UNIVERSITY OF CAMBRIDGE

ENGINEERING TRIPOS PART IIA

Thursday 6 May 2004 2.30 to 4

Module 3F6

SOFTWARE ENGINEERING AND DESIGN

*Answer not more than **three** questions.*

All questions carry the same number of marks.

*The **approximate** percentage of marks allocated to each part of a question is indicated in the right margin.*

There are no attachments.

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

(TURN OVER

1 (a) Engineering activities have a defined *process*, which provides a set of procedures for the people undertaking the activity to follow; software engineers follow a *software process*. What are the reasons for having a defined software process? Are there any arguments against? [25%]

(b) *Waterfall* and *Reusable Components* are two common models of the software process. Describe the Reusable Components process model. How does it differ from the Waterfall model? [25%]

(c) Can the Reusable Components model be described as a *top-down* or *bottom-up* process? [10%]

(d) Give two advantages and two disadvantages of the Reusable Components model as a software process when compared to other common models. [20%]

(e) Would Reusable Components be an appropriate process model for developing a safety-critical system? Explain your answer. [20%]

2 (a) Identify two key benefits that object-oriented software design has over a function-oriented approach. [10%]

(b) A college library is commissioning a software system to manage the loans of books in its collection. Part of the specification for the software reads as follows:

Specification: The system should maintain a list of all books in the collection and a list of all users of the library. Each book can be *on-shelf*, *awaiting-collection*, *on-loan*, or *overdue*. At the library check-out, a user may present a book which is recorded as being *on-shelf*, this book is then placed *on-loan* to that user. Each user may have a maximum of four books *on-loan*. When a user returns a book it is placed *on-shelf*. If a user keeps a book for more than one week, the book becomes *overdue* and a message is sent (by email) to the user. If a user has any books *overdue*, he is not permitted to borrow any more books (even if he has fewer than four *on-loan*). When a book is returned, a fine of 50p per *overdue* book per day is charged to the user and no more books may be borrowed until the fine is paid. A user may request a book using a web interface. If the book is *on-shelf* it is held at the checkout *awaiting-collection* by the user, otherwise the request is denied.

Using good design principles, draw a UML class diagram which shows the main classes that will be needed for this software, the relationships between these classes and the main attributes and operations that they should support. Identify any design patterns that you use in your design. You do not need to give any pseudocode in your answer. [70%]

(c) The library now wishes to maintain an additional collection of CDs for loan. CDs may only be borrowed for a maximum of three days, after which the fine is to be £1 per day.

Show what changes will be required to the software in order to support this additional functionality, give your answer by drawing class diagrams that show your modifications. [20%]

3 Figure 1 shows a UML class diagram for software which simulates a digital circuit.

(a) Identify the classes present and the relationships between them. [15%]

(b) Figure 2 shows a simple circuit diagram which is represented by this software simulator. When the `compute` function is called for the `Circuit`, the `Components` are processed from left to right as seen in Fig. 2. Draw a sequence diagram which shows what happens when `compute` is called. [40%]

(c) If the `Components` were processed in a different order, what might happen? [10%]

(d) It is now desired to allow the user to define a library of complex compound components which have multiple outputs, e.g. JK flip-flops or adders. The user should then be able to treat these compound components in the same manner as the existing primitive components.

Using good design principles, show the necessary changes to the software to allow this, drawing any modified parts of the class diagram, and identifying any design patterns used. You do not need to give any pseudocode in your answer. [35%]

