1 Relación de ejercicios. *Tema 4*

68. Categorizar las siguientes situaciones como: *fallo*, *defecto* o *error*. Para cada uno de ellos, explicar por qué pertenece a esa categoría.
    (a) Un ingeniero de software, trabajando apresuradamente, borra inintencionadamente una línea importante de código fuente Solución: Error, ya que resulta en un defecto del software y produce fallos posteriores de ese software.
    (b) El 1 de Enero de 2040 el sistema informa que la fecha es 1 Enero de 1940 Solución: Fallo que proviene de un defecto del software
    (c) No se suministra ninguna documentación de diseño o comentarios en el código fuente para un algoritmo muy complejo Solución: Defecto
    (d) Un array de tamaño fijo de longitud 10 se utiliza para mantener la lista de cursos de un estudiante durante un semestre. Los requisitos no se pronuncian acerca del número máximo de cursos que un estudiante puede cursar al mismo tiempo. Solución: Defecto
69. ¿Qué ocurre cuando el siguiente programa `java` es llamado: (a)sin argumentos,
    Solución:
    El programa captura en (19) la excepción provocada por el array `argv[]` sin asignar y se producirá la siguiente salida: `Debes especificar un argumento`.

    (b)con una cadena como argumento,
    Solución:
    El programa captura en la línea (23) la excepción producida por pasar una cadena ($\neq$ `int`) como argumento del programa y se producirá la siguiente salida `Debes especificar un argumento entero`.

    (c) con argumentos enteros 0, 1, 2 y 99?
    Solución:

    - Con argumento 0: se lanza `MiExcepcion` desde el método `c(...)` en la línea (58), que resulta capturada en la línea (33) del método `a(...)`, produciendo la siguiente salida:
      ```
      i=0
      MiExcepcion:  entrada demasiado baja
      Manejado en el punto 1
      ```
    - Con argumento 1: se lanza `MiSubExcepcion` desde el método `c(...)` en la línea (59), que resulta capturada en la línea (33) del método `a(...)`, produciendo la siguiente salida:
      ```
      i=1
      MiSubexcepcion:  entrada todavia muy baja
      Manejado en el punto 1
      ```
    - Con argumento 2: se ejecutará sin lanzar ninguna excepción y se produciría la siguiente salida:
      ```
      i=2
      c(i) = 4
      ```
    - Con argumento 99: se lanza `MiOtraExcepción` desde el método `c(...)` en la línea (60) que resulta capturada en (49) del método `b(...)` y se produciría la siguiente salida:
      ```
      i=99
      MiOtraExcepcion:  entrada muy alta
      Manejado en el punto 2
      ```

```
1 class MiExcepcion extends Exception{
2   public MiExcepcion(){ super();}
3   public MiExcepcion(String s){ super(s);}
4 }
5 class MiOtraExcepcion extends Exception{
6   public MiOtraExcepcion(){ super();}
7   public MiOtraExcepcion(String s){ super(s);}
8 }
```

```
class MiSubExcepcion extends Exception{
  public MiSubExcepcion(){ super();}
  public MiSubExcepcion(String s){ super(s);}
}
public class throwtest{
   public static void main(String argv[]){
     int i;
     try{
       i= Integer.parseInt(argv[0]);
     }
     catch(ArrayIndexOutOfBoundsException e){
       System.out.println("Debes especificar un argumento");
       return;
     }
     catch(NumberFormatException e){
       System.out.println("Debes especificar un argumento entero");
       return;
     }
     a(i);
   }
public static void a(int i){
  try{
    b(i);
  }
  catch(MiExcepcion e){//Punto 1
    if(e instanceOfMiSubExcepcion)
      System.out.print("MiSubEXcepcion:");
    else
      System.out.print("MiExcepcion:");
    System.out.println(e.getMessage());
    System.out.println("Manejado en el punto 1");
  }
}
public static void b(int i) throws MiExcepcion{
  int result;
  try{
    System.out.print("i="+i);
    resultado= c(i);
    System.out.print("c(i)="+resultado);
  }
  catch(MiOtraExcepcion e){  //Punto 2
    System.out.println("MiOtraExcepcion:"+e.getMessage());
    System.out.println("Manejado en el punto 2");
  }
  finally{
    System.out.print("\n");
  }
public static int c(int i) throws MiExcepcion, MiOtraExcepcion{
  switch(i){
    case 0: throw new MiExcepcion("entrada demasiado baja");
    case 1: throw new MiSubExcepcion("entrada todavia muy baja");
    case 99: throw new MiOtraExcepcion("entrada muy alta");
    default: return i*i;
  }
}
}
```

70. Sea P un componente de un programa que lee una lista de N registros y un *rango* de condición sobre la clave del registro. Por ejemplo, el rango: 'JONES' .. 'SMITH' producirá como archivo de salida que contiene todos los registros cuyas claves estén comprendidas lexicográficamente entre 'JONES' y 'SMITH'. Los primeros 7 caracteres del registro constituyen la clave. El componente P lee la clave y produce un archivo de salida que contiene únicamente aquellos registros que caen dentro del rango prestablecido. Escribir las condiciones de entrada y salida como afirmaciones para ser utilizadas para verificar si P es correcto. Preparar un diagrama de flujo para mostrar cómo podría ser el flujo lógico de P e identificar los puntos de transformación.

