Primer Examen Parcial 30 de Abril 2009
Ingeniería del Software II 4º Ingeniería Informática
Facultad de Ciencias Universidad de Cantabria

Apellidos, Nombre:
DNI:

PARTE I. Preguntas cortas teóricas (3 puntos, 40 minutos)

Responder con brevedad, pero con precisión y de forma justificada a cada una de las preguntas que se plantean. Cada pregunta tendrá un valor de 0.6 puntos. En esta parte no se dejan apuntes de ningún tipo. Se penalizará claramente la falta de síntesis.

1. Indique brevemente la relación entre los tres niveles de calidad: (1) SQM (Software Quality Management); (2) SQA (Software Quality Assurance) y (3) SQC (Software Quality Control); que se aplican en una organización.

2. Defina los conceptos de complejidad esencial y complejidad accidental y describa muy brevemente, o simplemente cite, un ejemplo de complejidad accidental relacionada con el desarrollo de sistemas software.

3. Defina los conceptos de traza y trazabilidad en el desarrollo de sistemas software. Cite un caso de uso o utilidad de la trazabilidad que NO sea análisis de impacto.

4. Enumere y relacione los distintos niveles de pruebas con las actividades de desarrollo. Indique además, de forma razonada, cuál sería el orden en el que deberían escribirse los casos de prueba y realizarse su ejecución.

5. Justificar brevemente la veracidad o falsedad de la siguiente afirmación: “El modelo CMMI y la Norma ISO 9126 son modelos equivalentes de mejora de procesos para evaluar la calidad del software como producto final”.

Primer Examen Parcial 30 de Abril 2009
Ingeniería del Software II 4º Ingeniería Informática
Facultad de Ciencias Universidad de Cantabria

Apellidos, Nombre:
DNI:

PARTE II. Casos prácticos (7 puntos, 70 minutos)

1. Calidad en sistemas software (2.5 puntos)

1.1 Aplicar la técnica GQM (Goal Question Metric) para establecer una o varias métricas para el siguiente objetivo (goal): "Mejorar la mantenibilidad de las aplicaciones OO desarrolladas" (1.25 puntos)

1.2 Para conseguir alta calidad del software tenemos dos empresas cuyos enfoques son los siguientes:

a) La empresa A lo entiende como la aplicación de una metodología de desarrollo rigurosa, controlando y midiendo el proceso completo de diseño, construcción, testeo e instalación del software pero obviando establecer métricas y controlar el proceso de captura y seguimiento de los requisitos ya que "…estos cambian constantemente porque el usuario final no tiene claro qué quiere y no merece la pena controlar por tanto ese proceso…".

b) La empresa B considera importante el control y medición del proceso de diseño, construcción, testeo e instalación del software pero considera vital y casi más importante el obtener mediciones de cómo se desarrolla el proceso de captura y seguimiento de requisitos.

¿Qué empresa consideras que conseguirá software de mayor calidad?, ¿Por qué?, ¿Crees que la empresa B está incurriendo en mayores costes sin conseguir mayor calidad del producto final por controlar más procesos? Justifica la respuesta. (1.25 puntos)

2. Pruebas en instalaciones deportivas (2.5 puntos)

Se está desarrollando un sistema de gestión de instalaciones deportivas que dispone de varios subsistemas, entre los cuales hay uno para el alquiler de instalaciones. Se dispone de una función encargada de calcular la tarifa que se ha de aplicar según el perfil del usuario que lo solicita. Las entradas de dicha función se especifican más abajo. Además se sabe que el sistema tiene en cuenta el tipo y edad de los socios en el cálculo de las tarifas, de modo que se establecen descuentos para socios "Gold" (50%), "Silver" (25%) y con edad inferior a 18 años (25% acumulable a los descuentos anteriores).

