
**Universidad Nacional de Rosario**

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

# Resolución ejercicios IS2

*Ingeniería de Software II*

Autor:
Arroyo, Joaquín

February 23, 2024

# Contents

# 1 Introducción

Este documento fue construido a medida que se fue cursando la materia en el año 2023, por lo que puede contener errores que en un principio no eran considerados como tales.

# 2 Diseño Orientado a Objetos

## 2.1 Parcial 2021

1. Definir un diseño para los siguientes requerimientos.

   El programa le presentará al usuario un formulario donde este debe ingresar su nombre y su DNI, y cuando pulsa el botón Aceptar se debe consultar en un repositorio de datos si el DNI existe y si corresponde con el nombre ingresado. El formulario tendrá un título y habrá una etiqueta para cada campo de texto (nombre y DNI) y una para el botón (Aceptar).

   Para el diseño se deben tener en cuenta los siguientes ítems de cambio:

   - El título del formulario y las etiquetas se deben poder presentar en tres idiomas (castellano, portugués e inglés).
   - El repositorio de datos podría estar implementado con una base de datos relacional o con un archivo de texto.

   **Documentación de diseño** Se debe entregar la siguiente documentación de diseño:

   - Especificación de interfaces
   - Estructura de módulos
   - Guía de módulos
   - Estrategia de cambio que indica cómo se incorporaría un nuevo idioma

**Especificación de interfaces.**

| **Module** | **Lang** |
|---|---|
| **exportsproc** | title():String<br>fields():Iterator<String><br>button():String |
| **comments** | La definición para cada variable de estado del módulo se realiza internamente. |

| **Module** | **EsLang inherits from** Lang |
|---|---|

| **Module** | **EnLang inherits from** Lang |
|---|---|

| **Module** | **PtLang inherits from** Lang |
|---|---|

| **Module** | **Form** |
|---|---|
| **imports** | Lang |
| **exportsproc** | setLanguage(**i** Lang)<br>draw()<br>getValues():Iterator<Value> |

| **Module** | **Repository** |
|---|---|
| **imports** | DB |
| **exportsproc** | checkData(**i** String, **i** String):Bool |
| **comments** | checkData() recibe como argumento el DNI y el nombre, y utiliza DB para hacer las consultas correspondientes y devolver un resultado. |

| **Module** | **DB** |
|---|---|
| **imports** | Value |
| **exportsproc** | processQuery(**i** String):Value |

| **Module** | **Client** |
|---|---|
| **imports** | Repository, Form |
| **exportsproc** | accept() |
| **comments** | accept() se ejecuta cuando el usuario presiona el botón del formulario y muestra la respuesta. |

**Estructura de módulos.**

| **Module** | **Root** |
|---|---|
| **comprises** | Client |
| | Frontend |
| | Backend |

| **Module** | **Frontend** |
|---|---|
| **comprises** | Form |
| | Lang |
| | EsLang |
| | EnLang |
| | PtLang |

| **Module** | **Backend** |
|---|---|
| **comprises** | Repository |
| | DB |

**Guía de módulos.**

**1. Root** Módulo raíz del sistema.

**1.1 Client**

**Función.** Se encarga de la conexión con el cliente y de los efectos colaterales.

**Secreto.** Esconde como se realiza la comuniación con el ciente.

**1.2 Frontend** Contiene los módulos relacionados a la representación del formulario.

**1.2.1 Form**

**Función.** Representación del formulario.

**Secreto.** Dicha representación.

**1.2.2 Lang**

**Función.** Representar un idioma para el formulario.

**Secreto.** La representación de un idioma en cuestión.

**1.3 Backend** Contiene los módulos relacionados a la consulta de datos del sistema.

**1.3.1 Repository**

**Función.** Implementación del algorimo de consulta de DNI + Nombre.

**Secreto.** Como se implementan dichas operaciones.

#### 1.3.2 DB

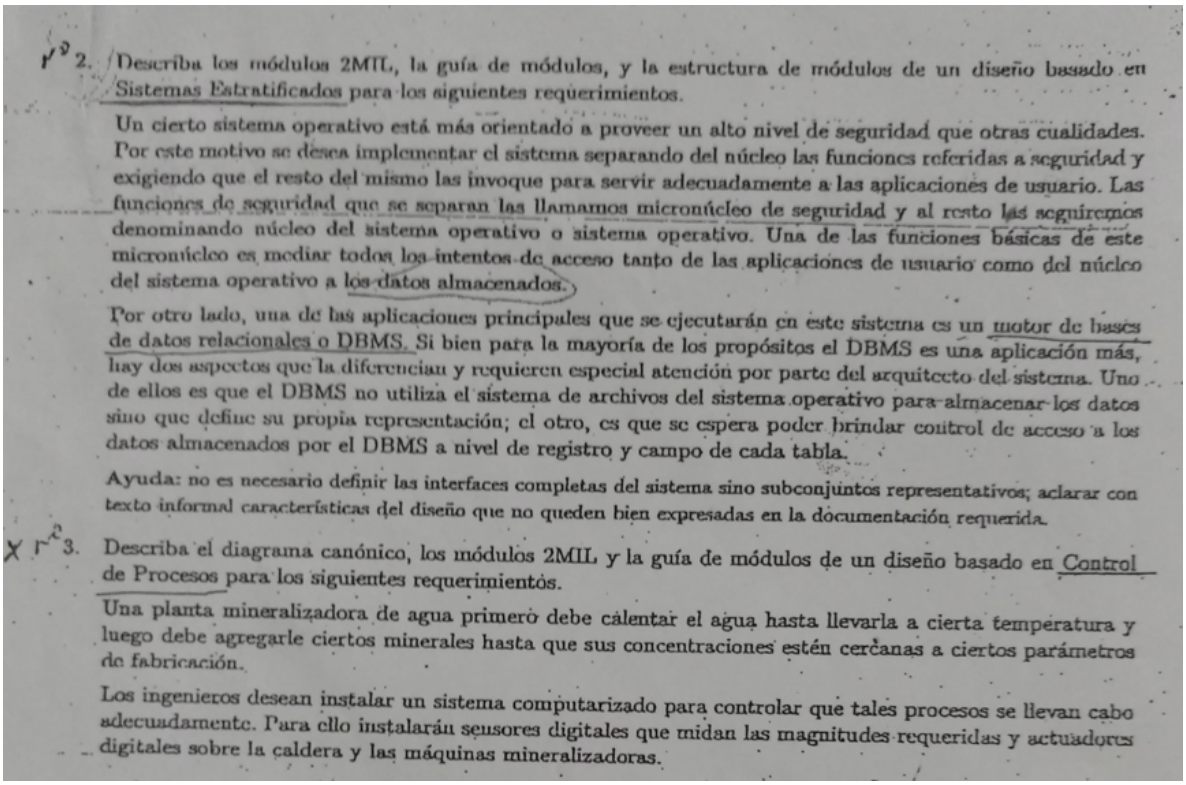
**Función.** Conexión con base de datos.

**Secreto.** Cómo se realiza la conexión con la base de datos..

**Estrategia para agregar un nuevo idioma.**

Para esto, se debe crear un nuevo módulo que herede de Lang y realizar las respectivas definiciones de su interfaz.

## 2.2 Parcial 2015



2. Describa los módulos 2MIL, la guía de módulos, y la estructura de módulos de un diseño basado en Sistemas Estratificados para los siguientes requerimientos.

Un cierto sistema operativo está más orientado a proveer un alto nivel de seguridad que otras cualidades. Por este motivo se desea implementar el sistema separando del núcleo las funciones referidas a seguridad y exigiendo que el resto del mismo las invoque para servir adecuadamente a las aplicaciones de usuario. Las funciones de seguridad que se separan las llamamos micronúcleo de seguridad y al resto las seguiremos denominando núcleo del sistema operativo o sistema operativo. Una de las funciones básicas de este micronúcleo es mediar todos los intentos de acceso tanto de las aplicaciones de usuario como del núcleo del sistema operativo a los datos almacenados.

Por otro lado, una de las aplicaciones principales que se ejecutarán en este sistema es un motor de bases de datos relacionales o DBMS. Si bien para la mayoría de los propósitos el DBMS es una aplicación más, hay dos aspectos que la diferencian y requieren especial atención por parte del arquitecto del sistema. Uno de ellos es que el DBMS no utiliza el sistema de archivos del sistema operativo para almacenar los datos sino que define su propia representación; el otro, es que se espera poder brindar control de acceso a los datos almacenados por el DBMS a nivel de registro y campo de cada tabla.

Ayuda: no es necesario definir las interfaces completas del sistema sino subconjuntos representativos; aclarar con texto informal características del diseño que no queden bien expresadas en la documentación requerida.

3. Describa el diagrama canónico, los módulos 2MIL y la guía de módulos de un diseño basado en Control de Procesos para los siguientes requerimientos.

Una planta mineralizadora de agua primero debe calentar el agua hasta llevarla a cierta temperatura y luego debe agregarle ciertos minerales hasta que sus concentraciones estén cercanas a ciertos parámetros de fabricación.

Los ingenieros desean instalar un sistema computarizado para controlar que tales procesos se llevan cabo adecuadamente. Para ello instalarán sensores digitales que midan las magnitudes requeridas y actuadores digitales sobre la caldera y las máquinas mineralizadoras.

**Ejercicio 2.**
**Items de cambio.**

- Cambio de representación de los datos almacenados.
- Cambio de procesamiento de accesos a los datos almacenados.
- Cambio de criterios para control de acceso a los datos almacenados por el DBMS.

**Especificación de módulos**

Se supone que:

- **Permit** es un tipo básico que representa a los permisos.
- **Value** es un tipo básico..

| **Module** | **DB** |
|---|---|
| | processSelectQuery(**i** String):Iterator<Value><br>processDeleteQuery(**i** String)<br>processUpdateQuery(**i** String)<br>processInsertQuery(**i** String)<br>processOtherQueries(**i** String) |
| **comments** | processOtherQueries() permite establecer permisos sobre tablas/registros. |

| | |
|---|---|
| **Module** | **DBMS** |
| **imports** | Value, Permit, Request, DB |
| **exportsproc** | processRequest(**i** Request):Iterator<Value> |
| **comments** | processRequest() recibe una request que sabe como procesar, y dependiendo los datos que esta tenga en el payload, realiza determinada acción sobre los datos almacenados a partir de la interfaz abstracta DB. Además procesa las peticiones según los permisos recibidos. |

| | |
|---|---|
| **Module** | **Request** |
| **imports** | Permit |
| **exportsproc** | setPayload(**i** String)<br>setPermit(**i** Permit) |
| **comments** | El Payload va a contener los datos correspondientes de la request (tipo, tabla, columnas, valores, etc.) |

| | |
|---|---|
| **Module** | **SystemCore** |
| **imports** | DBMS, Request |
| **exportsproc** | ... |

**Estructura de módulos**

| | |
|---|---|
| **Module** | **Root** |
| **comprises** | SystemCore<br>Persistence |

| | |
|---|---|
| **Module** | **Persistence** |
| **comprises** | Request<br>DBMS<br>DB |

**Guía de módulos**

**1. Root** Comprende los módulos principales del sistema operativo.

**1.1 SystemCore**

**Función.** Coordina los módulos que permiten el funcionamiento del sistema operativo.

**Secreto.** Algoritmo de coordinación de dichos módulos.

**1.2 Persistence** Módulo que contiene los módulos que interactúan con los datos almacenados y su seguridad.

**1.2.1 DBMS**

**Función.** Hace de enlace entre el sistema principal y sus datos almacenados.

**Secreto.** Oculta el manejo de los permisos sobre los datos almacenados.

**1.2.2 Request**

**Función.** Módulo que representa una petición. Es utilizado por el sistema principal para enviar peticiones al núcleo de seguridad.

**Secreto.** Oculta la representación de las peticiones.

**1.2.3 DB**

**Función.** Permite realizar operaciones básicas sobre los datos almacenados.

**Secreto.** Conexion con base de datos.

**Ejercicio 3.**

**Items de cambio.**

- Parámetros del agua (temperatura, concentración minerales).

- Hardware de sensores.
- Hardware de caldera.
- Hardware de máquina minelizadora.

**Especificación de módulos**

- Se supone que **Value** es un tipo basico.
- Se supone que **Mineral** es un tipo basico(enumerado) que representa a un tipo de mineral.

| **Module** | **Sensor** |
|---|---|
| **imports** | Value |
| **exportsproc** | calibrate()<br>getValue():Value |
| **comments** | Sensor pasivo. |

| **Module** | **Caldera** |
|---|---|
| **imports** | Sensor |
| **exportsproc** | on()<br>off()<br>subirTemperatura()<br>bajarTemperatura()<br>getTemperatura():Value |

| **Module** | **MaquinaMinelizadora** |
|---|---|
| **imports** | Sensor |
| **exportsproc** | on()<br>off()<br>agregarMineral(**i** Mineral, **i** Value)<br>descartarAgua(**i** Value)<br>getConcentracion(**i** Mineral):Value |
| **comments** | descartarAgua() permite descartar una determinada cantidad de agua si asi se desea. |

| **Module** | **System** |
|---|---|
| **imports** | MaquinaMinelizadora, Caldera |
| **exportsproc** | init()<br>setTemperatura(**i** Value)<br>setConcentracion(**i** Mineral, **i** Value)<br>setCheckTime(**i** Value)<br>checkWater() |
| **comments** | checkWater() se va a ejecutar segun el tiempo de chequeo establecido por default o por setCheckTime(). Este método es el que chequea la temperatura del agua y luego la concentración de los minerales. |

## 2.3 Ejercicio 1.

La celda consta de un robot y dos cintas transportadoras. El robot cuenta con dos brazos capaces de trabajar cada uno con bulto a la vez. Los bultos viajan sobre las dos cintas transportadoras, cada brazo trabaja sobre una cinta. Cada brazo y cada cinta funcionan independientemente del otro. La

tarea del robot es llevar los bultos a una prensa. La prensa solo puede prensar un bulto a la vez. La prensa tiene un sensor pasivo que inidca si esta libre o no y uno activo que señala el momento en que se levanta la prensa. El sistema debe indicarle a la prensa cuando prensar y cuando remover el bulto prensado.

