Nombre y apellido: Yamil Nicolás Soto Legajo: S-5522/1

Ingeniería de Software 2
LCC – FCEIA – UNR

Cristiá, Krapf, Scandolo
Lunes 19 de febrero de 2024

Examen Final

Nota: La interpretación de las consignas es parte del examen. **Cada parte** se aprueba con al menos dos problemas bien y no menos del 65 % del puntaje de esa parte. Problemas parcialmente correctos no necesariamente suman puntaje.

Primera parte

Duración aproximada 3,5 hs.

Se evalúa: Arquitecturas de software, estilos arquitectónicos, descripción de diseños en diferentes estilos; testing funcional y estructural.

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

   Se trata de un sistema que debe imponer una política de control de acceso sobre archivos. La política de control de acceso es el resultado de combinar políticas de control de acceso más específicas. Por ejemplo, una política específica determina si el usuario tiene permiso para abrir el archivo o no; otra determina si ese archivo se puede abrir ese día y en ese horario.

   Por otro lado, los archivos a los cuales se protege están almacenados en distintos sistemas de archivos. A su vez, cada sistema de archivos se monta sobre un hardware particular.

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

   Un sistema recibe registros con datos de personas que incluyen, nombre y apellido, fecha de nacimiento, edad, cantidad de hijos, estado civil, si está vivo o muerto, y en el último caso fecha del deceso. Como el sistema debe procesar los registros según sus diversos campos lo primero que debe hacerse es descomponer cada uno en sus campos. El último paso del procesamiento es almacenar los registros en distintos archivos de caracteres, un registro por línea y los campos separados por ":". Entre los requerimientos del sistema están:

   a) Comprobar la edad de la persona en relación a la fecha de nacimiento: en caso de error el registro se debe almacenar en el archivo "errorEdad".

   b) Las personas casadas y vivas que no tengan hijos se deben almacenar en el archivo "cvsh", y las que sí los tengan en el archivo "cvc123", en tanto que los que tengan más de 4 hijos se deben almacenar en el archivo "cvc4omas".

   c) Se debe comprobar que una persona muerta haya muerto luego de haber nacido, caso contrario debe almacenarse en el archivo "errorMuerte".

   d) Las personas vivas de entre 18 y 65 años deben almacenarse en el archivo "18-65", en tanto que las menores de 18 en el archivo "menos18", y las mayores de 65 en "mas65".

   Tener en cuenta no procesar inútilmente los registros pero al mismo tiempo mantener la simplicidad de los filtros.

3. Considere la siguiente especificación Z de una operación que agrega dos elementos a un conjunto.

$$ERROR :: ok \mid error$$

Estado

$$A : \mathbb{P}\mathbb{Z}$$

AgregarOk

$$\Delta Estado$$
$$x1?, x2? : \mathbb{Z}$$
$$err! : ERROR$$

$$\{x1?, x2?\} \cap A = \emptyset$$
$$A' = A \cup \{x1?, x2?\}$$
$$err! = ok$$

AgregarE

$\Xi Estado$
$x1?, x2? : \mathbb{Z}$
$err! : ERROR$

$\{x1?, x2?\} \cap A \neq \emptyset$
$err! = error$

$Agregar \mathrel{\widehat=} AgregarOk \lor AgregarE$

El siguiente fragmento de programa implementa erróneamente la operación *Agregar*.

```
.......... //Los puntos indican que el programa está incompleto
#define MAX 1000
#define ok 1
#define error 2
..........
int A[MAX], ult; //ult: primera posición de A que debe ocuparse
ult = 0;
..........
int agregar(int x1, int x2)
{int i = 0;

 while((A[i] != x1 || A[i] != x2) && i < ult) //Error en la condición
   i = i + 1;

 if(i == ult)  //Error: posible desborde de arreglo
   {A[ult] = x2; A[ult + 1] = x1; ult = ult + 2;
    return ok;}
 else
    return error;
}
..........
```

a) Aplique el criterio de cubrimiento de condiciones múltiples al programa.

