UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2003

COMPUTING AND INFORMATION SYSTEMS

IS52004A (CIS210) Software Engineering and Development

[Internal]

Duration: 3 hours

- *Full marks will be awarded for complete answers to FOUR questions. Do not attempt more than FOUR questions on this paper.*
- *Electronic calculators may be used. The make and model should be specified on the script. The calculator must not be programmed prior to the examination. Calculators which display graphics, text or algebraic equations are not allowed.*

THIS EXAMINATION PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Question 1: *Software Metrics*

a) State five reasons for measuring software. **[5]**

b) Explain briefly each of the following three properties that are required by an effective technical software metrics. The metric should be: **[6]**

simple and computable

empirically convincing;

programming language independent

c) According to these properties analyse the effectiveness of the Function-Points metric. **[6]**

d) Enumerate five information domain characteristics normally used in function-oriented software project metrics. Give the formula used to compute function points, on the basis of these information characteristics. **[8]**

Question 2: *COCOMO*

a) Consider the following table of coefficients that corresponds to an example semidetached software project of approximate size 43.21 KLOC:

Project class	a_b	b_b	c_b	d_b
organic	2.4	1.05	2.5	0.38
semidetached	3.0	1.12	2.5	0.35
embedded	3.6	1.20	2.5	0.32

Calculate the basic COCOMO estimates of the software development effort and the software development time for this system (You do not have to compute the exponential functions). **[12]**

b) How do we compute the recommended number of people for a software engineering project according to the Basic COCOMO model? **[4]**

c) Describe the three kinds of software project classes to which the COCOMO model for software project estimation can be applied. **[9]**

Question 3: *Scheduling*

Consider the task of developing a software system for handling telephone calls between two users connected to a switching mechanism. The switching mechanism connects incoming and outgoing subscribers so that they can talk to each other. The scheduling of this task must account for the following requirements (the subtasks are given in italic):

- initially *a subscriber* class has to be designed. (T1) This should take no more than a working day;
- next, the classes for *incoming* (T2) and *outgoing subscribers* (T3) should be designed in parallel. Neither of them should take more than a day; subtasks T2 and T3 should be carried out in parallel with the *management of the network operations* class (T5); the subscribers can be made for two days each; the network software needs also two days but it should be preceded by a subtask of *establishing the network protocol* (T4) which requires one day;
- after the completion of all these subtasks, a database called *telephone directory* should be created (T6). This takes two days approximately;
- the *design of the overall control* (T7) should take up to seven days as it has to deal with two connect-caller-to-callee cases that have to be ready at the seventh day: one for *handling the incoming subscriber* (T8) lines which should take five days, and one for *handling the outgoing subscriber* (T9) lines which takes three days;
- along with the control design the *software interface* (T10) should be created for no more than four days;
- the software engineering process terminates with the *testing* (T11) and *quality assurance* (T12) phases conducted sequentially for two days each.

a) Produce a Task Network for scheduling the development of this task, where each subtask is associated with its starting time, assuming that the start time for the whole system is 1/9/2002. **[12]**

b) Produce a Gantt Chart for scheduling the development of this task, including a milestone before each subtask in the main working sequence. Use a common milestone for the subtasks that are parallel. **[13]**

Question 4: *Testing*

a) Distinguish between white-box and black-box testing. **[4]**

b) Consider the following program fragment:

```
int example( float desired, int a[] )
{
   int x;
   int f;

   f = 0; x = 0;
   while ( x < 5 )
   {
      if ( desired == a[ x ] )
         f = 1;
      x = x+1;
   }
   if ( f == 1 )
      printf( "Found" );
   else
      printf( "failed" );

   return f;
}
```

i) Produce the flow-graph of this program fragment; **[5]**

ii) Determine the cyclomatic complexity of this fragment using the number of regions in the flow graph; **[4]**

iii) Identify the number of testing paths and enumerate them; **[4]**

iv) Develop test cases for the examination of all testing paths. **[8]**

Questions 5: *UML and Object-Oriented Design*

Note that questions 5 and 6 both refer to the specifications given in the Appendix.