**Especificación de módulos.**
**Items de cambio.**

- Hardware cinta.
- Hardware brazo.
- Hardware sensor activo.
- Hardware sensor pasivo.
- Hardware prensa.
- Proceso de coordinación de prensado.
- Proceso de coordinación entre brazo y prensa.

| **Module** | **Cinta** |
|---|---|
| **exportsproc** | encender()<br>apagar() |

| **Module** | **HBrazo** |
|---|---|
| **exportsproc** | arriba()<br>abajo()<br>izquierda()<br>derecha()<br>agarrar()<br>soltar() |

| **Module** | **Brazo** |
|---|---|
| **imports** | HBrazo, Cinta |
| **exportsproc** | irCinta(**i** Cinta)<br>irPrensa()<br>init()<br>agarrar()<br>soltar() |
| **comments** | agarrar() y soltar() delegan a HBrazo |

| **Module** | **SensorPasivo** |
|---|---|
| **exportsproc** | signal():Boolean<br>activar()<br>desactivar() |

| **Module** | **SensorActivo** |
|---|---|
| **exportsproc** | init(**i** *f())<br>activar()<br>desactivar() |
| **private** | doAction() |
| **comments** | init() recibe el callback que ejecuta doAction() cuando el sensor detecta una señal. |

| **Module** | **Prensa** |
|---|---|
| **exportsproc** | init()<br>prensar() |
| **comments** | Se supone que prensar prensa y vuelve con T unidades de tiempo. |

| **Module** | **ProcesoPrensa** |
|---|---|
| **imports** | Prensa, SensorPasivo |
| **exportsproc** | inicializar()<br>prensaLibre():Boolean |

| **Module** | **System** |
|---|---|
| **imports** | ProcesoPrensa, SensorActivo, Brazo, Cinta |
| **exportsproc** | encender()<br>apagar() |
| **private** | useArm() |
| **comments** | useArm() es el método que se envía como callback al sensor activo. |

**Estructura de módulos.**

| **Module** | **Root** |
|---|---|
| **comprises** | System<br>PressSystem<br>ArmSystem |

| **Module** | **PressSystem** |
|---|---|
| **comprises** | ProcesoPrensa<br>Prensa<br>SensorPasivo<br>SensorActivo |

| **Module** | **ArmSystem** |
|---|---|
| **comprises** | Cinta<br>HBrazo<br>Brazo |

**Guía de módulos.**
**1. Root.** Carpeta raiz del sistema
**1.1 System.**
**Función.** Coordina el sistema.
**Secreto.** Algoritmo de coordinación del sistema.
**1.2 PressSystem** Carpeta que contiene los módulos involucrados en el sistema de prensado.

#### 1.2.1 ProcesoPrensa

**Función.** Coordina el sistema de prensado y permite el acceso a su estado.

**Secreto.** Algoritmo de coordinación del sistema de prensado.

#### 1.2.2 Prensa

**Función.** Se encarga de la conexión con el hardware de la prensa.

**Secreto.** Como se realiza la conexión con el hardware de la prensa.

#### 1.2.3 SensorPasivo

**Función.** Se encarga de la conexión con el hardware del sensor pasivo.

**Secreto.** Como se realiza la conexión con el hardware del sensor pasivo.

#### 1.2.4 SensorActivo

**Función.** Se encarga de la conexión con el hardware del sensor activo.

**Secreto.** Como se realiza la conexión con el hardware del sensor activo.

**1.3 ArmSystem.** Contiene los módulos que están involucrados en el proceso de tomar objetos de una cinta.

#### 1.3.1 Cinta.

**Función.** Conexión con hardware de la cinta.

**Secreto.** Como se realiza la conexión con el hardware de la cinta.

#### 1.3.2 HBrazo.

**Función.** Conexión con hardware de brazo.

**Secreto.** Como se realiza la conexión con el hardware de un brazo del robot.

#### 1.3.3 Brazo.

**Función.** Implementa las funciones básicas que necesita el sistema con respecto al brazo a partir de la interfaz abstracta HBrazo.

**Secreto.** Oculta como se implementan dichas funciones.

## 2.4 Ejercicio 2.

Administrador de eventos. Diseñe un módulo que que represente un administrador de eventos. Los eventos que administra tienen un nombre, una cantidad de parámetros fija pero (posiblemente) de diferentes tipos. El módulo debe mantener una tabla dinámica que relaciona eventos con procedimientos a invocar.

**Especificación de módulos.**
**Items de cambio.**

- Política de llamados de Eventos.
- Parámetros de un evento.

| **Module** | **Params is** Iterator(Value) |
|---|---|

| **Module** | **Event** |
|---|---|
| **imports** | Params |
| **exportsproc** | setName(**i** String)<br>getName():String<br>addParam(**i** Value)<br>removeParam(**i** Value)<br>getParams():Params |

| **Module** | **EventsAdm** |
|---|---|
| **imports** | Event, Params |
| **exportsproc** | announce(**i** String, **i** Params)<br>suscribe(**i** String, **i** *F(**i** Params))<br>unsuscribe(**i** String, **i** *F(**i** Params)) |

**Estructura de módulos.**

| | |
|---|---|
| **Module** | **Root** |
| **comprises** | Params |
| | Event |
| | EventsAdm |

**Guía de módulos**
**1. Root** Carpeta raíz del sistema.
**1.1 Params**
**Función.** Representa a los parámetros de un evento.
**Secreto.** Oculta la cantidad de parámetros de un evento.
**1.2 Event**
**Función.** Representa a un evento.
**Secreto.** Oculta la representación de un evento.
**1.3 EventsAdm**
**Función.** Se encarga de la administración de suscripciones a eventos.
**Secreto.** Algoritmo de administración de eventos.

## 2.5 Ejercicio 3.

Museo. (Leer con atención al finalizar los requerimientos) Un museo de bellas artes desea instalar un sistema de iluminación automático que mantenga la cantidad de luz dentro de cierto rango que permite a los visitantes apreciar las obras y al mismo tiempo la luz no les produce daños serios. El museo cuenta con ventanas e iluminación artificial. A las ventanas se les colocan percianas tipo americanas que pueden ser movidas por motores eléctricos de la siguiente manera: — Las varillas horizontales pueden moverse desde su posición inicial (ventana cerrada) de a un grado hasta que el motor emite una señal que indica que ya no es posible seguir girando, y viceversa — La perciana puede elevarse o bajarse de a un centímetro hasta que el motor emite una señal que indica que no es posible seguir bajando o subiendo Por otro lado, los artefactos de iluminación artificial son modificados de manera tal que es posible regular su intensidad (aumentándola o dismunuyéndola) discretamente. Al mismo tiempo cerca de cada obra de arte se ha instalado un sensor pasivo que mide la cantidad de luz llega. En primer lugar, el sistema de software deberá mantener la cantidad de luz sobre cada obra dentro de ciertos límites; y en sengundo lugar, el sistema deberá optar primero por utilizar la luz proveniente de las ventanas y si esta no es suficiente deberá hacer uso de iluminación artificial.

1. Haga un diseño para una obra, una ventana, un sensor de iluminación, y un artefacto de iluminación regulable.
2. ¿Podría hacer un diseño BOI para el problema más general? Explique su respuesta.

**Especificación de módulos.**
**Items de cambio:**

- Rango de cantidad de luz a obras.
- Motor persianas.
- Artefactos de iluminación.
- Sensores.

| | |
|---|---|
| **Module** | **Sensor** |
| **imports** | Value |
| **exportsproc** | calibrate() |
| | getValue():Value |

| **Module** | **LuzArtificial** |
|---|---|
| **exportsproc** | init()<br>aumentar()<br>disminuir() |

| **Module** | **LuzNatural** |
|---|---|
| **imports** | MotorHorizontal, MotorVertical |
| **exportsproc** | init(**i** *F())<br>aumentar()<br>disminuir() |
| **comments** | init() recibe un callback que se invoca cuando la luz natural no se pued aumentar/disminuir mas. |

| **Module** | **MotorVertical** |
|---|---|
| **exportsproc** | init(**i** *F())<br>subir()<br>bajar() |
| **comments** | init() recibe un callback que invoca cuando el motor no puede subir/bajar más la persiana. |

| **Module** | **MotorHorizontal** |
|---|---|
| **exportsproc** | init(**i** *F())<br>abrir()<br>cerrar() |
| **comments** | init() recibe un callback que invoca cuando el motor no puede abrir/cerrar más la persiana. |

| **Module** | **Obra** |
|---|---|
| **imports** | Value, LuzArtificial, LuzNatural, Sensor |
| **exportsproc** | setBotRange(**i** Value)<br>setTopRange(**i** Value)<br>getBotRange():Value<br>getTopRange():Value<br>check() |
| **comments** | check() es el método que chequea que la obra este en los rangos definidos a partir de sus sensores. |

**Estructura de módulos.**

| **Module** | **Root** |
|---|---|
| **comprises** | Obra<br>Sensor<br>LuzArtificial<br>LuzNaturalM |

| **Module** | **LuzNaturalM** |
|---|---|
| **comprises** | LuzNatural<br>MotorVertical<br>MotorHorizontral |

**Guía de módulos.**
**1. Root** Carpeta raíz del sistema
**1.1 Obra**
**Función.** Representa a una obra.
**Secreto.** Algoritmo de calibración de nivel de luz a obra.
**1.2 Sensor**
**Función.** Conexión con hardware de sensor.
**Secreto.** Como se realiza dicha conexión.
**1.3 LuzArtificial**
**Función.** Conexión con hardware de lámpara.
**Secreto.** Como se realiza dicha conexión.
**1.4 LuzNaturalM** Contiene los módulos que se encargan de la regulación de la luz natural sobre una obra.
**1.4.1 LuzNatural**
**Función.** Se encarga de regular la luz natural sobre una obra.
**Secreto.** Algoritmo de regulación.
**1.4.2 MotorVertical**
**Función.** Conexión con hardware del motor vertical de la persiana.
**Secreto.** Como se realiza dicha conexión.
**1.4.3 MotorHorizontal**
**Función.** Conexión con hardware del motor horizontal de la persiana.
**Secreto.** Como se realiza dicha conexión.

# 3 Patrones de Diseño

## 3.1 Ejercicio 3, 4, 5 y 7

Aplicar patrones *Composite*, *Iterator*, *Visitor* y *Strategy* sobre el próximo ejercicio.

> **Mantenimiento de catálogos.** Una compañía de venta por correo ha decidido establecer una serie de almacenes regionales relativamente autónomos. Cada uno de ellos adquiere responsabilidad local para la gestión de catálogos y el procesamiento de pedidos. Cada almacén mantiene catálogos lo más adecuados posibles al mercado local. La línea específica de productos que gestiona cada almacén pude cambiar de una región a otra; además, la línea de productos gestionada por una región dada tiende a actualizarse una vez al año según los gustos de los consumidores. La compañía desea tener un sistema de seguimiento de catálogos y pedidos común para todos los almacenes. Las funciones claves del sistema incluyen:
>
> — Gestionar mercancías al entrar al almacén, expedidas por distintos proveedores.
> — Gestionar los pedidos según se reciben de una organización de venta a distancia, central pero remota; los pedidos también pueden recibirse por correo, y se procesan localmente.
> — Gestionar los pedidos de informes hechos por los clientes.
> — Generar listas de embalaje, utilizadas para dirigir al personal del almacén en la confección y despacho de un pedido.
> — Generar facturas y controlar las facturas recibidas.
> — Generar peticiones de suministro y controlar las facturas a pagar.
> — Utilidad de generación de informes general y abierta.

**COMPOSITE.** Definir con detalle la estructura de los catálogos. Cada producto ofertado en un catálogo tiene las siguientes características: nombre, código, descripción, varios niveles de secciones (por ejemplo, Ferretería, Tornillos y Bulones, Tornillos, Tornillos de acero, Tornillos de 1,5mm, etc.), precio, marca, color, etc.

| | |
|---|---|
| **Module** | **CatalogueComponent** |
| **exportsproc** | search()<br>addChild(**i** CatalogueComponent)<br>delChild(**i** CatalogueComponent)<br>getChild(**i** Int) |

| | |
|---|---|
| **Module** | **Catalogue inherits from** CatalogueComponent |

| | |
|---|---|
| **Module** | **Section inherits from** CatalogueComponent |

| | |
|---|---|
| **Module** | **Product inherits from** CatalogueComponent |
| **exportsproc** | name():String<br>id():Long<br>description():String<br>price():Int<br>brand():String |

| | |
|---|---|
| **Module** | **Store** |
| **imports** | CatalogueComponent, ... |
| **exportsproc** | ...<br>getCatalogue():CatalogueComponent |

| | |
|---|---|
| **Pattern** | **Catalogue** |
| **based on** | Composite |
| **because** | Permite tratar los distintos tipos de objetos del catálogo de forma uniforme.<br>Permite expandir/modificar la estructura del catálogo de forma sencilla. |
| **where** | Cliente **is** Store<br>Componente **is** CatalogueComponent<br>Compuesto **is** Catalogue<br>Compuesto **is** Section<br>Hoja **is** Product<br>operacion() **is** search()<br>anadir() **is** addChild()<br>eliminar() **is** delChild()<br>obtenerHijo() **is** getChild() |

**ITERATOR.** Aplicar el patrón para recorrer la estructura de un catálogo.

| | |
|---|---|
| **Module** | **CatalogueIterator is** Iterator(CatalogueComponent) |

**NOTA:** Se extiende el módulo Store

| | |
|---|---|
| **Module** | **Store** |
| **imports** | CatalogueComponent, CatalogueIterator, ... |
| **exportsproc** | ... |
| | getCatalogue():CatalogueComponent |
| | createIterator():CatalogueIterator |

| | |
|---|---|
| **Pattern** | **CatalogueIterator** |
| **based on** | Iterator |
| **because** | Permite ocultar cómo se recorre el árbol del catálogo. |
| **where** | Iterador **is** Iterator |
| | IteradorConcreto **is** CatalogueIterator |
| | AgregadoConcreto **is** Store |

**VISITOR.** Defina operaciones de búsqueda por precio, código, sección y palabra en la descripción aplicando el patrón Visitor sobre cada catálogo. Muestre código de ejemplo de cómo se implementaría una de las búsquedas.

| | |
|---|---|
| **Module** | **CatalogueVisitor** |
| **imports** | Catalogue, Section, Product |
| **exportsproc** | visitCatalogue(**i** Catalogue) |
| | visitSection(**i** Section) |
| | visitProduct(**i** Product) |

| | |
|---|---|
| **Module** | **VisitByPrice inherits from** CatalogueVisitor |

| | |
|---|---|
| **Module** | **VisitByCode inherits from** CatalogueVisitor |

| | |
|---|---|
| **Module** | **VisitBySection inherits from** CatalogueVisitor |

| | |
|---|---|
| **Module** | **VisitByKeyword inherits from** CatalogueVisitor |

