UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, DECEMBER 1999

CSC444F - SOFTWARE ENGINEERING I
Examiner: Steve Easterbrook

Exam type: A (Closed Book; no aids permitted)
Calculator type: 1 (any programmable and non-programmable calculator)

Duration: 2.5 hours.

This exam paper consists of 8 questions on 4 pages.
Answer ANY 5 questions. All questions have equal weight (20 marks per question)

QUESTION 1 (Procedural and Data Abstractions)

a) What is *abstraction*? Why is it so useful in software engineering? [3 marks]
b) A procedure can be said to have *referential transparency* if it behaves in the same way no matter where it is used, and *encapsulation* if the procedure can be used correctly without knowing how it is implemented. Why are these properties important for procedures, and how can they be achieved? [4 marks]
c) A *procedural abstraction* has five elements: '*parameters*', '*requires*', '*effects*', '*modifies*', and '*raises*'. What is the role of each of these elements, and why is each element needed? [5 marks]
d) What is the difference between *object-oriented design* and *object-oriented programming*? How can object oriented designs be implemented in a procedural programming language? [4 marks]
e) You are assigned to work on a project to develop a controller for a new satellite. To save money, your manager estimates you could reuse about 90% of the software from one of two similar satellites the company built in the past. One system has no known bugs, but the code is undocumented and uncommented. The other has 25 known non-critical bugs, but every procedure is documented with a procedural abstraction. Which system would you choose to work with, and why? [4 marks]

QUESTION 2 (Design Representations)

a) The four key viewpoints for representing software design information are *structural*, *behavioural*, *functional* and *data modelling*. For each of these four, explain what types of information the viewpoint represents, and give examples of notations used. [4 marks]
b) What are the advantages and disadvantages of using *viewpoints* to separate out different kinds of information in software design? [4 marks]
c) Some design notations lend themselves well to measuring different software quality indicators. Pick a design notation you are familiar with and describe how well it supports measurement of design quality. [4 marks]
d) Software architectural styles, such as *pipe-and-filter* and *layered systems* are structural viewpoints. How would you add behavioural and data-modelling viewpoints to these architectural descriptions? [4 marks]
e) The company you work for has a large amount of legacy code, for which no documentation exists. You are assigned to lead a team to develop a reverse engineering tool to automatically extract some design representations from the legacy code. What design representations would you choose to extract? Why? [4 marks]

QUESTION 3 (Testing)

a) Why is random testing insufficient even for relatively small programs? [2 marks]

b) *Unit testing* is the process of testing a single program unit (e.g. a procedure) in isolation from the rest of the program. How would you go about choosing test cases for unit testing? [4 marks]

c) *Integration testing* can be tacked top-down or bottom-up. Describe each of these strategies. Why is integration testing harder than unit testing? [4 marks]

d) Explain the purpose of each of the following. What types of error is each likely to find?
   i) Endurance testing
   ii) Recoverability testing
   iii) Regression testing [6 marks]

e) The company you work for develops internet applications. To reduce time to market, the company is considering dispensing altogether with integration testing. Instead, the company plans to rely on Beta testing, in which free trial versions of new software will be sent to existing, trusted customers to try out, with the agreement that they will report any problems they encounter. What are the advantages and disadvantages of this approach? [4 marks]

QUESTION 4 (Verification and Validation)

a) Define the terms Verification and Validation. Why is each insufficient on its own? What makes validation particularly hard? [5 marks]

b) The traditional approach to software verification is testing. However, studies have indicated that inspection can be more cost effective at detecting errors. Why is this? [4 marks]

c) *Formal verification* applies mathematical reasoning (e.g. predicate logic) to prove various properties about a program, including correctness. What advantages and disadvantages does formal verification have compared to testing? [4 marks]

d) In theory, formal verification could be automated if the original specification is stated completely and precisely. Why is this hard to achieve in practice? [3 marks]

e) For many safety critical systems, a separate contractor is engaged to carry out *Independent V&V*, where the V&V contractor is financially, managerially and technically independent from the software development contractor. Why is this independence important? [4 marks]

[TURN OVER]

