PAPER CODE NO.
COMP 319

EXAMINER : Mr Sebastian Coope
DEPARTMENT: Computer Science

JANUARY 2016 EXAMINATIONS ANSWERS

COMP319 : SOFTWARE ENGINEERING II

TIME ALLOWED : Two and a Half Hours

INSTRUCTIONS TO CANDIDATES

Answer ALL questions in Section A

(Section A is worth 70%)

Answer One question from Section B

(Section B is worth 30%)

Section A

Answer ALL questions in Section A. Section A is worth 70% of the marks available.

A1 Describe cross-cut concerns, the problems they cause and explain how Aspect Orientated Programming approach can be used to help overcome this problem.

[20 marks]

Cross-cut concerns are functional aspects of the code that effect most of not all the code in the program, examples of this are logging or security concerns. **[2 Marks]**

Problems: Code ends up being duplicated in many places, this is called code cloning and leads to issues with maintenance and debugging, since the same code may have be fixed many times. **[4 Marks]**

Scattering Since the concern is spread throughout the software, it is hard to know where it is being executed. **[2 Marks]**

Tangling, the code is inter-twined with code from the main concern it is applied to and other cross cut concerns and sometimes one cross cut concern may call another cross cut concern, for example security with logging. **[2 Marks]**

Aspect orientated programming brings all the functionality of one cross cut concern together in a single structure called an Aspect. **[2 Marks]**

This aspect contains the code segments that will be applied to execute the concern (called the advice code) .**[2 Marks]**

To determine where the advice code will be added to the main target code, there is defined a point cut, this is a pattern which can be based in the name of a method being executed, a particular class being the container for the code or the target for the call. [4 Marks]

The structuring of the code into AOP advice means it can be defined once and once only and be removed from the main target code making it easier to read. [2 Marks]

A2 With the aid of class diagrams or code examples, describe and explain the application of the following OO design patterns

Singleton **[5 Marks]**

A singleton is a class which only supports a single instance. The singleton class has to control its own creation and not allow client code to make new instances external to the class. This allows multiple threads to share the data in the class instance. [2 Marks]

```
public class DatabaseHelper {
   private static final DatabaseHelper INSTANCE = new DatabaseHelper();

   private DatabaseHelper() {}

   public static DatabaseHelper getInstance() {
      return INSTANCE;
   }
}
```

[2 Marks]

In the example code, the instance of the singleton instance is created when the class definition is loaded. Note the use of the private keyword on the class constructor. [1 Mark]

Memento **[5 Marks]**

The memento pattern is a software design pattern that provides the ability to restore an object to its previous state (undo via rollback). The memento does not allow external classes to view the state [1 Mark]

The memento pattern consists of 3 parts an originator, this is the class that has internal state, a memento object which stores the state and a care taker which manipulates the originator object. [2 Mark]

The originator can be asked to generate a memento class for its current state and restore its state from a memento object. The importance of the memento pattern is that is allows the object to be persisted without breaking its encapsulation. [2 Marks]

Chain of responsibility **[5 Marks]**

This is a number of classes work together to handle a message or command (call).

If the class doesn't want to handle the message, it passes the messages down to next class in the chain [2 Marks]

An example of chain of responsibility would be logging in an application, each logger can send message to the next logger in the list and depending on the level of the message and the current logging level the logger would deal with the message. The loggers can be organized into a linked list. [2 Marks]

```
abstract class Logger {
protected Logger next;

   public Logger(int level) {
      this.logger_level=level;
      setNext();
   }

   private void setNext() {
      if (lastLogger!=null) {
         lastLogger.next=this; // add this into chain
      }
      lastLogger=this;
   }

}                                   // 1 Mark
```

Double lock checking **[5 Marks]**

This is a form of thread safety which tries to reduce the amount of time needed when locking an object by first using a lock hint to determine if the lock is required. [2 Marks]

