

Concordia University, Department of Computer Science
COMP 354 (Software Engineering) — Section X
Fall 1997 — Quiz 1

This quiz has six questions. Answer all questions and answer each completely. The relative weight of each question is given in brackets. Time allowed is 1 hour and 15 minutes.

1. [15%] Describe 3 ambiguities or omissions in the following statement of requirements for part of a ticket issuing system which is intended to automate the sale of train tickets. For each ambiguity that you discover, describe a way to resolve it in some appropriate way.

When the user presses the start button, a menu display of potential destinations is activated along with a message to the user to select a destination. Once a destination has been selected, users are requested to input their credit card. Its validity is checked and the user is then requested to input a personal identifier, after which the ticket is issued.

2. [15%] Describe how the systems requirements document (SRD) is used by each of the following:
 - (a) the customer/user/client
 - (b) the V&V team
3. [15%] You are the leader of a software development team in a large organization. One of the developers in your team tells you that she thinks “all this documentation we have to do takes away all the fun and skill from programming.” Give three reasons you would give her to support the need for documentation.
4. [15%] Give one example of how each of the following principles can be applied during the requirements analysis phase of software development.
 - (a) rigor and formality
 - (b) separation of concerns
 - (c) abstraction
5. [15%] Describe three differences between an evolutionary prototype and a throw-away prototype.

6. [25%] An automatic teller machine (ATM) at a bank allows a customer to deposit and withdraw money, pay bills, and query the status of their accounts. Consider each of the following software qualities:

- (a) robustness
- (b) security
- (c) user friendliness
- (d) data integrity
- (e) reliability

Define each of these terms; and illustrate your definition by explaining how it applies to an ATM.

COMP 354/554 Software Engineering

116AN

Fall Term, 1993

ANSWERS

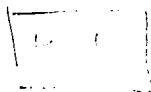
Quiz 2

Duration: 1 hour. You are not allowed to consult notes, books, or the invigilator. Attempt all of the questions.

- ✓ 1. [20%] What is a "throwaway prototype"? Describe circumstances in which you would recommend using a throwaway prototype during software development.
- ✓ 2. [20%] Consider the statement: "code inspection by a team is more effective than debugging by an individual". What does "more efficient" mean in this statement? What should an inspection team do when someone finds an error?
- ✓ 3. [20%] Parnas says that we should "design for change". List some of the things that might change after the first version of a software product is released to customers. For each item in your list, explain briefly how the design can anticipate the change.
- ✓ 4. [20%] Briefly describe the process of "top down design". What characteristics of top down design make it unsuitable for large-scale software development?
- ✓ 5. [20%] Assume that x is not a member of the set S . Using the equations below, prove that $Delete(S, x) = S$. [Hint: use induction on the cardinality of S . The symbol $\stackrel{\text{def}}{=}$ means "equal by definition."]

$$\begin{aligned} Delete(Empty, m) &\stackrel{\text{def}}{=} Empty \\ Delete(Insert(S, m), n) &\stackrel{\text{def}}{=} \begin{cases} \text{if } m = n \\ \text{then } Delete(S, n) \\ \text{else } Insert(Delete(S, n), m) \end{cases} \end{aligned}$$

no. 11



#1 "Throwaway prototype": This is only a trial involving
of * building ST that contains some of the desired
features of the final product, but that omits important
parts of the final product. ✓

* Delivering that thing to the customer to get
feedback. ✓

* Based on feedback, measuring "added value"
to cost, forming more precise & profoundly based
requirements. ✗

* Designing & adjusting objectives. ✓
* Throwing away that thing (not part of the
final product). ✓

Recommended circumstances:

When the application is too large & complex, better
use "throwaway prototype" to deliver incremental
trial things after a certain # of stages. This
ensures the more precise requirements & specifications.
Hence, ensuring a building of the right product
(i.e. meet exactly the customer's expectations.)

incremental development is different

#2 In the state, "more efficient" means:

x Code inspectn. by a team gives more precise feedback since "more heads are always better than one"

ie: a team of people, when examining a sectn. of a program (or a program) tends to find out more incorrect, unclear, & imprecise code than (1 person).

x When errors found, the team can suggest many alternative solutions, among them the best can be chosen out at last.

x Among ^{people} a "team", many ideas are discussed to find out the best solution.

x When errors found, first thing to do for the team is to ask the person who wrote the code for more explanation. This is good for the code-writer to explain his/her logic step by step. Thanks to this, he/she can find out the flaws in his/her logic & errors in code.

