IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2004

EEE/ISE PART II: MEng, BEng and ACGI

SOFTWARE ENGINEERING 2

Corrected Copy

Tuesday, 8 June 2:00 pm

Time allowed: 2:00 hours

There are FOUR questions on this paper.

Q1 is compulsory.
Answer Q1 and any two of questions 2-4.

Q1 carries 40% of the marks. Questions 2 to 4 carry equal marks.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : L.G. Madden, L.G. Madden
Second Marker(s) : J.V. Pitt, J.V. Pitt

This page is blank.

1. (a) There are 4 main characteristics of an Object Oriented Programming language. Identify the 2 that are:

(i) Shared with Structured Programming.
(ii) Specific to Object Oriented Programming.

[4]

(b) In Object Oriented Analysis and Design, briefly describe:

(i) What do the terms Verification and Validation mean?
(ii) How is Verification and Validation used in the context of a Use Case diagram? [4]

(c) For each of the 4 classes in Figure 1.1 specify how many data-members are in each class. [4]

(d) Using the class hierarchy shown in Figure 1.2 and the C++ statements shown below, for each of the 2 assignment statements, state whether it will compile and justify your conclusion.

```
Y *y1;
X *x1;
y1 = x1;  // assignment #1
x1 = y1;  // assignment #2
```

[3]

(e) Boehm's Spiral is described as a *meta*-process development cycle?
(i) Briefly explain the significance of the term *meta*-process.
(ii) Describe how the Boehm's Spiral maps to 2 prevalent development models.

[3]

(f) Redraw the UML activity diagram in Figure 1.3 as a UML statechart diagram.

[2]

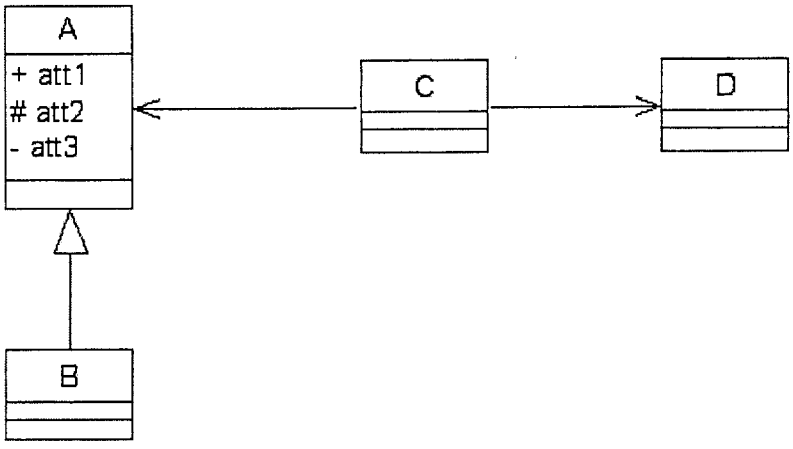


Figure 1.1

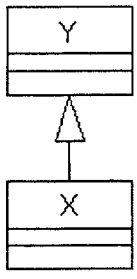


Figure 1.2

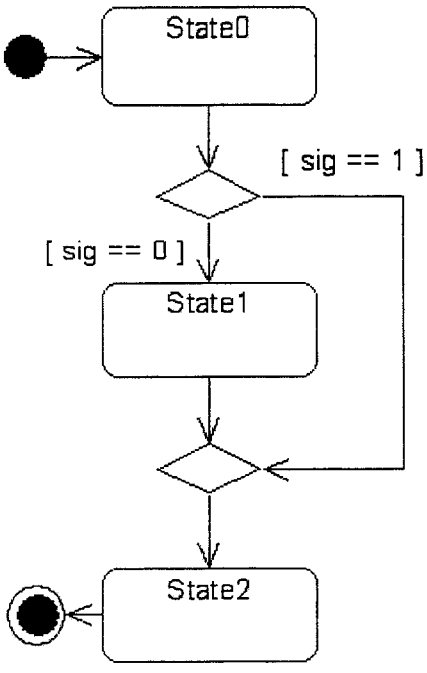


Figure 1.3

2. (a) The Ariane 5 rocket crashed 40 seconds after takeoff owing to a floating point conversion error.

