Preguntas De Finales

Pregunta 1

Usted es asignado en su nueva empresa al mantenimiento de un **sistema Enterprise** del cual depende gran parte del negocio de la empresa. Debe aprenderlo ya que además de mantenerlo deberá **incorporar nuevas funcionalidades**. Para implementar estos cambios debe **conocer el impacto** (grande, chico, nulo) que **tendrán en cada parte del sistema a extender.** Qué **métrica** utilizará para obtener **una visión primera del sistema y qué aportaría la métrica que usted proponga.**

- **Herencia ANDC:** Esta métrica indica el número de clases diferentes que se heredan entre sí, hasta la clase base. Cuanto mayor sea ese número, mayor será la profundidad de la herencia y por lo tanto mayor será la posibilidad de realizar modificaciones a la clase base para poder realizar un cambio. Si el número es bajo esto implica que el modelo es extensible, es decir, permite la incorporación de nuevas funcionalidades mientras que si es muy alto esto significa que se van a tener que realizar cambios en el modelo para poder agregar nuevas funcionalidad y eso es algo que se trata de evitar.
    - ANDC: promedio de clases base (mide is hay baja/alta uso de herencia)
    - AHH: altura jeraquica promedio (mide qué tan profundo es una jerarquia)

- **Acoplamiento:** Esa métrica mide el acoplamiento a clases únicas por medio de parámetros, llamadas a funciones, variables locales, valores devueltos en una función , instancias genéricas, interfaces, clases base, etc. [CALLS mide las llamadas a funciones, FANOUT son la cant de clases dependientes]
  Para poder tener un buen diseño se requiere de tener una alta cohesión y un bajo acoplamiento. Esta métrica nos ayuda a medir el acoplamiento, si este valor da alto eso implica que el diseño es difícil de reutilizar (no permite que se pudiera utilizar código para una nueva funcionalidad) y de mantener.
    - FANOUT / CALLS: cant de llamadas hacia clases dependientes / cantidad de llamadas a cualquier metodod
    - CALLS / NOM: cantidad de llamadas totales dividida la cantidad de métodos por clase

  NOM: la cantidad de métodos por clase

- El **tamaño**: Una métrica es para medir el tamaño de nuestro sistema que puede ser por medio de cantidad de líneas de código. Esta métrica nos permite medir la cantidad de líneas por función (se estima que no deben ser más de 20). Si este número es muy alto, se debe revisar el código para ver si hay responsabilidades mixtas o es confuso el bloque de código.
    - NOM/NOC : cantida de metodos por clase / cantidad de clase (mide el tamaño de la clase)
    - LOC/NOM: cantidad de lineas de codigo / cantidad de metodos por clase (mide el tamaño de un metodo)
    - CYCLE/LOC: número de deciciones x linas de codigo / cantidad de lineas de codigo (mide la complejidad del bloque de código)

Pregunta 2

Usted es consultado para definir la arquitectura de un producto de software que debe entre otras cosas **interactuar con sistemas ya en funcionamiento**. Estos sistemas **externos fueron definidos y pertenecen a un grupo de tecnologías determinadas**. Además y por necesidades de negocio se ha decidido lanzarlo al mercado con los requerimientos con que se cuenta a la fecha. Sin embargo, se estima que deberán **generar una segunda versión el próximo año con funcionalidades específicas para nuevos roles**. Estos deberán ser incorporados ya que por falta de tiempo y conocimiento de este aspecto del negocio no estarán incluídos en esta primera versión. Qué **criterios/patrones de diseño** propondría utilizar y que esperaría lograr con ello. Incluya un diagrama simple de paquetes o clases.

“debe entre otras cosas **interactuar con sistemas ya en funcionamiento**. Estos sistemas **externos fueron definidos y pertenecen a un grupo de tecnologías determinadas**” ->Facade

Como se debe interarcuar con sistemas que ya se encuentran en funcionamiento, un patron que podriamos utilizar es facade ya que este implementa una interfaz que nos permite aislar el nuevo sistema de los que ya existen asi no se “ensuia” el concepto del nuevo sistema y nos permite la comunicazcion entre los subsistemas.