**NOTA:** Se extiende el módulo CatalogueComponent y Store pasa a importar el Visitor.

| | |
|---|---|
| **Module** | **CatalogueComponent** |
| **imports** | CatalogueVisitor |
| **exportsproc** | search() |
| | addChild(**i** CatalogueComponent) |
| | delChild(**i** CatalogueComponent) |
| | getChild(**i** Int) |
| | accept(**i** CatalogueVisitor) |

| | |
|---|---|
| **Pattern** | **CatalogueVisitor** |
| **based on** | Visitor |
| **because** | Preserva la estructura ya funcional del catálogo.<br>Permite añadir funcionalidades al catálogo sin modificar su interfaz. |
| **where** | Visitante **is** CatalogueVisitor<br>VisitanteConcreto **is** VisitByPrice<br>VisitanteConcreto **is** VisitByCode<br>VisitanteConcreto **is** VisitBySection<br>VisitanteConcreto **is** VisitByKeyword<br>Elemento **is** CatalogueComponent<br>ElementoConcreto **is** Catalogue<br>ElementoConcreto **is** Section<br>ElementoConcreto **is** Product |

**STRATEGY.** La empresa desea que los catálogos puedan mostrarse ordenados de diferente forma. Por ejemplo, orden alfabético de productos, de secciones de nivel 1, por precio, etc.

| | |
|---|---|
| **Module** | **DisplayCatalogueStrategy** |
| **exportsproc** | display() |

| | |
|---|---|
| **Module** | **DisplayAlph inherits from** DisplayCatalogueStrategy |

| | |
|---|---|
| **Module** | **DisplayBySection inherits from** DisplayCatalogueStrategy |

| | |
|---|---|
| **Module** | **DisplayByPrice inherits from** DisplayCatalogueStrategy |

**NOTA:** Se extiende el módulo Store.

| | |
|---|---|
| **Module** | **Store** |
| **imports** | CatalogueComponent, CatalogueIterator, DisplayCatalogueStrategy, ... |
| **exportsproc** | ...<br>getCatalogue():CatalogueComponent<br>createIterator():CatalogueIterator<br>setDisplayCatalogueStrategy(**i** DisplayCatalogueStrategy) |

| | |
|---|---|
| **Pattern** | **DisplayCatalogueStrategy** |
| **based on** | Strategy |
| **because** | Permite implementar fácilmente distintas maneras de recorres/mostrar el catálogo.<br>Permite que los distintos algoritmos de la estrategia puedan ser cambiados en tiempo de ejecución. |
| **where** | Contexto **is** Store<br>Estrategia **is** DisplayCatalogueStrategy<br>EstrategiaConcreta **is** DisplayAlph<br>EstrategiaConcreta **is** DisplayBySection<br>EstrategiaConcreta **is** DisplayByPrice |

## 3.2 Ejercicio 8 y 10

Aplicar patrones *AbstractFactory* y *Decorator* sobre el problema del banco de la práctica de diseño.

**ABSTRACT FACTORY.** Suponga que el banco ha comprado otros bancos. Cada banco comprado tiene su propia forma de definir una cuenta (caja de ahorro, cuenta corriente, en pesos y dólares), plazo fijo, cliente, etc. Estudie las posibilidades de aplicar (y eventualmente aplique) el patrón *AbstractFactory* para que el sistema compuesto cree los diferentes productos.

| | |
|---|---|
| **Module** | **CA** |
| **exportsproc** | id():Long<br>getSaldo():Int<br>extraer(**i** Int)<br>depositar(**i** Int) |

| | |
|---|---|
| **Module** | **Banco1CA inherits from** CA |

| | |
|---|---|
| **Module** | **Banco2CA inherits from** CA |

| | |
|---|---|
| **Module** | **CC** |
| **exportsproc** | id():Long<br>getSaldo():Int<br>extraer(**i** Int)<br>depositar(**i** Int) |

| | |
|---|---|
| **Module** | **Banco1CC inherits from** CC |

| | |
|---|---|
| **Module** | **Banco2CC inherits from** CC |

| | |
|---|---|
| **Module** | **AccountFactory** |
| **imports** | CA, CC |
| **exportsproc** | createCA():CA |
| | createCC():CC |

| | |
|---|---|
| **Module** | **Banco1AccountFactory inherits from** AccountFactory |
| **imports** | Banco1CA, Banco1CC |

| | |
|---|---|
| **Module** | **Banco2AccountFactory inherits from** AccountFactory |
| **imports** | Banco2CA, Banco2CC |

| | |
|---|---|
| **Pattern** | **AccountFactory** |
| **based on** | Abstract Factory |
| **because** | Nos permite utilizar uniformemente la creacion de los productos de los bancos a partir de la interfaz abstracta de la Factory. |
| **where** | FabricaAbstracta **is** AccountFactory |
| | FabricaConcreta **is** Banco1AccountFactory |
| | FabricaConcreta **is** Banco2AccountFactory |
| | ProductoAbstracto **is** CA |
| | ProductoAbstracto **is** CC |
| | ProductoConcreto **is** Banco1CA |
| | ProductoConcreto **is** Banco1CC |
| | ProductoConcreto **is** Banco2CA |
| | ProductoConcreto **is** Banco2CC |

**DECORATOR.** Ahora toca el turno de aplicar el patrón Wrapper para implementar resoluciones del BCRA referidas a las transacciones que pueden realizar los clientes de cualquier banco sobre sus cuentas corrientes.

| | |
|---|---|
| **Module** | **CCWrapper inherits from** CC |
| **imports** | CC |

| | |
|---|---|
| **Module** | **CCA2156 inherits from** CCWrapper |

| | |
|---|---|
| **Module** | **CCA3401 inherits from** CCWrapper |

| | |
|---|---|
| **Pattern** | **CCWrapper** |
| **based on** | Decorator |
| **because** | Permite agregar funcionalidades a las cuentas corrientes sin modificar su interfaz.<br><br>Permite tratar uniformemente tanto a las CC como a algun Wrapper. |
| **where** | Componente **is** CC<br>Decorador **is** CCWrapper<br>DecoradorConcreto **is** CCA2156<br>DecoradorConcreto **is** CCA3401<br>operacion() **is** extraer()<br>operacion() **is** depositar() |

## 3.3 Ejercicio 9

Continuando con el problema de la estación climatométrica, suponga que la empresa que la desarrolla tiene varios modelos diferentes con sensores de distinta precisión, robustez y precio. Utilice el patrón Bridge para diseñar un sistema que se auto configure en tiempo de ejecución de acuerdo a la plataforma donde está ejecutando. Muestre código de ejemplo de cómo se implementaría este aspecto de la aplicación.

| | |
|---|---|
| **Module** | **ISensor** |
| **exportsproc** | encender()<br>apagar()<br>getValue():Value |

| | |
|---|---|
| **Module** | **ISensor1 inherits from** ISensor |

| | |
|---|---|
| **Module** | **ISensor2 inherits from** ISensor |

| | |
|---|---|
| **Module** | **ASensor** |
| **imports** | HSensor<br>encender()<br>apagar()<br>calibrar()<br>getValue():Value |

| | |
|---|---|
| **Pattern** | **SensorBridge** |
| **based on** | Bridge |
| **because** | Permite agregar un nuevo sensor facilmente debido a que el sistema trabaja con la interfaz ISensor, y para realizar esto solo debemos crear un nuevo heredero de dicha clase.<br><br>Además dicho cambio puede ser en tiempo de ejecución, y también permite intercambiar sensores (ya definidos) en tiempo de ejecución |
| **where** | Abstraccion **is** ASensor<br>Implementador **is** ISensor<br>ImplementadorConcreto **is** ISensor1<br>ImplementadorConcreto **is** ISensor2 |

## 3.4 Ejercicio 11, 12, 13, 14, 15 y 16

**11.** Considere el problema relativo al sistema de archivos de Linux visto en la práctica de diseño. El VFS da una representación jerárquica a los archivos y directorios como usted sabe. Utilice el patrón Composite para representar los directorios y archivos.
**12.** Aplique el patrón Iterator para recorrer el sistema de archivos de diferentes formas.
**13.** Suponga que el VFS provee un módulo que dado un código de error emitido por alguna de sus subrutinas (llamadas al sistema) retorna el mensaje correspondiente. Aplique el patrón Bridge para para poder configurar el VFS para retornar los errores en diferentes idiomas (internacionalización).
**14.** Una utilidad provista por Linux es du que retorna el espacio en disco utilizado por un directorio dado. Estas utilidades se proveen a nivel de usuario. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Visitor para implementar este tipo de servicios (find, etc.).
**15.** Otra utilidad provista por Linux es el comando fscheck que permite reparar el sistema de archivos ante algunos daños. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Strategy para implementar diferentes estrategias de reparación. Haga lo mismo con el patrón Visitor. Notar que se requiere recorrer el sistema de archivos completo.
**16.** Suponga que se desean implementar diferentes modelos de control de acceso a los archivos y directorios del sistema de archivos de Linux. Analice la posibilidad de aplicar (y eventualmente aplique) el patrón Wrapper para implementar esos modelos de seguridad.

| | |
|---|---|
| **Module** | **VFSComponent** |
| **imports** | VFSIterator, VFSVisitor |
| **exportsproc** | enter()<br>addChild(**i** VFSComponent)<br>delChild(**i** VFSComponent)<br>getChild(**i** Int):VFSComponent<br>getName():String<br>getSize():Int<br>createIterator():VFSIterator<br>accept(**i** VFSVisitor) |

| | |
|---|---|
| **Module** | **Directory inherits from** VFSComponent |

| | |
|---|---|
| **Module** | **File inherits from** VFSComponent |

| | |
|---|---|
| **Pattern** | **VFSComposite** |
| **based on** | Composite |
| **because** | Permite agregar fácilmente nuevos componentes al compueso si así fuera necesario.<br>Permite trabajar de manera uniforme con todos los componentes del compuesto. |
| **where** | Componente **is** VFSComponent<br>Compuesto **is** Directory<br>Hoja **is** File<br>operacion() **is** enter() |

| | |
|---|---|
| **Module** | **VFSIterator is** Iterator(VFSComponent) |

| | |
|---|---|
| **Module** | **VFSIterator1 inherits from** VFSIterator |

| | |
|---|---|
| **Pattern** | **VFSIterator** |
| **based on** | Iterator |
| **because** | Permite definir distintas formas de recorrer el VFS. |
| **where** | Agregado **is** VFSComponent<br>Iterador **is** VFSIterator<br>IteradorConcreto **is** VFSIterator1 |

| | |
|---|---|
| **Module** | **ErrorMessage** |
| **imports** | IntErrorMessage, Code |
| **exportsproc** | getMessage(**i** Code):String |

| | |
|---|---|
| **Module** | **IntErrorMessage** |
| **imports** | Code |
| | getMessage(**i** Code):String |

| | |
|---|---|
| **Module** | **EsErrorMessage inherits from** IntErrorMessage |

| | |
|---|---|
| **Module** | **EnErrorMessage inherits from** IntErrorMessage |

| | |
|---|---|
| **Pattern** | **IntErrorBridge** |
| **based on** | Bridge |
| **because** | Permite cambiar de idioma en tiempo de ejecución.<br>Permite agregar nuevos idiomas fácilmente. |
| **where** | Abstraccion **is** ErrorMessage<br>Implementador **is** IntErrorMessage<br>ImplementadorConcreto **is** EsErrorMessage<br>ImplementadorConcreto **is** EnErrorMessage |

| | |
|---|---|
| **Module** | **VFSVisitor** |
| **imports** | Directory, File |
| **exportsproc** | visitDirectory(**i** Directory)<br>visitFile(**i** File) |

| | |
|---|---|
| **Module** | **Du inherits from** VFSVisitor |

| | |
|---|---|
| **Module** | **Find inherits from** VFSVisitor |

| | |
|---|---|
| **Pattern** | **VFSVisitor** |
| **based on** | Visitor |
| **because** | Permite agregar funcionalidades sobre el VFS sin tocar su interfaz. |
| **where** | Visitante **is** VFSVisitor<br>VisitanteConcreto **is** Du<br>VisitanteConcreto **is** Find<br>Elemento **is** VFSComponent<br>ElementoConcreto **is** Directory<br>ElementoConcreto **is** File |

| | |
|---|---|
| **Module** | **VFSContext** |
| **imports** | VFSComponent, FSCheck |
| **exportsproc** | setStrategy(**i** FSCheck)<br>setComponent(**i** VFSComponent) |

| | |
|---|---|
| **Module** | **FSCheck** |
| **imports** | VFSComponent |
| **exportsproc** | repair(**i** VFSComponent) |

| | |
|---|---|
| **Module** | **FSCheck1 inherits from** FSCheck |

| **Module** | **FSCheck2 inherits from** FSCheck |
|---|---|

| **Pattern** | **VFSStrategy** |
|---|---|
| **based on** | Strategy |
| **because** | Permite implementar distintas variantes de un algoritmo sobre el VFS. Y a su vez, intercambiar estos en tiempo de ejecución. |
| **where** | Contexto **is** VFSContext<br>Estrategia **is** FSCheck<br>EstrategiaConcreta **is** FSCheck1<br>EstrategiaConcreta **is** FSCheck2<br>interfazAlgoritmo() **is** repair() |

| **Module** | **VFSComponentWrapper inherits from** VFSComponent |
|---|---|
| **imports** | VFSComponent |

| **Module** | **AccessControl1 inherits from** VFSComponentWrapper |
|---|---|

| **Module** | **AccessControl2 inherits from** VFSComponentWrapper |
|---|---|

| **Pattern** | **VFSDecorator** |
|---|---|
| **based on** | Decorator |
| **because** | Permite agregar funcionalidad al VFS sin necesidad de tocar su interfaz.<br>Permite agregar facilmente nuevas funcionalidades. |
| **where** | Componente **is** VFSComponent<br>ComponenteConcreto **is** Directory<br>ComponenteConcreto **is** File<br>Decorador **is** VFSComponentWrapper<br>DecoradorConcreto **is** AccessControl1<br>DecoradorConcreto **is** AccessControl2<br>operacion() **is** enter() |

## 3.5 Parcial 2021

La capa de transporte de un protocolo de comunicación debe interactuar con la placa de red de una computadora para proveer datos a las capas superiores del protocolo. Hay muchos fabricantes de placas de red que proveen interfaces diferentes para los programadores de protocolos. Por otro lado, se desea que las capas superiores del protocolo no estén al tanto de las diferencias entre los diversos modelos de placas de red.

