UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

FINAL EXAMINATION, DECEMBER 2001

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 (Project Planning and Management)

a) Software Project Management has been described as a difficult and thankless task. Why? [2 marks]

b) A *Work Breakdown Structure* gives a detailed list of all the tasks involved in a software project. How would you go about defining a work breakdown structure for a new project, and how would you estimate the time and number of people to devote to each task? [4 marks]

c) A *PERT Chart* allows a manager to determine which tasks lie on the critical path of a project. Why is this useful information? Illustrate your answer with an example PERT chart. [4 marks]

d) During a software project, a manager can adjust four key variables to ensure the project meets its goals: *schedule*, *resources*, *scope* (of the product) and *risk*. For each of these four variables, describe the circumstances under which a manager would increase it, and the circumstances under which a manager would decrease it. [6 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 2 (Software Process Modelling)

a) Leon Osterweil published a famous paper entitled "*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 development 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]

QUESTION 3 (Software Lifecycles)

a) Managers often use the *waterfall model* to plan a software project. Describe the waterfall model. What limitations does this model have for project planning? [4 marks]

b) Describe two alternatives to the waterfall model. Describe the advantages and disadvantages of using each model to plan a software development project. [5 marks]

c) In *throwaway prototyping*, the prototype is used to explore requirements with the customer, and to investigate feasibility. The prototype is discarded before the real system is built. Why might it be cost-effective to build software that will be discarded? [2 marks]

d) The IEEE standards define *verification* as the process of evaluating whether software components developed in a given lifecycle phase meet the conditions identified in the previous phase. They define *validation* as the process of evaluating the software at the end of the development process, to determine whether it satisfies the requirements. What is wrong with these definitions? Suggest better definitions. [4 marks]

e) A friend is enrolled in a software engineering course, but has missed most of the lectures. He hopes to get hired by a large software company that is recruiting specialists in all different parts of the software lifecycle. Unfortunately his interview is in two days! What parts of the software engineering lifecycle would you advise him to study in detail, to give him the best prospects of getting hired into an interesting and rewarding career? [5 marks]

QUESTION 4 (Software Testing)

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

b) *Endurance testing* involves running the software for long durations (weeks or even months), to determine if latent errors show up only on long runs. What types of software benefit from endurance testing? [2 marks]

c) Explain the purpose of each of the following. What types of error is each likely to find?

i) Stress testing

ii) Recoverability testing

iii) *Customer acceptance testing* [6 marks]

d) *Regression testing* involves repeating earlier tests each time the software is modified, to ensure that it still passes all the tests it passed before. However, regression testing can be expensive and time-consuming. How can the costs of regression testing be reduced? What are the risks of not performing it? [3 marks]

e) Imagine you work as a tester for a company that produces application software for personal computers. The company employs a test team, who are responsible for *all* software testing. This team is kept separate from the teams of programmers that develop the software. However, recently the company has found it hard to recruit software testers. To overcome the personnel shortfall, the management have proposed that programmers shall do their own *unit testing*, while the independent test teams shall concentrate only on *integration testing*. What are the disadvantages of this solution? What other solutions would you propose? [5 marks]

QUESTION 5 (Software Architectures)

a) A *software architecture* describes a high-level design view of a software system. What are the advantages of explicitly describing the architecture independently from the implementation? [2 marks]

b) *Blackboard architectures* allow a complex task to be broken down so that several cooperating agents can tackle it, sharing a central data repository, or *blackboard*. Give an example of a system that uses a blackboard architecture. What are the advantages and disadvantages of this style? [4 marks]

c) *Object oriented architectures* can be difficult to maintain because each object needs to know which other objects exist and how to call their methods. This limitation can be overcome using *implicit invocation*. Describe how an implicit invocation scheme works, and give two examples of its use. [4 marks]

d) An *architectural description language (ADL)* is a general purpose language for describing software architectures. What kinds of basic components and basic connectors would you expect an ADL to provide? What advantages are there in using an ADL? [4 marks]

e) The *Free Software Foundation* proposes to develop an *open source* suite of tools for developing and debugging Java programs. What architectural styles would you consider for this project, and what aspects of this project would affect your choice of style? [6 marks]