(i) Which syntactic feature of C++ might have been used to avoid the crash.

(ii) Write a short extract of C++ code to demonstrate your answer.

(iii) Briefly describe what aspect of Design by Contract (DbC) would have revealed the need for aforementioned code. [5]

(b) Figure 2.1 shows a syntactically correct C++ program. You may assume that OCCF has been implemented in all classes.

(i) List the objects that are *actually* being used as polymorphic variables.

(ii) Would you expect the program to compile successfully if the definition of the member function `TFilter::getGain()` were changed to the code shown below? Briefly justify your answer.

```
virtual double getGain(double)=0;
```

[6]

(c) Figure 2.2 shows a class hierarchy involving the classes `Series`, `ArithmeticSeries`, `GeometricSeries` and `LinearSeries` which you may assume have been implemented in C++. From the perspective of the `LinearSeries` class there are 2 problems that need to be resolved.

(i) What are the 2 specific problems?

(ii) To solve the 2 problems requires some changes to the C++ code. Describe in words (or provide a C++ code extract) the changes to the C++ code required to solve each problem. [5]

(d) Figure 2.3 shows a simple iteration of an Aggregate. Write an extract of C++ code that:

(i) Creates iterators for the start and finish of the iteration.

(ii) Uses the start and finish iterators in the while statement [4]

```
class TFilter{
public:
   virtual double getGain(double){};
};

class TLowPassFilter : public TFilter{
public:
    double getGain(double){}
};

   void main(void){
     TFilter *f1, *f2, f3, *f4, *f5;
     TLowPassFilter t1, *t2=new TLowPassFilter();
     f1 = new TLowPassFilter();
     f2 = &t1;
     f3 = t1;
     f5 = t2;
   }
```

Figure 2.1

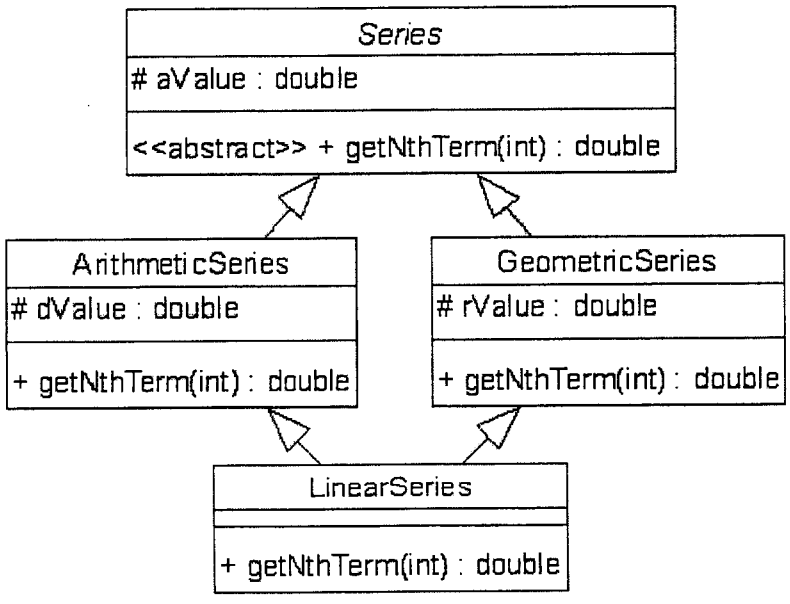


Figure 2.2

```
vector <int> v(10);
int i=0;
while (i < 10)
    cout << v[i++];
```

Figure 2.3

3. (a) Draw a Use-Case diagram from the perspective of a user of a standard laboratory Digital Multimeter (DMM).