A nivel de la placa de red existen secuencias de bytes con bytes de control. Precisamente las placas de red difieren en la interfaz que proveen para acceder a estas secuencias de bytes y en la forma en que codifican los bytes de control. Estas secuencias deben ser interpretadas por la capa de transporte de forma tal de proveer a las capas superiores tres tipos de paquetes de datos: inicio de conexión, paquete de datos (*payload*) y fin de conexión.

(a) Aplique el patrón de diseño Bridge para diseñar la capa de transporte del protocolo mencionado suponiendo que hay dos modelos de placa de red diferentes.

(b) Explique cómo deberían utilizarlo los programadores de las capas superiores.

| **Module** | **Package** |
|---|---|
| **imports** | PackageType |
| **exportsproc** | setSequenceType(**i** PackageType)<br>setSequence(**i** *List* <Int>) |

| **Module** | **ProtocoloCapaTransporte** |
|---|---|
| **imports** | PlacaDeRed, Package |
| **exportsproc** | interpretPackage():Package<br>sendPackage(**i** Package) |

| **Module** | **PlacaDeRed** |
|---|---|
| **imports** | PackageType |
| **exportsproc** | getPackageType():PackageType<br>getByteSequence():*List* <Int><br>receivePackage(**i** *List* <Int>, **i** PackageType) |

| **Module** | **PlacaACME inherits from** PlacaDeRed |
|---|---|

| **Module** | **PlacaEMCA inherits from** PlacaDeRed |
|---|---|

| | |
|---|---|
| **Pattern** | **ProtocoloCapaTransporte** |
| **based on** | Bridge |
| **comments** | Permite que las capas superiores del protocolo interactuen con la placa de red sin necesidad de conocerla.<br><br>Permite realizar cambios de placa de red facilmente. |
| **where** | Abstraccion **is** PlacaAbstracta<br>Implementador **is** PlacaDeRed<br>ImplementadorConcreto **is** PlacaACME<br>ImplementadorConcreto **is** PlacaEMCA |

Considere una agenda electrónica corporativa donde los empleados de una empresa pueden coordinar reuniones entre sí. Uno de los problemas al coordinar una reunión es que los participantes tienen otras reuniones programadas y resulta complicado encontrar un horario libre para todos o re-programar algunas reuniones para que haya un horario en que todos puedan participar. Suponga que hay una lista de usuarios cada uno de los cuales mantiene una lista de sus reuniones. Utilice el patrón de diseño Strategy para implementar dos formas de coordinación automática de una reunión:

- Falla. Si alguno de los participantes a la nueva reunión tiene ocupado el horario para ella, se les avisa a todos los participantes que deben buscar un nuevo horario.
- Postergación. Si alguno de los participantes a la nueva reunión tiene ocupado el horario para ella, se fija como horario para la nueva reunión el primer día y hora que todos los participantes tengan libre.

| | |
|---|---|
| **Module** | **MeetingContext** |
| **imports** | CoordinateMeetingStrategy, User, Meeting |
| **exportsproc** | setStrategy(**i** CoordinateMeetingStrategy)<br>addUserMeeting(**i** User, **i** Meeting)<br>getUserMeetings(**i** User):*List* <Meeting> |

| | |
|---|---|
| **Module** | **CoordinateMeetingStrategy** |
| **exportsproc** | coordinate() |

| | |
|---|---|
| **Module** | **FailMeetingStrategy inherits from** CoordinateMeetingStrategy |

| | |
|---|---|
| **Module** | **PostergateMeetingStrategy inherits from** CoordinateMeetingStrategy |

| | |
|---|---|
| **Pattern** | **CoordinateMeetingStrategy** |
| **based on** | Strategy |
| **comments** | Permite definir distintas formas de coordinar reuniones y tratarlas a todas uniformemente.<br>Permite que dichas formas de coordinar reuniones puedan ser elegidas en tiempo de ejecución.<br>Permite agregar fácilmente nuevas formas de coordinar reuniones. |
| **where** | Estrategia **is** CoordinateMeetingStrategy<br>EstrategiaConcreta **is** FailMeetingStrategy<br>EstrategiaConcreta **is** PostergateMeetingStrategy<br>Contexto **is** MeetingContext |

Teniendo en cuenta el problema anterior, utilice el patrón de diseño Decorator para implementar las siguientes funciones:

- Dar aviso al jefe de cada empleado que participará de una reunión una vez que esta se programa.
- Dar aviso a cada participante de una reunión cuando se modifica su horario o el lugar.

| | |
|---|---|
| **Module** | **Meeting** |
| **imports** | User, ... |
| **exportsproc** | ...<br>setUsers(**i** *List* <User>) |

| | |
|---|---|
| **Module** | **User** |
| **imports** | Meeting, ... |
| **exportsproc** | ...<br>notify(**i** Meeting)<br>setBoss(**i** User)<br>getBoss():User |

| | |
|---|---|
| **Module** | **UserWrapper inherits from** User |
| **imports** | User |
| **comments** | Setea el objeto User con el cuál compone en su constructor. |

| | |
|---|---|
| **Module** | **UserNotifyBoss inherits from** UserWrapper |

| | |
|---|---|
| **Module** | **UserNotifyMeetingChange inherits from** UserWrapper |

| **Pattern** | **UserMeetingNotifications** |
|---|---|
| **based on** | Wrapper |
| **comments** | Permite agregar funcionalidades sobre los procesos de User sin necesidad de modificar su interfaz.<br><br>Permite agregar distintos comportamientos para los procesos de la interfaz User fácilmente. |
| **where** | Componente **is** User<br>Decorador **is** UserWrapper<br>DecoradorConcreto **is** UserNotifyBoss<br>DecoradorConcreto **is** UserNotifyMeetingChange |

## 3.6 Parcial 2022

Considere los siguientes requerimientos: Un arreglo de diferentes tipos de sensores producen un flujo más o menos continuos de datos. Estos datos deben ser interpolados y filtrados para poder ser graficados de diversas formas (pero siempre las mismas). Las gráficas se muestran todas continuamente en diferentes sectores de la pantalla. Las gráficas pueden mostrar los datos de un único sensor o combinar los datos de varios de ellos. Dentro de los sensores los hay pasivos y activos.

**a)** Documente apropiadamente un diseño para que un software que implemente estos requerimientos tenga en cuenta los puntos que siguen.

**b)** Aplique el patrón de diseño *Iterator* para recorrer el arreglo de sensores.

**c)** Aplique el patrón de diseño *Visitor* para producir las diferentes gráficas que se necesitan.

**d)** Aplique el patrón de diseño *Command* para recibir las señales provenientes de los sensores activos.

**e)** Aplique el patrón de diseño *Strategy* para implementar diferentes funciones de interpolación y filtrado de datos.

**f)** Aplique el patrón de diseño *Decorator* para agregar un título, un borde y un color de fondo a cada gráfica.

**g)** Aplique el patrón de diseño *Abstract Factory* para que los títulos de todas las gráficas puedan escribirse en castellano, inglés, francés, según configuración inicial del sistema.

| **Module** | **Main** |
|---|---|
| **imports** | SensorArray, Sensor, SensorIterator, InterpolateAndFilterDataStrategy, ActiveSensorOrder, Graphic, SensorArrayVisitor, PacifyActiveSensorOrder, GraphicDecorator, GraphicAbstractFactory |
| **exportsproc** | ...<br>init()<br>draw()<br>setInterpolateAndFilterDataStrategy(**i** InterpolateAndFilterDataStrategy) |

| **Module** | **SensorArray** |
|---|---|
| **imports** | Sensor, SensorIterator, InterpolateAndFilterDataStrategy |
| **exportsproc** | addSensor(**i** Sensor)<br>removeSensor(**i** Sensor)<br>getSensor(**i** Int):Sensor<br>createIterator():SensorIterator |

| | |
|---|---|
| **Module** | **Sensor** |
| **imports** | Value, SensorArrayVisitor |
| **exportsproc** | on() |
| | calibrate() |
| | accept(**i** SensorArrayVisitor) |

| | |
|---|---|
| **Module** | **ActiveSensor inherits from** Sensor |
| **imports** | ActiveSensorOrder |
| **exportsproc** | sendValue() |
| | setActiveSensorOrder(**i** ActiveSensorOrder) |

| | |
|---|---|
| **Module** | **PassiveSensor inherits from** Sensor |
| **imports** | Value |
| **exportsproc** | getValue():Value |

| | |
|---|---|
| **Module** | **PacifiedSensor inherits from** Sensor |
| **imports** | Value |
| **exportsproc** | setValue(**i** Value) |
| | getValue():Value |

| | |
|---|---|
| **Module** | **SensorIterator** |
| **imports** | Sensor |
| **exportsproc** | first() |
| | next() |
| | hasNext():Boolean |
| | actualValue():Sensor |

| | |
|---|---|
| **Module** | **Graphic** |
| **imports** | Sensor |
| **exportsproc** | setValues(**i** *List* <Value>) |
| | draw() |

| | |
|---|---|
| **Module** | **SensorArrayVisitor** |
| **imports** | ActiveSensor, PassiveSensor, Graphic |
| **exportsproc** | visitActiveSensor() |
| | visitPassiveSensor() |
| | visitPacifiedSensor() |

| | |
|---|---|
| **Module** | **SimpleGraphicVisitor inherits from** SensorArrayVisitor |

| | |
|---|---|
| **Module** | **CombinedGraphicVisitor inherits from** SensorArrayVisitor |

| | |
|---|---|
| **Module** | **PacifyActiveSensorOrder** |
| **imports** | ActiveSensor, PacifiedSensor |
| **exportsproc** | execute() |

| | |
|---|---|
| **Module** | **InterpolateAndFilterDataStrategy** |
| **exportsproc** | interpolateAndFilterData() |

| | |
|---|---|
| **Module** | **OnlyInterpolateData inherits from** InterpolateAndFilterDataStrategy |

| | |
|---|---|
| **Module** | **OnlyFilterData inherits from** InterpolateAndFilterDataStrategy |

| | |
|---|---|
| **Module** | **InterpolateAndFilterData inherits from** InterpolateAndFilterDataStrategy |

| | |
|---|---|
| **Module** | **GraphicDecorator inherits from** Graphic |
| **imports** | Graphic |
| **exportsproc** | setGraphic(**i** Graphic) |

| | |
|---|---|
| **Module** | **TitleGraphic inherits from** GraphicDecorator |
| **exportsproc** | setTitle(**i** String) |
| | drawTitle() |

| | |
|---|---|
| **Module** | **Edge** |
| **exportsproc** | setLenght(**i** Int) |
| | setWidth(**i** Int) |

| | |
|---|---|
| **Module** | **EdgeGraphic inherits from** GraphicDecorator |
| **imports** | Edge |
| **exportsproc** | setEdge(**i** Edge) |
| | drawEdge() |

| | |
|---|---|
| **Module** | **BackgroundColorGraphic inherits from** GraphicDecorator |
| **exportsproc** | setBackgroundColor(**i** Int, **i** Int, **i** Int) |
| | drawBackgroundColor() |
| **comments** | El color se pasa en formato RGB. |

| | |
|---|---|
| **Module** | **ESGraphic inherits from** Graphic |

| | |
|---|---|
| **Module** | **ENGraphic inherits from** Graphic |

| | |
|---|---|
| **Module** | **FRGraphic inherits from** Graphic |

| | |
|---|---|
| **Module** | **GraphicAbstractFactory** |
| **imports** | Graphic |
| **exportsproc** | createGraphic():Graphic |

| | |
|---|---|
| **Module** | **ESGraphicFactory inherits from** GraphicAbstractFactory |
| **imports** | ESGraphic |

| | |
|---|---|
| **Module** | **ENGraphicFactory inherits from** GraphicAbstractFactory |
| **imports** | ENGraphic |

| | |
|---|---|
| **Module** | **FRGraphicFactory inherits from** GraphicAbstractFactory |
| **imports** | FRGraphic |

**b)**

| | |
|---|---|
| **Pattern** | **SensorIterator** |
| **based on** | Iterator |
| **because** | Permite recorrer el arreglo de sensores de distintas formas sin exponer su representacion interna. |
| **where** | Agregado **is** SensorArray<br>Iterador **is** SensorIterator |

**c)**

| | |
|---|---|
| **Pattern** | **SensorArrayVisitor** |
| **based on** | Visitor |
| **because** | Permite definir nuevas operaciones sobre una estructura de objetos sin modificar su interfaz. |
| **where** | Visitante **is** SensorArrayVisitor<br>VisitanteConcreto **is** SimpleGraphicVisitor<br>VisitanteConcreto **is** CombinedGraphicVisitor<br>EstructuraDeObjetos **is** SensorArray<br>Elemento **is** Sensor<br>ElementoConcreto **is** ActiveSensor<br>ElementoConcreto **is** PassiveSensor<br>ElementoConcreto **is** PacifiedSensor |

**d)**

| | |
|---|---|
| **Pattern** | **ActiveSensorCommand** |
| **based on** | Command |
| **because** | Nos permite "pacificar" a los sensores activos para poder acceder a sus datos. |
| **where** | Invocador **is** ActiveSensor<br>Orden **is** PacifyActiveSensor<br>ejecutar() **is** execute()<br>Receptor **is** PacifiedSensor<br>accion() **is** setValue() |

**e)**

| | |
|---|---|
| **Pattern** | **InterpolateAndFilterDataStrategy** |
| **based on** | Strategy |
| **because** | Nos permite definir facilmente nuevos algoritmos para interpolar y filtrar los datos de los sensores.<br><br>Estos nuevos algoritmos pueden ser intercambiados en tiempo de ejecución. |
| **where** | Contexto **is** Main<br>Estrategia **is** InterpolateAndFilterDataStrategy<br>EstrategiaConcreta **is** OnlyInterpolateData<br>EstrategiaConcreta **is** OnlyFilterData<br>EstrategiaConcreta **is** InterpolateAndFilterData<br>interfazAlgoritmo() **is** interpolateAndFilterData() |

