PAPER CODE NO.
COMP 319

EXAMINER : Mr Sebastian Coope
DEPARTMENT: Computer Science

UNIVERSITY OF LIVERPOOL

JANUARY 2014 EXAMINATIONS

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 All questions in 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 (a) In software engineering what is the constraint triangle and comment on when it might be ignored? (10 Marks)

The constraint triangle described by Rucker (2003) captures implicitly the idea behind most costing and estimations methodologies and presents an equation in 3 terms that must be in balance. Graphically it is represented as a triangle where the vertices are labelled; Cost, Time, and Quality. Cost is in terms of the number of people and other resources that are required to complete the software. Quality is a measure of the number of features the software is to include and of how extensively it will be tested. Time is a measure of how long the project will take to complete. The aim in software engineering is to properly balance cost, quality and time. Control of time ensures that the project is delivered to a deadline. Control of cost means that the project is affordable. Quality control ensures that the software developed is attractive to users. Only in an ideal world may the constraint triangle be ignored. In such an ideal world a software project would have an infinite number of features, will cost nothing and be available instantly. In all other worlds, time to complete a project may be decreased by increasing personnel, reducing features and/or test quality. Cost may be reduced by using fewer programmers; but this may involve longer development times and reduction in quality. A higher level of quality may take more time and/or cost more.

(b) Explain with the aid of an example and class diagrams the application and benefits of the use of the following object patterns:

Factory **(10 Marks)**

Factory classes are classes that are used to generate instances of a class. The factory class approach has advantage over standard object creation using a constructor, is that is allows the actual object type to be deferred to run-time. For example, imagine you are creating a class which will handle and render image data on a screen. You might want a different class type depending on the image data contained, so for png format you might have a class called ImagePNG for a jpg format you could have a class called ImageJPG. This becomes a problem when creating instances of the classes, since only at runtime you will know the type of the object concerned. The solution is to define an interface, called Image, and this will be the type handled by the calling code, like so: **[6 Marks]**

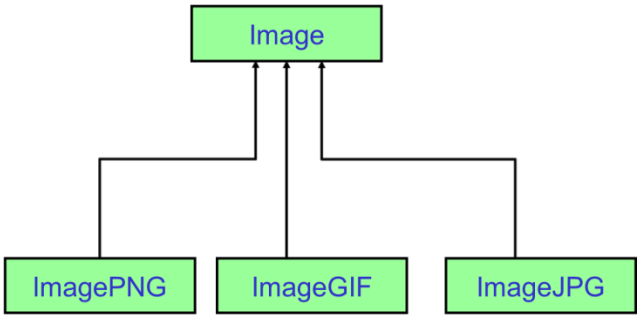
```
public interface Image {

        public Image(String filename) {};

        public drawImage(Graphics g) { };

}
```

Each particular concrete class shall implement this interface, see Figure



[2 Mark for diagram]

We can now declare a static factory method, which will return a different class instance depending on the input parameters.

```
public interface ImageFactory {
        public static createImage(String fname) throws Exception {
            if (fname.endsWith(“.gif”)) {
                return( (Image) new ImageGIF(fname) );
            }
            if (fname.endsWith(“.png”)) {
                return( (Image) new ImagePNG(fname) );
            }
            if (fname.endsWith(“.jpg”)) {
                return( (Image) new ImageJPG(fname) );
            }
            throw new Exception("Unknown image type for file "+fname);
        }
}
```

[2 Marks for code example]

Builder (10 Marks)

The builder pattern is a creational object pattern, which translates from an abstract formally defined structure to some rendering of this structure (commonly as a string). One example of this would be a builder which produced SQL query strings or output of XML or HTML. **[2 Marks] In this example, different sub-classes could be used to build different types of queries for example, SELECTS, DELETES, INSERTs etc. [2 Marks]**

The benefits of this are many:

1 The client of the builder will not have to remember the syntax of the underlying expression [1 Mark]
2 Accidental mistakes in the output format will be eliminated. [1 Mark]
3 The builder can be adapted for different syntax formats, for example between MySQL and MySQL.
4 Extra functionality can be added, for example to produce queries which can be sharded across different tables or to provide a layer of database security. [2 Marks]
5 The builder can validate arguments such as table names to ensure validation before the query is executed on the database. [1 Mark]

A2 (a) Discuss the benefits of the use of AOP, include in your answer descriptions and explanations of: cross cutting concerns, joint points, point cuts, advice and aspects. (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. [2 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 (this called the advice code)

[2 Marks]. The Aspect contains a set of patterns which define where the advice code is to be inserted into the main code, these

patterns are called pointcuts and can be defined based on the signature of a call, the accessing of data member or other execution

events such as the throwing of an exception. [4 Marks]

The pointcuts are linked to the main code in the advice section, the advice code can be run before or after the execution of the join points and it is also possible to skip over the target code. [2 Marks]

One of the problems with AOP is that it can make the main code harder to test since the structure of the actual code produced is somewhat obfuscated by the process of weaving the advice the core code. [2 Marks]

(b) Look at the code segment in show in Appendix A. For this program construct forward slices for the following statements

pension_rate=6; (5 Marks)

```
float pension_rate=6;

    float pension_amount=monthly_pay*pension_rate/100;
    taxable_pay=monthly_pay-pension_amount;

if (taxable_pay>HIGH_TAX_THRESHOLD) {
    tax=HIGH_TAX_THRESHOLD*standard_rate+(monthly_pay-
HIGH_TAX_THRESHOLD)*high_tax_rate;
}
```

high_tax_rate=0.40 (5 Marks)

```
float high_tax_rate=0.40;
tax=HIGH_TAX_THRESHOLD*standard_rate+(monthly_pay-
HIGH_TAX_THRESHOLD)*high_tax_rate;
```