“**generar una segunda versión el próximo año con funcionalidades específicas para nuevos roles**.”

Como se deben agrena nuevas funcionalidades, el patron que se podria usar es el decorator ya que este patron permite agregar nuevas funcionalidades a los objetos y permite mucha reusabilidad.

Quizas un microkernel? porque tenes external servers pero no se si hay más de una razón de cambio, dice lo de generar la segunda versión pero no se

Pregunta 3

- ¿Cuál es la diferencia entre un paradigma/lenguaje de **programación declarativo** vs uno **imperativo**? La **programación funcional**, a qué grupo pertenece? ¿Por qué?

  El paradigma de **programación imperativa** describe instrucciones a ejecutarse paso a paso para variar el estado del programa donde el estado final debe ser la solución del problema. Describe de forma explícita qué pasos deben llevarse a cabo y en qué secuencia para llegar a la solución final.

  El paradigma de **programación declarativa** describe el problema que se quiere solucionar. El sistema usa la descripción para hallar un algoritmo que resuelva el problema.

Los lenguajes de programación imperativa se distinguen de los lenguajes declarativos en un aspecto básico**: la programación imperativa** se centra en el **"cómo**", y la **declarativa**, en el **"qué**".

La **programación funcional** nos permite escribir código declarativo en lenguajes imperativos

- De un **ejemplo** de **programación declarativa** y el mismo en programación **imperativa**. Marque la esencia, en el código/pseudo código que presente, que represente el mecanismo de su razonamiento al enfrentar el problema de resolución.

  Ejemplo **programación declarativa**

```
SELECT name FROM people AS kids WHERE age <= 12;
```

  En este caso, indicamos lo que queremos obtener pero no nos importa como se deberia hacer.

Ejemplo **programación imperativa**

```
var people = [
  {
    name: 'Aang',
    age: 12,
  },
  {
    name: 'Sokka',
    age: 15,
  },
  {
    name: 'Toph',
    age: 12,
  },
  {
    name: 'Iroh',
    age: 55,
  }
];


var kid_names = [];

for (let person of people) {
  if (person.age <= 12) {
    kid_names.push(person.name);
  }
}

console.log(kid_names);
```

Mientras que en este caso indicamos los pasos de como debemos hacer para obtener el resultado deseado.

- En el desarrollo de un sistema ¿utilizaría ambos? ¿para qué? ¿en qué forma?

Ambas formas tienen sus ventajas y desventajas dependiendo del contexto, y es común desarrollar aplicaciones con una mezcla de lenguajes que soportan diversas formas de programar. No se recomienda que se usen en el mismo módulo, podrian utilizarse en distintos paquetes segun el contexto del problema.