**f)**

| | |
|---|---|
| **Pattern** | **GraphicDecorator** |
| **based on** | Decorator |
| **because** | Nos permite agregar funcionalidades a las gráficas sin modificar su interfaz.<br><br>Estas nuevas funcionalidades pueden ser agregadas/quitadas en tiempo de ejecución. |
| **where** | Componente **is** Graphic<br>Decorador **is** GraphicDecorator<br>DecoradorConcreto **is** TitleGraphic<br>estadoAñadido **is** title<br>comporamientoAñadido() **is** drawTitle()<br>DecoradorConcreto **is** EdgeGraphic<br>estadoAñadido **is** edge<br>comporamientoAñadido() **is** drawEdge()<br>DecoradorConcreto **is** BackgroundColorGraphic<br>estadoAñadido **is** backgroundColor<br>comporamientoAñadido() **is** drawBackgroundColor() |

g)

| | |
|---|---|
| **Pattern** | **GraphicAbrtractFactory** |
| **based on** | Abstract Factory |
| **because** | |
| | Nos permite crear fácilmente una familia de productos sin especificar sus clases concretas. |
| | Nos permite agregar un nuevo idioma fácilmente. |
| **where** | FabricaAbstracta **is** GraphicAbstractFactory<br>FabricaConcreta **is** ESGraphicFactory<br>FabricaConcreta **is** ENGraphicFactory<br>FabricaConcreta **is** FRGraphicFactory<br>ProductoAbstracto **is** Graphic<br>ProductoConcreto **is** ESGraphic<br>ProductoConcreto **is** ENGraphic<br>ProductoConcreto **is** FRGraphic |

# 4 Estilos Arquitectónicos

## 4.1 Preguntas teóricas

**Pregunta 1.** Se establece a menudo una analogía entre la arquitectura de software y la construcción de edificios. Determine los puntos fuertes de dicha analogía. ¿Con qué se corresponden las estructuras arquitectónicas (de software) dentro de la analogía mencionada? ¿Y los estilos? Determine los puntos débiles de la analogía. ¿En qué punto se torna imposible pensar en esos términos?
**Pregunta 2.** Establezca la diferencia entre arquitectura de referencia y estilo arquitectónico. ¿Qué es lo que puede hacer con uno y no con el otro en términos de (a) planificación de la organización y (b) análisis arquitectónico?
**Pregunta 3.** Un gran número de estilos están diseñados para soportar la cualidad de modificabilidad. Indique, mediante ejemplos, los cambios específicos que soporta cada uno de los estilos vistos en clase.
**Pregunta 4.** Haga lo mismo que en el problema anterior pero para el caso de la seguridad.
**Pregunta 5.** Explique la relación entre atributos de calidad y la arquitectura de un sistema.
**Pregunta 6.** Explique la relación entre tácticas de diseño y la arquitectura de un sistema.

## 4.2 Invocación Implícita

**Ejercicio 1.**

Un termómetro para uso medicinal posee:

- Una tapa que puede abrirse y cerrarse
- Debajo de la tapa hay un display y dos botones: uno para encender el termómetro y otro para seleccionar la escala de temperaturas (°F o °C);
- En un extremo del termómetro hay otro botón que se utiliza para efectuar las mediciones
- Una batería cuya carga máxima es conocida
- Una punta capaz de reaccionar a los cambios de temperatura del medio que la rodea; esta punta se introduce en el oído del paciente
- Un chip que correrá un software para controlar el funcionamiento del termómetro

El termómetro se debe apagar al cerrar la tapa. El termómetro se apaga cuando se termina la batería. Una vez pulsado el botón de encendido, el termómetro debe verificar su funcionamiento durante 5 segundos y luego emite los datos iniciales al display (cualquier acción que se intente antes de esos 5 segundos debe ser obviada).

El botón de medición debe ser pulsado al menos un segundo para que la lectura sea exitosa. La punta que se introduce en el oído del paciente funciona sólo si el software se lo indica. Deben mediar 10 segundos entre dos mediciones consecutivas; pero si el usuario lo intenta antes se obtendrá una medición fallida.

En cuanto finaliza la inicialización, el display muestra un indicador de batería (puede considerarse que es un número natural) y la escala de temperaturas seleccionada (estos datos deben ser "recordados" por el software de control aun luego de ser apagado). Si una medición de temperatura es fallida, se muestra una ∃; si la medición es exitosa se muestra la temperatura (puede pensarse también como un número entero). El indicador de batería debe ser actualizado para mostrar el estado de la misma.

La batería pierde un quanto de energía cada un segundo si sólo se usa el display, 3 quantos si se cambia la escala y 5 en cada medición errónea o no de temperatura. Cuando quedan 10 quantos de energía se deben obviar todas las acciones del usuario, excepto el apagado del termómetro.

| **Module** | **Cover** |
|---|---|
| **announces** | close() |

| **Module** | **ButtPower** |
|---|---|
| **announces** | powerPulse() |

| **Module** | **ButtTemperature** |
|---|---|
| **announces** | tempPulse() |

| **Module** | **ButtMeasure** |
|---|---|
| **announces** | measureHold() |

| **Module** | **Tip** |
|---|---|
| **exportsproc** | measure():Int |
| **announces** | measureValue(Int temp) |

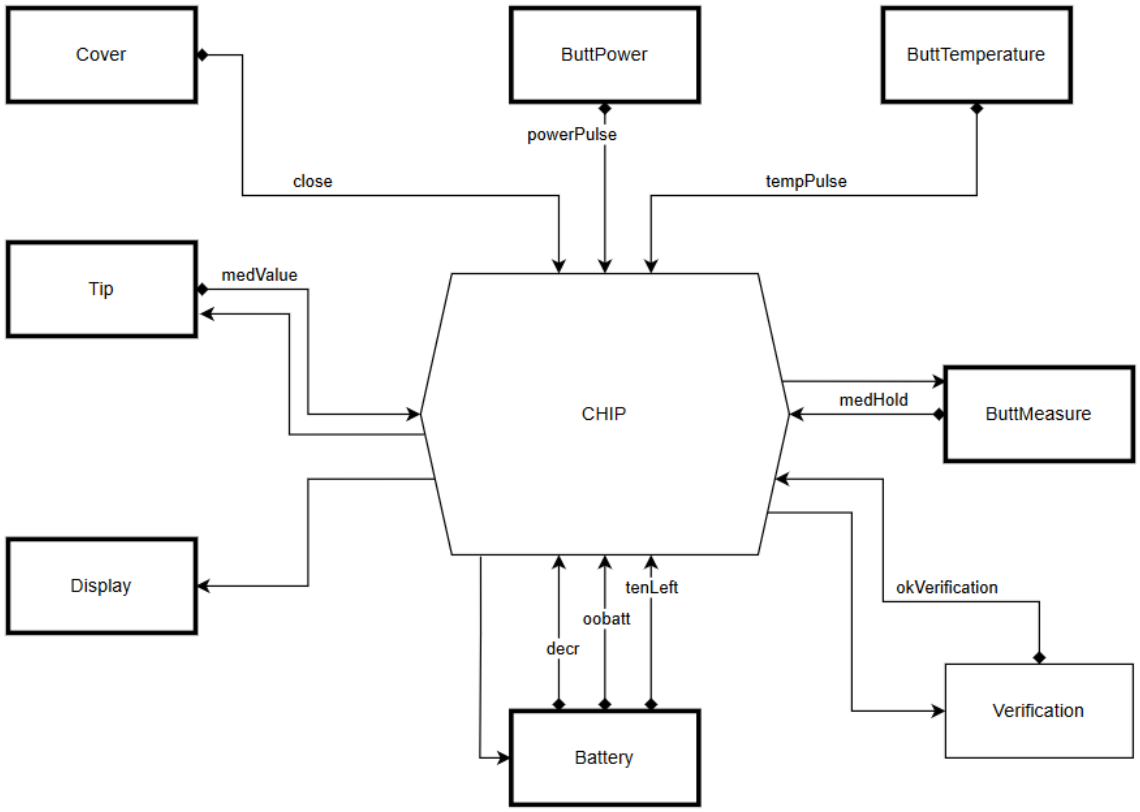
| **Module** | **Display** |
|---|---|
| **imports** | Unit |
| **exportsproc** | on()<br>off()<br>frame(**i** Int, **i** Unit)<br>draw(**i** Int)<br>drawErr()<br>drawCanMeasureMsg() |
| **comments** | Unit representa a las dos escalas de temperaturas del problema Farenheit y Celcius. |

| **Module** | **Battery** |
|---|---|
| **exportsproc** | on()<br>off()<br>current():Int |
| **announces** | oobatt()<br>tenLeft()<br>decr(Int batt) |

| **Module** | **Verification** |
|---|---|
| **exportsproc** | verify() |
| **announces** | okVerification() |
| **comments** | **verifiy()** debe verificar el sistema por 5 segundos. |

| **Module** | **Chip** |
|---|---|
| **imports** | Unit, Verification, Battery, Display, Tip, ButtMeasure, ButtTemperature, ButtPower, Cover |
| **exportsproc** | annClose()<br>annPowerPulse()<br>annTempPulse()<br>annMeasureHold()<br>annMeasureValue(**i** Int)<br>annOOBatt()<br>annTenLeft()<br>annDecr(**i** Int)<br>annOkVerification()<br>onRoutine()<br>offRoutine()<br>verRoutine()<br>measureRoutine() |
| **callonevent** | close **calls** offRoutine()<br>oobatt **calls** offRoutine()<br>okVerification **calls** onRoutine()<br>powerPulse **calls** verRoutine()<br>measureHold **calls** measureRoutine() |
| **comments** | Se supone que Chip lleva un estado interno con el cual puede obviar eventos. |

**Diagrama.**



**Estructura de módulos.**

| **Module** | **Main** |
|---|---|
| **comprises** | TADs |
| | Toolies |
| | Chip |

| **Module** | **TADs** |
|---|---|
| **comprises** | Cover |
| | ButtPower |
| | ButtTemperature |
| | ButtMeasure |
| | Tip |
| | Display |
| | Battery |

| **Module** | **Toolies** |
|---|---|
| **comprises** | Verification |

**Guía de módulos.**

**COMPLETAR**

## 4.3 Tubos y filtros

3. Un sistema debe procesar un flujo más o menos continuo de imágenes. Las imágenes pueden ser de tres tipos diferentes: color, B/N y 3D. Cada tipo pasa por un proceso diferente aunque puede haber pasos en común.

   A las imágenes tipo color se les aplica el siguiente proceso (en el orden indicado): se aumenta el brillo y disminuye el contraste; si el tamaño es menor que cierta cota se la amplía hasta que supere la cota y si es superior a otra cota se la disminuye hasta que esté por debajo; si el rojo o el verde es muy brillante se suavizan.

   A las imágenes tipo B/N se les aplica el siguiente proceso (en el orden indicado): si hay zonas muy blancas o muy negras se las suaviza; se hace una copia en negativo de cada imagen. Las originales y sus negativos siguen los siguientes pasos: se aumenta el brillo y disminuye el contraste; si el tamaño es menor que cierta cota se la amplía hasta que supere la cota.

   A las imágenes tipo 3D se les aplica el siguiente proceso: si son en blanco y negro se les aplica el proceso para imágenes B/N; caso contrario el proceso para imágenes tipo color.

| **Generic Module** | **Tubo(X)** |
|---|---|
| **exportsproc** | read():X |
| | write(**i** X) |
| | isEmpty():Boolean |

| **Module** | **TuboImage is** Tubo(Image) |
|---|---|

| **Module** | **Catalogador** |
|---|---|
| **imports** | Image |
| **inports** | image : Image |
| **outports** | bn : Image |
| | color : Image |
| | 3D : Image |

| **Module** | **FiltroPipeline** |
|---|---|
| **imports** | Image |
| **inports** | inImage : Image |
| **outports** | outImage : Image |

| **Module** | **FiltroColorUno inherits from** FiltroPipeline |
|---|---|

| **Module** | **FiltroColorDos** |
|---|---|
| **exportsproc** | setCota(**i** Int) |
| **inports** | inImage : Image |
| **outports** | outImage : Image |

| **Module** | **FiltroColorTres inherits from** FiltroPipeline |
|---|---|

| **Module** | **FiltroColor** |
|---|---|
| **comprises** | FiltroColorUno |
| | FiltroColorDos |
| | FiltroColorTres |

| **Module** | **FiltroBNUno inherits from** FiltroPipeline |
|---|---|

| **Module** | **FiltroBNDos** |
|---|---|
| **inports** | inImage : Image |
| **outports** | outNegImage : Image |
| | outImage : Image |

| **Module** | **FiltroBNTres** |
|---|---|
| **inports** | inNegImage : Image |
| | inImage : Image |
| **outports** | outImage : Image |

| **Module** | **FiltroBN** |
|---|---|
| **comprises** | FiltroBNUno |
| | FiltroBNDos |
| | FiltroBNTres |

| | |
|---|---|
| **Module** | **Filtro3D** |
| **inports** | 3D : Image |
| **outports** | color : Image |
| | bn : Image |

**Declaraciones.**

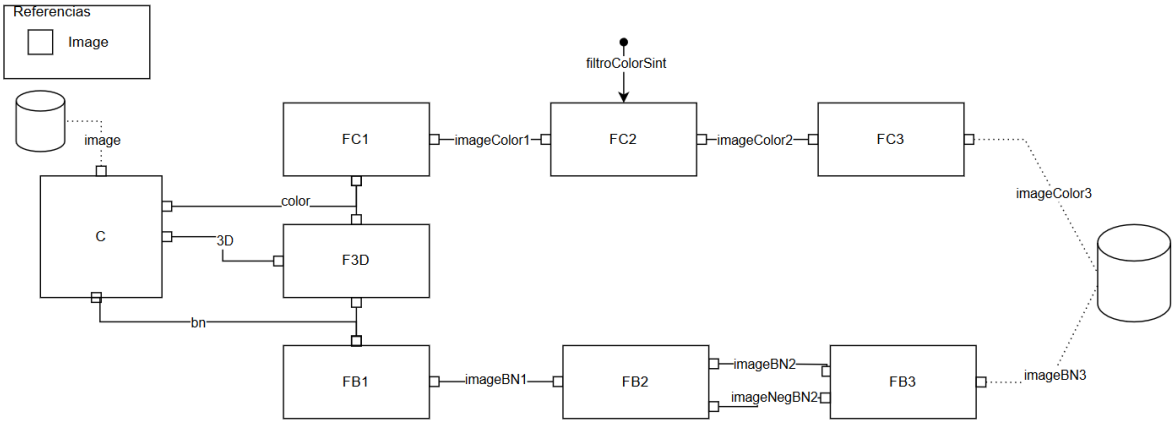
- C : **Catalogador**
- FC1, FC3, FB1 : **FiltroPipeline**
- FC2 : **FiltroColorDos**
- FB2 : **FiltroBNDos**
- FB3 : **FiltroBNTres**
- F3D : **Filtro3D**
- image, color, bn, 3D, imageColor1, imageColor2, imageColor3, imageBN1, imageBN2, imageNegBN2, imageBN3, colorFilterSint : **TuboImage**

**Asignaciones.**

- **C**.bn **as** bn.write
- **C**.color **as** color.write
- **C**.3D **as** 3D.write
- **F3D**.3D **as** 3D.read
- **F3D**.color **as** color.write
- **F3D**.bn **as** bn.write
- **FC1**.inImage **as** color.read
- **FC1**.outImage **as** imageColor1.write
- **FC2**.inImage **as** imageColor1.read
- **FC2**.outImage **as** imageColor2.write
- **FC2**.setCota **as** colorFilterSint.read
- **FC3**.inImage **as** imageColor2.read
- **FC3**.outImage **as** imageColor3.write
- **FB1**.inImage **as** bn.read
- **FB1**.outImage **as** imageBN1.write
- **FB2**.inImage **as** imageBN1.read
- **FB2**.outNegImage **as** imageNegBN2.write
- **FB2**.outImage **as** imageBN2.write
- **FB3**.inNegImage **as** imageNegBN2.read
- **FB3**.inImage **as** imageBN2.read
- **FB3**.outImage **as** imageBN3.write

**Diagrama.**

- Se utilizó un **Sintonizador** para que el usuario pueda establecer la cota para uno de los filtros a color.
- Se utilizaron Puertos y Tubos tipados.
- Los filtros son activos es decir ellos son quienes leen y escriben en los tubos correspondientes. Además los tubos tienen el método *isEmpty* el cuál nos indica si el tubo está ocupado o no, esto para que los filtros tengan el control sobre el procesamiento en paralelo de las imágenes.




