1. Atributos, Escenarios y Tácticas

1.1) (10 puntos) Describa el concepto de **Escenario**. Adicionalmente clasifique los siguientes **escenarios** en función del **atributo de calidad** al que corresponden indicando para cada uno: *Fuente, Estimulo, Artefacto, Respuesta y Medida*.

- *Un desarrollador desea cambiar el código de UI en tiempo de diseño, la modificación es realizada sin efectos secundarios en un tiempo de tres horas.*
- *Un formato de archivo (no anticipado) es recibido por un proceso del sistema encargado de leer una carpeta del S.O. El proceso audita el error e informa al operador mediante un mensaje. El sistema continúa operando archivos en forma normal.*

Solución:

*Debido a los problemas relacionados con las definiciones, taxonomías y las diferentes interpretaciones sobre los atributos de calidad de sistema, surge el concepto de **Escenario** como una "caracterización de un atributo de calidad"*

Atributo: **Modificabilidad**.
Fuente: Desarrollador
Estimulo: Desea cambiar la UI
Artefacto: Código
Respuesta: Modificación realizada sin efectos secundarios.
Medida: tres horas.

Atributo: **Disponibilidad**.
Fuente: Externa al Sistema (archivo)
Estimulo: Formato de archivo no anticipado.
Artefacto: Proceso encargado de leer.
Respuesta: Auditar e informar al operador.
Medida: El sistema no se cae.

1.2) (15 puntos) Durante el diseño de la arquitectura de una solución el arquitecto del software tomó las siguientes acciones.

a) Implementó un sistema en el cual los componentes que ejecutan en los dispositivos clientes (terminales móviles) envían al servidor un mensaje con el código "WRKING" cada 10 segundos.

b) Adquirió un producto servidor de base de datos que cuenta con un componente primario que recibe las solicitudes, las procesa e informa a otros componentes redundantes sobre las solicitudes que recibió.

c) Introdujo un componente que distribuye los mensajes que recibe el servidor entre distintas instancias del mismo.

d) Introdujo copias de determinados datos - en medios rápidos - en los componentes cliente.

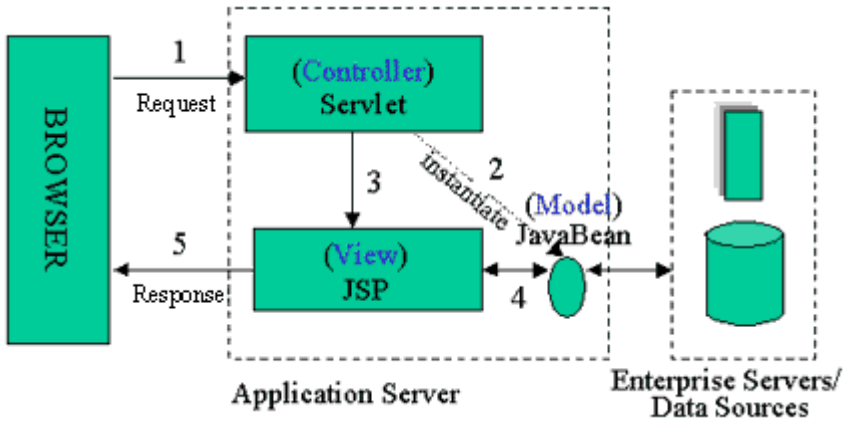
Se pide: Identifique (mediante su nombre) los mecanismos de arquitectura utilizados e indique para cada uno ante qué atributo de calidad el arquitecto se estaba enfrentando.

Solución

a) pulso – confiabilidad
b) redundancia pasiva - confiabilidad
c) Balanceador de carga – eficiencia
d) caches – eficiencia.

2. Patrones / Estilos

2.1) (10 puntos) Indique exactamente cuales son las **responsabilidades** de cada uno de los componentes del diagrama (Servlet, JSP y JavaBean) si el patrón arquitectónico es **MVC**.



***Observación**: Nótese que no es necesario conocer la implementación de estos componentes web (Servlets, JSP) para responder la pregunta.*

Solución:

MVC divide en tres grandes áreas funcionales:
--Vista (JSP): la presentación de los datos.
--Controlador (Servlet): el que atenderá las peticiones y componentes para toma de decisiones de la aplicación.
--Modelo (JavaBean): la lógica del negocio o servicio y los datos asociados con la aplicación.

El propósito del MVC es aislar los cambios. Es un patrón arquitectónico que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados componentes.
El MVC consta de:
--Una o más vistas de datos.
--Un modelo, el cual representa los datos y su comportamiento.
--Un controlador que controla la transición entre el procesamiento de los datos y su visualización.

2.2) (15 puntos) Se le encomendó el diseño de un sistema en el cual es necesario contar con un conjunto de componentes que interactúen mediante eventos. Algunos componentes son productores de información y otros consumidores de información. Se deben poder asociar a los consumidores de información --en tiempo de ejecución-- a determinados tipos de eventos. Cuando se produzca un evento de ese tipo deben recibir notificaciones de estos eventos, también en tiempo de ejecución.

