IC470, Software Engineering
Six Week Problem Set

Due date: At the start of class as per the syllabus

Collaboration: You may collaborate on this problem set as a homework with other midshipmen currently enrolled in this course. If you do collaborate, you must turn in only a SINGLE solution with the names of all the midshipmen that collaborated on the solution in the top right corner. The graded work will be handed back in the same section that it was submitted. If your name appears on more than one solution, you will receive the lowest score of the work on which your name appears.

Note: For questions requiring a calculation, show your work to receive possible partial credit if you make a simple math error but otherwise set the problem up correctly. It is fine to use a calculator to assist with calculations, but still show your work. Remember that only non-programmable calculators may be used on the exams in this course, so obtain one prior to the 6 week exam if you want to have one available during the exam.

Some potentially useful figures from the text are provided at the bottom of this file.

Scope of Software Engineering.

1.1. (4 pts) You are in charge of software development for the Joint Fire Control System (JFCS), with a development budget of $370,000. Assume that your development budget does *not* include post-delivery maintenance costs. Using the most recent historical data available in the text, indicate how much money should be allocated to each phase of the JFCS development lifecycle. Base your answer on the most recent historical data available in the text.

1.2. (4 pts) Assuming the system in question 1.1 is built on time and within budget, and using the approximate average costs discussed in the text, how much do you expect future maintenance to cost?

1.3. (4 pts) Fifteen months after delivery a fault is detected in the system from question 1.1. The cost of fixing the fault is $14,594. The fault was traced to an ambiguous sentence in the specification document. Basing your answer on the most recent historical data available in the text, approximately how much would it have cost to have corrected the fault during the specification/analysis phase?

1.4. (4 pts) Approximately how much would it have cost to have corrected the fault described in 1.3 if it had been identified during the implementation phase vice the specification/analysis phase?

Software Life-Cycle Models.

2.1 (4 pts) What type of system is the Spiral life-cycle model best suited for? Why?

2.2 (4 pts) What is the primary measure of progress using Agile Software Development?

2.3 (4 pts) What similarities (if any) are there between Agile Software Development and the Waterfall Model?

2.4 (4 pts) What differences (if any) are there between Agile Software Development and the Waterfall Model?

2.5 Functional Requirements. Identify each of the following Acceptance Test Plan test cases as Normal, Abnormal, or Not Useful (see Lab 1). Explain your answers.

Functional Requirement	**Acceptance Test Plan** test cases (scenarios showing that the Functional Requirement has been met, includes both normal and abnormal uses of the system).	**Circle one per row** to identify each Acceptance Test Plan test case on the left as Normal, Abnormal, or Not Useful. In each case, **explain why**.
<u>1.0 Dice roll input.</u> The game must allow the user to simulate randomly rolling two standard six-sided die. The consecutive sums produced from the dice rolls must agree with established frequency distributions.	1. User rolls two dice. **Expected result ->** Get a resulting sum in the range of 2..12. 2. User rolls two dice. **Expected result ->** Get a resulting sum outside the range of 2..12. 3. User rolls two dice 10,000 times. **Expected result ->** The resulting sums agree with established frequency distributions. 4. User does not roll any dice for 10 minutes. **Expected result ->** No gaming action occurs. 5. User attempts to roll only one die. **Expected result ->** Unable to roll only one die since dice are rolled together (consecutively).	1. (2 pts) (circle one) Normal, Abnormal, Not Useful Explain why: 2. (2 pts) (circle one) Normal, Abnormal, Not Useful Explain why: 3. (2 pts) (circle one) Normal, Abnormal, Not Useful Explain why: 4. (2 pts) (circle one) Normal, Abnormal, Not Useful Explain why: 5. (2 pts) (circle one) Normal, Abnormal, Not Useful Explain why:

Object-Oriented Analysis

Consider the below ATM system description (from howstuffworks.com), and answer the following questions. *Note: For questions 13.1. .. 13.5 ignore for now the last paragraph that starts with "Besides the electric eye …"*

13.1 (10 pts) Give a UML Use Case Diagram for the entire ATM system described below. Be sure to show all actors and their uses of the system.