(i) Include at least 3 Use-Cases. You can assume that the system will be implemented in software.

(ii) Make allowance for coupling (AC or DC). [4]

(b) Perform a textual analysis of the text below to identify the classes, then draw a Class-Association diagram. Include only information that can be derived from the text below:

Electronic systems may be analogue or digital. Some digital systems are composed of state machines. There are two kinds of state machine, Mealy and Moore. Each state machine has a number of states. Each state is associated with a number of input or output signals. [6]

(c) Figure 3.1 shows a fragment of a class model and Figure 3.2 shows 5 object diagrams, labelled (a), (b) etc.

In each case, explain why the object diagram is *or* is not a valid configuration of objects from the class diagram in Figure 3.1. [6]

(d) A Full-adder circuit can be constructed from 2 Half-adders and 1 OR-gate. This is expressed in 3 ways using UML in Figure 3.3, labelled (a), (b) and (c).

(i) For each figure state in words the interpretation of the UML.

(ii) Which of the 3 UML expressions is the most precise and why? [4]

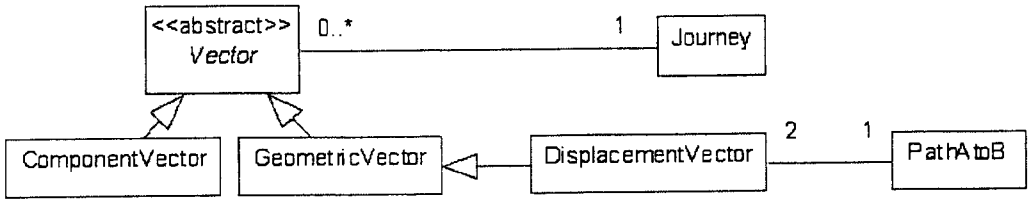


Figure 3.1

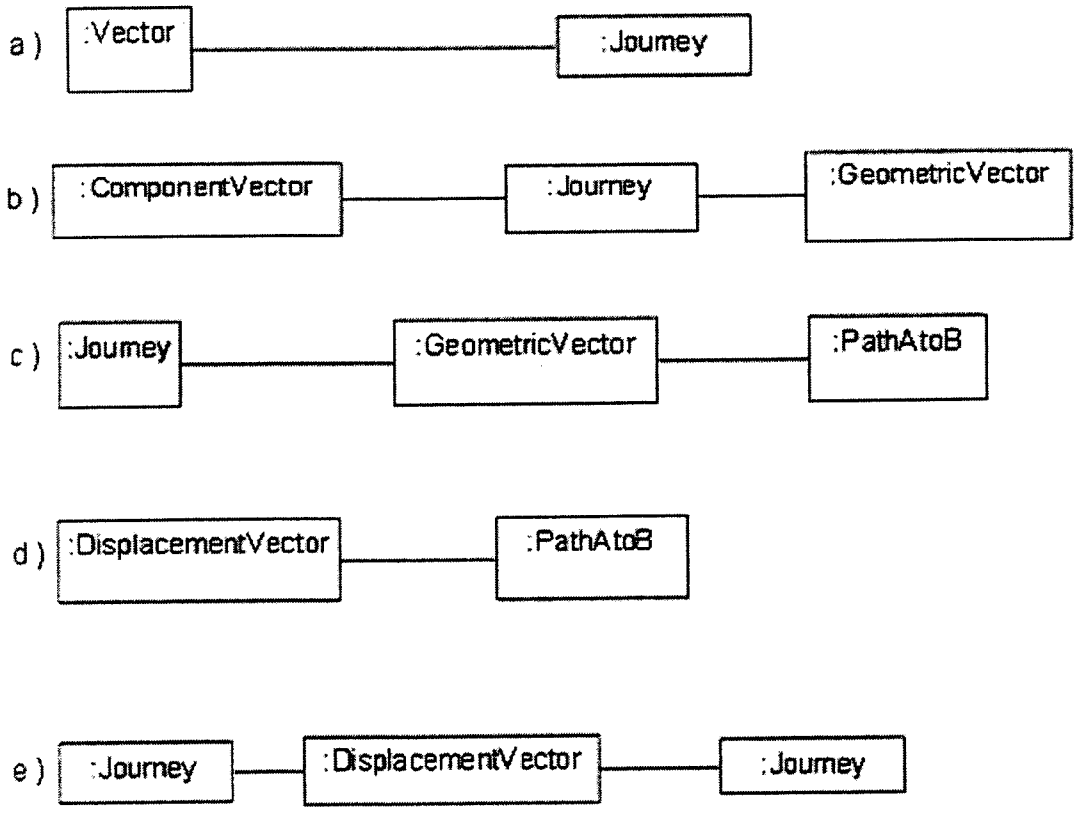


Figure 3.2

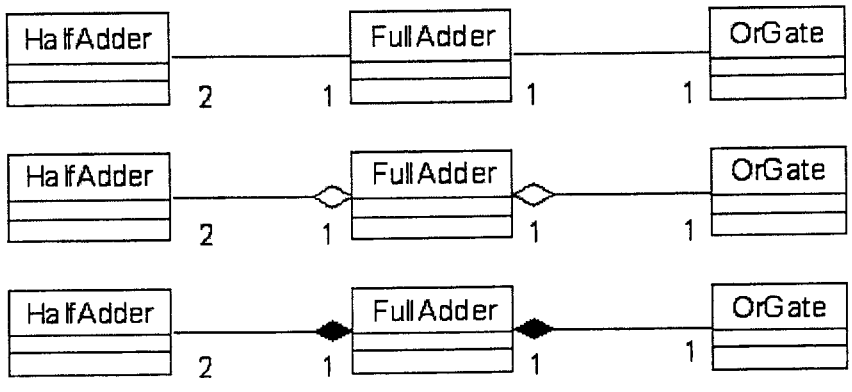


Figure 3.3

4. (a) Figure 4.1 shows a simple purely resistive circuit. Given voltage and resistor values, a program is required to calculate the currents flowing through the resistors.