Se pide:
Describe el patrón o estilo más apropiado para este tipo de problema.
Diagrama su topología y describa las responsabilidades de los componentes.

Solución
Publicador suscriptor. Referirse al material de clase por más detalle.

3. Principios de Diseño de Componentes

3.1) (10 puntos) Defina el principio de diseño **"Dependencias a cíclicas"**.
Se pide:

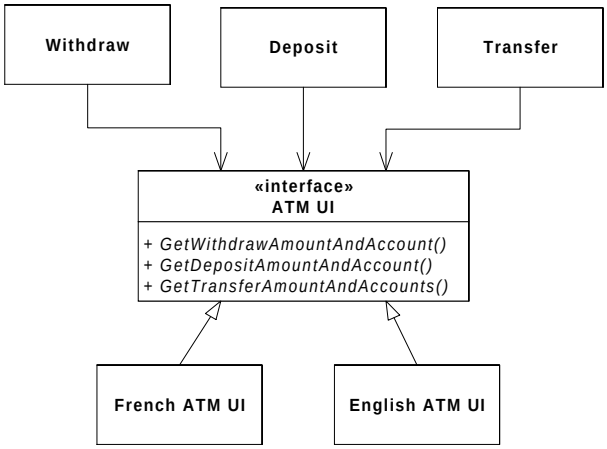
- Explique que tipos de problemas de diseño intenta resolver este principio.
- Explique detalladamente cómo se pueden eliminar los ciclos.

Solución
La estructura de dependencia entre paquetes debe formar un grafo dirigido y acíclico.
En caso que existiera ciclos el cambio se propaga, se pueden invertir las dependencias y o poner los elementos que generan los ciclos en paquetes separados

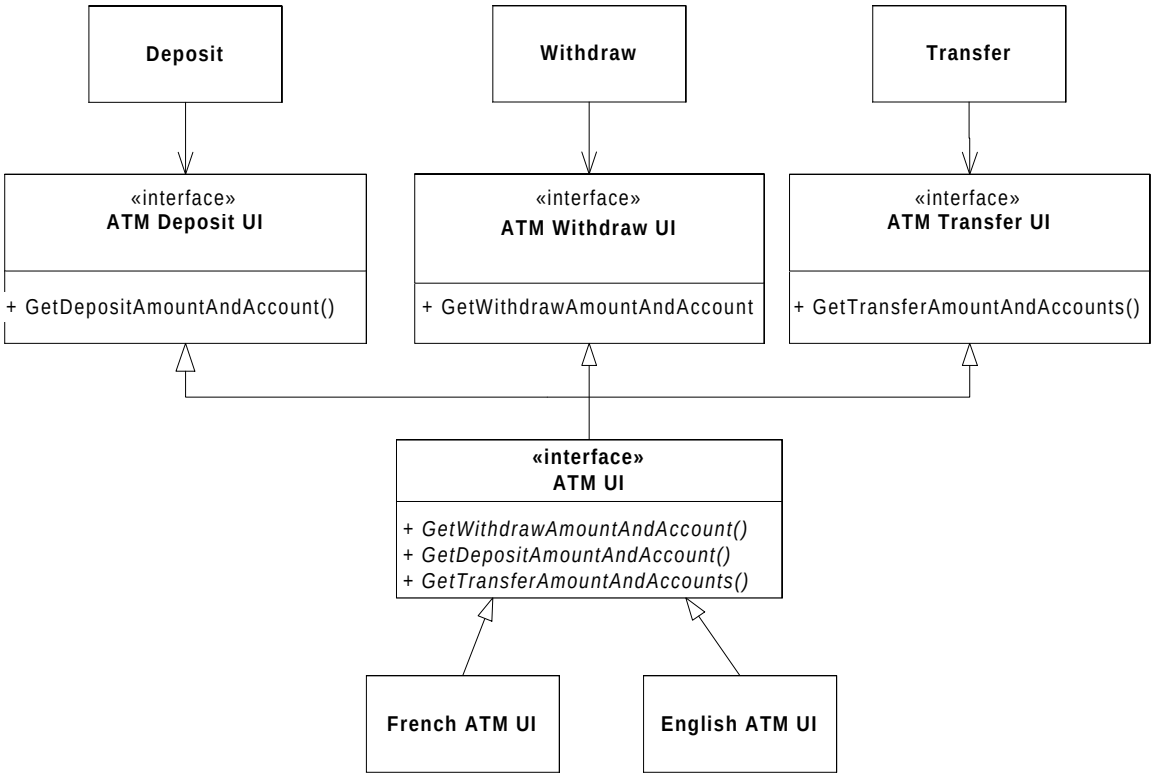
3.2) (10 puntos) El principio de segregación de interfaces se relaciona con las interfaces inapropiadas o "gordas".
Se pide:

i) Si las operaciones y la UI se empaquetaran en componentes independientes **Operación** (*Withdraw, Deposit, Transfer*) y **UI** (*ATMUI, FrenchATMUI, EnglishATMUI*), muestre un diagrama de UML completo que permita identificar las **interfaces provistas / requeridas** entre los componentes.

ii) Aplique el **principio de segregación** para la interface ATMUI desde la perspectiva de las entidades que la emplean.