    Solución:

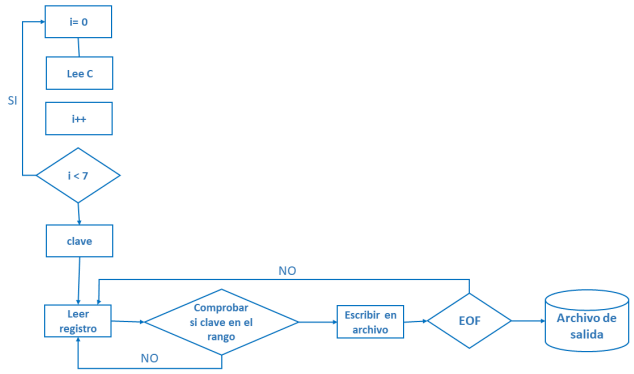


Figura 1: Diagrama de flujo del programa de identificación de clave.

71. Examinar las categorías de defectos en el esquema de clasificación de Hewlett-Packard, mostrado en la figura 2 ¿Corresponde a una clasificación ortogonal de defectos? si no lo fuera, explicar por qué y proponer una manera de convertirla en ortogonal.

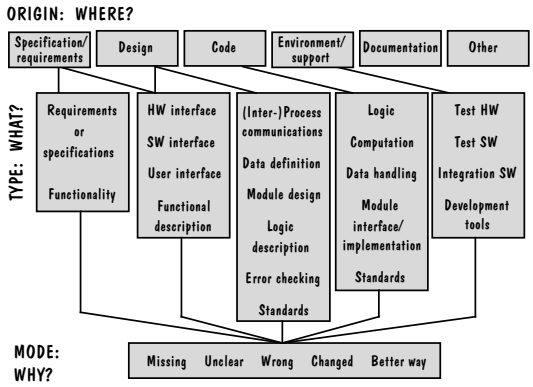


Figura 2: Clasificación de defectos de Hewlett-Packard

Solución:
La clasificación de la figura 2 no es ortogonal porque hay tipos de errores que son comunes a varias categorías. Ver una posible clasificación ortogonal en tabla 1.

1	Función	capacidad, estructura de datos, interfaces SW/HW
2	Interfaces	defecto en la interacción con otro componente
3	Comprobación	defecto en la lógica del programa
4	Asignación	estructuras de datos o inicialización
5	Sincronización	recursos compartidos
6	Construcción	defectos en repositorios o gestores de cambios
7	Documentación	incompleta o mala
8	Algoritmos	defecto en la eficiencia o en la exactitud

Tabla 1. Clasificación ortogonal de defectos de IBM.

72. Completar la prueba del ejemplo ilustrado en la figura 3. En otras palabras, escribir los enunciados que se corresponden con el diagrama de flujo. Encontrar también los caminos entre la condición de entrada y la de salida.

73. Suponer que un programa contiene n puntos de decisión, cada uno de los cuales tiene dos ramas ¿Cuántos casos de prueba se necesitan para realizar la prueba de caminos?¿Puede deducirse ese número gracias a la estructura del programa? Dar un ejemplo que confirme y justifique las respuestas.
Solución: Si es un árbol, el número de caminos = E - N +1

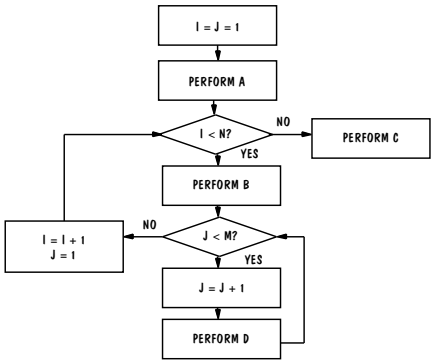


Figura 3: Ejemplo de estructura lógica expresada como diagrama de flujo

74. Considerar un diagrama de flujo de un programa como un grafo dirigido en el cual los rombos y cajas del programa se consideran como nodos y las flechas de flujo lógico entre ellos como aristas orientadas del grafo. Por ejemplo el programa de la figura 3 siguiente puede graficarse como se muestra al lado. Probar que la prueba de sentencias de un programa es equivalente a encontrar el conjunto de caminos del grafo que contienen a todos los nodos del grafo. Probar que la prueba de ramificación es equivalente a encontrar el conjunto de caminos cuya unión cubre el conjunto de aristas del grafo. Por último, probar que la prueba de caminos es equivalente a encontrar todos los caminos posibles a través del grafo.