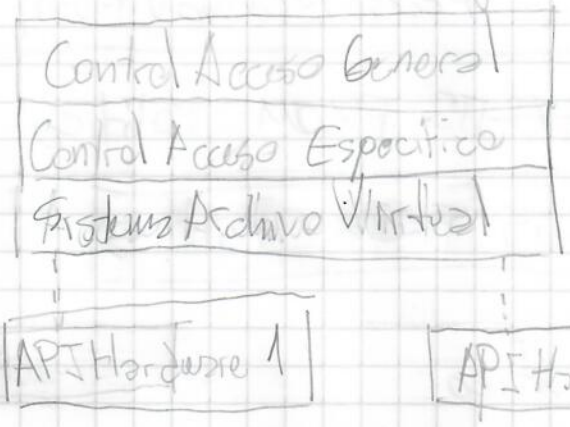
b) Determine si cualquier conjunto de prueba que satisfaga dicho criterio encontraría los errores que se han señalado. Justifique su respuesta.

4. Explique lo más formalmente posible la táctica de testing funcional denominada mutación de especificaciones.

Hoja 1/5

Saur, Yamil Nicolás S-5522/1	Final Ingeniería de Software 2	19/07/24 L.C.C.

1)




```
Module ControlAccesoGeneral
IMPORTS Politicas
EXPORTS abrir(¿UID, ¿String):FD
        cerrar(¿UID, ¿FD)
        leer(¿UID, ¿FD, ¿Int):Bytes
        escribir(¿UID, ¿FD, ¿Bytes).
        setPoliticas()
```

POR QUE NO HEREDA??!

```
Module AccesoArchivoVirtual
EXPORTS abrir(¿UID, ¿String):FD
        cerrar(¿UID, ¿FD)
        leer(¿UID, ¿FD, ¿Int):Bytes
        escribir(¿UID, ¿FD, ¿Bytes)
```

```
Module SistemaArchivo1 INHERITS FROM AccesoArchivoVirtual
IMPORTS API Hardware 1
```

```
Module SistemaArchivo2 INHERITS FROM AccesoArchivoVirtual
IMPORTS API Hardware 2
```

Module PolAcc INHERITS FROM AccesoArchivoVirtual
EXPORTS set AccesoArchivo (i AccesoArchivoVirtual)

Module PolHorario INHERITS FROM Politicas

Module PolíticasHorario INHERITS FROM Políticas

Module ControlAccesoEspecífico
COMPRISES Políticas
PolíticasPermiso
PolíticasHorario.

Module SistemaArchivoVirtual
COMPRISES AccesoArchivoVirtual
SistemaArchivo1
SistemaArchivo2

Guía de Modulos

1. ControlAccesoGeneral

Función: Provee servicios a los usuarios para acceder a archivos mediante un control de acceso con una composición de Políticas específicas que se puede configurar con setPolíticas().

Secreto: Oculta la implementación de las operaciones de acceso y sus políticas.

Szuc, Yani Nicolás
5-5522/1

Final Ingeniería de Software 2

19/02/24
C.C.C.

2. Control Acceso Específico

Módulo que contiene a las políticas que controlan cuestiones en particular.

2.1 Políticas

Función: Otorga operaciones de archivos, teniendo en cuenta si se cumple algunas condiciones especificadas.

Secreto: Oculta la implementación del acceso a archivos y su condición de acceso.

2.2 Políticas Permiso

Función: Verifica si un determinado usuario puede acceder mediante una lista para un archivo dado.

Secreto: Lo mismo que Políticas.

2.3 Políticas Horaria

Función: Verifica si un determinado usuario puede acceder a un archivo dependiendo del día y la hora en que se quiere acceder.

Secreto: Lo mismo que Política.

3. Sistema Archivo Virtual

Módulo que contiene la virtualización de distintos sistemas de archivos que utilizan APIs de distinto hardware.

3.1 AccesoArchivoVirtual

Función: Brinda una interfaz uniforme para acceder a archivos sin importar el sistema o el hardware que utilizan.

Secreto: Oculta la implementación de las operaciones de acceso a archivos según un sistema de archivos con su hardware específico.

3.2 SistemaArchivo1

Función: Virtualización de un sistema de archivos específico.

