing reusable code.
2. OLE architecture - as noted in 1 above, this is a really powerful architecture for providing complex but extremely effective software for a wide variety of systems.
3. Windows applications - Using any one for the commercial frameworks, user developed windows application are usually developed using object-oriented architectures.

Layered architecture - layered architecture is where the software system is decomposed from large blocks into smaller blocks. Each subblock provides individual actions and information which can then be recombined provide the necessary overall system activity. Alternatively, layering in object-oriented systems is represented by base and derived classes. The base class provides general features; the derived class may specialize to enable particular features.

1. Network architecture - a perfect example where the design of network systems is provided by 7 layers - application, service, transport, media and physical.

2. Operating system architecture - again here, we have lower level activities to hardware peripherals using drivers, upper level activities such as the file system, GUI system, etc., and still higher level activities such as application programs.

Problem 20.4. Using your own words and a few examples, define the terms *class*, *encapsulation*, *inheritance*, and *polymorphism*.

*Class* - class is the reserved word for defining an object in the language C++. In more general terms class represents a software structure. The structure provides a number of important features like:

1. Contains both data and functions to process the data.
2. Permits the controlled use of data and functions.
3. Through the controlled use of data and functions provides interfaces so that the class can be thought of as providing a service.

*Encapsulation* - encapsulation is the feature embodied in the class features above. It enables control of class activity by controlling the manner in which the user is able to use the data and/or the services provided by the class function interface. An example is an Array class where the user is able to access each Array element using an index variable, but is unable to access individual elements through link list processing.

*Inheritance* - inheritance is the mechanism by which classes can relate to other classes in an orderly manner. The original method by which one structure could relate directly to other structure was by embedding the latter. Inheritance provides a different mechanism. Here Class B can be related by adopting all the features of Class A. B after adopting all A's features can:

1. Add unique features of its own, and/or
2. Override or modify features adopted from class A.

Class B can do so without regard to any other class which inherits features from A. Finally, class B can be inherited in its own right by Class C which now adopts both A and B features. Therefore, inheritance provides an extremely effective mechanism by which class can interact.

*Polymorphism* - polymorphism is the mechanism by which dynamic (run-time) binding can be enabled in C++. Run-time binding provides a mechanism whereby function calls can be decided while the program is running as opposed to when the program is compiled. Thus, in different runs, the program is able to call different functions based on run-time parameters. This adds significant flexibility to programming. In one instance, after the program has been developed a new feature can be added without recompiling the whole code. The obvious disadvantage of run-time binding is that the program will run slower since the resolution of the call is made during the run and not at compile time.

Problem 20.5. Review the object defined for the SafeHome system. Are there other objects that you feel should be defined as modeling begins. (Use Fig. 21.7)

In my opinion, Figure 21.7 does not show the interface or connectivity to the SafeHome company (external) monitoring link. In our design of the SafeHome system self-test is not shown explicitly. As far as the drawing is concerned, I don't understand why the audible alarm is sending information to the system; it should be reversed. Second, the sensor event should be connected to the system. Actually, the sensor event should occur as a polling of the various sensors directly from the system. Then, the system through the keypad can activate/deactivate any sensor handle sensor-alarm timing, etc.

Problem 22.2. How do OOD and structured design differ? What aspects of these two designs are similar?

Structured design deals with functions and data as separate entities whereas in objects they can be incorporated within a unique structure. This is the major reason why differences can occur. That is, objects can provide different interfaces. In most cases structured design decomposes the functionality with the data being structured to work with this functionality (and vice versa). In OOD, the Pressman defines the objects' responsibilities (service) as the upper level. From that point, the functionality of the object is broken down into its messages (interface) and then class and subclass (possibly derived class) design. I would observe that the OOD design provides more flexibility design. In the lecture, I illustrated this point by examining the concept of COM based upon the object's flexible interface.

Both systems design using decomposition. Both have functions and data incorporated. Both enable the user to progress from specifications through design development and testing using structuring as the enabling mechanism.

Problem 17.11. Give three examples in which black-box testing might give the impression that "everything's OK", while

white box testing might uncover an error. Give at least three examples in which white-box testing might give the impression that “everything’s OK”, while black-box testing might uncover an error.

**Black-box vs white-box:**

1. White-box testing is directed toward verification of specifications. In some cases, the specification may not represent a valid system. White box testing may not exercise the situation where the specs may not work in the real world.
2. White-box testing does not usually examine data and processing rates or data volumes. Black-box testing should be designed to examine the conditions where overloads, overflows, etc., occur.
3. Black-box testing may extend the testing on systems which are different from those used during white-box testing. This can lead to errors due to external support features.

**White-box vs black-box testing:**

1. Black-box testing even though it is designed to examine the wide range of system inputs may fail to exercise a particular sequence of code. White-box testing is specifically designed to execute all feasible code sequences.
2. Black-box testing is usually developed to examine potential classes of errors using a range of inputs. The tests may not encompass all the classes or members of a particular class. For example, some errors may be caused by the environment which is not usually an input that is easily controlled. In the black-box tests that environment may not exist. However, in white-box testing the actual fault or error can be setup during the debugging process and the fault control code exercised to determine its effectiveness.
3. In many situations, running or response time may be important. While run-time errors during black-box testing may occur, the cause(s) of the error is usually discovered when the code is *profiled*, that is, when timing studies are done to determine and tailor the performance.

Problem 18.5. Why is a highly coupled module difficult to test?

A highly coupled module is one who has at least part of its interface closely tied to one or more other modules. This means that the closely related modules can not be completely tested independently (nominal unit testing) but must be tested as a unit (more nearly sub-integrated testing). The larger (in terms of code size and complexity) the code the more difficult it is to test. Further, action occurring in different modules is harder to trace, code read, design realistic tests for, etc.

A downloadable copy of the solution can be found in:

[http://www.cs.odu.edu/~foudr\\_e/cs451sp02/pdfhtmfolder/probassigns/probassign4.rtf](http://www.cs.odu.edu/~foudr_e/cs451sp02/pdfhtmfolder/probassigns/probassign4.rtf)