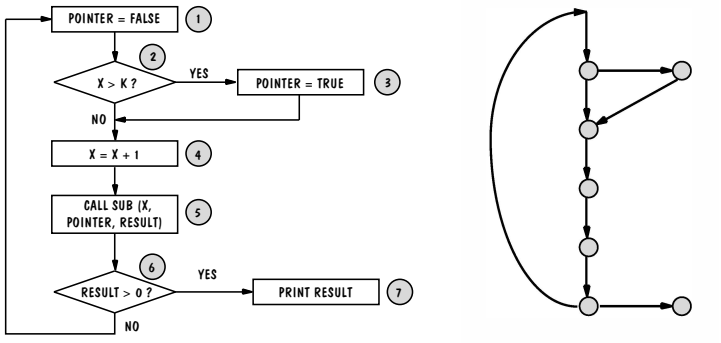


Figura 3: (a) Diagrama de flujo de un programa simple (b) Grafo orientado equivalente

Solución:

- número de caminos = E - N + 2
- E= 8, N= 7 y caminos= 3

75. *Problema programable:* escribir un programa que acepte como entrada los nodos y aristas de un grafo orientado e imprima como salida todos los posibles caminos a través del grafo ¿Cuál es la principal consideración de diseño para este programa?¿Cómo afecta la complejidad del grafo al algoritmo utilizado?

76. La figura 4 muestra la jerarquía de componentes de un sistema software. Describir la secuencia de pruebas de integración de los componentes utilizando los diversos enfoques: ascendente, descendente, descendente modificado, *big-bang*, emparedado y emparedado modificado.

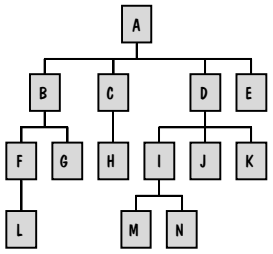
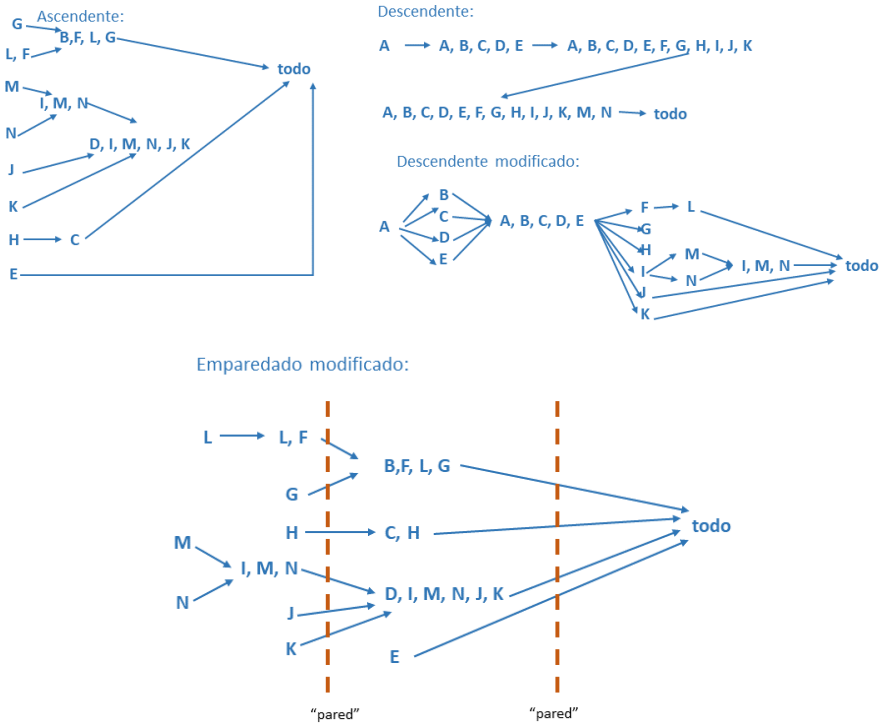


Figura 4: Ejemplo de jerarquía de componentes

Solución:



77. ¿Cuáles son las explicaciones posibles para el comportamiento de la gráfica presentada en la figura 5?

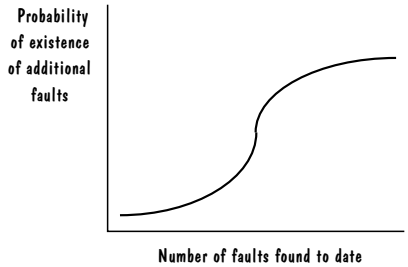


Figura 5: Probabilidad de localizar defectos durante el desarrollo

Solución:

- los valores en abcisas representan los fallos encontrados en ese software
- A partir del punto de *torsión* de la curva, la probabilidad de que todavía existan un número excesivamente alto de fallos en el software es muy grande

78. Un programa se siembra con 25 defectos. Durante la prueba, se detectan 18 defectos, 13 de los cuales son defectos sembrados y 5 son defectos autóctonos, ¿Cuál es la estimación del número de defectos autóctonos que permanecen sin detectar en el programa? Solución:
    - Correlación que guardan los defectos detectados (s:sembrados y n:autóctonos) y los totales incluidos en el programa (S:sembrados y N:autóctonos)
    - $\frac{S}{s} = \frac{N}{n}$
    - $N = 5 \cdot \frac{25}{13}$
    - $N = 9$