https://styde.net/programacion-declarativa-vs-imperativa/

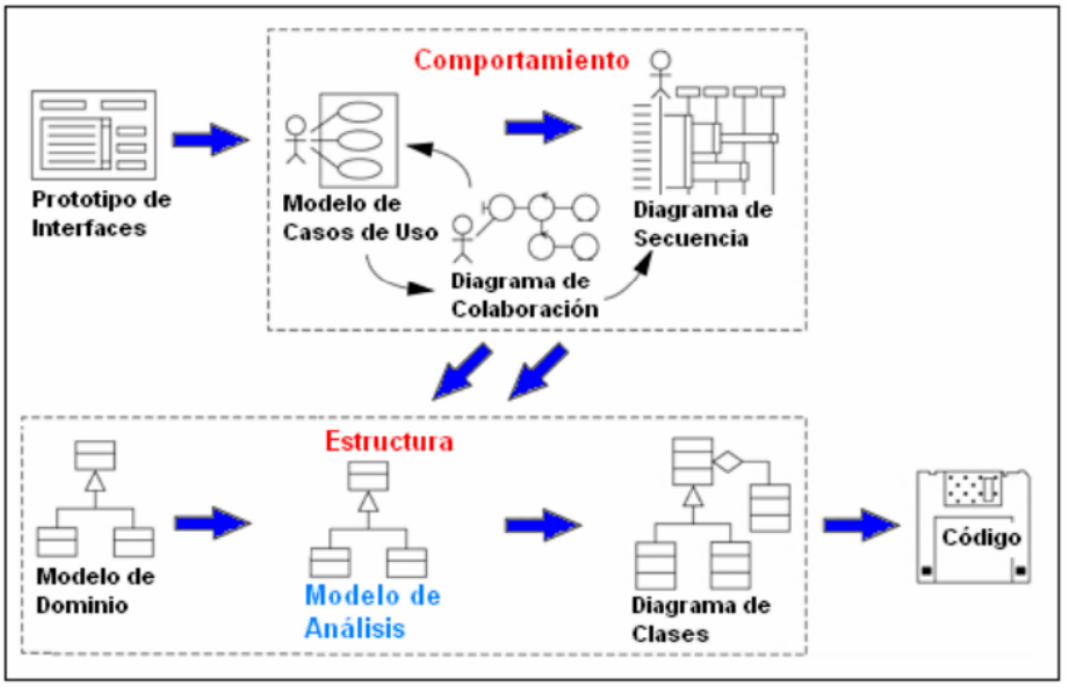
Coloquio 4

Pregunta 1

Explique en frases cortas el proceso de resolución de un problema de diseño de software utilizando programación orientada a objetos. Indique punto de partida, condiciones necesarias, criterios a aplicar y modo de validación del producto resultante.

- Como primer paso, se debe analizar la problemática del problema, entender de lo que se trata, ya que sin este paso podría ser que se tuviera una solución no deseada.
- Luego se analiza el comportamiento del problema(los requerimientos), planteando casos de usos, diagramas de secuencia, diagramas de colaboración. Esto nos permite poder entender mejor el problema.
- Una vez hecho esto, se pasa al diseño de la estructura del problema. Arrancando por un modelo de dominio, en esta etapa se particiona el problema. Luego se definen las reglas de negocio y los requerimientos hasta lograr diseñar un diagrama de clases
- Luego se pasa a la etapa de desarrollo, que es donde se implementa el diagrama de clases, es decir, se lo lleva a código.
- Finalmente testeamos/ evaluamos el resultado verificando que se cumplan todas las reglas de negocio y requerimientos.

Primero analizamos la problemática para definir las reglas de negocio y poder plantear un diagrama de dominio. Luego procedemos a plantear los requerimientos y junto con lo anterior diseñamos un diagrama de clases.
Luego pasamos a la etapa de desarrollo teniendo en cuenta lo anterior
Y finalmente evaluamos el resultado probando los casos de uso de las reglas de negocio y verificamos que se cumplan y no se viole ninguna.



Pregunta 2

Lo mismo (escribanlo ustedes) utilizando programación orientada a prototipos.
??

Pregunta 3

Para un sistema en desarrollo se ha definido la arquitectura utilizando los siguientes patrones: MVC, pipe filter, EA y layers en la capa de negocio. Según esta definición indique en qué contexto y qué problemas se buscó resolver en el sistema en cuestión con la utilización de los patrones de arquitectura mencionados

La utilización del patrón **MVC** es para cuando se tiene múltiples interfaces de usuarios que requieren diferentes funcionalidades.

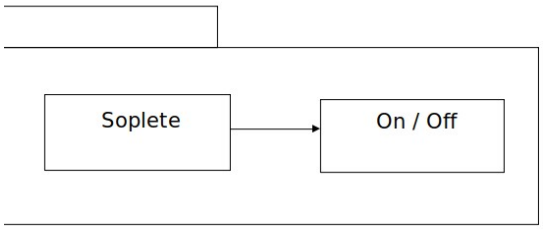
El patrón **EA** es utilizado cuando hay un sistema complejo cuya estructura probablemente sea cliente-servidor, con muchos usuarios concurrentes que requieren múltiples interfaces de usuarios, existen varias reglas de negocio y se reqequiere persistir los datos.