Rather than just meet the given requirements, it has been decided to develop aspects of the program as general purpose components. For example, rather than just generate code to do matrix algebra with fixed dimension matrices (i.e. 2x2) and vectors (i.e. 2x1), code will be written to work with variable dimension matrices and vectors, in order to handle more complex circuits in other projects.

Briefly describe 2 other aspects of the circuit that could developed as general purpose components and could be used in other software projects. [4]

(b) Identify the 2 architectural styles shown in Figure 4.2, labelled (a) and (b). Name any 2 other architectural styles not shown in Figure 4.2 [4]

(c) Describe a solution to the requirements shown below that is based upon the MVC framework. Your answer should be brief but should clearly map elements of the requirements to named elements from the MVC framework.

- A program is required to interactively create and edit UML diagrams
- The primary mode of creating and editing the UML diagrams is through the visual interface.
- The underlying data is stored in a text format (e.g. XML).
- There is secondary text editor interface to the underlying data that may be used by a knowledgeable user.
- A change to the underlying data made using either interface should be visible instantly in the visual interface. [6]

(d) Suggest a Design Pattern that is ideally suited to meet the requirements shown below and describe how the Design Pattern would be used.

Software is required for a TV remote control that is used interactively to present textual information on the TV. Initially, the text will be in English but at some future date it is expected that the user will be able to select other languages that the information will be presented in. [6]