79. Un programador afirma que sus programas están libres de defectos con un nivel de certeza del 95%. El plan de prueba afirma que los programas han de ser probados hasta encontrar la totalidad de los defectos sembrados ¿Con cuántos defectos debe sembrarse el programa antes de la prueba a fin de probar la afirmación anterior? Si por alguna razón el programador no piensa encontrar todos los defectos sembrados, ¿cuántos defectos sembrados requiere la fórmula de Richards en su lugar?
    Solución:
    - Fórmula de Richards: $C = 1\ si\ n > N; en\ otro\ caso = \frac{S}{S-N+1}$
    - $\frac{S}{S-N+1} = \frac{95}{100}$
    - $S = 19$

80. Explicar por qué la gráfica de la figura 5 puede interpretarse para que signifique que, si se encuentran demasiados defectos en el código al momento de la compilación, lo mejor es descartar el código y escribirlo de nuevo.
    Solución:
    Porque la etapa de compilación del programa estaría en la parte izquierda de la curva, mucho antes de llegar al punto de torsión de la misma, por tanto, se esperaría un número excesivamente alto de fallos más adelante.

81. Crearse una tabla de clases de equivalencia para cada uno de los siguientes problemas de única entrada. Ayuda: situar una entrada en una clase de equivalencia distinta si existe incluso una remota posibilidad de que algún algoritmo razonable pudiera tratar la entrada de una manera especial.
    (a) Un número telefónico para ser utilizado por un marcador automático
    (b) El nombre de una persona escrito con caracteres latinos
    (c) Una franja horaria, que puede ser especificada bien nuéricamente dando su diferencia con UTC (GMT) o alfabéticamente utilizando un conjunto de códigos estándar (EST,BST,PDT, etc.)
    (d) La velocidad de un vehículo, que es un entero de 3 dígitos, que podría ser seguido por las unidades: km/h, m/s, o mph (km/h es la unidad por defecto)
    (e) El número de una tarjeta de crédito
    (f) Una frecuencia de radio FM
    (g) Un URL

    Solución:
    (a) Clases de equivalencia de
        - datos válidos: cualquier cadena de caracteres ASCII imprimibles (incluyendo letras, números y símbolos especiales)
        - datos inválidos: la cadena vacía, suponiendo que se obliga a realizar una entrada

(b) Clases de equivalencia de

- datos válidos: cadena de caracteres alfabéticos de longitud 2–30 con 1 ó más espacios simples entre ellos, pero no al comienzo o al final; cadenas 2–30 de longitud con caracteres con acentos y otros signos
- datos inválidos: cadena vacía, cadenas de caracteres no latinos, cadenas con números, espacios, barras, etc.

(c) Clases de equivalencia de

- datos válidos: secuencias de 3 letras mayúsculas + `shh[:]mm`, con $\texttt{s} = \prime+\prime|\prime-\prime$, $\texttt{h} \in [0, 14]$ (14 usos horarios) y $\texttt{mm} \in \{0, 30\}$
- datos inválidos: cadena vacía, cadena de longitud 1 ó 2, zona sin $\texttt{s} = \prime+\prime|\prime-\prime$, zona numérica con una hora > 14, con minutos $\neq \{0, 30\}$, secuencia de caracteres de longitud mayor que 3 o una cadena con letras y números

(d) Clases de equivalencia de

- datos válidos: secuencia de dígitos de longitud 1–3, secuencias de dígitos de longitud 1–3 seguidos por una de las cadenas: *Km/h, m/s, mph*
- datos inválidos: cadena vacía, secuencia de dígitos mayor que 3, una cadena que incluya caracteres distintos de las subcadenas: *Km/h, m/s, mph*

(e) Clases de equivalencia de

- datos válidos: una cadena de 7 ó más dígitos, opcionalmente con espacios que separen los números
- datos inválidos: cadena vacía, secuencias que tengan caracteres distintos de dígitos o espacios, cadenas de longitud menor que 7

(f) Clases de equivalencia de

- datos válidos: una cadena $\in [87.5, \ldots, 108.0]$ que contenga hasta 2 decimales, pero el segundo ha de ser $\{0, 5\}$
- datos inválidos: cadena vacía, cadenas no numéricas o aquellas que no sigan la especificación anterior

(g) Clases de equivalencia de (sólo `http`, `https`, `ftp` y archivos)

- datos válidos:
  1. Una cadena con el formato: `http://{dominio}`, incluyendo caracteres no occidentales
  2. (1a) Clase 1 seguida por '/'
  3. (1b) Clase 1 seguida por un texto arbitrario de longitud indefinida
  4. Lo mismo que las de clase 1 pero sustituyendo el prefijo por `https`
  5. Lo mismo que las de clase 1 pero sustituyendo el prefijo por `ftp`
  6. Lo mismo que las de clase 1 pero sustituyendo el prefijo por `file`
- datos inválidos: Una cadena con un prefijo distinto de `http://` , `https://` , `ftp://` o `file://`

82. Describir un buen conjunto de equivalencia de clases de test para un formulario que pregunta información personal: apellidos, nombre, fecha de nacimiento, calle, ciudad, país,código postal y teléfono particular.
Solución:

(a) Clases de equivalencia de fecha de nacimiento
Año:

- datos válidos: consideramos esto como 3 entradas separadas con un formato definido como dd/mm/aaaa
  - Un entero válido desde 1890 hasta 2025 (suponiendo que este sistema es para personas vivas mientras esté en uso)
  - Valores frontera: 2000, 2001, 1890, 2025

- datos inválidos:
  - Enteros en el rango [0..99] (lo tratamos como una caso aparte porque podría existir algún código que maneje fechas de 2 dígitos incorrectamente, ya que hemos decidido no permitir este formato
  - Valores frontera: 0, 1, 99
  - Enteros en el rango [100..1889]
  - Valores frontera: 100, 1889
  - Cadena vacía
  - Cadena con caracteres no numéricos

Mes:

- datos válidos:
  - Enteros en el rango [01..12]
  - Valores frontera: 01, 12
- datos inválidos:
  - Enteros en el rango [1..9] (suponer que necesitamos explícitamente 2 dígitos)
  - Valores frontera: 1, 9
  - Los valores: 00 y 0
  - Los valores en el rango [13..99]
  - Valores frontera: 13, 99
  - Cadena vacía
  - Cadena con caracteres no numéricos

Día:

- datos válidos:
  - Enteros en el rango [01..28]
  - Valores frontera: 01 y 28
  - El 29 de Febrero en años bisiestos o en cualquier otro mes de cualquier año
  - El valor 30 en cualquier mes que no sea Febrero
  - El valor 31 en: Enero, Marzo, Mayo, Julio, Agosto, Octubre y Diciembre
- datos inválidos:
  - El valor 29 de Febrero en un año no bisiesto
  - El valor 30 en el mes de Febrero
  - El valor 31 en los meses: Febrero, Abril, Junio, Septiembre y Noviembre
  - Los valores en el rango [32..99]
  - Los valores 00 y 0
  - Los valores en el rango [1..9] porque los valores han de ser de 2 dígitos
  - Valores frontera: 1 y 9
  - Cadena vacía
  - Cadena con caracteres no numéricos

(b) Dirección postal:

- datos válidos:
  - Cadenas de caracteres de longitud 2–40 (suponemos estos límites) que contienen letras mayúsculas y minúsculas, números, comas, puntos, guiones, espacios simples, acentos y signos diacríticos (~ ` ^)
  - Valores frontera: Cadenas de longitudes 2 y 40
- datos inválidos:
  - Cadenas que contienen caracteres no imprimibles
  - Cadenas que contienen internamente más de un espacio contiguo
  - Cadenas de longitud 0 ó 1

(c) Ciudad, País y Código Postal:

- La misma especificación que para la dirección postal salvo que el país y el código postal pueden tener 0 caracteres

(d) Número de teléfono particular:

- datos válidos:
  - Cadena vacía
  - Cadena de caracteres que incluya dígitos, '+','-','(',')','-','x' y espacio
- datos inválidos:
  - Cualquier otra cadena de caracteres

83. Ampliar la respuesta del ejercicio anterior para incluir valores frontera de clases de equivalencia que hayan de ser probados.

Solución: Incluída en la solución del ejercicio anterior.

84. Crear las tablas que sean precisas para ayudarnos a probar las condiciones correspondientes con los siguientes requerimientos.
    (a) Los solicitantes de menos de 18 años deben utilizar el formulario A, mientras que los que tengan 18 o más deben utilizar el formulario B. Las personas minusválidas de cualquier edad, excepto aquellos con asistencia social, deben utilizar el formulario C. Las personas con asistencia social deben completar el formulario D, además del A o del B. Los ciudadanos mayores deben completar el formulario E además de otros formularios, salvo que tengan asistencia social o estén viviendo en una residencia. Cualquiera que gane más de 15.000 EUR al año debe rellenar el formulario F además de los otros

    Solución:

Discapacitado	Asistencia	Menor de 18	Formulario
No	?	No	B
No	?	Sí	A
Sí	No	?	C
Sí	Sí	No	B
Sí	Sí	Sí	A

Tabla 4. Para formularios A, B y C.

Asistencia	Formulario
Sí	Sí
No	No

Tabla 5. Para formulario D.

Mayor	Asistencia	en Residencia	Formulario
No	?	?	No
Sí	No	No	Sí
Sí	?	Sí	No
Sí	Sí	?	No

Tabla 6. Para formulario E

Discapacitado	Mayor	Ingresos altos	Formulario
Sí	?	?	No
?	Sí	?	No
No	No	Sí	Sí
No	No	No	No

Tabla 7. Para formulario F

    (b) El usuario C envía un mensaje al usuario A. El mensaje debe ser reenviado de A a B si las siguientes condiciones son ciertas: 1) A ha solicitado que sus mensajes sean reenviados a B; 2) B está conectado; y 3) B no ha pedido que se bloqueen los mensajes que le lleguen de A o C. Además si se declara el mensaje como de emergencia, entonces no puede ocurrir ningún bloqueo.