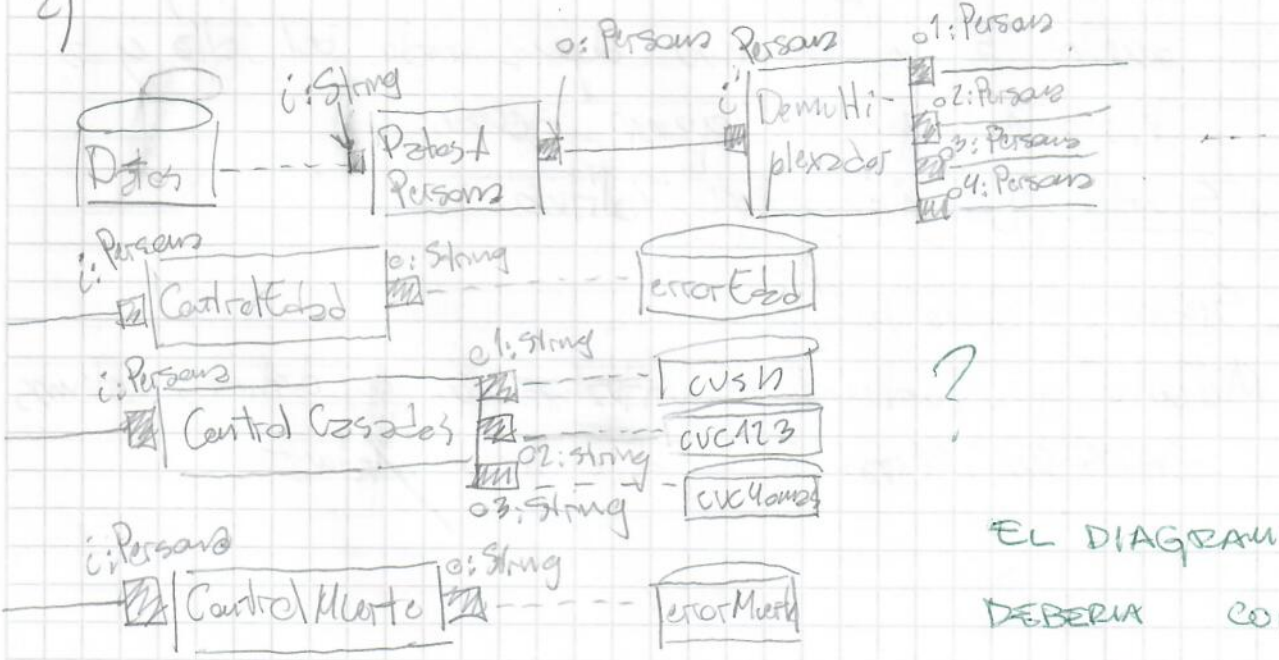
Secreto: Lo mismo que AccesoArchivoVirtual.

3.3 SistemaArchivo2

Función: Virtualización de un sistema de archivos específico.

Secreto: Lo mismo que AccesoArchivoVirtual.

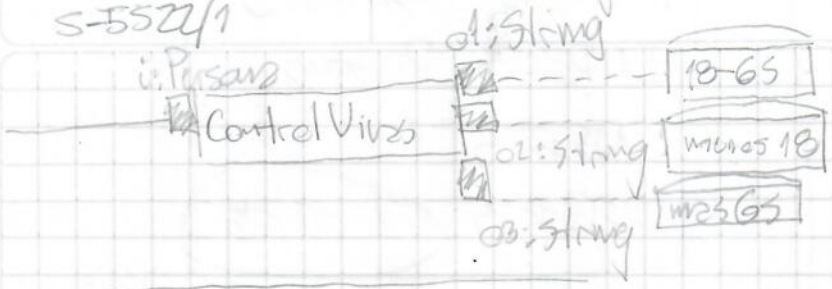
2)



?

EL DIAGRAMA DEBERIA CONTEMPLAR UN FLUJO ESPECIFICO

Saez, Yamil Nicolás Final Ingeniería de Software 2 19/02/24
S-5522/1 C.C.C.



Module Persona
IMPORTS Fecha
EXPORTS setNombre(i String)
getNombre():String
setFechaNac(i Fecha)
getFechaNac():Fecha
setEdad(i Int)
getEdad():Int.
setCantHijos(i Int)
getCantHijos():Int
setEstadoCivil(i String)
getEstadoCivil():String.
setFechaDeceso(i Fecha)
getFechaDeceso():Fecha
isFallecido():Bool.

Module DatosAPersona
IMPORTS Persona
INPORTS i
OUTPORTS o

Module DemultiplexadorI
IMPORTS Persona
INPORTS i
OUTPORTS o1, o2, o3, o4

MEJOR NOMBRES

Module Control1
IMPORTS Persona
INPORTS i
OUTPORTS o

Module Control3
IMPORTS Persona
INPORTS i
OUTPORTS o1, o2, o3

?