Esta función tiene como entradas:

`Nombre`	Una cadena de caracteres con el nombre del usuario que solicita el servicio
`Password`	Una cadena alfanumérica con la contraseña de entre 8 y 20 caracteres
`Edad`	Un número positivo con la edad del socio
`Tipo_de_socio`	Indica el tipo de socio (`Gold`, `Silver` o `Standard`)

Se pide:

1.1. Realizar una tabla con las clases de equivalencia indicando las clases válidas y no válidas para cada variable de entrada. Numerar las clases obtenidas. (0.75 puntos)

1.2. Obtener casos de prueba para dicha tabla indicando qué clases de equivalencia cubriría cada caso. Indicar dos ejemplos de casos de prueba que apliquen la técnica de análisis de valores límite y pruebas de robustez. (0.5 puntos)

Dado el pseudocódigo de la Figura 1, se pide:

```
validado ← comprobar_password(usuario, password)
if(validado == FALSE) {
   tarifa ← -1
} else {
   tarifa ← Precio_Base – descuento_socio(tipo_de_socio)
   i ← 1
   while(lista_extras(i)){
      elemento ← lista_extras(i)
      tarifa ← tarifa + precio_extra(elemento)
      i++
   }
   If(edad < 18) { tarifa ← tarifa * 0.75 }
}
```

Figura 1 Pseucódigo

2.1. Obtener su correspondiente grafo de flujo y calcular la complejidad ciclomática por las tres fórmulas vistas en clase (señalando las regiones sobre el grafo). (0.75 puntos)

2.2. Definir un conjunto de caminos que garanticen la cobertura de sentencias. (0.5 puntos)

3. Refactorización (2 puntos).

Refactorice el siguiente diseño software que adolece de diversos defectos, principalmente porque ha sido creado por un diseñador que no conoce ni maneja el concepto de patrón de diseño software. Dicho diseño se ilustra en la Figura 2.

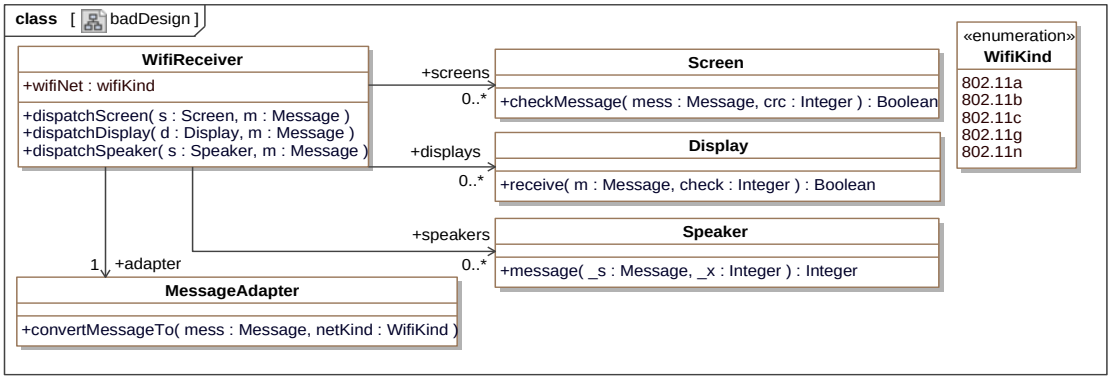

Figura 2 Diseño mal realizado a refactorizar

En este caso, una clase *WifiReceiver* (ver Figura 2) se encarga de recoger mensajes que llegan a través de una red WiFi. Esta red Wifi puede obedecer distintos estándares, más concretamente 802.11a, b, c , g y n. La clase *WifiReceiver* está conectada a tres tipos diferentes de dispositivos que pueden recibir mensajes *Screen, Display* y *Speaker*. La clase *WifiReceiver* tiene un atributo *wifiNet* que indica el tipo de red WiFi donde está desplegada. Dependiendo del tipo de red, hay que seguir un determinado proceso de descompresión y desencriptación del mensaje. Para ello, la clase WifiReceiver usa una clase *MessageAdapter* que ofrece un método, *convertMessage(..),* para convertir y traducir un mensaje dependiendo del tipo de red. El código de este método se describe en la Figura 3.

```
void convertMessageTo(Message mess, WifiKind netKind) {
     switch(netKind){
     case 802.11a:
       // Operations related to conversion in 802.11a
      break;
     case 802.11b:
       // Operations related to conversion in 802.11b
        break;
     case 802.11c: …
     } //endSwitch
} //endConvertMessage
```