Solución: Operación requiere de la interfaz UI, mientras que esta la provee.
Las interfaces provistas se anotaran mediante "Loolli pops" mientras que las requeridas, con los "sokets". Cuando se muestran juntas las provistas junto con las requeridas quedan "Ball and Basket".



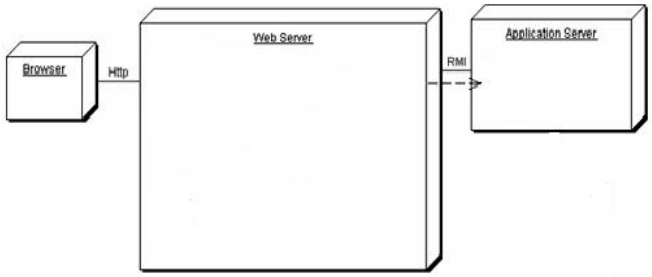
4. Enterprise Java Beans (20)

Dado el siguiente SessionBean se pide:

5.1) Bosqueje el código de las **interfaces home, remote** y anote el código necesario para invocar los métodos del Bean desde una clase Cliente.
5.2) Anote un **diagrama de clases** que considere todas las entidades de las partes anteriores.
5.3) Suponiendo que **SolicitudEmpleadoBean** esta distribuido en un nodo diferente al de los clientes que lo emplean *¿Cómo podría optimizar las llamadas a los métodos este Bean?*
5.4) ¿En que nodo tiene sentido realizar el deployment del Componente *(Browser, WebServer, Application Server)*?

```
import javax.ejb.*;
public class SolicitudEmpleadoBean implements SessionBean {

public String getNombre() {
      //.....
}
public String getApellido() {
      //.....
}
public int getNroEmpleado(int nro) {
      //.....
}
public void ejbActivate(){}
public void ejbPassivate(){}
public void ejbRemove(){}
public void ejbCreate(){}
}
```



Solución:

```
public interface SolicitudEmpleado extends EJBObject {
      public String getNombre()throws RemoteException;
      public String getApellido()throws RemoteException;
      public int getNroEmpleado()throws RemoteException;
}
```

```
public interface SolicitudEmpleadoHome extends EJBHome {
      public SolicitudEmpleado create()
                        throws CreateException,RemoteException;
}
```

```
public class Cliente {
```

```
public static void main(String[] args) {
try {

Context ictx = new InitialContext();
Object objref =
ictx.lookup("java:comp/env/ejb/SolicitudEmpleado");

SolicitudEmpleadoHome
home=(SolicitudEmpleadoHome)PortableRemoteObject.narrow(objref,SolicitudEmpleadoHom
e.class);
SolicitudEmpleado solicitud = home.create();

System.out.println(solicitud.getNroEmpleado(100));
System.out.println(solicitud.getNombre());
System.out.println(solicitud.getApellido());

}catch (Exception ex) {
      System.err.println("Caught an exception!");
      ex.printStackTrace();
}
}
}
```

II) Se infiere del código.

III) **SolicitudEmpleadoBean** podría recibir/retornar un objeto Empleado, de forma tal que no existirían invocaciones para cada uno de los atributos de un empleado; Adicionalmente tal entidad debe ser *Serializable*.

IV) Application Server.

5. .NET Remoting (10)

5.1) Indique la diferencia entre **Marshal-by-Value**, y **Marshal-by-Reference** en .Net Remoting.

5.2) Mencione los posibles **canales** de comunicación que se pueden emplear entre los clientes y el servidor en .Net Remoting.

Solución:

Marshal-By-Value	Marshal-By-Reference
El proceso Servidor crea una copia del estado del Objeto existente en el mismo y transfiere dicha copia al proceso Cliente, quien crea un nuevo Objeto basado en tal estado.	El proceso Cliente crea un representante (proxy) del Objeto real existente en el Servidor y utiliza dicho representante para acceder al Objeto.
Cualquier cambio realizado en el Objeto Servidor no afectará la estructura del Objeto Cliente.	Cualquier cambio realizado en el Objeto Servidor afectará al Cliente.
Para implementar Marshal-By-Value, se debe hacer que la Clase deseada implemente la interfaz ISerializable ó utilice el atributo [Serializable].	Para implementar Marshal-By-Reference, se debe hacer que la Clase deseada herede de System.MarshalByRefObject.

Para el resto de las respuestas, ver la Guía de Remoting del curso.

Duración: **3 horas**
Con material: **No**
Puntaje máximo: **100 puntos**