UNIVERSITY OF COLOMBO, SRI LANKA

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY (EXTERNAL)

Academic Year 2022 – 3rd Year Examination – Semester 6

IT6505(R) – Middleware Architecture (Repeat)
Structured Question Paper

(TWO HOURS)

To be completed by the candidate

BIT Examination Index No: ....

Important Instructions:

- The duration of the paper is **2 (Two) hours**.
- The medium of instruction and questions is English.
- This paper has **4 questions** on **15 pages**.
- **Answer all questions.** All questions carry **equal** marks.
- **Write your answers** in English using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book, used or unused, be removed from the Examination Hall by a candidate.
- Note that questions appear on both sides of the paper.
  If a page is not printed, please inform the supervisor immediately.
- All kinds of electronic devices including calculators are **not** allowed.

Questions Answered

Indicate by a cross (×), (e.g. ☒) the numbers of the questions answered.

	Question numbers				
To be completed by the candidate by marking a cross (×).	1	2	3	4	
To be completed by the examiners:					

1)
(a) Using an example each, explain *four (4)* **reliability types (or aspects)** expected in a distributed system.
(8 marks)

ANSWER IN THIS BOX

Fault tolerance is the ability of a distributed computing system to recover from the failure of some component. A component is considered faulty once its behavior is no longer consistent with its specification. *Eg. In a cloud storage, if one server is faulty another should make the data available for the client.*

High availability is a system provides uninterrupted service in spite of failures

Consistency is the ability of a distributed computing system to coordinate failures. It behaves like a non-distributed system.

Security is the ability of a system to protect data, services and resources against unauthorized access

Privacy is the ability of a system to protect user identity and data from other users

➔ **1 mark for each definition + 1 mark for example**

(b) Distinguish between the concepts of **fault**, **error**, **failure, and a defect** in relation to a system.

(6 marks)

ANSWER IN THIS BOX

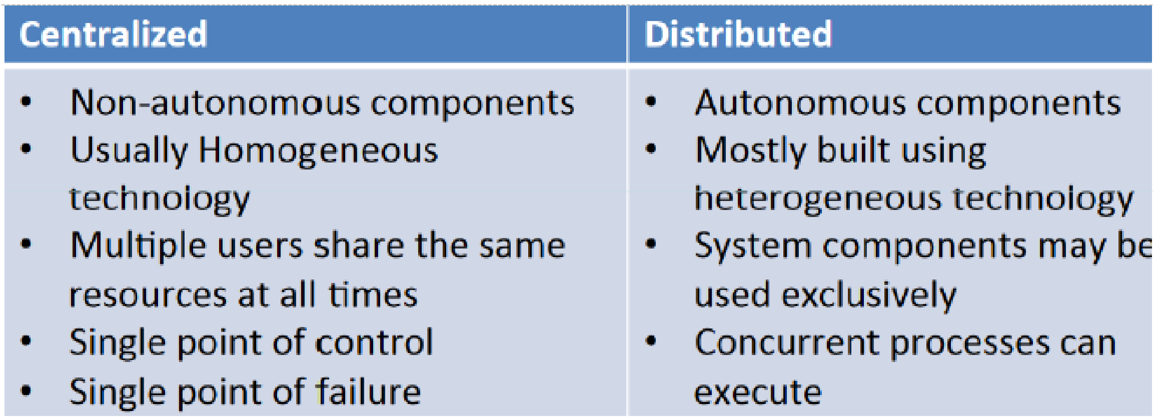
Fault: It is a condition that causes the software to fail to perform its required function.
Error: Refers to difference between Actual Output and Expected output.
Failure: It is the inability of a system or component to perform required function according to its specification.
Defect: The departure of a quality characteristic from its specified value that results in a product not satisfying its normal usage requirements. The 'Error' introduced by programmer inside the code are known as a 'Defect'.

➔ **1.5 marks for each definition**

(b) Outlining *three (3)* aspects in each, compare and contrast **Centralized Systems** and **Distributed Systems**.