El patrón **Layer**, capa, se utiliza cuando hay distintos niveles de abstracción dentro de un sistema de gran volumen. Por ende hay distintas capas con distinto nivel de abstracción y cada capa utiliza un servicio de la capa superior. Permite reutilizar partes. La cantidad de las capas depende del requerimiento del problema

El patrón **pipe filter** es utilizado para distribuir el procesamientos de un stream de datos y este procesamiento se puede ir descomponiendo en N proceamientos atomicos. Este proceso brinda reusabilidad. Se van teniendo diferentes etapadas donde se aplican filtros y/o tranformaciones. Al llegar a la última etapa vamos a tener los datos riginales pero filtrados/transformados.

Pregunta 4

Para una simulación se escribió una librería de prueba y ensayo de máquinas herramientas eléctricas. Entre el código de la misma puede verse el correspondiente al diseño que se muestra más abajo. Se trata de un interruptor de tipo on/off y un soplete eléctrico para remover pintura. Ahora ha surgido la necesidad de simular una agujereadora que posee además de este tipo de interruptor uno por contacto, es decir que permanece activado mientras se oprime. Analice el diseño actual en relación con los criterios de buen diseño, modifiquelo si lo considera necesario al momento de agregar la nueva funcionalidad. Justifique cada aseveración y modificación que realice.



Con respecto al diseño actual, en vez de qué soplete dependa de la clase on/off convendria que soplete implemente una interfaz “Utilizable” que extenderian tanto el soplete y la agujereadaro como otras posibles herramientas y la interfaz interruptor donde extenderian los dos interrupciones(on/off y por contacto). De esta forma evitamos tener una dependencia de clases concetras y pasamos a tener una dependencia entre interfaces, y al mismo tiempo perimitimos agregar esta nueva funcionalidad.
A su vez como también se está agregando una agujeriadora, convendria que estén en paquetes separados ya que son cosas distintas por un lado tener el paquete que tiene las herramientas (soplete y agujereadora) y por otro lado el de interruptores, asi esto permitira que los interruptores puedan ser utilizados para otras cosas, y lo mismo las máquinas

Diagrama planteado:
https://drive.google.com/file/d/18kEjdf7qDxtfAubldmfZZHdwHdr3Gm-C/view?usp=sharing

Coloquio 1

Pregunta 1 -

- ¿Podría el patrón de arquitectura Model View Controller ser considerado un patrón de arquitectura 3TIER? ¿Podría el patrón de arquitectura Model View Controller ser considerado un patrón de arquitectura 3Layer? Si, no, ¿por qué?
  El patrón MVC no posee una arquitectura 3tier ya que este posee una arquitectura lineal mientras que el mc posee una arquitectura triangular, es decir, en mcv la vista envía actualizaciones al controlador, este actualiza el modelo, y la vista se actualiza drectamente desde el modelo mientras que en la arquiterua 3tier, el cliente no se comunica directamente con el data tier, si no que tiene que pasar pro el tier del medio.

  El patrón MVC no posee una arquitectura 3layer porque no

- ¿Cuál de los componentes del patrón MVC es más reusable? ¿por qué?

  El modelo ya que este contiene los datos y las reglas de negocio.

- El patrón de diseño de la Facade encapsula las clases que definen las interfaces que se simplifican por su uso. Indique si es Verdadero o Falso.

Verdadero, su intencion es porveer una interfaz unificada a un conjunto de interfaces en un subsistema. Este brinda una interfaz para evitar tener alto acomplamiento

- El patrón de diseño Facade introduce nuevas interfaces. Indique si es Verdadero o Falso.

  Verdadero, facade en sí es una nueva interfaz para el usuario.