    Solución:

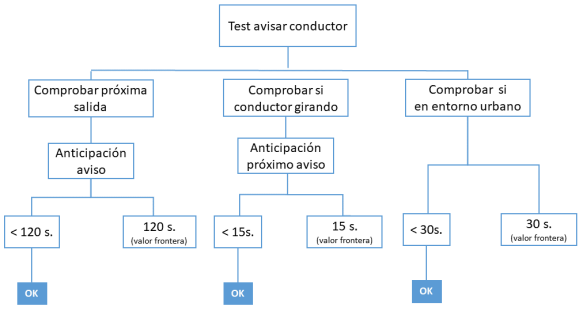
A-solicita	B-conectado	B-solicita	Emergencia	Reenviar
No	?	?	?	No
?	No	?	?	No
Sí	Sí	?	Sí	Sí
Sí	Sí	No	No	Sí
Sí	Sí	Sí	No	No

Tabla 8. Envío de mensajes.

(c) El sistema de navegación debe anunciar a los conductores que tienen que girar hacia una nueva salida dos minutos antes. Se pueden producir excepciones en las siguientes circunstancias. Primera, si un conductor está completando todavía un giro anterior, entonces el aviso del nuevo giro debe ser retrasado hasta que el primer giro termine, pero aún en ese caso se le debe dar un aviso de 15 segundos. Segunda, en un entorno urbano (donde el conductor está circulando bajo un límite de velocidad) encontraría, al menos, un posible giro cada 30 segundos, entonces el sistema avisará al conductor sólo 30 segundos antes que se necesite girar.

Solución:

	Anunciar conductor	**Próxima salida**	**Tiempo anticipación**	**Completar giro**	**Entorno urbano**
	No	No	$\leq$ 120 s.	No	No
	Sí	Sí	$\leq$ 120 s.	No	No
	Sí	Sí	$\leq$ 30 s.	No	Sí
	No	Sí	$\leq$ 15 s.	Sí	No
	No	Sí	$\leq$ 15 s.	Sí	Sí
	Sí	Sí	$\leq$ 15 s.	No	No
	Sí	Sí	$\leq$ 15 s.	No	Sí



85. Java posee una capacidad de clasificación incluida, que se encuentra en las clases `Array` y `Collection`. Probar experimentalmente si estas clases contienen o no algoritmos estables y eficientes.

86. Describir el tipo de defectos numéricos presentes cuando el siguiente código es programado en un programa. Suponiendo que no se conocía la implementación, explicar cómo haríamos para intentar detectar tales defectos:

    (a) `double x, y; ... if(x/3.0==y){...}`

    (b) `int activosCorporativosTotales; //En EUR`

    (c) `short precioGasolina; // En EUR decimas de centimo por litro`

    (d) Administramos una página Web que implementa micro–pagos; le factura a los usuarios 1/3 de céntimo por cada click en la página. Acumulamos la factura de cada cliente en un entero donde cada unidad representa 1/10 de céntimo. Sin embargo, registramos el dinero ganado por cada página con un valor en coma flotante.

Solución:

(a) Está suponiendo que el valor en coma flotante de una variable va a ser exactamente igual que 3.0 cuando su valor real podría ser ligeramente diferente.

(b) No está utilizando suficientes bits pata almacenar los valores máximos: los `activos corporativos totales` podrían muy probablemente exceder el valor del mayor `int`. Para comprobar que no exista este defecto, simplemente cambiar el tipo de datos para que pueda contener un valor mucho más grande de activos corporativos. Téngase en cuenta que algunas personas podrían requerir que sus activos se indiquen al céntimo de Euro, pero pero la mayoría de agencias no registrarían el volumen de sus activos hasta este nivel de precisión.

(c) No está utilizando suficientes bits para almacenar los valores máximos que puede alcanzar el precio de la gasolina, ya que dicho precio podría subir indefinidamente.

(d) El defecto más evidente en este caso consiste en que guarda el precio total con un valor en coma flotante. Cuando la página Web reciba un número muy grande de visitas, el valor en coma flotante dejará de funcionar correctamente porque se estaría sumando un valor entero muy pequeño a un número muy grande y por consiguiente el número en coma flotante no cambiaría su valor.

87. Si estuviéramos diseñando los siguientes tipos de software, a qué pruebas de *stress* y situaciones excepcionales someterías el sistema:

(a) Un navegador Web nuevo
Solución:
- Cargar una página Web muy cargada y compleja
- Cargar una página Web que contenga mucho código desarrollado con *javascript*
- Cargar una página Web que contenga varios applets de Java *pesados*
- Cargar una página Web llena de errores en `html`
- Cargar una página Web que contenga código `html` obsoleto
- Cargar una página Web que contenga el `html` más complejo y reciente así como varias hojas de estilo anidadas
- Cargar una página Web que contenga un número muy grande de niveles de anidación de etiquetas `html`
- Cargar una página Web que contenga un número muy grande de columnas y filas en tablas
- Cargar una página Web que demande un campo muy ancho para la salida de resultados
- Cargar una página Web que contenga un número muy grande de objetos incrustados
- Probar el navegador utilizando una conexión de red extremadamente lenta
- Cargar un número muy grande de páginas Web en varias ventanas de una sola vez
- Ejecutarlo para diferentes versiones del sistema operativo
- Ejecutarlo en una máquina con muy poca memoria
- Ejecutarlo en una máquina con muy poco espacio en disco para ser utilizado como cache
- Utilizar un *driver* para cargar páginas aleatorias una y otra vez durante mucho tiempo
- Ejecutarlo para una pantalla muy pequeña
- Tratar con fallos de conexión que se repitan