"Code inspectn." is "more efficient"

- finding more errors per person/hr.

What inspectors do:

- document errors (*)

- do not fix it

- may not affect other modules

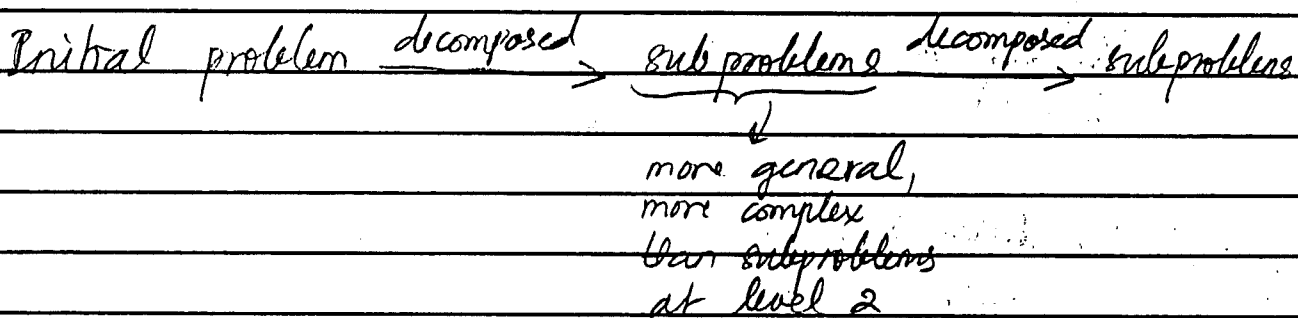
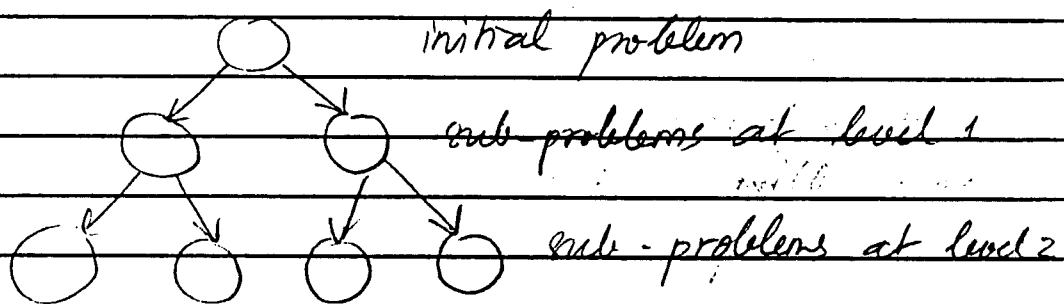
#3 Things that might change after 1st version of a SW product is released to customers!

1. Change in data representation: ✓
ex: a discovery of a new data structure that, if being applied in the SW. developm., will produce a more "efficient" product in terms of speed, cost, qualities, etc...
2. Change in algorithm: ✓
ex: a better sorting algorithm to produce ^{much} faster output
3. Change in peripheral device: ✓
ex: module interface to terminal; disk; LAN
4. Change in social environment: ✓
ex: change in units of measure, in interest rate, in legislation of re-tax, etc...

To anticipate these change, design should (for each type of change numbered above):

1. use Informal. hiding, ^{and high cohesion, low coupling} as much as possible. When this change occurs, just need to change the data structure in the appropriate module. This does not affect either other modules, or interfaces. ✓
2. apply info. hiding, ^{and high cohesion, low coupling} to the utmost for this change (if occurs), just modify the module (probably the library module) that contains the sort, w/o affecting anything else in the whole system.
3. also use info-hiding, high cohesion & low coupling. Only modify the interface modules & those containing any code that manipulates secondary storage & terminal.
4. use info. hiding, high cohesion & low coupling.

#4 Top-down design:



→ modules → writes

- x Top-down design is good for small problems, but
- x Not suitable for large-scale SW development, b/c:
 - The recursive decomposition is based on functions, no attention is paid to data \Rightarrow cohesion is not high ✓
 - Informal hiding is not applied much in this method \Rightarrow poor reusability. ✓

Since a SW development (in general), especially a large-scale one requires high cohesion, low coupling, reusability of modules & maintainability (besides other qualities such as correctness, robustness, etc...), "Top-down design" does not meet these requirements.