(c) Explain the benefits of using backward and forward program slices when developing software. (5 Marks)
Backward slices are useful when wants to find out which statements have Effected the target statement, this is very useful for code debugging, we can construct a backward slice from the point where the data was found to be wrong and search backwards looking for the source of the bug. (3 Marks)
Forward slices allow the programmer to see what will be the effect of changing a particular line of code. So if we need to maintain the code to change a line, we can construct a slice which only contains the code that will be effected by this change. (2 Marks)

(d) Explain how the Horowitz, Prins, & Rep algorithm is used to analyse the merging of two separate pieces of code. (5 Marks)
Dependency graphs are built from each of the separate pieces of code.
[1 Mark] The two dependency graphs are then merged together using the idea of a graph union. [1 Marks] Backward program slices of the data under inspection of the merged code are then compared with all equivalent slices of the original code segments, if there is a change in the slices then the change of slice will indicate the conflicting code statements. If the slices do not change it can be assumed the code has merged without issues. [3 Marks]

Section B

Answer 1 question in Section B. Section B is worth 30% of the marks available.

B1 Two organisations are involved in the production of software and have been measuring their project development performance over a period of 3 years. Each organisation made an estimated timescale for each project The results are shown in Table 1.

Organisation	Year 1	Year 2	Year 3	Year 4	Year 5
Company A	45	50	45	50	80
Company B	79	98	95	96	95

Table 1 Percentage of projects completed within estimated time schedule with all features implemented

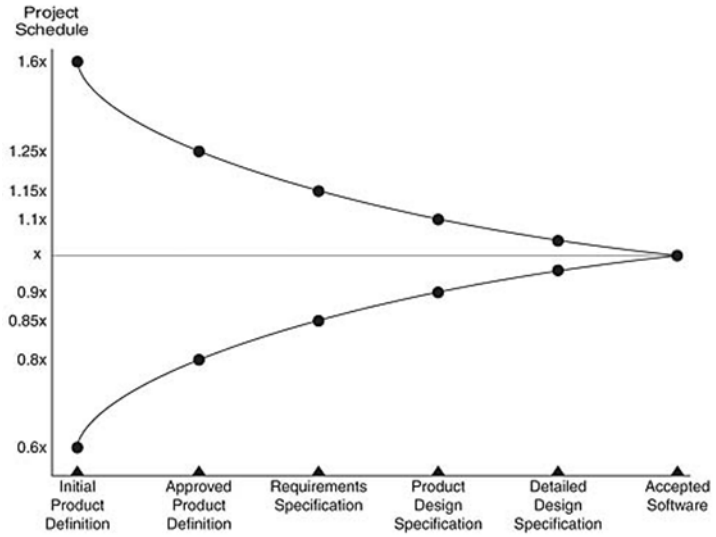
The EQF for organisation A is 10.5 and for organisation B is 2.2

a) Discuss in detail what conclusions you can draw from this data if any about the performance of each of these organisation in terms of success in delivering software projects. [15 Marks]

Answer should include:
Discussion of whether failing to deliver within estimated timescale is a good measure of performance, is this a failure to deliver or a failure to estimate.
Discussion fact that a totally unbiased estimation technique would expect to get project to overrun estimation about 50% of the time (due to the even range of errors above and below the estimation line).
Discussion of different EQF values for each organisation, with low EQF indicating a possible bias which results in the organisation performing better relative to its estimation (i.e. Company B may be systematically over-estimating the time required for each project).

b) What technique could you use to determine what if any bias was present in the estimation approach used by each of the companies? [10 Marks]

To determine bias in the estimation one would need to get hold of the raw data from the companies in terms of their estimations and there actual time to deliver the projects. They would then plot the forecast/actual times for all the projects against the actual project delivery times to get a graph. The figure below shows the typical pattern of the range of estimation values [4 Marks]



[2 Marks for diagram]

By measuring the average deviance of the estimates positive or negative from the actual delivered project time we can work out if there is a particular bias. [2 Marks]
If the estimation technique is totally unbiased we would expect this deviance to be very small with the estimations above the line to cancelling out estimations below the line.
If most of the points are above the line, this indicates a f/a high bias and means the projects are generally overestimated. [4 Marks]

c) What other approach could you use to try and accurately measure the relative software productivity of these 2 companies. [5 Marks]

Another better approach to evaluate productivity would be to get an indendent company to do size estimates of the delivered projects. This should be done by analysing the products to look at the amount of functionality delivered. One possible metric to use here could be function points. Once this has been done, a productivity metric could be produced in by looking at the staffing of the project, how many function points were delivered and how long it took. [5 Marks]

Appendix A

```
float pay=12000;

float monthly_pay=pay/12;

float pension_rate=6;

float high_tax_rate=0.40;

float standard_tax_rate=0.20;

float tax=0.0;

boolean pension=true;

if (pension) {
    float pension_amount=monthly_pay*pension_rate/100;
```

```
      taxable_pay=monthly_pay-pension_amount;
} else {
     taxable_pay=monthly_pay;
}
if (taxable_pay>HIGH_TAX_THRESHOLD) {
    tax=HIGH_TAX_THRESHOLD*standard_rate+(monthly_pay-
HIGH_TAX_THRESHOLD)*high_tax_rate;
}
```