13.2 (10 pts) Produce Stage III of the Noun Extraction technique for the ATM system description below in a 3-column table. In the first column, list the initial set of results of the noun extraction (i.e. list all the nouns). Draw a line through (strikethrough) all nouns in the first column that are outside the problem boundary. For all remaining nouns, either list the abstract nouns in the second column or identify them as candidate classes in the third column. The candidate classes in the third column must be used in your initial UML Class Diagram in 13.3 below.

13.3 (10 pts) Give a UML Class Diagram for the entire ATM system. Your class diagram must make at least one appropriate use EACH of inheritance (is-a), aggregation (has-a), and an association. Include all abstract nouns culled from the noun extraction above as attributes where appropriate in your UML class diagram. Note that since this is OOA, no methods appear on the UML Class Diagram (they will, however, appear on the UML *Detailed* Class Diagram described as part of the eventual Object Oriented Design – a subject of a later chapter).

13.4 (8 pts) Give a one paragraph discussion of how your UML class diagram would need to change if your client wanted you to add a database that stored the account number entered, pin number entered, photograph of user, and time of all invalid pin entry attempts.

13.5 (10 pts) Using a different color, etc, to clearly differentiate any additions, modify your UML Class diagram from 3 above to accommodate your client's requests regarding invalid pin entry attempts. Your modified UML Class Diagram *must promote reuse* by abstracting the database, the information stored in the database, and the system from each other.

13.6 (10 pts) Non-UML modeling. Give a data flow diagram that models the information flow and processes, etc, described in just the last paragraph of the ATM system description, the one that starts with "Besides the electric eye …" Use labeled ovals to model processes, labeled rectangles to model data sources/sinks, labeled parallel lines to model data stores and labeled arrows to model the flow of data.

An ATM is simply a data terminal with two input and three output devices. A normal data terminal has some sort of keyboard for input, some sort of screen for output, and a network connection that lets it talk to a server somewhere on the network. An ATM adds a card reader as an input device, along with a printer and an amazing money dispenser as output devices, to create a complete package.

Settlement Funds

Let's say you want to get some money from an ATM at a convenience store. Chances are that the merchant who owns the store either owns or rents the ATM. So the merchant fills the ATM with cash each day, and it is the merchant's cash that you receive when you get money from the ATM.

You walk up to the ATM, insert your card, and type your password. The card tells the machine your bank and account information. The ATM forwards this information to the host processor, which routes the transaction request to your bank.

If you're requesting cash, the host processor causes an electronic funds transfer to take place from your checking account to the host processor's account. Once the funds are transferred to the host processor's bank account, the processor sends an approval code to the ATM authorizing the machine to dispense the cash. The host processor then sends your funds into the merchant's bank account by automated clearing house, usually the next bank business day. So when you request cash, the money moves electronically from your account to the host's account to the merchant's account, and you get the merchant's cash.

Now you know what the virtual process is, but what's actually going on inside the machine?

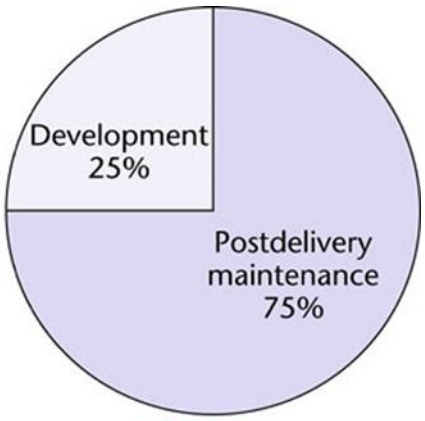
Parts of the Machine

An ATM has two input devices, the card reader and the keypad. The card reader captures the account information stored on the magnetic stripe on the back of an ATM card. The keypad lets the cardholder tell the bank what kind of transaction is required (cash withdrawal, balance inquiry, or whatever) and for what amount. Also, the bank requires the cardholder's personal identification number (PIN) for verification.