QUESTION 5 (Requirements Engineering & Specification)

a) Zave defines Requirements Engineering as *"the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems [and] the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families"*. Why is Requirements Engineering considered to be the most important part of software engineering? [4 marks]

b) Requirements should state what a system should do, without stating how it should do it. Why is this distinction useful? [2 marks]

c) *Structured Analysis* proceeds by modeling the current physical system, abstracting out a model of the current logical system, and then modeling the new logical system. What are the advantages and disadvantages of building these three separate models? What representations are used for each of these models? [4 marks]

d) Explain why each of the following is an important property of a software specification, and explain how it can be achieved when writing specifications:
   i) unambiguousness
   ii) traceability
   iii) verifiability. [6 marks]

e) Project managers sometimes regard work put into writing high quality specifications as "gold plating", and claim that it is unnecessary as it doesn't contribute to producing program code. Under what circumstances is this view sensible, and under what circumstances is it foolish? In the latter case, how would you persuade such a manager that the specification does need to be high quality? [4 marks]

QUESTION 6 (Software maintenance and evolution)

a) Why is software maintenance difficult? Why is it necessary? [4 marks]

b) In his paper on the laws of software evolution, Lehman introduced three types of software:
   i) *S-type* (specifiable), where the problem to be solved can be stated precisely and is independent of the real world,
   ii) *P-type* (problem-solving) where the software is required to solve some real world problem that cannot be stated precisely;
   iii) *E-type* (embedded) where the software is to be embedded in the world.
   Give an example of each type of software. Why is change intrinsic to P-type and E-type software but not to S-type? [6 marks]

c) Why does software tend to increase in complexity as it evolves? [2 marks]

d) Software maintenance activities can be classified as *corrective* (fixing errors), *adaptive* (responding to change) and *perfective* (improving the original software). Why is this distinction useful? How would you expect the proportion of time spent on each activity to change as the software ages? [4 marks]

e) The company you work for has traditionally kept its software maintenance teams separate from development teams. It now wants to move to a mission orientation where a single team will be responsible for the development and maintenance of each software product. What advantages should your company expect from the re-organisation, and what problems might it encounter? [4 marks]

[TURN OVER]

QUESTION 7 (Software Measurement)

a) DeMarco states that "you cannot control what you cannot measure". What does this mean for software project managers? [2 marks]

b) A *metric* is a measurable characteristic of software, and a *model* is a mathematical relationship between metrics. Why is validity important for both metrics and models? Why is the validity of most models used in software measurement disputed? [4 marks]

c) Software quality measurement generally starts with high level quality goals, and then identifies metrics that can be used to indicate satisfaction of the quality goals. For each of the following quality goals, explain why the goal is important, and identify a metric that could be used to measure it:
   i) Reliability
   ii) Efficiency
   iii) Usability [6 marks]

d) Describe two *design* metrics that can be used as *predictors* of software quality before the code is written. For each metric, explain why this metric indicates satisfaction of the high level quality goals and how it can be measured. [4 marks]

e) An internet start-up company plans to use COCOMO to estimate the cost of development of their planned Java-based web tools. What advice would you give them about the appropriateness and accuracy of COCOMO for these projects? [4 marks]

QUESTION 8 (Software Process Modelling)

a) Leon Osterweil published a famous paper with the title "*Software Processes Are Software Too*", in which he argued that software development processes could be modelled at a fine level of detail, as algorithms or even programs. What are the advantages of modelling software processes to this level of detail? What are the difficulties? [4 marks]

b) The *Capability Maturity Model* (CMM) rates software companies according to how well they identify and manage their software processes. The model has five levels: Initial, Repeatable, Defined, Managed, and Optimising. Briefly describe each of the five levels. What advantages are there for a company to move up to the top level? [6 marks]

c) Why is process improvement unlikely to occur unless an organisation defines and manages its processes? [2 marks]

d) What notations would you use to define your software development process? What are the advantages and disadvantages of these notations? [4 marks]

e) The company you work for is considering instituting a company wide software process modelling effort, and plans to set up a Software Engineering Process Group to take responsibility for this effort. What advice would you give to this group to help ensure its success? [4 marks]

[LAST PAGE!]