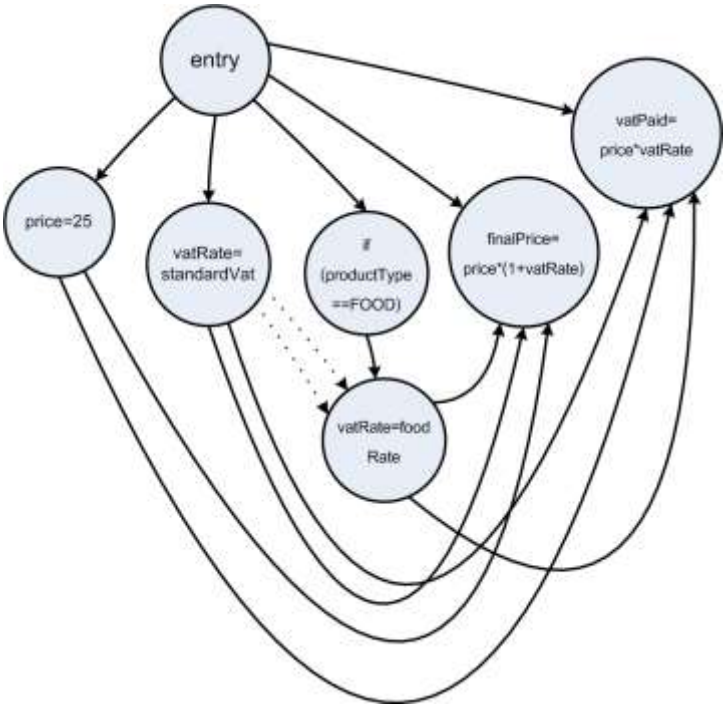
```
class Singleton {
private static Object lock=new Obect();
   private volatile instance;
   public Helper getInstance() {
      if (instance == null) {
         synchronized(lock) {
            if (instance == null) {
```

```
            instance = new Singleton();
          }
        }
      }
      return instance;
    }
}
```

[3 Marks]

A3 Look at the code shown in Appendix A.

a) Draw a dependency graph for this code **[10 Marks]**



[10 Marks]

b) Explain how the dependency graph and program slicing can be used to help when

i) Debugging the software **[5 Marks]**

The dependency graph can be used to construct a backward slice from a point of interest in the code. This allows the program to see all code which effects the data used at the point of interest. This simplifies the listing making it easier to debug.]5 Marks]

ii) Maintaining the software (handling software changes**)** **[5 Marks]**

The dependency graphic can be used to construct a forward slice from the point where the code is about to be modified. This allows the programmer to see all points where the code they are changing could have an effect and highlight points where the output can be tested.

[5 Marks]

A4 The review at the end of the second month of a software engineering project determines that there has been slippage of one month. Assume that the project must be completed on time and that the original estimates were 3 people for 12 person months. Explain what strategies are available to remedy the situation? **[10 Marks]**

The original estimate was that 3 people could complete the project in 12 person months thus the assumption was that it would take 4 months to complete. The key issue is where the delay occurred. If all the delay was due to project start-up in the first month and this is unlikely to recur, then the calculation that could be made is:
One months delay for 3 people leaving 9 person months of effort to be completed in 2 months; ceiling(9/2) is 5, so 2 extra staff are required.

Secondly if the whole estimate was uniformly low and it has taken 6 person months of effort to get one months work completed then it will take 6*3 = 18 person months to complete the entire project. With two months for completion the extra effort required is thus; 18/2 = 9 people. Thus we need 6 extra staff to complete the project. (Both these strategies have not allowed for project related training)
A final strategy might be to negotiate a lower level of project functionality that would then match project completion by the existing staff in the remaining 2 months.
[10 Marks]

Section B

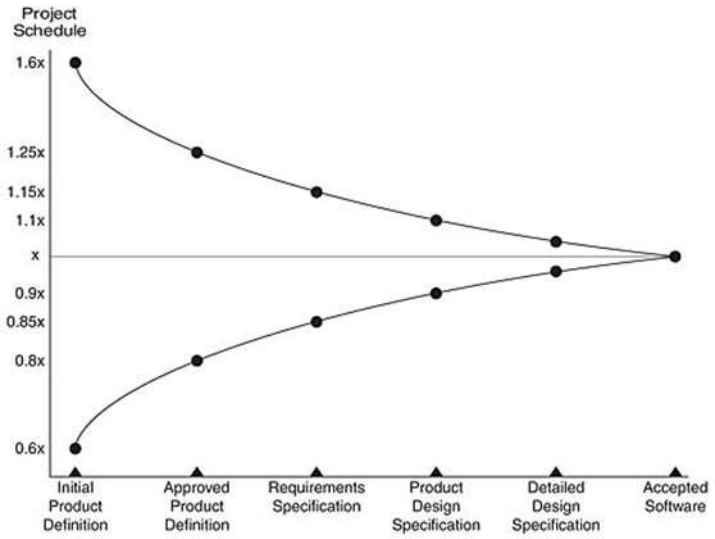
Answer ONE question from Section B.

Section B is worth 30% of the marks available.

B1 **(a)** Explain with the aid of a diagram Barry Boem's cone of uncertainty and how it is relevant to working with risk management in software estimation.