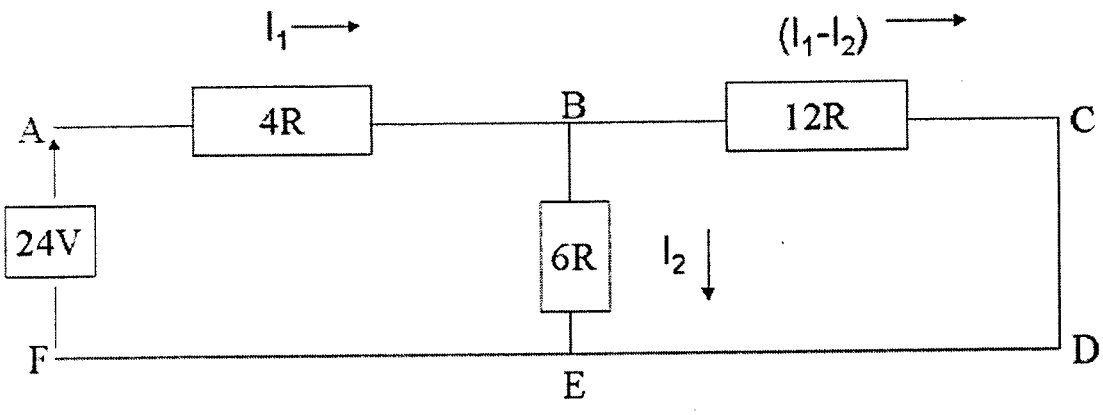
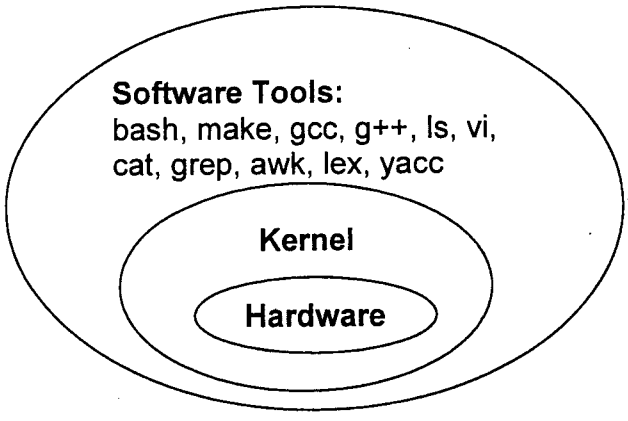


Figure 4.1

(a)



(b)

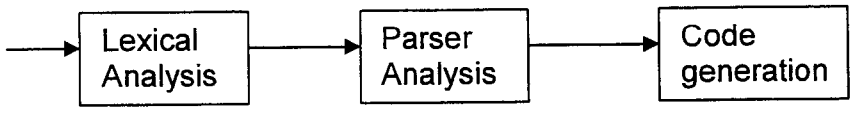


Figure 4.2

Solutions -
Software Engineering 2
2004

A1(a) "bookwork" [4]

i) Abstraction, Encapsulation [2]
ii) Polymorphism , Inheritance [2]

A1(b) "bookwork" [4]

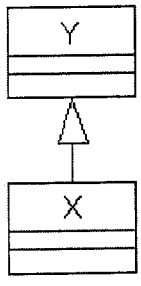
i) validation i.e. are we building the right system?
verification i.e. are we building the system right? [2]
ii) If all the scenarios are verified then the Use Case is validated [2]

A1(c) "bookwork" [4]

Class A has 1 private, 1 protected & 1 public i.e. **3** [1]
Class B inherits public and protected i.e. **2** [1]
Class C has 2 unidirectional associations i.e. **2** [1]
Class D has **0** DM [1]

A1(d) "bookwork/ new computed example" [3]

```
Y *y1;
X *x1;
y1 = x1;  // assignment #1
```



Will compile[0.5], since an instance of X can do everything an instance of Y can do (uses Liskov's principle of substitution) [1]