QUESTION 6 (Software Quality)

a) *Software quality* can be defined as fitness for purpose. Give two reasons why software quality is hard to measure. Why is there no absolute measure for software quality? [3 marks]

b) Software quality measurement generally starts by identifying high level quality goals, and then finding measurable attributes of software 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) Maintainability
   iii) *Usability* [6 marks]

c) It is often desirable to predict software quality from design models, before the code is written. Describe two *design* metrics that can be used as *predictors* of software quality. For each metric, explain why this metric indicates satisfaction of the high level quality. [4 marks]

d) 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 the two design metrics you described in part (c). [2 marks]

e) NASA Jet Propulsion Lab (JPL) uses a Quality Assurance (QA) team to check that various quality controls are followed throughout the software development process. However, JPL recently lost a Mars probe due to a navigation error that was traced to a data file in the ground support system, in which navigation data was provided in imperial units, rather than the specified metric units. Despite noticing a discrepancy in computed navigation results during the mission, JPL was unable to identify this problem during the mission. What does this failure suggest about the quality of the software developed at JPL, and about the effectiveness of JPL's QA process? [5 marks]

QUESTION 7 (Requirements Modelling)

a) A recent conference described Requirements Engineering (RE) as *"concerned with identifying the purpose of a software system, and the contexts in which it will be used. Hence, RE acts as the bridge between the real world needs of users, customers, and other constituencies affected by a software system, and the capabilities and opportunities afforded by software-intensive technologies"* Why is requirements engineering hard? [2 marks]

b) *Modelling* plays a central role in requirements engineering. Briefly describe each of the following modelling notations, and give an example of when you would use each for requirements modelling:
   i) *Dataflow Diagrams*
   ii) *Object Class Hierarchies*
   iii) *Entity Relationship Diagrams* [6 marks]

c) *Formal Methods* provide modelling languages that have precise mathematical definitions of their syntax and semantics. What are the advantages and disadvantages of using formal methods for requirements modelling? [4 marks]

d) The formal modelling language *SCR* models the requirements for a real time control system as a set of modes. To build an SCR model, one provides tables defining the precise circumstances under which the system switches between modes, and the outputs the system is required to give in each mode. What types of analysis can be performed on this type of model? [3 marks]

e) A company that manufactures fighter aircraft is finding it hard to maintain the onboard flight software for one of its most popular jets, as they don't really know what the software does. No requirements models exist. What do you suggest they should do to make maintenance easier? [5 marks]

QUESTION 8 (Re-use and Re-engineering)

a) Dave Parnas coined the term "software geriatrics" for the problem of maintaining software that is reaching the end of its useful life. What special challenges are there in software geriatrics that make it harder than other aspects of software engineering? [3 marks]

b) Software Re-use has the potential to cut the cost of software development. One approach to re-use is to develop libraries of re-usable program components. Another approach is to re-use artefacts from other phases of development, such as requirements models, architectural designs, test drivers, etc. What are the advantages and disadvantages of these two competing approaches? Give examples of when you would use each. [5 marks]

c) Software Rejuvenation involves restructuring existing software without altering its functionality. Why might this become necessary? Under what circumstances is rejuvenation likely to be cost effective? [4 marks]

d) Reverse engineering tools analyse existing source code, and generate various diagrams showing the design of the system. This is especially useful for software for which documentation has been lost, or is of poor quality. What kinds of design information would you expect such a tool to generate? How would automatically generated design models differ from those generated during the original design process? [4 marks]

e) The European Space Agency's Ariane-5 launch vehicle failed on its maiden flight because a component that was re-used from the (extremely successful) Ariane-4 rocket generated a floating point exception when used in Ariane-5. The exception handler was disabled in this component because analysis had showed that the overflow was physically impossible on Ariane-4. What lessons for code re-use can be drawn from this experience? [4 marks]