- Cuando se utiliza el patrón de diseño Facade, las clases que implementan a éste son conocidas por el subsistema al cual controla el acceso. Indique si es Verdadero o Falso.

  Falso, justamente una de las intenciones de usar este patrón es evitar el acomplamiento entre el cliente y el subsistema, por lo que el cliente no conoce que hay detras de facade.

- ¿Qué patrón de diseño es utilizado conjuntamente con el patrón facade en un gran porcentaje de los diseños de software? ¿Por qué? ¿Qué función cumple?
  Segun lo que encontre facade es un “singleton” entonces se podria decir que se utilizan juntos.

- Para controlar el acceso a un subsistema cual de los patrones de diseño utilizaría, marque solo con una cruz:
  - facade X
  - adapter
  - mediador
  - proxy X

    El patrón proxy proporciona un sustituto de otro objeto para controlar el acceso al mismo

- ¿Por qué el método de actualización de la interfaz de Observadores incluye una referencia al objeto observado? ¿Tiene el observer saber qué objeto se está observando?
  confirmar esto.
  El método de actualización de la interfaz de Observadores incluye una referencia al objeto observado para que cuando a este le cambie el estado se ejecute la actualización de los observadores. 1 El observador tiene que saber que objeto está observando ya que cuando le llegue la actualización va a querer saber cual es el nuevo estado del objeto observado.

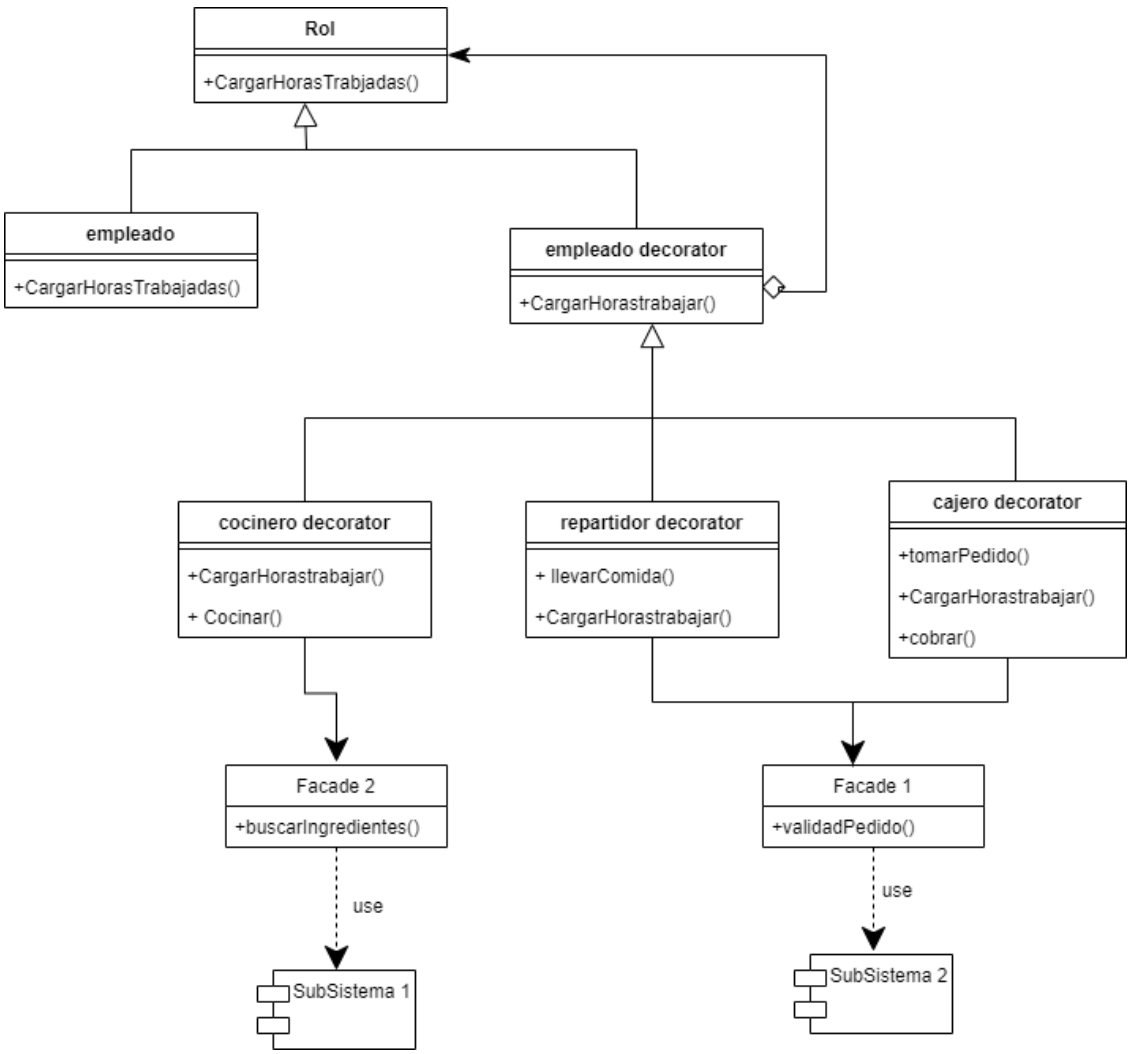
- interface Observer {
  void update (Observable model, Object object);
  }