Figura 3 Implementación de la operación *convertMesssage(..)*

Cuando llega un nuevo mensaje a través de la red Wifi, se invoca uno de los métodos "*dispatch"* de la clase WifiReceiver, dependiendo del dispositivo destinario. A continuación, la clase *WifiReceiver* invoca el método *checkMessage(..)*, *receive(..) ó message(..)* de la clase que corresponda al dispositivo destino. Estos tres métodos realizan una función parecida: reciben como parámetros un mensaje y un código de control de errores calculado a partir del mensaje original. Si tras realizar unos cálculos con el mensaje recibido, el resultado es igual al código de control significa el mensaje no está corrupto, y el método devuelve un valor a la clase WifiReceiver indicando que el mensaje ha llegado correctamente. En caso contrario, se devuelve un valor a la clase *WifiReceiver* indicando que el mensaje está corrupto, por lo que la clase *WifiReceiver* deberá reenviarlo de nuevo. La Figura 4 ilustra la traza de ejecución descrita.

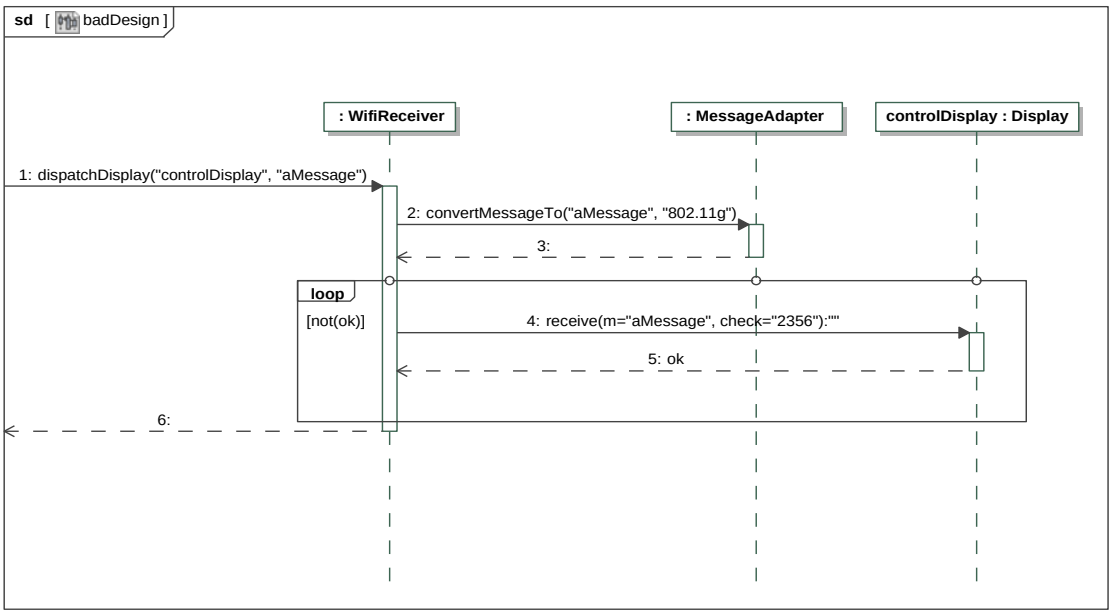

Figura 4 Traza de ejecución del código a refactorizar

Al aplicar un algoritmo de detección de clones al código correspondiente a este diseño, se detecta una similitud, a nivel de sintaxis abstracta, del 90% entre las operaciones *checkMessage(..)*, *receive(..) y message(..).*

Con esta información, muestre el diagrama de clases resultado de la refactorización y describa que refactorizaciones se deben realizar. Basta con indicar el nombre de la refactorización a aplicar, las clases o métodos afectados y el objetivo de tal refactorización. Se valorará muy positivamente la precisión y brevedad en las respuestas.