**Estructura de módulos.**

**COMPLETAR**

**Guía de módulos.**

**COMPLETAR**

## 4.4 Sistemas Estratificados

Un dispositivo electrónico posee una placa de comunicación tipo Ethernet. Se desea un sistema que permita a un usuario enviar mensajes de texto, sonido o imágenes, usando diversos programas, a otros usuarios que poseen dispositivos similares. Sin embargo, no todos los otros dispositivos usan el mismo tipo de placa. Peor aun, distintos fabricantes de dispositivos han propuesto e impuesto diferentes protocolos de comunicación. Incluso el fabricantes de software para el dispositivo no tiene por qué ser el mismo fabricantes del hardware, con lo cual un fabricante de software querría poder proveer su implementación a distintos fabricantes de hardware.

| **Module** | **App** |
|---|---|
| **imports** | SO, String, Sound, Image |
| **exportsproc** | sendMessage(**i** String) |
| | sendSound(**i** Sound) |
| | sendImage(**i** Image) |
| | receiveData(**i** String) |

| | |
|---|---|
| **Module** | **SO** |
| **imports** | CapaProtocolo, Packet, Bus |
| **exportsproc** | SO(**i** f*(**i** String)) |
| | startConnection() |
| | searchFile(**i** String) |
| | sendData(**i** Packet) |
| | announceReceivedSomething() |
| | receiveData() |
| **callonevent** | receivedSomething **calls** receiveData() |
| **comments** | SO recibe un callback para enviar el mensaje en formato string, el cual puede representar al mensaje o a la ruta absoluta del archivo. |

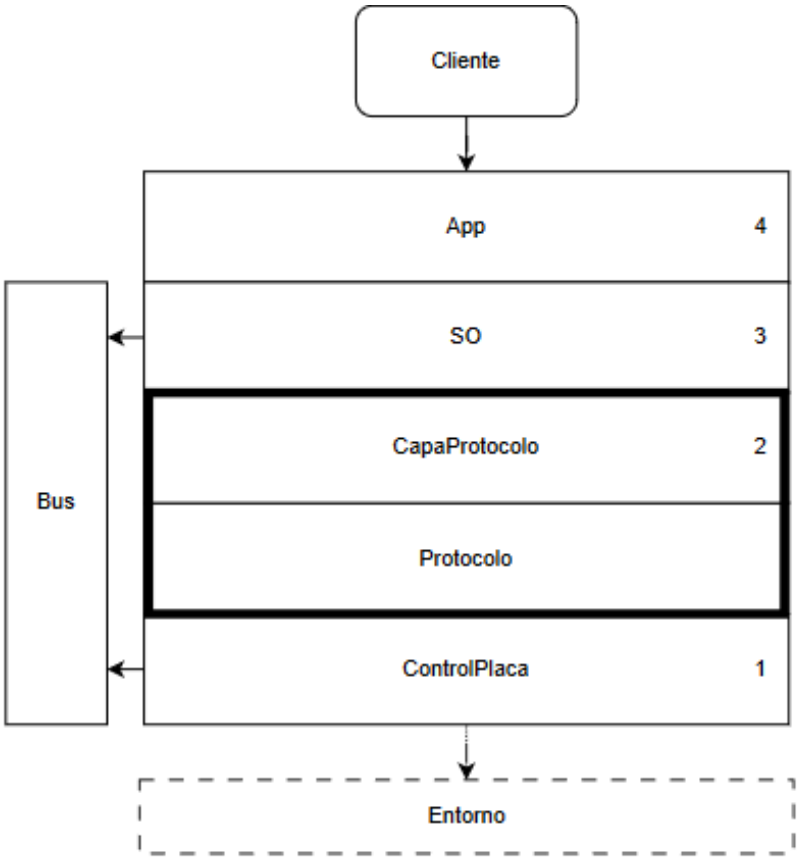
| | |
|---|---|
| **Module** | **CapaProtocolo** |
| **imports** | Protocolo, Packet |
| **exportsproc** | setProtocolo(**i** Protocolo) |
| | startConncection() |
| | endConnection() |
| | sendData(**i** Packet) |
| | readData():Packet |

| | |
|---|---|
| **Module** | **Protocolo** |
| **imports** | ControlPlaca |
| **exportsproc** | startConncection() |
| | endConnection() |
| | sendPacket(**i** List<Int>) |
| | readPacket():List<Int> |

| | |
|---|---|
| **Module** | **ControlPlaca** |
| **imports** | Bus |
| **exportsproc** | beginLooking() |
| | stopLooking() |
| | emit(**i** List<Int>) |
| | receivingSomething():List<Int> |
| **announces** | receivedSomething() |

| | |
|---|---|
| **Module** | **Bus** |
| **exportsproc** | broadcast(**i** List<Int>) |
| **comments** | Los eventos se envían codificados como una cadena de bytes por el bus. |

**Diagrama.**




Notar que la parte de *Bus* no es necesaria en el diagrama canónico. Se decidió agregarla para que se entienda más como se utiliza dicho *Bus*.

**Estructura de módulos.**

**COMPLETAR**

**Guía de módulos.**

**COMPLETAR**

## 4.5 Control de Procesos

**Ejercicio 1.**

> **Robot de limpieza.** Un robot para limpieza de pisos deberá recorrer un pasillo relativamente ancho en forma de U. La misión del robot es recoger papeles y otros objetos pequeños tirados en el piso y depositarlos en un cesto que forma parte del robot. El robot debe ser capaz de detectar paredes y otros obstáculos, y girar para recorrer la U. En consecuencia esta máquina posee sensores de diverso tipo (activos y pasivos), cuatro ruedas, el tren delantero con capacidad de giro, posibilidad de ir hacia adelante y atrás, un brazo prensil y un motor eléctrico alimentado con una batería. Inicialmente el robot asume que su batería está con la carga máxima y debe llevar la cuenta de la energía consumida con el fin de avisar, a través de una placa de comunicación Wi-Fi, a una sala de control que su batería necesita ser recargada; cuando esto ocurre el robot se detendrá por completo.
>
> Tener en cuenta que puede ser necesario que ciertos eventos o datos sean comunicados rápidamente desde los estratos inferiores a los superiores (por ejemplo la aparición de una persona).

| **Module** | **TrenDelantero** |
|---|---|
| **exportsproc** | derecha() |
| | izquierda() |
| | initialize() |

| **Module** | **Motor** |
|---|---|
| **exportsproc** | on() |
| | off() |
| | acelerar() |
| | desacelerar() |
| | initialize() |

| **Module** | **BrazoAbstraccion** |
|---|---|
| **imports** | BrazoImplementador |
| **exportsproc** | agarrar() |
| | depositar() |
| | setBrazo(**i** BrazoImplementador) |

| **Module** | **BrazoImplementador** |
|---|---|
| **exportsproc** | izquierda() |
| | derecha() |
| | arriba() |
| | abajo() |
| | agarrar() |
| | soltar() |
| | desplegar() |
| | retraer() |
| | rotarIzq() |
| | rotarDer() |

| **Module** | **Bateria** |
|---|---|
| **exportsproc** | on() |
| | off() |
| **announces** | oobattery() |

| **Module** | **TuboInt is** Tubo(Int) |
|---|---|

| **Module** | **SensorActivo** |
|---|---|
| **imports** | TuboInt |
| **exportsproc** | on() |
| | off() |
| | calibrate() |
| | setTubo(**i** TuboInt) |

| **Module** | **SensorProximidad inherits from** SensorActivo |
|---|---|

| **Module** | **SensorObjeto inherits from** SensorActivo |
|---|---|

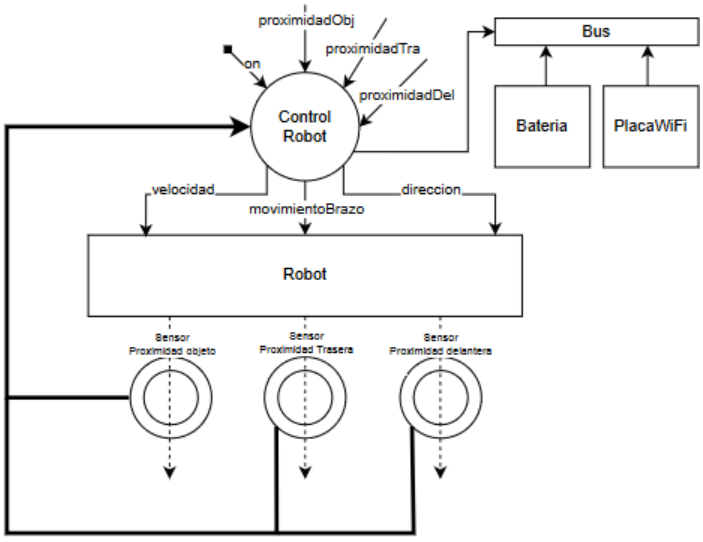
| **Module** | **PlacaWifi** |
|---|---|
| **imports** | PlacaWifiImplementador |
| **exportsproc** | sendOobSignal() |
| | setPlacaWifiImplementador(**i** PlacaWifiImplementador) |
| **callonevent** | oobattery **calls** sendOobSignal() |

| **Module** | **PlacaWifiImplementador** |
|---|---|
| **exportsproc** | send(**i** Bytes) |
| | receive():Bytes |

| **Module** | **ControlRobot** |
|---|---|
| **imports** | MovementStrategy |
| **exportsproc** | on() |
| | off() |
| | setPickupDistante(**i** Medida) |
| | setCrashDistance(**i** Medida) |
| | setMaxSpeed(**i** Medida) |
| | setMovementStrategy(**i** MovementStrategy) |
| | setProximidadDelantera(**i** Medida) |
| | setProximidadTrasera(**i** Medida) |
| | setProximidadObjeto(**i** Medida) |
| | getProximidadDelantera():Medida |
| | getProximidadTrasera():Medida |
| | getProximidadObjeto():Medida |
| **callonevent** | on **calls** on() |
| | oobattery **calls** off() |
| **comments** | **on** es un evento proveniente del boton de encendido del robot. |

| **Module** | **MovementStrategy** |
|---|---|
| **imports** | TrenDelantero, Motor, Brazo, ControlRobot |
| **exportsproc** | move(**i** ControlRobot) |

**Diagrama.**




El control se lleva a cabo en un ciclo cerrado de retroalimentación hacia atrás. Las variables controladas son la proximidad delantera y trasera del robot con respecto a obstáculos, y la proximidad a un objeto a agarrar. Por último las variables del proceso son la dirección y velocidad del robot, y el movimiento del brazo recolector de objetos.

Notar que la parte de *Bus* no es necesaria en el diagrama canónico. Se decidió agregarla para que se entienda más como se utiliza dicho *Bus*.

**Estructura de módulos.**

**COMPLETAR**

**Guía de módulos.**

**COMPLETAR**

## 4.6 Blackboard Systems

El sistema recibe una imagen que contiene texto y dibujos y debe escribir un archivo de texto con la porción de la imagen que es texto. El texto-imagen es siempre negro sobre fondo blanco. El texto-imagen se puede transformar en curvas, luego en letras, luego en palabras y finalmente en el texto.