(6 marks)

ANSWER IN THIS BOX



Centralized	Distributed
• Non-autonomous components	• Autonomous components
• Usually Homogeneous technology	• Mostly built using heterogeneous technology
• Multiple users share the same resources at all times	• System components may be used exclusively
• Single point of control	• Concurrent processes can execute
• Single point of failure	

(d) Outlining at least three (3) aspects, distinguish between **Operating Systems** and **Middleware**. **(5 marks)**

ANSWER IN THIS BOX

– The distinction between operating system and middleware functionality is, to some extent, arbitrary.
– Core kernel functionality can only be provided by the operating system itself.
– Middleware provides a set of specific features for a given problem domain.
- Usually, middleware sits between the Operating System and the Application Layer.
– However, some functionality previously provided by separately sold middleware is now integrated in operating systems. A typical example is the TCP/IP stack for telecommunications, nowadays included in virtually every operating system.

2.

(a) What is meant by a **Data Access Middleware**? Give an example for a data access middleware API.

(5 marks)

ANSWER IN THIS BOX

Provide a uniform interface to relational and nonrelational data using sql. Requests to access data from a dbms are sent to a data access driver rather than directly to the dbms. The data access driver converts the sql statement into the sql supported by the dbms and then routes the request to the dbms.

➔ 4 mark

The two leading data access middleware are the open database connectivity (odbc) supported by microsoft and the java database connectivity (jdbc) supported by oracle.

➔ 1 mark

(b) List *four (04)* **functions** of middleware services.

(4 marks)

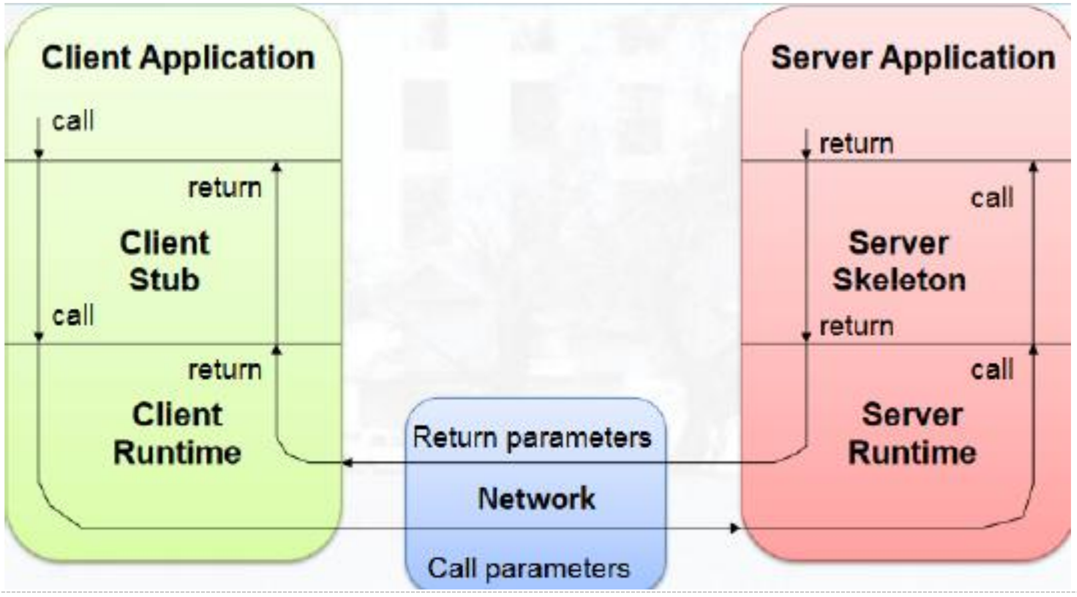
ANSWER IN THIS BOX

- Locate transparently across the network, thus providing interaction with another service or application
- Filter data to make them friendly usable or public via anonymization process for privacy protection
- Be independent from network services
- Add complementary attributes like semantics
- Be reliable and always available

(c) Draw a diagram of the typical **RPC model** and briefly describe main sequential steps of the communication process between the client and the server.