Pregunta 2 -

Usted es consultado para definir la arquitectura de un producto de software que debe entre otras cosas **interactuar con sistemas ya en funcionamiento**. Estos sistemas externos fueron definidos y pertenecen a un grupo de tecnologías determinadas.
Además y por necesidades de negocio se ha decidido lanzarlo al mercado con los requerimientos con que se cuenta a la fecha. Sin embargo, se estima que **deberán generar una segunda** versión el próximo año con funcionalidades específicas **para nuevos roles**. Estos deberán ser incorporados ya que por falta de tiempo y conocimiento de este aspecto del negocio no estarán incluídos en esta primera versión.

Qué criterios/patrones de diseño propondría utilizar y que esperaría lograr con ello. Incluya un diagrama simple de paquetes o clases.

- Utilizaria el patrón facade ya que este patrón me permite aislar el sistema nuevo de los sistemas ya existentes y así eviatr tener alto acoplamiento.
- Utilizaria el patrón decorator ya que este patrón permite extender la funcionalidad lo cual permitiria agregar nuevas funcionalidades a nuevos roles(patrón flexible).



Pregunta 3 -

Para un sistema de asistencia en el desarrollo de software por computadora se piensa desarrollar las siguientes funcionalidades: d**etección de entidades relacionadas** y **construcción automática del modelo de dominio** usando patrones de colaboración a partir de lenguaje natural expresado por texto, **validación de la asignación de las reglas de negocio** a las entidades del modelo según la estrategia definida en clase y **validación de la consistencia lógica** entre reglas de negocio (detectar contradicciones). El sistema debe presentar una **interfaz** de edición del texto de entrada que define el dominio del problema a resolver **y vistas gráficas para cada uno de los resultados** de salida mencionados. Para procesar el texto de entrada y validar su correcta expresión según el estándard SBVR (Semantics of business vocabulary and rules specification) se piensa utilizar el **parser de la Universidad de Stanford implementado en Java**. No obstante el **procesamiento de la salida de este parser deberemos implementarlo** en el proyecto, así como la construcción del modelo con las entidades y relaciones detectadas y la validación de la asignación de las reglas. Para la validación de la consistencia lógica **se usará un SAT** Solver (Problema de satisfacibilidad booleana) implementado en Java que **operará sobre** un modelo lógico construido a partir del modelo de dominio, esta funcionalidad debe construirse. El sistema **deberá poder extenderse** agregando otras funcionalidades en forma continua como por ejemplo la generación automática de código u otras. Es decir que **deberá evolucionar con frecuencia**. El funcionamiento se pensó interactivo, **casi en tiempo real**. **El uso debe ser lo más flexible posible** en el sentido de que el usuario elegirá solo construir el modelo de dominio, o además asignar las reglas y validar su asignación o solo validar la consistencia lógica de las reglas sin ver ningún modelo o todos los anteriores. El sistema será utilizado por un analista con escasos conocimientos de programación. Se está analizando la posibilidad de implementar cada aspecto del problema que resuelve con diferentes tecnologías.