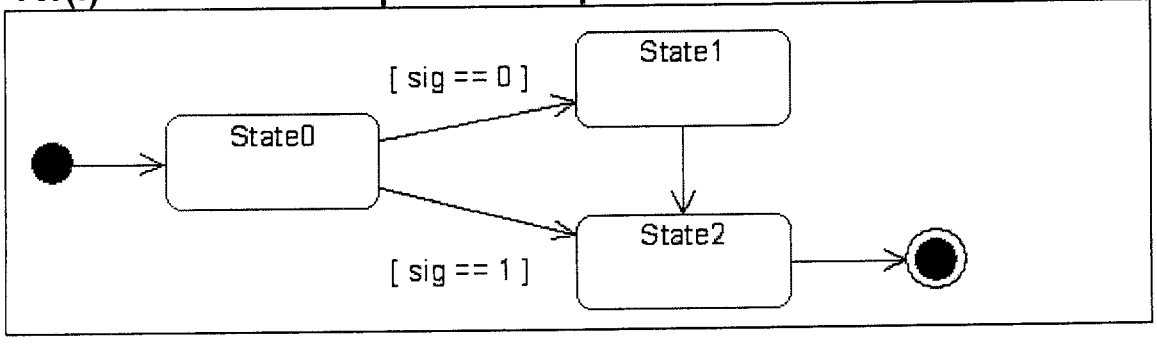
```
x1 = y1;  // assignment #2
```

Will not compile [0.5], since an instance of Y can not do everything an instance of X can [1]

A1(e) "bookwork" [3]

The numbers of iterations determines whether the model maps [1] to Waterfall (i.e. one iteration [1]) or Evolutionary/Iterative and Incremental (i.e. more than one iteration[1]).

A1(f) "new computed example" [2]



A2(a) "new computed example" [5]

(a) Use a try-catch block to trap the floating-point error [1]

(b) [2]

```
try {
    doFloatingPointConversion();
    etc . . .
}
catch (EConvertError &e) {
    doFloatingPointConversionErrorhandling(e);
```

(c) Checking of pre-conditions would have ensured the integrity of the converted data [2]

A2(b) "new computed example" [6]

```
TFilter *f1, *f2, f3, *f4, *f5;
TLowPassFilter t1, *t2=new TLowPassFilter();

f1 = new TLowPassFilter();                          [1]
f2 = &t1;                                           [1]
f3 = t1; // Not a PV since not declared as pointer
         // f4 is a PV but is not actually used
f5 = t2;                                            [1]
extra mark if only these 3 given                    [1]
```

The change causes the `TFilter` class to become pure virtual i.e. abstract [1]
Creating instances (e.g. `f1-5`) is illegal resulting in a compilation error. [1]

A2(c) "bookwork" [5]

i) The `LinearSeries` class inherits the (not overridden) data member `Series::aValue` from both the `ArithmeticSeries` and `GeometricSeries` classes. The common grandparent must be made a virtual base class to resolve the ambiguity. [1]

ii) The `LinearSeries` class inherits the (overriden) member function `getNthTerm()` from the `Series`, `ArithmeticSeries` and `GeometricSeries` classes. To resolve the ambiguity the member function is made virtual in the common grandparent and the parent classes. Then the scope resolution operator is used e.g.
`GeometricSeries:: getNthTerm()` [1]

i) Common grandparent becomes virtual base class:

```
class ArithmeticSeries: virtual public Series { etc...
class GeometricSeries: virtual public Series { etc...
```

[1]

ii) SRO used to address specific member function [1]

```
virtual double Series::getNthTerm(int i);
virtual double ArithmeticSeries::getNthTerm (int i);
virtual double GeometricSeries::getNthTerm (int i);
virtual double LinearSeries::getNthTerm (int i);
```