(8 marks)

ANSWER IN THIS BOX



➔ **3 mark**

1. The client calls the local stub procedure. Parameters are marshalled.
2. Networking functions in the O/S kernel are called by the stub to send the message.
3. The kernel sends the message(s) to the remote system. This may be connection-oriented or connection-less.
4. A server skeleton unmarshals the arguments from the network message.
5. The server skeleton executes a local procedure call.
6. The procedure completes, returning execution to the server skeleton.
7. The server skeleton marshals the return values into a network message.
8. The return messages are sent back.
9. The client stub reads the messages using the network functions.
10. The message is unmarshalled and the return values are set on the stack for the local process.

➔ **5 mark**

(d) In what aspects would **Remote Procedure Calls (RPC)** be different from **Local Procedure Calls (LPC)**? Explain briefly.

(4 marks)

ANSWER IN THIS BOX

While local procedure calls (LPCs) provide a mechanism for enabling different parts of an application located on a single computer to communicate with each other, RPCs involve communication between different computers.

(b) Explain the expected functionality of the following code.

(4 marks)

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface Hello extends Remote {
   void printMsg() throws RemoteException;
}
```

ANSWER IN THIS BOX

This code is brought from a Java RMI implementation. Java Remote Method Invocation (Java RMI) is a Java API that performs remote method invocation, the object-oriented equivalent of remote procedure calls (RPC), with support for direct transfer of serialized Java classes and distributed garbage-collection.

This code is defining the Remote Interface. A remote interface provides the description of all the methods of a particular remote object.

To create a remote interface the following steps to be done in general: Create an interface that extends the predefined interface Remote which belongs to the package, Declare methods that can be invoked by the client in this interface, Since there is a chance of network issues during remote calls, an exception named RemoteException may occur; throw it.

3.

(a) Explain *two (2)* differences between **COM** and **DCOM**.

(4 marks)

ANSWER IN THIS BOX

COM (Component Object Model) is a computing environment where reusable functions or the business logic of an application are bundled as a component (as a Dynamic Link library or another local process) to be invoked whenever necessary by the presentation layer of the application. COM is executed at a local level, at the client's machine. On the other hand, DCOM (Distributed Component Object Model) runs at the server end, where the client passes instructions to the DCOM object and get it executed at the server over the network.

(b) Briefly explain *two (2)* benefits of using **in-process servers (DLLs)** in DCOM.

(4 marks)

ANSWER IN THIS BOX

In-process servers (Dynamic Link Libreries or DLLs) can be dynamically load into the program when they are required.

Practical Usages:

- No additional requirement of memory at runtime or disk storage as a common binary file will be there,
- All the application or clients will be using one single version of the common code provided by the DLL,
- Operating system loads only one instance of the DLL when the first application/client loads it then for every subsequent application it shares the memory pages of the DLL with their process address space. Thus, there is no unnecessary memory overload in the system.

(c) What is meant by **Access** and **Location Transparency** in CORBA?

(4 marks)

ANSWER IN THIS BOX

Any CORBA-compliant object broker directly achieves access and location transparency.

Access transparency is achieved because the client stubs have exactly the same interface as the server objects. Hence client programs do not have to be changed when they switch from invoking a client stub to invoking the server object directly.

➔ **2 mark**

CORBA achieves location transparency because all the client needs to make an object request is a reference of the server object

➔ **2 mark**

(d) The following code segment is taken from an IDL file of a CORBA example implementation.

(i) Explain the use of **IDL** files in CORBA.

(3 marks)

(ii) What is the **intended functionality** of the below IDL file?

(4 marks)