| | |
|---|---|
| **Module** | **FC** |
| **imports** | Control |
| **exportsproc** | initialize() |
| **private** | identificar() |
| | precondicion():Boolean |

| | |
|---|---|
| **Module** | **IdentificadorDePixelesNegros inherits from** FC |
| **imports** | Imagen, PixelesNegros |

| | |
|---|---|
| **Module** | **IdentificadorDeCurvas inherits from** FC |
| **imports** | PixelesNegros, Curvas |

| | |
|---|---|
| **Module** | **IdentificadorDeFiguras inherits from** FC |
| **imports** | Curvas, Figuras |

| | |
|---|---|
| **Module** | **IdentificadorDeLetras inherits from** FC |
| **imports** | Figuras, Letras |

| | |
|---|---|
| **Module** | **IdentificadorDePalabras inherits from** FC |
| **imports** | Letras, Palabras |

| | |
|---|---|
| **Module** | **IdentificadorDeTexto inherits from** FC |
| **imports** | Palabras, Texto |

| | |
|---|---|
| **Module** | **Letras is** Lista(Letra) |

| | |
|---|---|
| **Module** | **Figuras is** Lista(Figura) |

| | |
|---|---|
| **Module** | **Curvas is** Lista(Curva) |

| | |
|---|---|
| **Module** | **Palabra is** Lista(Palabras) |

| | |
|---|---|
| **Module** | **PixelesNegros is** Lista(PixelNegro) |

| | |
|---|---|
| **Module** | **ElemBB** |
| **imports** | Coord |
| **exportsproc** | setVertices(**i** Coord, **i** Coord) |
| | getVerticeSI():Coord |
| | getVerticeID():Coord |

| | |
|---|---|
| **Module** | **Texto inherits from** ElemBB |
| **imports** | Palabras |
| **exportsproc** | setPalabras(**i** Palabras) |
| | getPalabras():Palabras |

| | |
|---|---|
| **Module** | **Palabra inherits from** ElemBB |
| **imports** | Letras |
| **exportsproc** | setLetras(**i** Letras) |
| | getLetras():Letras |

| | |
|---|---|
| **Module** | **Letra inherits from** ElemBB |
| **imports** | Figuras |
| **exportsproc** | setFiguras(**i** Figuras)<br>getFiguras():Figuras |

| | |
|---|---|
| **Module** | **Figura inherits from** ElemBB |
| **imports** | Curvas |
| **exportsproc** | setCurvas(**i** Curvas)<br>getCurvas():Curvas |

| | |
|---|---|
| **Module** | **Curva inherits from** ElemBB |
| **imports** | RectangulosNegros |
| **exportsproc** | setRectangulosNegros(**i** RectangulosNegros)<br>getRectangulosNegros():RectangulosNegros |

| | |
|---|---|
| **Module** | **PixelNegro inherits from** ElemBB |

| | |
|---|---|
| **Module** | **Imagen inherits from** ElemBB |

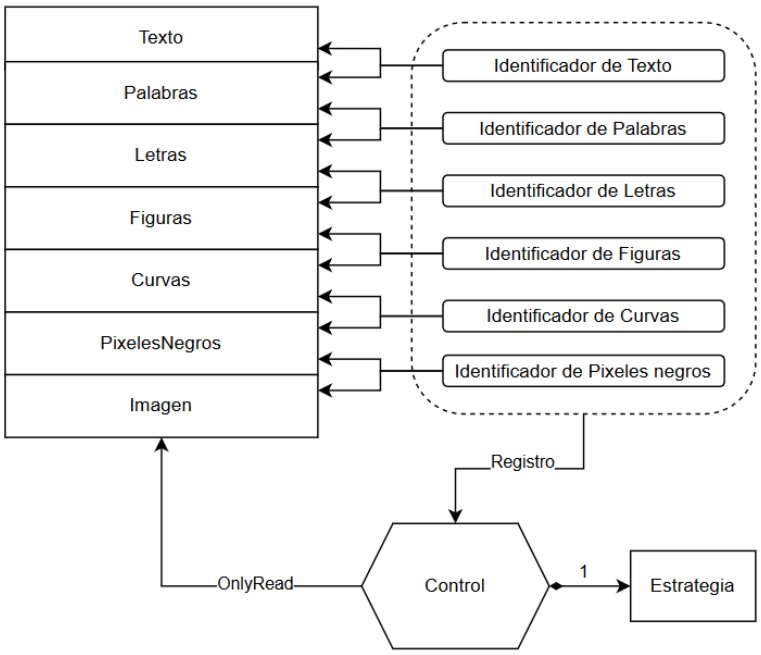
| | |
|---|---|
| **Module** | **Control** |
| **imports** | FCId, Metadata, SolutionStrategy |
| **exportsproc** | register(**i** FCId, **i** *F(), **i** *F(), **i** Metadata)<br>desregister(**i** FCId)<br>setSolutionStrategy(**i** SolutionStrategy)<br>start() |

| | |
|---|---|
| **Module** | **SolutionStrategy** |
| **imports** | Blackboard |
| **exportsproc** | solution() |

| | |
|---|---|
| **Module** | **Blackboard** |
| **comprises** | Imagen<br>PixelesNegros<br>Curvas<br>Figuras<br>Letras<br>Palabras<br>Texto |

| Module | FCs |
|---|---|
| comprises | IdentificadorDePixelesNegros |
| | IdentificadorDeCurvas |
| | IdentificadorDeFiguras |
| | IdentificadorDeLetras |
| | IdentificadorDePalabras |
| | IdentificadorDeTexto |

**Diagrama.**




**Conector Control-FC.**

Los FC se registran ante el control pasando sus método de precondición y su algoritmo de análisis/modificación del Blackboard. Luego, dependiendo la estrategia de solución, esta misma elegirá a que FC invocar dependiendo del estado del Blackboard y de la información que cada FC pueda aportar a la solución (Los metadatos de la FC tienen esta información).

Notar que la parte de *Estrategia* no es necesaria en el diagrama canónico. Se decidió agregarla para que se entienda más como se utiliza dicha *Estrategia*.

**Estructura de módulos.**

**COMPLETAR**

**Guía de módulos.**

**COMPLETAR**

## 4.7 Cliente/Servidor

Una firma dedicada al negocio inmobiliario tiene numerosas sucursales distribuidas en una ciudad muy grande. La inmobiliaria vende, compra y alquila inmuebles propios y de clientes. Además provee el servicio de cobro de alquiler y gestión de reparaciones a edificios. La empresa tiene un sitio web donde la gente puede consultar las propiedades disponibles. Además tiene que haber un sistema para que los empleados utilicen dentro de las sucursales. Por otro lado, la inmobiliaria le da a cada vendedor de la empresa una laptop que le permite al empleado consultar datos de la propiedad cuando está fuera de la empresa.

| | |
|---|---|
| **Module** | **WebApp** |
| **imports** | RestAPI |
| **exportsproc** | loginForm(**i** String, **i** String)<br>fetchFrontend()<br>fetchPropiedades()<br>reparacionesForm(**i** String, **i** Date)<br>fetchCuenta() |

| | |
|---|---|
| **Module** | **DesktopApp** |
| **imports** | DesktopAPI |
| **exportsproc** | loginForm(**i** String, **i** String)<br>fetchFrontend()<br>fetchPropiedades()<br>fetchClientes() |

| | |
|---|---|
| **Module** | **RestAPI** |
| **imports** | Propiedades, Usuarios, Lector, Escritor |
| **exportsproc** | authenticate(**i** String, **i** String)<br>getPropiedades():Json<br>getCuenta(**i** String):Json |

| | |
|---|---|
| **Module** | **DesktopAPI** |
| **imports** | Propiedades, Usuarios, , Lector, Escritor |
| **exportsproc** | authenticate(**i** String, **i** String)<br>getPropiedades():Xml<br>getCuenta(**i** String):Xml<br>getClientes():Xml |

| | |
|---|---|
| **Module** | **Propiedad** |
| **imports** | PropMetadata |
| **exportsproc** | Propiedad(**i** String, **i** String, **i** PropMetadata)<br>getName():String<br>getDireccion():String<br>getMetadata():PropMetadata |

| | |
|---|---|
| **Module** | **Usuario** |
| **imports** | UserMetadata |
| **exportsproc** | Usuario(**i** String, **i** String, **i** UserMetadata)<br>getUsername():String<br>getPassword():String<br>getMetadata():UserMetadata |
| **comments** | Dentro de UserMetadata tenemos info sobre rol de usuario y el estado de sus cuentas (en caso que tenga). |

| | |
|---|---|
| **Module** | **Usuarios is** Lista(Usuario) |

| **Module** | **Propiedades is** Lista(Propiedad) |
|---|---|

| **Module** | **Lector** |
|---|---|
| **imports** | Propiedades, Usuarios, Usuario, DBMS |
| **exportsproc** | leerUsuario(**io** Usuario):Bool |
| | leerPropiedades(**io** Propiedades):Bool |
| | leerCuenta(**io** Usuario):Bool |
| | leerClientes(**io** Usuarios):Bool |

| **Module** | **Escritor** |
|---|---|
| **imports** | Propied, Usuario, DBMS |
| **exportsproc** | nuevoUsuario(**io** Usuario):Bool |
| | nuevaPropiedad(**io** Propiedad):Bool |
| | nuevaCuenta(**io** Usuario):Bool |

| **Module** | **DBMS** |
|---|---|
| **exportsproc** | configurate(**i** Json) |
| | executeQuery(**i** String) |

**Estructura de módulos.**

| **Module** | **LogicaDePresentacion** |
|---|---|
| **comprises** | WebApp |
| | DesktopApp |

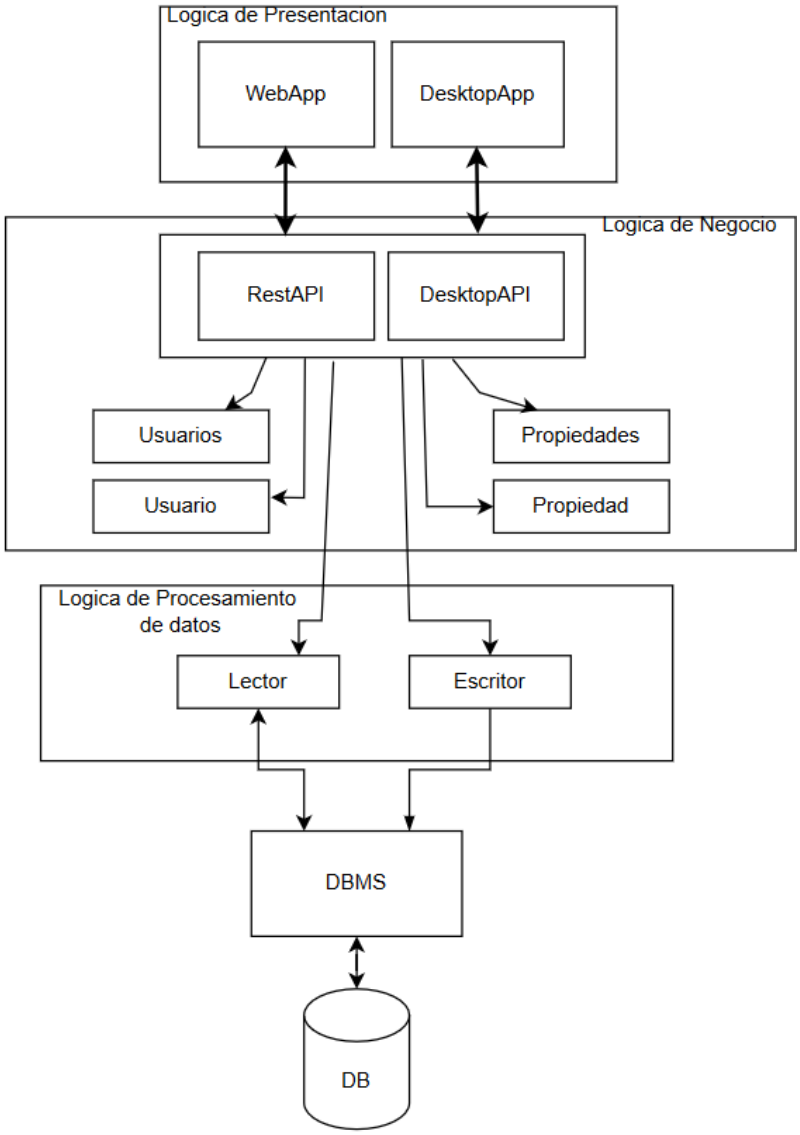
| **Module** | **LogicaDeNegocio** |
|---|---|
| **comprises** | APIs |
| | Entidades |

| **Module** | **APIs** |
|---|---|
| **comprises** | RestAPI |
| | DesktopAPI |

| **Module** | **Entidades** |
|---|---|
| **comprises** | Propiedad |
| | Usuarios |
| | Propiedades |
| | Usuarios |

| **Module** | **LogicaDeProcesamientoDeDatos** |
|---|---|
| **comprises** | Lector |
| | Escritor |

**Diagrama.**




**Guía de módulos.**

**COMPLETAR**

# 5 Testing

## 5.1 Teoria

**1.** Discuta las diferencias entre validación y verificación y explique por qué la validación es un proceso particularmente difícil.

La verificación consiste en corroborar que el programa respeta su especificación, mientras que validación significa corroborar que el programa satisface las expectativas del usuario.

La validación es un proceso particularmente difícil debido a que se necesita estar en contacto con los usuarios para ver si efectivamente el programa satisface sus necesidades, y además en caso de que esto no suceda, los usuarios pueden llegar a solicitar requerimientos que no son fáciles de llevar a cabo. En caso de tener un especialista del campo que se está tratando como intermediario entre los usuarios y los desarrolladores esto último se facilita, pero no es sencillo conseguir un perfil como el mencionado.

**2.** Una forma de trabajo común respecto del testeo de un sistema es testearlo hasta agotar el presupuesto destinado a esa fase del ciclo de vida y entregarlo a los clientes. Discuta el aspecto ético de esa forma de trabajo.

**3.** Volviendo sobre las cualidades del software vistas en la Unidad I de Análisis de Sistemas, clasifíquelas con respecto a las siguientes cuestiones relativas a V&V: ¿Son subjetivas u objetivas? ¿Son binarias o no? ¿Son más o menos relevantes en diferentes aplicaciones y entornos?

**4.** Explique la racionalidad detrás del testing estructural.

La racionalidad detrás del testing estructural es complementar al testing funcional donde este se realiza en paralelo con el desarrollo del código en sí, y eventualmente es acompañado con el testing estructural el cuál brinda garantías de la correcta ejecución del programa.

La racionalidad detrás de esta estrategia: no se puede encontrar un error si no se ejecuta la línea de código donde se encuentra ese error.

**5.** Determine las diferencias y semejanzas entre el testing estructural y el testing basado en especificaciones.

- Diferencias:
  - El testing funcional se hace sobre la especificación del programa, antes de que este esté en desarrollo/terminado, en cambio el testing estructural se realiza sobre el código fuente del programa.
  - El testing funcional prueba lo que el programa se supone que debe hacer, y el testing estructural prueba lo que el programa efectivamente hace.
- Semejanzas:
  - Ambos cuestan recursos, tanto en personal como en tiempo y dinero.

## 5.2 Práctica

**1.** Considere el siguiente fragmento de programa

```
read(x); read(y);
if x > 0 or y > 0 then
    write("1");
else
    write("2");
endif;
if y > 0 then
    write("3");
else
    write("4");
endif;
```

Genere casos de prueba utilizando el criterio de cubrimiento de sentencias.