¿Cuál sería su elección? ¿Por qué? Si es necesario haga supuestos para fundamentar sus decisiones. Ayudese con un diagrama de contexto para explicar su propuesta y definir las distintas partes de la arquitectura.
Recomendación – sea breve y directo, no supere los 4 renglones para cada justificación.

Como se requiere que el sistema debe evolucionar con frecuencia(hay mas de una razon de cambio) y se debe hacer una integración con dos sistemas externos se puede utilizar un **microkernel** ya que esta permite adaptarse a cambios.
También propondría utilizar un **MVC** para manejar las distintas vistas de los distintos resultados y para la edición de texto. Este patrón es extensible por si se obtienen nuevos resultados.
Por otro lado sería conveniente usar el patrón **Layer** ya que este permite un nivel de abstracción para el desarrollo del sistema de software: construcción del modelo de dominio, construcción del modelo lógico, validación de reglas de negocio.

Como hay más de una razon de cambio ->Microkernel donde las cosas externas seran external servers y lo otro internal

Recu:

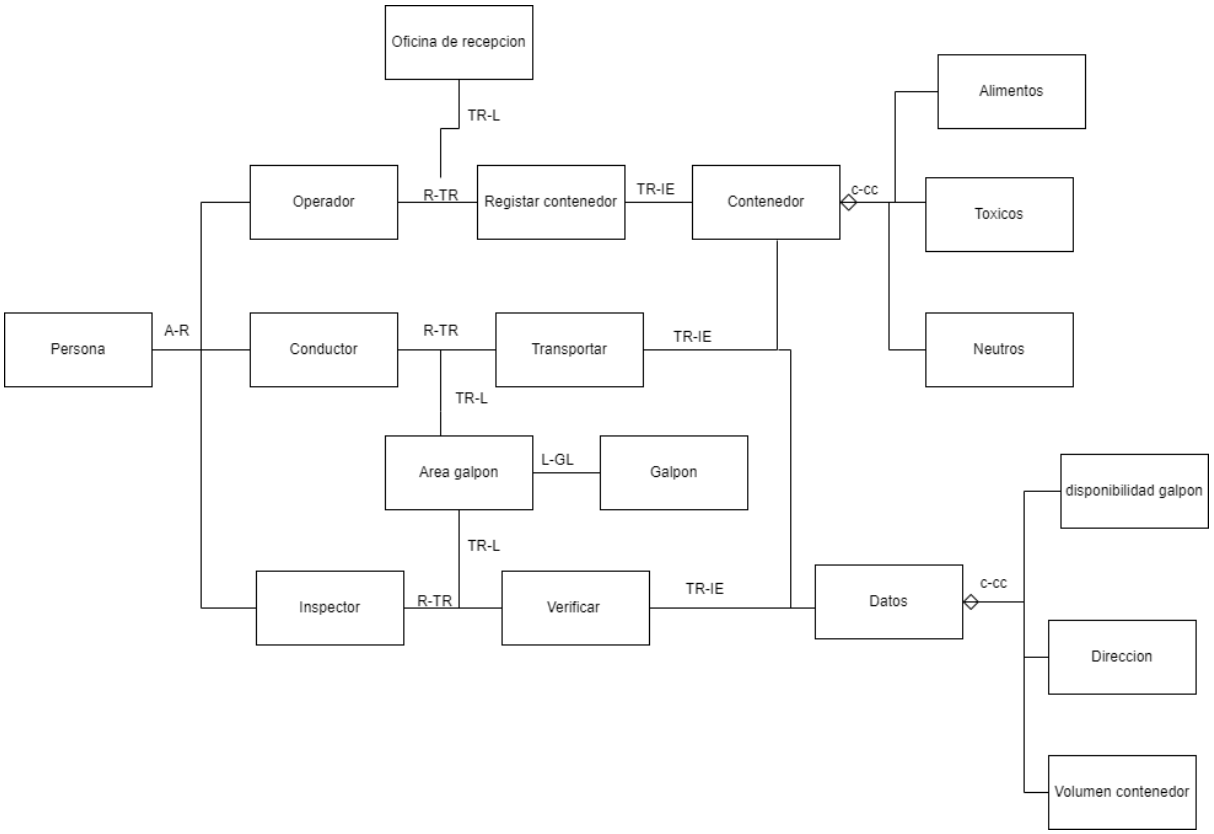
Ejercicio nro. 1

Se debe modelar el dominio del siguiente contexto de un proyecto que abarca varios problemas del cual éste es solo uno. La oficina de recepción de contenedores está atendida por un operador que debe registrar la llegada de los mismos. En estos contenedores se transportan los productos que este galpón almacena, los cuales pueden ser alimentos, tóxicos o neutros. Los contenedores se deben conducir al área del galpón que corresponda con su tipo de contenido. Se debe verificar que los contenedores poseen la dirección de despacho correcta y que coincide con la del galpón, que no estén abiertos, que posean un volumen máximo determinado y que el área correspondiente del galpón no esté completa.