```
struct Person
{
      long pid;
      string name;
      string middle;
      string last_name;
}

struct Account
{
      Person person;
      short age;
      double income;
}

double loanAmount;

enum cardType {AMEX, VISA, MC, DISCOVER, DINERS};
typedef sequence<cardType> CreditCards;

interface LoanAnalyzer
{
      boolean approve( in Account, in CreditCards);
}
```

ANSWER IN THIS BOX

(i)

CORBA objects are described in Interface Definition Language (IDL) files, and these IDL files are used to configure the CORBA message flow nodes. The IDL file is stored in a message set project, in a folder called CORBA IDLs.

➔ 3 mark

(ii)

The above code shows an example of using a LoanAnalyzer CORBA object. This object determines whether an applicant is approved for a loan based on the information that is supplied.

➔ 2 mark

The LoanAnalyzer CORBA interface has one method, which takes two in arguments Account and Credit Cards

➔ 2 mark

(e) Describe what is meant by **Portable Object Adapter (POA)** in CORBA.

(6 marks)

ANSWER IN THIS BOX

In CORBA, object adapter connects a request using an object reference with the proper code to service that request. The Portable Object Adapter (POA) is a particular type of object adapter that provides following functionalities:

- POA allows programmers to construct object implementations that are portable between different ORB products.
- It supports for objects with persistent identities.
- It supports transparent activation of objects.
- Associate policy information with objects.
- Allow multiple distinct instances of the POA to exist in one ORB.

4.

(a) Refer the given code below and answer the questions (i) and (ii).

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body>
  <m:GetPrice xmlns:m="https://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
</soap:Envelope>
```

i. Briefly interpret the ***SOAP request message*** given above.

(3 Marks)

ii. Write down the expected **SOAP response message** for this message.

(5 Marks)

ANSWER IN THIS BOX

(i)

The above SOAP message requests the price of apples. m:GetPrice and the Item elements above are application-specific elements. They are not a part of the SOAP namespace.

➔ 3 mark

(ii) A SOAP response could look something like this:

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="https://www.w3schools.com/prices">
    <m:Price>190</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

➔ **5 mark**

(b) Identify the most suitable **HTTP verb** to be used in the following RESTful URIs.

(5 marks)

ANSWER IN THIS BOX

URI	**HTTP Method**
/person/{id}	GET
/person/add	POST
/person/delete/{id}	DELETE
/person/getAll	GET
/person/update/{id}	PUT

(c) The following piece of code was taken from a **Controller class** in a Restful backend application.

Explain the intended **functionality of the code** given below.

(6 marks)

```
@RestController
class EmployeeController {
  private final EmployeeRepository repository;
  EmployeeController(EmployeeRepository repository) {
    this.repository = repository;
  }
  @PutMapping("/employees/{id}")
```

```
    Employee   replaceEmployee(@RequestBody   Employee   newEmployee,
   @PathVariable Long id) {
        return repository.findById(id)
      .map(employee -> {
        employee.setName(newEmployee.getName());
        employee.setRole(newEmployee.getRole());
        return repository.save(employee);
      })
      .orElseGet(() -> {
        newEmployee.setId(id);
        return repository.save(newEmployee);
      });
  }
  @DeleteMapping("/employees/{id}")
  void deleteEmployee(@PathVariable Long id) {
    repository.deleteById(id);
  }
}
```

ANSWER IN THIS BOX

@RestController indicates that the data returned by each method will be written straight into the response body instead of rendering a template.

➔ **1 mark**

An EmployeeRepository is injected by constructor into the controller.

➔ **1 mark**

There are routes for two operations (@PutMapping and @DeleteMapping, corresponding to PUT, and DELETE calls).

➔ **2 mark**

During the update if the employee Id is not existing, a new record will be created in the database.

➔ **2 mark**

(d) In the context of an MVC application, briefly explain the usefulness of a **Service** class?

(3 marks)

ANSWER IN THIS BOX

Service classes implements the business logic (methods to create, retrieve, and manipulate data) for a particular entity. E.g. User Service will provide the creating, updating, deleting, and retrieving methods for users.

(e) In the context of an MVC application, briefly explain the usefulness of a **Model class**?

(3 marks)

ANSWER IN THIS BOX

The model contains all the data-related logic that the user works with, like the schemas and interfaces of a project, the databases, and their fields.

*****