The most important output device is the heart of an ATM—the safe and cash-dispensing mechanism. The entire bottom portion of most small ATMs is a safe that contains the cash. The cash is stored in a series of cassettes—a big ATM in a high-traffic area can hold up to $100,000. The bill count and all of the information pertaining to a particular transaction is recorded in a journal. The journal information is printed out periodically and the machine owner maintains a hard copy for two years.

Besides the electric eye that counts each bill, the cash-dispensing mechanism also has a sensor that evaluates the thickness of each bill. If two bills are stuck together, then instead of being dispensed to the cardholder they are diverted to a reject bin. The same thing happens with a bill that is excessively worn or torn, or is folded. So, while it's not likely you're going to get an extra 20-dollar bill with your next withdrawal, you'll be happy to know you won't get half of a bill either.

Figures from the text that you may find helpful.



1992-1998

Figure 1.3b

	Various Projects between 1976 and 1981	132 More Recent Hewlett-Packard Projects
Requirements and analysis (specification) phases	21%	18%
Design phase	18	19
Implementation phase		
Coding (including unit testing)	36	34
Integration	24	29

Figure 1.4

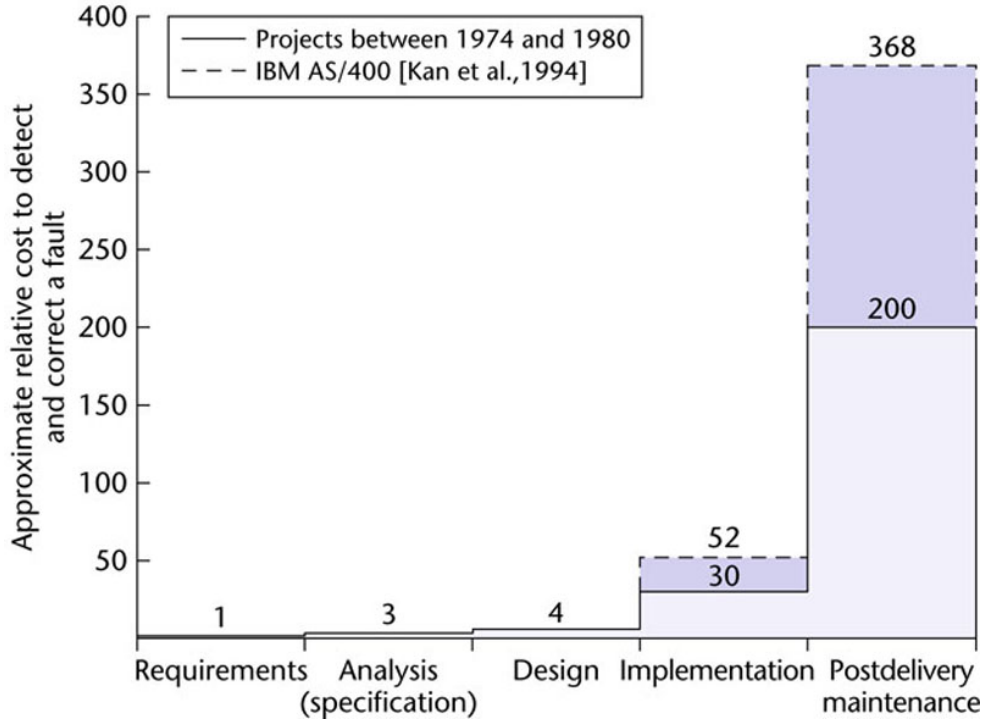

Projects between 1974 and 1980
IBM AS/400 [Kan et al.,1994]
Approximate relative cost to detect and correct a fault
400
350
300
250
200
150
100
50
368
200
52
30
1
3
4
Requirements
Analysis (specification)
Design
Implementation
Postdelivery maintenance

Figure 1.6