$CS = \{\langle x = 1, y = 1\rangle, \langle x = 0, y = 0\rangle\}$

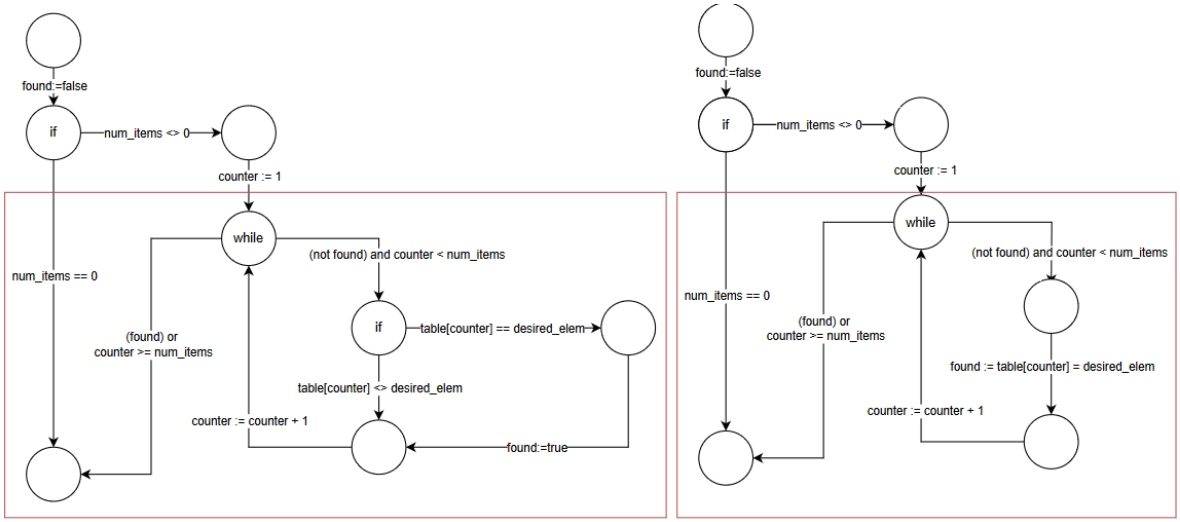
**2.** Considere los dos fragmentos de programa siguientes:

```
found := false;
if num_items <> 0 then counter := 1;
    while (not found) and counter < num_items loop
        if table[counter] = desired_elem then
            found := true;
        endif
        counter := counter + 1;
    endloop
endif;
if found then
    write("the desired element exists in the table");
else
    write("the desired element doesn't exists in the table");
endif;
```

```
found := false;
if num_items <> 0 then counter := 1;
    while (not found) and counter < num_items loop
        found := table[counter] = desired_elem;
        counter := counter + 1;
    endloop
endif;
if found then
    write("the desired element exists in the table");
else
    write("the desired element doesn't exists in the table");
endif;
```

Determine si el criterio de cubrimiento de flechas genera conjuntos de prueba diferentes en cada caso.

El criterio de cubrimiento de flechas genera dos conjuntos de prueba distintos, debido a que en el segundo caso, el caso de prueba en el cuál *desired_elem* no esté en *table* cubre todas las flechas del while, en cambio en el primer caso esto no sucede, por lo que necesita si o si un caso de prueba en el cuál *desired_elem* si esté en *table*.



**3.** Sea $\{C_i\}$ el conjunto de todas las condiciones lógicas usadas en un programa $P$ para gobernar el flujo de ejecución. El criterio de cubrimiento de flechas requiere que el conjunto de prueba $\{d_j\}$ deber ser tal que cada $C_i$ sea falsa y verdadera para algún $d_j$ al menos una vez.

Para cada $C_i$ sean $D_i$ y $\overline{D_i}$ los conjuntos de entradas que hacen que $C_i$ sea verdadera y falsa, respectivamente. Luego el criterio de cubrimiento de flechas es satisfecho por conjuntos de prueba $T$ que deben contener al menos un elemento de $D_i$ y uno de $\overline{D_i}$ para cada $C_i$. Explique por qué esto no determina una partición de $D$.

$C_1 = x \neq 3$
$D_1 = \{x \mid x \neq 3\}, \overline{D_1} = \{3\}$
$T_k = \{\langle x = 1\rangle, \langle x = 3\rangle\}$ $T_j = \{\langle x = 2\rangle, \langle x = 3\rangle\}$

Viendo el ejemplo, podemos ver que $T_k \cap T_j \neq \emptyset$ por lo que $T$ no determina una partición de $D$.

Como consecuencia debe haber diferentes conjuntos de prueba con diferente cardinalidad que satisfagan el criterio de cubrimiento de flechas. Así, aparece el problema de encontrar conjuntos de prueba minimales que satisfagan el criterio. Para los siguientes fragmentos, encuentre conjuntos de prueba mínimos compatibles con el criterio de cubrimiento de flechas:

```
if x > z then
    y := 3;
else
    y := 2;
endif;
if x > z + 1 then
    w := 3;
else
    w := 2;
endif;
```

$CS = \{\langle x = 2,\ z = 0\rangle, \langle x = 0,\ z = 0\rangle\}$

**4.** Sea $\{C_i\}$ definido como en el problema anterior. Considere la definición del criterio de asignación de valores de verdad:

Sea $t = \langle tr_1, ..., tr_n\rangle$ una asignación de valores de verdad a las condiciones $\{C_i\}$, es decir $\langle tr_1, ..., tr_n\rangle$ es un vector $n$-dimensional de valores booleanos que será asignado a cada condición de $P$. En otras palabras $t$ relaciona las condiciones (compuestas) de cada estructura de control de $P$ (y no las condiciones de una misma estructura de control). Para cada una de estas asignaciones, sea $D_t$ el subconjunto de $D$ que hace todas las $C_i$ sean verdaderas o falsas según la asignación. Muestre que $\{D_t\}$ es una partición de $D$.

- Qvq $\bigcup_i D_i = D$
- Qvq $D_i \cap D_j = \emptyset\ \forall\ i, j$

**5.** Sea $\{C_i\}$ definido como en el problema anterior. El criterio de cubrimiento de condiciones múltiples puede definirse como sigue: cada conjunto de prueba debe hacer todas las condiciones $C_i$ verdaderas o falsas de todas las formas posibles, basándose en los valores de las proposiciones simples que las componen. Por ejemplo, si $C_5$ es $c_{51}$ **and** $c_{52}$, entonces debemos generar cuatro casos de prueba que hacen a $c_{51}$ verdadera, $c_{52}$ verdadera, $c_{51}$ verdadera, $c_{51}$ falsa, etc.

Determine un conjunto de prueba (posiblemente minimal) que satisfaga el criterio de cubrim- iento de condiciones múltiples para el siguiente fragmento de programa:

```
if x > z and x > 3 then
    a := 1;
else
    a := 2;
endif;
if a > b or z < x then
    w := 1;
else
    z := x;
endif;
```

$CS = \{\langle x = 4,\ z = 3\,,\ b = 0\rangle, \langle x = 4,\ z = 5\,,\ b = 0\rangle, \langle x = 3,\ z = 4\,,\ b = 2\rangle,$
$\langle x = 3,\ z = 2\,,\ b = 2\rangle\}$

**6.** Verifique el siguiente programa utilizando el criterio de asignaciones de valores de verdad y determine si revela el error existente.

```
if x <> 0 then
    y := 5;
else
    z := z - x;
endif;
if z > 1 then
    z := z / x;
else
    z:= 0;
endif;
```

$CS = \{\}$

**7.** Considere el siguiente fragmento de un programa de ordenación:
**Ayuda:** considere a todas las componentes del arreglo como una única variable, **a**.

```
for i in 2..n loop
    x := a[i];
    a[0] := x;
    j := i - 1;
    while x < a[j] loop
        a[j + 1] := a[j];
        j := j - 1;
    endloop;
    a[j + 1] := x;
endloop;
```

- **a.** Cubrimiento de sentencias: $CS = \{\langle a = [2, 4, 3], n = 2\rangle\}$
- **b.** Cubrimiento de flechas: $CS = \{\langle a = [2, 4, 3], n = 2\rangle, \langle a = x, n = 1\rangle, \langle a = [2, 1, 3], n = 2\rangle\}$
- **c.** Cubrimiento de condiciones: $CS = \{\langle a = [2, 4, 3], n = 2\rangle, \langle a = x, n = 1\rangle, \langle a = [2, 1, 3], n = 2\rangle\}$

- **d.** Asignación de valores de verdad:
- **e.** Cubrimiento de condiciones múltiples: $CS = \{\langle a = [2,4,3], n = 2\rangle, \langle a = x, n = 1\rangle, \langle a = [2,1,3], n = 2\rangle\}$

**8.** Calcule los conjunto de casos de prueba de los programas del problema 3 que satisfagan los criterios

```
found := false;
if num_items <> 0 then counter := 1;
    while (not found) and counter < num_items loop
        if table[counter] = desired_elem then
            found := true;
        endif
        counter := counter + 1;
    endloop
endif;
if found then
    write("the desired element exists in the table");
else
    write("the desired element doesn't exists in the table");
endif;
```

```
found := false;
if num_items <> 0 then counter := 1;
    while (not found) and counter < num_items loop
        found := table[counter] = desired_elem;
        counter := counter + 1;
    endloop
endif;
if found then
    write("the desired element exists in the table");
else
    write("the desired element doesn't exists in the table");
endif;
```

- **a.** Cubrimiento de condiciones:
- **b.** Cubrimiento de valores de verdad:
- **c.** Cubrimiento de condiciones múltiples:

## 5.3 Testing en Z

**1.** Base de datos cinematográficos. Considere la especificación Z escrita en Análisis de Sistemas para este problema. Genere casos de prueba para las operaciones:
**a.** Que obtienen todas las películas de un director dado
**b.** Que modifican el nombre de un director de una película dada.

$[DIRID, MOVID, STRING]$

*BD*

$dirsNames : DIRID \pfun STRING$
$dirsMovies : DIRID \pfun \mathbb{P}\, MOVID$
$movies : MOVID \pfun DIRID$

*BDInit*

$$BD$$

---

$$dirsNames = \emptyset$$
$$dirsMovies = \emptyset$$
$$movies = \emptyset$$

*GetMoviesOk*

$$\Xi BD$$
$$did? : DIRID$$
$$movies! : \mathbb{P}\, MOVID$$

---

$$did? \in \mathrm{dom}\, dirsMovies$$
$$movies! = dirsMovies(did?)$$

*DirectorNotFound*

$$\Xi BD$$
$$did? : DIRID$$

---

$$did? \notin \mathrm{dom}\, dirsMovies$$

$$GetMovies == GetMoviesOk \lor DirectorNotFound$$

*ModifyDirName*

$$\Delta BD$$
$$mid? : MOVID$$
$$name? : STRING$$

---

$$mid? \in \mathrm{dom}\, movies$$
$$dirsNames' = dirsNames \oplus \{movies(mid?) \mapsto name?\}$$

*MovieNotFound*

$$\Xi BD$$
$$mid? : MOVID$$

---

$$mid? \notin \mathrm{dom}\, movies$$

**Casos de prueba.**

```
loadspec specs/bdcine.tex
selop GetMovies
genalltt
addtactic GetMovies_DNF_1 SP \in did? \in \dom dirsMovies
addtactic GetMovies_DNF_2 SP \notin did? \notin \dom dirsMovies
genalltt
prunett
genalltca
```

*GetMovies_SP_1_TCASE*

$$GetMovies\_SP\_1$$

---

$$movies = \emptyset$$
$$dirsNames = \emptyset$$
$$dirsMovies = \{(dIRID1 \mapsto \{mOVID2\})\}$$
$$did? = dIRID1$$

*GetMovies_SP_2_TCASE*

*GetMovies_SP_2*

---

$movies = \emptyset$
$dirsNames = \emptyset$
$dirsMovies = \{(dIRID1 \mapsto \{mOVID2\}), (dIRID3 \mapsto \{mOVID2\})\}$
$did? = dIRID1$

*GetMovies_SP_3_TCASE*

*GetMovies_SP_3*

---

$movies = \emptyset$
$dirsNames = \emptyset$
$dirsMovies = \emptyset$
$did? = dIRID1$

*GetMovies_SP_4_TCASE*

*GetMovies_SP_4*

---

$movies = \emptyset$
$dirsNames = \emptyset$
$dirsMovies = \{(dIRID2 \mapsto \{mOVID1\})\}$
$did? = dIRID1$

**5.** Genere casos de prueba para la siguiente operación.

*State*

$f : X \pfun \mathbf{Z}$

*Operation*

$\Delta State$
$new? : \mathbb{P}\, X$

---

$f' = f$
$\quad \cup \{x : X \mid x \in new? \setminus \operatorname{dom} f \bullet x \mapsto 0\}$
$\quad \setminus \{x : X \mid x \in new? \cap \operatorname{dom} f \bullet x \mapsto f\, x\}$

**6.**

$[DNI, DOMICILIO]$

*SeguridadSocial*

$viveEn : DNI \pfun DOMICILIO$
$ingresosPorDom : DOMICILIO \pfun \mathbf{Z}$

*SeguridadSocialInv*

*SeguridadSocial*

---

$\operatorname{ran} viveEn = \operatorname{dom} ingresosPorDom$

*CambioDeDom*

$\Delta SeguridadSocial$
$x? : DNI;\ nuevoDom? : DOMICILIO;\ z? : \mathbf{Z}$

---

$x? \in \operatorname{dom} viveEn$
$nuevoDom \neq viveEn\ x?$
$viveEn' = viveEn \oplus \{x? \mapsto nuevoDom?\}$
$ingresosPorDom' =$
**if** $viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn)$
**then** $\{viveEn\ x?\} \lhd ingresosPorDom \oplus \{nuevoDom? \mapsto z?\}$
**else** $ingresosPorDom \oplus \{nuevoDom? \mapsto z?\}$

El **ifthenelse** se puede transformar a lo siguiente:

$(viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn) \land \{viveEn\ x?\} \lhd ingresosPorDom \oplus \{nuevoDom? \mapsto z?\})$
$\lor$
$(\neg\ (viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn)) \land ingresosPorDom \oplus \{nuevoDom? \mapsto z?\})$

*CambioDeDom*1

$\Delta SeguridadSocial$
$x? : DNI;\ nuevoDom? : DOMICILIO;\ z? : \mathbf{Z}$

---

$x? \in \operatorname{dom} viveEn$
$nuevoDom \neq viveEn\ x?$
$viveEn' = viveEn \oplus \{x? \mapsto nuevoDom?\}$
$viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn)$
$ingresosPorDom' = \{viveEn\ x?\} \lhd ingresosPorDom \oplus \{nuevoDom? \mapsto z?\}$

*CambioDeDom*2

$\Delta SeguridadSocial$
$x? : DNI;\ nuevoDom? : DOMICILIO;\ z? : \mathbf{Z}$

---

$x? \in \operatorname{dom} viveEn$
$nuevoDom \neq viveEn\ x?$
$viveEn' = viveEn \oplus \{x? \mapsto nuevoDom?\}$
$\neg\ (viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn))$
$ingresosPorDom' = ingresosPorDom \oplus \{nuevoDom? \mapsto z?\}$

$CambioDeDom == CambioDeDom1 \lor CambioDeDom2$