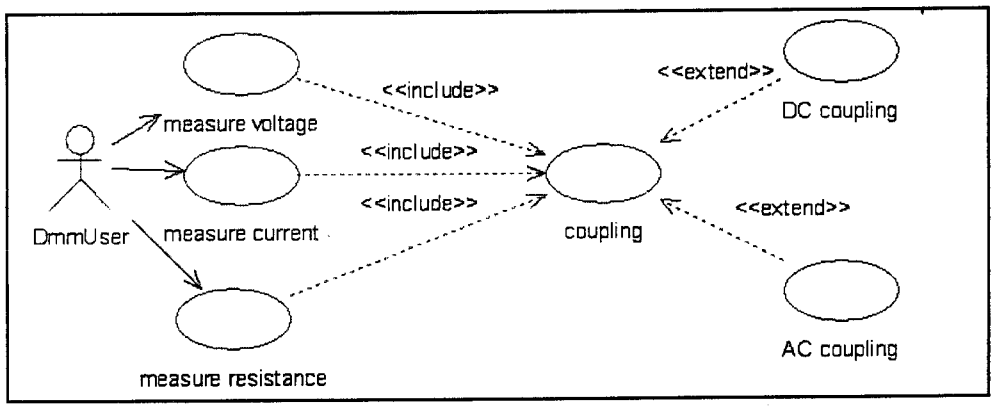
[1]

A2(d) "new computed example" [2]

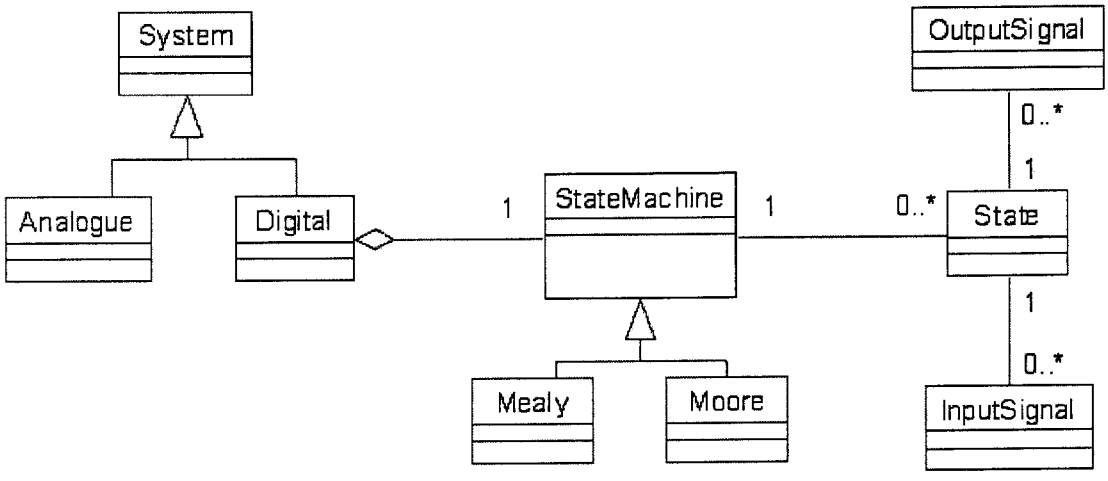
```
vector <int>::iterator start = v.begin();
vector <int>::iterator finish = v.end();                    [2]

while (start != finish)
    cout << *start++;                                       [2]
```

A3(a) "new computed example" [4]

three primary Use Case's (others: select units, set attenuation) [1]
handling coupling [2]
system boundary required since system will be implemented in software [1]



A3(b) "new computed example" [6]



Classes [3], Associations [1], Multiplicites [1], Generalisations [1]

A3(c) "new computed example" [6]

(a) Incorrect - `Vector` is abstract thus it can not have any instances [1]
(b) Correct - `Journey` is associated with many instances of concrete implementations of `Vector`. `ComponentVector` and `GeometricVector` are concrete implementations of the abstract class `Vector`. [1]
(c) Incorrect - A `GeometricVector` can only be associated with a `PathAtoB` instance if it is also a `DisplacementVector` OR `PathAtoB` is associated with *exactly* 2 instances of `GeometricVector` in the form of `DisplacementVector` [1]
(d) Incorrect - `PathAtoB` is associated with *exactly* 2 instances of `DisplacementVector` [1]
(e) Incorrect - Each concrete implementation of `Vector` (e.g. `DisplacementVector`) can only be associated with a single instance of `Journey`. [1]
extra mark for all correct [1]