En el diagrama del modelo de dominio enumere los patrones que utilice y asigne las reglas de negocio que encuentre a las clases correspondientes.

Ejercicio nro. 2

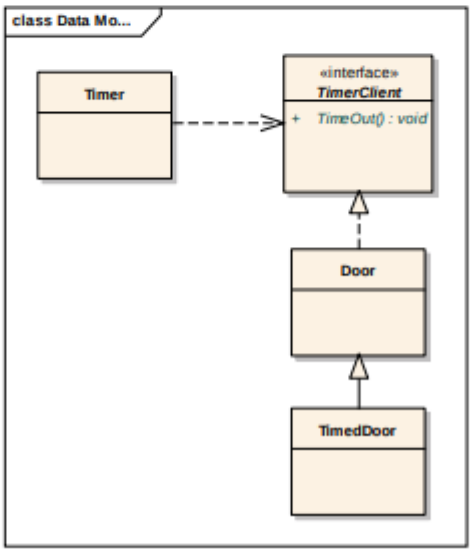
Se ha revisado la arquitectura de un sistema legacy y se ha encontrado que los técnicos que lo han desarrollado han utilizado los siguientes patrones: Microkernel, Pipe-Filter y MVC.
En base al contexto y problemas que resuelven estos patrones describa usted las características primeras del sistema en revisión.



2)

Ejercicio nro. 3

En el diagrama de abajo se muestran las clases del modelo de análisis de un negocio de seguridad en el cual es necesario modelar el problema de la administración de puertas que se abren después de un tiempo determinado (TimeOut). Con esta intensión se creó una clase (TimedDoor) que interactúa con un reloj (Timer) el cual oficia de temporizador. Sin embargo una de las clases que se muestran (Door) adolece del problema que para ser utilizadas en el mismo proyecto pero en otro contexto no son las adecuadas. Indique qué criterio de buen diseño viola y cómo lo resolvería.



Viola el diseño de principio de segregación de interfaces, es decir, puerta implementa timeout() cuando en realidad la que implementa este método es timedDoor ya que puede haber otros tipos de puerta “la puerta clásica” que no utiliza un timer sino que su mecanismo es de empuje.
Por un lado tendría la clase puerta donde timerDoor hereda de puerta(eso está bien) y luego tendria una interfaz mecanismo de la cual dor la utiliza(dependendica entre interfaces) donde timer implementa la funcion de timeout.