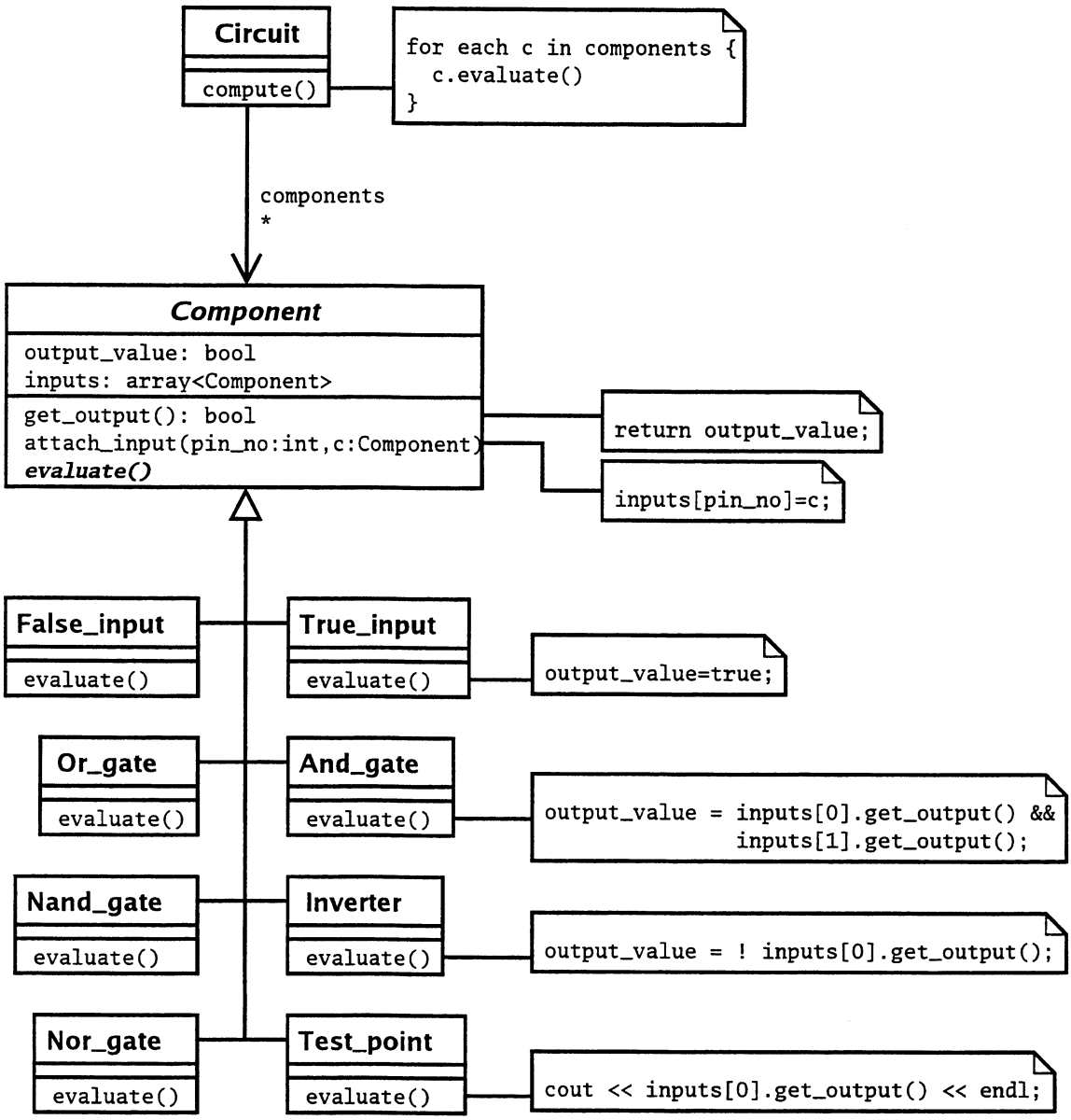
Circuit
compute()
for each c in components {
c.evaluate()
}
components
*
Component
output_value: bool
inputs: array<Component>
get_output(): bool
attach_input(pin_no:int,c:Component)
evaluate()
return output_value;
inputs[pin_no]=c;
False_input
evaluate()
True_input
evaluate()
output_value=true;
Or_gate
evaluate()
And_gate
evaluate()
output_value = inputs[0].get_output() &&
inputs[1].get_output();
Nand_gate
evaluate()
Inverter
evaluate()
output_value = ! inputs[0].get_output();
Nor_gate
evaluate()
Test_point
evaluate()
cout << inputs[0].get_output() << endl;

Fig. 1

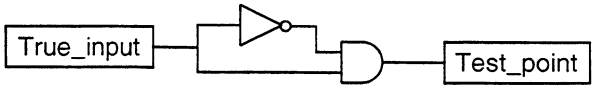
True_input
Test_point

Fig. 2

4 St Cedds College have recently installed a system of networked terminals which enable students to select and book particular seats for Formal Hall. This is implemented using CORBA. The students log onto the system and select a particular meal, and the client is then provided with an object which exposes the `BookSeats` interface, representing the seats available at that meal. The `.idl` file for this interface is shown in Fig. 3; this interface allows the seat allocation to be viewed and any seat to be reserved (by seat number). Figure 4 shows part of the C++ implementation for this object.

(a) What is the purpose of a CORBA `.idl` file, and how does the `BookSeats` interface relate to the C++ class `BookSeats_i`? [15%]

(b) If two clients try to access the server at the same time, the CORBA server creates two concurrent threads to handle the two requests. Why does the CORBA server process requests concurrently? What would happen if it didn't? [10%]

(c) Explain what could go wrong if two students simultaneously tried to reserve the same seat by calling the `book_seat` method at the same time. Using the semaphore defined in the `BookSeats_i` class, suggest a modified version of the `book_seat` method which addresses this problem. [30%]

(d) The College now wishes to add functionality to allow a student to reserve up to 5 seats at the same time. Write a new `.idl` file which extends the interface with an extra function to allow this. Design the interface in such a way that any existing clients will work with the new interface, but will not need to be recompiled. [20%]

(e) The new function is to be implemented in such a way as to avoid concurrency problems. If only per-seat semaphores were used, as defined in the `BookSeats_i` class of Fig. 4, there is a potential problem if two people try to book multiple seats at the same time. What could happen? How could this be resolved? [25%]

(Cont.

```
const long NUM_SEATS = 140;

struct BookingSheet{
  string name[NUM_SEATS];
};

interface BookSeats{
  BookingSheet get_seats();
  boolean book_seat(in long seat_no, in string name);
};
```

Fig. 3

```
class BookSeats_i : public POA_BookSeats,
                    public PortableServer::RefCountServantBase
{
public:
  BookSeats_i();
  virtual BookingSheet* get_seats();
  virtual CORBA::Boolean book_seat(CORBA::Long seat_no, const char* name);
private:
  BookingSheet my_seats;
  semaphore my_seat_mutex[NUM_SEATS];
};


CORBA::Boolean BookSeats_i::book_seat(CORBA::Long seat_no, const char* name) {
  if(my_seats.name[seat_no] == "") {            // if not currently booked
    my_seats.name[seat_no] = string_dup(name);  // copy name into the sheet
    return true;                                // and return success
  }
  else
    return false;                               // else return failure
}

[ + Other member functions not shown ]
```

Fig. 4

END OF PAPER