Question 5: The Use-case view of the system.
(Say which specification you are working with)

a. Give a high-level use case diagram for your system. **[6]**

b. Write a specification of each of your use cases. Your specification should include pre and post-conditions and a description of a scenario that represents the use case in action. **[12]**

c. Draw one Activity Diagram for your system. **[7]**

If you are using Specification A, your diagram should follow the changes of the status of the `CompanyDocument` object.

If you are using Specification B, your diagram should follow the changes of the status of the `PublishingArticle` object.

If you are using Specification C, your diagram should follow the changes of the status of the `MedicalPatientFile` object.

Question 6: *UML and Object-Oriented Design*

a. Draw a Class Diagram for your system. **[9]**

b. Describe your classes including, at least, the attributes and methods associated with each class. **[9]**

c. Draw either one Sequence Diagram or one Collaboration Diagram for your system: **[7]**

If you are using Specification A, concentrate onthe publish() function, checking the `CompanyStatus` first and then changing it to 'published'.

If you are using Specification B, concentrate on. the publish() function, checking the `PublishingStatus` first and then changing it to 'published'.

If you are using Specification C, concentrate on. the addATest() function, checking the `MedicalStatus` first and then changing it to 'tested'.

Appendix: Specifications

Specification A: *Document Management System*

The document management system has the following user requirements:

- A CompanyUser must be able to create a CompanyDocument. This CompanyDocument is an object with some properties and the path of a real document attached to it.
    - All CompanyUsers are part of exactly one CompanyDepartment.
    - The real document attached to the CompanyDocument can be whatever the CompanyUser chooses, a Word document, a java file, etc. including a link to a web page.
    - The properties of the CompanyDocument object are: a description, a list of keywords.
    - A number of CompanyDepartments are associated with this CompanyDocument. The CompanyDocument object will be visible only for the CompanyUsers that are part of those CompanyDepartments.
- A CompanyUser must be able to update a created CompanyDocument. This means he can change the properties of the object, the path of the real document attached to it, and the CompanyDepartments associated with it.
- A CompanyUser must be able to publish a created/updated CompanyDocument object. This means the CompanyStatus will change from 'created' or 'updated' to 'published', and the CompanyDocument object will be visible to all the CompanyUsers in the associated CompanyDepartments.

Specification B: *Publishing System*

The publishing system has the following user requirements:

- A PublishingUser must be able to create a PublishingArticle. This PublishingArticle is an object with some properties:
    - A PublishingArticle has a title, a text, a photo, a publicationDate.
    - A PublishingArticle is associated with some PublishingPapers, in which it will be published.
    - A PublishingArticle is associated with exactly one PublishingLayout, which defines how the title, text and photo will be arranged when printed.
    - A PublishingArticle is associated with exactly one PublishingCity, the PublishingCity of which the PublishingUser is part of.
    - All PublishingUsers are part of exactly one PublishingCity.
- A PublishingUser must be able to update one of his PublishingArticles. All properties can be changed, except the PublishingCity associated with the PublishingArticle.
- A PublishingUser must be able to publish one of his PublishingArticles. This means only that an email will be send to the PublishingPapers with the properties of the PublishingArticle, and the PublishingStatus of the PublishingArticle will change from 'created' or 'updated' to 'published'.

Specification C: *Medical System*

The medical system has the following user requirements:

- A MedicalDoctor must be able to create a MedicalPatientFile. The MedicalPatientFile is an object with some properties:
    - A MedicalPatientFile has a name, an address, a list of medicines and a list of illnesses.
    - A MedicalPatientFile is associated with exactly one MedicalDoctor.
- A MedicalDoctor must be able to update a MedicalPatientFile. All properties can be changed, except the MedicalDoctor associated with the MedicalPatientFile.
- A MedicalDoctor must be able to addATest to a MedicalPatientFile. This means:
    - He must be able to *chose* from a list of MedicalTests ("Test1", "Test2", …).
    - He must be able to *create* a MedicalTestResult for the MedicalPatientFile. A MedicalTestResult is an object with some properties:
        - It is associated with exactly one MedicalTest.
        - It is associated with exactly one MedicalPatientFile.
        - It has a testDate, a testLength (in milliseconds) and a testScore (a score from 1 to 10).
    - The MedicalStatus of the MedicalPatientFile object will change from 'created' or 'updated' to 'tested'.