Need only change the appropriate module that do the conversion, nothing ^{else} required modification.

17

#5

Given:

$$x \notin S$$

Prove: $\text{Delete}(S, x) = S$

Given: ① $\text{Delete}(\text{Empty}, m) = \text{Empty}$

② $\text{Delete}(\text{Insert}(S, m), n) =$ if $m = n$ then

a) $\text{Delete}(S, n)$

else b) $\text{Insert}(\text{Delete}(S, n), m)$

Proof:

Basis: $S = \emptyset$ (empty set)

$\text{Delete}(S, x) \stackrel{\text{from ①}}{=} \emptyset = S \rightarrow \underline{P[|S|=0] \text{ is T.}}$

Hypo: assume $P[|S|=k]$ is true

(k is an integer) (ie: $\text{del}(S, x) = S$ is T.)

Induct: have to prove $P[|S|=k+1]$ is also T.

$$\begin{aligned} \text{del}(\text{Insert}(S, m), x) &\stackrel{\text{from ②}}{=} \text{Insert}(\text{del}(S, x), m) \\ &= \text{Insert}(S, m) \leftarrow \text{from hypothesis.} \\ &= S \text{ (including element 'm')} \end{aligned}$$

(see next page before conclusion) (ie: $|S|=k+1$)

$\Rightarrow P[|S|=k+1]$ is T

\therefore By induct., $P[|S|=k+1]$ is T $\Rightarrow P[|S|=k]$ is true

Hence, it is true for all sets.

COMP 354/554: SOFTWARE ENGINEERING

Section ZZ
Winter 1993

Quiz 2

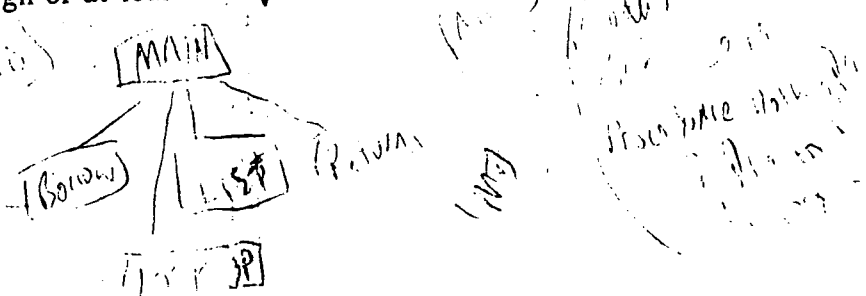
Answer both questions. Each question carries equal marks. Time allowed: 60 minutes.

- (a) What is a software *module*? What characteristics does a good module possess?
(b) Software modules enable the *reusability* of software. Explain the difference between the following types of module, from the point-of-view of reusability:
 - abstract object module
 - abstract data type (ADT) module
 - generic ADT module
 - object-oriented design module

2. A simple software system for a university library provides the following services to users:

- ✓ Borrow: Record the fact that a specified *user* has borrowed a specified *book*.
- ✓ Return: Record the fact that a specified *user* has returned a specified *book*.
- List: List all books currently borrowed by a specified *user*.
- Look-up: List all books by a specified *author*, or with a specified *title*.

Produce a software design for this system, including a complete architectural design, a complete module interface specification, and the internal module design of at least one module.



Concordia University
Department of Computer Science
COMP 354 (Software Engineering) — Section X
Fall 1993 — Quiz 2

Answer all questions and answer each completely. The relative weight of each question is given in brackets. Time allowed is 60 minutes.

Question 1.[30%] Consider the design of software for an automatic teller machine (ATM) for a bank.

(a) Describe two “objects” that might exist in an object-oriented design of the ATM software. For each object, describe one operation that is either required of it, or suffered by it.

(b) Describe two data flows that might exist in a functional design of the ATM software. For each data flow, describe one process/transformation that it is involved in.

Question 2.[30%] Describe how each of the following is related to the principle of “design for change”:

- (a) information hiding
- (b) cohesion
- (c) coupling
- (d) anticipated changes
- (e) traceability of the requirements in SRD to the AD document

Question 3.[20%] Define each of the following terms:

- (a) IS_A relation
- (b) USES relation
- (c) INHERITS_FROM relation
- (d) IS_COMPONENT_OF relation

Question 4.[20%] Give three examples of how the principle of “separation of concerns” is applied in the design phase.