(b) Un simulador de vuelo
Solución:
- Ejecutarlo con una CPU que posea poca memoria y velocidad
- Ejecutarlo con diferentes sistemas operativos y tarjetas gráficas
- Ejecutarlo durante un periodo largo de tiempo (para comprobar si se producen o no *fugas* de memoria)
- Ejecutarlo con una pantalla que tenga una resolución en pixels muy alta o bien juntar varias pantallas para crear una pantalla virtual más grande
- Ejecutarlo con una pantalla muy pequeña
- (en el caso de tratarse de un simulador que pueda operar con varios usuarios que vuelan cada uno su avión) ejecutarlo con número grande de aviones concurrentemente

(c) Un sistema de chat simple
Solución:
- Conectar un número muy grande de usuarios a un determinado servidor
- Que un driver envíe un número muy grande de mensajes de una sola vez
- Ejecutarlo con tantos sistemas operativos como sea posible

(d) Un sistema de navegación

Solución:
- Tratar errores que pudieran aparecer en la información de ubicación
- Pruebas de navegación en movimiento rápido a través de ciudades densas y complejas para circular
- Tratar con conexiones a Internet intermitentes cuando se han de cargar mapas
- Tratar con una señal GPS intermitente ocasionada por edificios altos y túneles
- Probar a navegar durante una larga distancia por una carretera complicada

88. Escribir casos de test completos para cada una de las situaciones de prueba listadas en el ejercicio (87)

89. Expandir la siguiente tabla en un plan de pruebas completo:

Etapa de vuelo	Visibilidad	Tren de aterrizaje	Resultado requerido
2-3 min. del despegue	$\geq$ 1000 ft	Desplegado	Sin alarma
2-3 min. antes aterrizar	$\geq$ 1000 ft	Desplegado	Sin alarma
2-3 min. del despegue	$\geq$ 1000 ft	No desplegado	Sin alarma
2-3 min. antes aterrizar	$\geq$ 1000 ft	No desplegado	Sin alarma
2-3 min. del despegue	$<$ 1000 ft	Desplegado	Sin alarma
2-3 min. antes aterrizar	$<$ 1000 ft	Desplegado	Sin alarma
2-3 min. del despegue	$<$ 1000 ft	No desplegado	Alarma!
2-3 min. antes aterrizar	$<$ 1000 ft	No desplegado	Alarma!

Tabla 9. Requerimientos de seguridad del sistema de control del tren de aterrizaje de aviones-1.

Etapa de vuelo	Altitud	Tren de aterrizaje	Resultado requerido
2-3 min. del despegue	$\geq$ 2000 ft	Desplegado	Sin alarma
2-3 min. antes aterrizar	$\geq$ 2000 ft	Desplegado	Sin alarma
2-3 min. del despegue	$\geq$ 2000 ft	No desplegado	Sin alarma
2-3 min. antes aterrizar	$\geq$ 2000 ft	No desplegado	Sin alarma
2-3 min. del despegue	$<$ 2000 ft	Desplegado	Sin alarma
2-3 min. antes aterrizar	$<$ 2000 ft	Desplegado	Sin alarma
2-3 min. del despegue	$<$ 2000 ft	No desplegado	Alarma!
2-3 min. antes aterrizar	$<$ 2000 ft	No desplegado	Alarma!

Tabla 10. Requerimientos de seguridad del sistema de control del tren de aterrizaje de aviones-2.

Solución:
Suponemos que si el avión se encuentra a una altitud < 2000ft, o hasta 2 min desde el despegue o hasta el aterrizaje, o la visibilidad es < 1000ft, entonces el tren de aterrizaje ha de estar desplegado en cualquier caso.

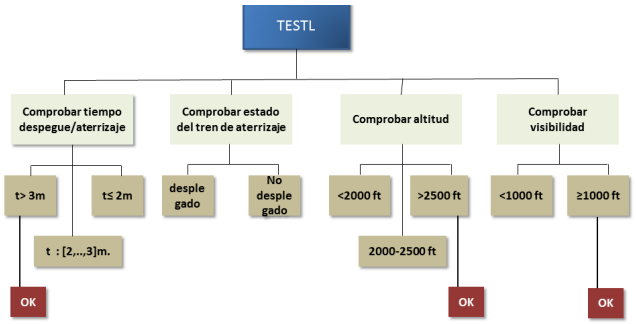


Figura 6 (a). Solución ejercicio 89(a)