[10 Marks]

Barry Boem's cone of uncertainty (derived from real world estimate data) shows how the uncertainty of an estimate changes depending on what part of the lifecycle the estimate is carried out. Earlier estimates (at earlier stages in the lifecyle e.g. initial specification) are a lot more uncertain that estimates carries out later on. This relationship is shown in the Figure, the top and bottom limits of the graph show the range of uncertainty depending on what stage the estimate is carried out. This allows on to introduce a range of certainty to the estimation process. For example if one was doing and estimate for a product at the at the approved product definition stage, one could multiple this by 0.8 to 1.25 respectively to produce an estimated range of schedules. [5 Marks]



[5 Marks]

(b) Appendix B, shows time estimates for organisations A and B

i) Calculate EQF and bias figures for these estimates. **[12 Marks]**

EQF = 1/(Normalised Average deviation size)
Org A
Deviations are 14 4 6 16 6 1
Average = (14+4 + 6 + 16 + 6 + 1)/6 = 7.8
Normalised = 7.8. / 24 = 0.326
1/0.326 = 3.06 → EQF = 3.06
Bias average deviation = ((10+20+30+40+30+25-(24*6))) / (24*6) = 0.0764
Or 7.6% positive bias

Org B
Estimates 45, 47, 49, 52 , 50
Deviations are 5 3 1 2 0
Average = (5 + 3 + 1+ 2 + 0)/5 = 2.2
Normalised = 2.2/50 = 0.044
EQF = 22
Bias((45+47+49+52+50) – (50x5))/(50x5) = -0.028

ii) Comment on the results you obtained for i) for each organisation. **[8 Marks]**

We can see organisation A has a fairly poor EQF around 3 (which means a typical error of around 30% in there estimation values). Also organisation A has a positive bias of 7% in their estimates. This would indicate a tendency to over estimate project time. This could lead to projects being over resourced and costing for projects being consistently over estimated.
Organisation B has a lot higher performance in terms of estimation and seems to have a successful estimation technique (EQF 22). Also organisation B has a very small bias, this would imply that the estimation tends to be not effected by external pressures.
[8 Marks]

B2 Examine listing shown in Appendix C. The application is a banking application which transfers money between accounts.

i) Explain the sequence of events which could result in this code dead locking. **[8 Marks]**

If the code is called with same pair of Account objects but in reversed order by different threads. For example if one thread is calling transfer(account1,account2,am) And a second thread is calling transfer(account2,account1,am). If after locking account 1 but before locking account2, thread 2 pre-empts thread 1. Then thread 2 will get a lock on account 2 but cannot proceed because it cannot get a lock on acc2. Then both threads will be locked waiting for the other to release the lock on the other account. [8 Marks]

ii) Why is it hard to test for dead lock in applications such as shown in the example code. **[4 Marks]**

Looking at the code it can be seen the timing requirements and the data requirements for the bug are very exacting. This will happen very rarely and may be difficult for the tester to produce in a simulated environment. [4 Marks]

ii) What modifications could be made to the code to remove this problem. **[8 Marks]**

One way of getting round this locking problem would be to always lock the accounts in a fixed order. For example if the locks were applied to accounts with lower account numbers first, them the situation as shown in i) would not occur as both threads would try to lock acc1 first. [8 Marks]

iv) Explain how the Actor model could be used to provide the basic accounting services described and what the benefits would be in such a design. **[10 Marks]**

Answer should include:
Clear description of actor model, including description of Actors, messaging, mailboxes and clear explanation of how this approach avoids the problems encountered with the shared memory model. Description of how Account can be modelled as a Actor with instructions such as removal of addition to account can be carried out by sending messages to the actor.

[10 Marks]

Appendix A

```
int price=25;

float vatRate=standardVat;

if (productType==FOOD) {
        tax_rate=foodVat;
}
finalPrice=price*(1+taxRate)
vatPaid = price * taxRate;
output(finalPrice)
output(vatPaid)
```

Appendix B

Organisation A

Estimates (days)
10, 20, 30,40,30,25
Actual time to complete project
24 days

Organisation B

Estimates (days)
45, 47, 49, 52 , 50
Actual time to complete project
50 days

Appendix C

```
public void transfer(Account fromAccount, Account toAccount,   int amount) {
        synchronized (fromAccount) {

                synchronized (toAccount) {

                        if (fromAccount.hasSufficientBalance(amountToTransfer) {

                                fromAccount.debit(amountToTransfer);

                                toAccount.credit(amountToTransfer);
        }
}
```