MODULE ControlEdad INHERITS FROM Control1

MODULE ControlMuerte INHERITS FROM Control1

MODULE ControlCasados INHERITS FROM Control3

MODULE ControlVivos INHERITS FROM Control3

Module FiltroDatos
COMPRISES DatosAPersona
DemuHipexador

Module FiltroControl
COMPRISES Control1
Control3
ControlEdad
ControlMuerte
ControlCasados
ControlVivos

Muy
Fijo

Saez, Yamil Nicolás
5-5522/1

Final Ingeniería de Software 2

19/02/24
L.C.C.

Guía de Modulos

1. Persona

Función: Guarda la información correspondiente al registro de una Persona.

Secreto: Oculta la forma en que se guarda la información.

2. FiltroDatos

Módulo que guarda los Filtros que afectan el flujo de datos.

2.1 DatosAPersona

Función: Convierte los datos de los registros en un objeto de tipo Persona y lo transmite en el tubo de salida.

Secreto: Oculta el algoritmo de conversión.

2.2 Demultiplexador

Función: Pasa objetos Persona que recibe como entrada los replica en los cuatro puertos de salida.

Secreto: Oculta el algoritmo de demultiplexación.

3. FiltroControl

Modulo que aactiva o los Filtros que controlan los campos de cada registro.

3.1 Control 1

Función: Realiza algún control sobre el objeto de entrada y de ser necesario lo escribe en el flujo de salida hacia un archivo.

Secreto: oculta la implementación del control.

3.2 Control 3

Función: Realiza algún control sobre el objeto de entrada y lo escribe en alguna de los tres puertos de salida hacia un archivo.

Secreto: Lo mismo que Control 1

3.3 Control Edad

Función: Escribe en la salida si la fecha de nacimiento no es correcta

Secreto: Lo mismo que Control 1.

3.4 Control Muerte

Función: Escribe en la salida si la fecha de deceso es inconsistente.

Secreto: Lo mismo que Control 1.

3.5 Control Casados

Función: Escribe en una de las tres salidas dependiendo de la cantidad de hijos de una persona casada.

Secreto: Lo mismo que Control 1.

Sar, Yamil Nicolás | 19/02/24
S-5522/1 | Final Ingeniería de Software 2 | C.C.C.

3.6 Control Vidas

Función: Escribe en una de las tres salidas dependiendo de la edad de la Persona vieja.

Secreto: Lo mismo que Control 1.

3) a)

	A[i] != X1	A[i] != X2	i < cH
①	1	1	1
②	1	1	0
③	1	0	1
④	1	0	0
⑤	0	1	1
⑥	0	1	0
⑦	0	0	1
⑧	0	0	0

- $\langle A=[1],\ X_1=2,\ X_2=3,\ cH=1\rangle$ ①
- $\langle A=[\ ],\ X_1=1,\ X_2=2,\ cH=0\rangle$ ②
- $\langle A=[1],\ X_1=2,\ X_2=1,\ cH=1\rangle$ ⑤⑥
- $\langle A=[1],\ X_1=1,\ X_2=2,\ cH=1\rangle$ ③④
- $\langle A=[1],\ X_1=1,\ X_2=1,\ cH=1\rangle$ ⑦⑧

b) Analizando los casos de prueba podemos encontrar el primer error. Si tomamos $\langle A=[1],\ X_1=1,\ X_2=2,\ cH=1\rangle$ podemos ver que el programa devuelve OK ya que la condición $A[0] != X_2$ Y $i < cH$ es verdadera lo que permite que la operación termine correctamente

cuando $\{1\} \cap \{1,2\} \neq \emptyset$.

En lo que respecta al segundo error no se puede encuadrar con el criterio ya que el error tiene que ver con problemas de desborde y límites en la representación numérica de los tipos.

4) Dada una especificación de un programa y una implementación del mismo hecha por un programador. La técnica de testing llamada mutación de especificaciones requiere escribir la especificación de la implementación que el programador realizó o intuyó sobre la especificación original, para luego encontrar discrepancias entre la especificación original y la "mutada".

Una forma de lograr esto es creando mediante técnicas de testing casos de prueba sobre la nueva especificación y comparar con esos casos en la especificación original.

Esta técnica es muy poderosa porque permite encontrar errores en la implementación de forma muy clara y además nos permite ver si los requisitos están bien representados en las especificaciones, pero también es una técnica complicada de formalizar a la hora de escribir la especificación mutada. ?

FALTA