A3(d) "new computed example" [4]

(i) (a) A Full-adder is associated with 2 Half-adders and one Or-gate (b) A Full-adder is an aggregation of 2 Half-adders and 1 Or-gate OR 2 Half-adders (is-part-of) and 1 Or-gate (is-part-of) a Full-adder. (c) A Full-adder is a composition (is-part-of) of 2 half-adders and 1 Or-gate. OR EXACTLY 2 Half-adders (is-part-of) and 1 Or-gate (is-part-of) a Full-adder. [3]
(ii) (c) All 3 are expressions of (a) but (c) added the extra precision in that we know that the FA is composed exactly of 2 HA's and 1 OG and nothing else. In other words the 3rd description is the only one that asserts that a FA consists of exactly 3 components. [1]

A4(a) "new computed example" [4]

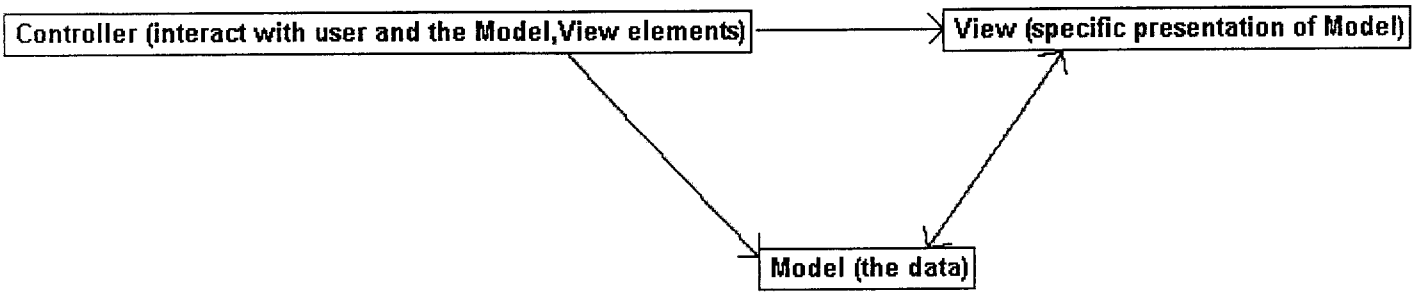
i) Complex components, current/voltage sources (AC/DC) [2]
ii) Algorithm used e.g. Kirchoff (KVL/KCL), Thevenin equivalent [2]

A4(b) "new computed example" [4]

i) Virtual [1]
ii) Data flow [1] AND • data-centred, • call and return, • independent software components. [2]

A4(c) "bookwork/new computed example " [6]

Model:	XML data representation of UML,	[2]
Views:	displayed in Textual and Visual Interface	[2]
Controller:	input from Textual and Visual Interface	[2]



A4(d) "bookwork/new computed example " [6]

Decorator (Wrapper) Design Pattern [1]

Use the Decorator Design Pattern
Problem: How can you attach additional responsibilities to an object dynamically?
Solution: Decorator (Wrapper)design pattern combines *is-A* and *has-A* relations, create an object that wraps around an existing value, adding new behaviour without changing its interface.
Discussion: ... A decorator wraps around an existing object and satisfies the same requirements (for example, is subclassed from the same parent class or implements the same interface). The wrapper delegates much of the responsibility to the original but occasionally adds new behaviour.
OO programming in Java – Timothy Budd
For example, a multi-lingual browser (similar to question):
* Browser and ChineseLanguage share the same protocol by inheritance of the interface BrowserInterface thus messages sent to either concrete class are polymorphic
* ChineseLanguage has an instance variable (i.e. an instance of Browser) which is an implementation of the interface BrowserInterface. Thus, the concrete decorator can be swapped in at run-time and will respond to the same messages sent by the client, it then do the additional processing required and send on the message that was originally destined for the Browser