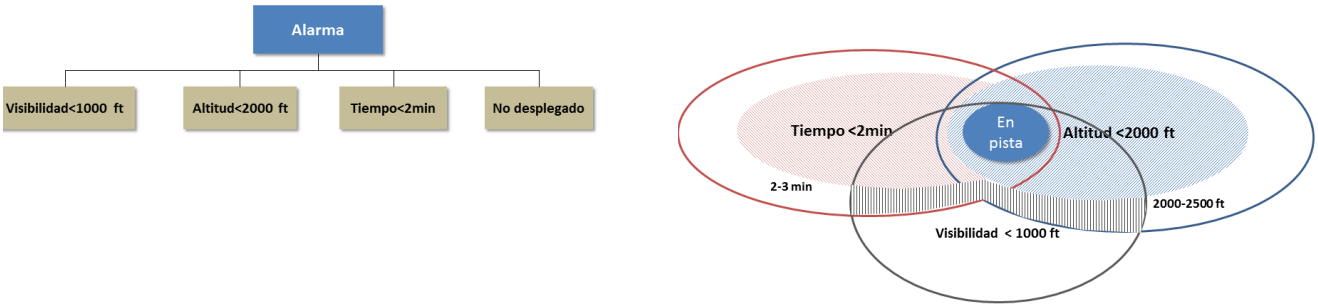


Figura 6(b)(c). Solución ejercicio 89: (b) prueba de integración y (c) alarma en las zonas sombreadas

90. Discutir cómo la *prueba de integración* podría realizarse en un sistema de chat simple ¿Qué *stubs* o *drivers* podrían escribirse para permitir realizar una prueba independiente de las capas de componentes?

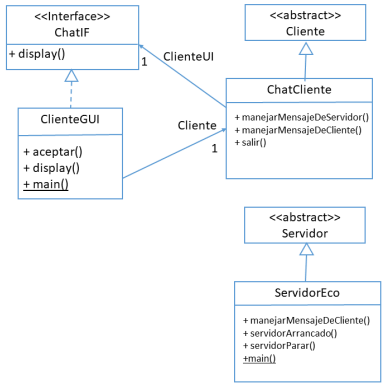


Figura 6: Posible implementación de un sistema de chat simple

Solución:
Suponiendo la implementación mostrada en la figura 6:

- Parte servidora:
  - el método main() crea una nueva instancia de ServidorEco y comienza escuchar de las conexiones establecidas con el servidor,
  - también llamará al método `sendToAllClients()` para hacer eco de cualquier mensaje recibido de 1 cliente a todos los demás
- Parte del cliente:
  - ChatCliente tiene 2 métodos que son llamados por la interfaz con el cliente: ClienteGUI.
  - La interfaz de cliente ClienteGUI está separada cuidadosamente de la parte funcional del cliente.
  - Cualquier clase que implemente el método `display()` podría sustituir a la clase ClienteGUI.
  - El método `main()` de ClienteGUI se ejecutará al comenzar un nuevo cliente: creará instancias de ClienteGUI y ClienteChat, el cual se ejecutará como 1 hebra independiente, después llamará al método `aceptar()` para esperar una entrada del cliente.
  - `aceptar()` se ejecuta en bucle hasta la terminación del programa y envía todas las entradas del cliente a ClienteChat, que llamará a su método `manejarMensajeDeCliente()` que, a su vez, llamará al método `sendToServer()`.
  - La comunicación que proviene del servidor se gestiona con `manejarMensajeDeServidor()`, que llama a la operación `display()` de ChatIF, lo que provoca una llamada al método `display()` de la clase ClienteGUI.

Prueba de integración solicitada:
Escribir una clase `DriverClientes` que permitirá comunicar directamente los clientes con el servidor

sin utilizar la clase ChatCliente. Para lo cual, se creará una instancia de ServidorEco a la que se pasarán las referencias de ClienteGUI y de la clase `DriverClientes`.

La clase ClienteGUI ha de ser modificada para que envíe todas los mensajes que proceden de la entrada de los clientes al servidor y ha de ignorar todos los mensajes que emanan de ClienteChat.

ServidorEco ha de ser modificado para enviar los mensajes que reciba a todos los clientes que se hayan conectado al servidor hasta ese momento. La clase DriverClientes ignorará los mensajes que se envían a ClienteGUI y ésta ignorará los mensajes que se envíen a la primera.

91. ¿Por qué puede ser muy complicado realizar pruebas unitarias a un módulo altamente acoplado?

    Solución:
    porque no se pueden probar sus operaciones y sus métodos de forma independiente.

92. ¿Es siempre posible desarrollar una estrategia para probar software que utiliza la secuencia de pasos de prueba descrita más abajo?
    (a) Pruebas unitarias
    (b) Pruebas de integración
    (c) Pruebas de validación
    (d) Pruebas del sistema

    Solución:
    No. Si los componentes software de la aplicación son *poco cohesivos*, las *pruebas unitarias* no siempre se pueden llevar a cabo. En el caso de las *pruebas de integración* hay que considerar más cosas que las secuencias de casos de prueba resultado de las pruebas unitarias. Las *pruebas de validación* dependen del estado del sistema, ya que un conjunto de datos de prueba podría producir como resultado el correcto funcionamiento del sistema para un estado del mismo. Siempre se puede desarrollar una estrategia para realizar las *pruebas del sistema*.
    ¿Qué posibles complicaciones podrían aparecer en los sistemas empotrados?
    Solución:
    Para poder desarrollar la secuencia de pasos de prueba descritas anteriormente hay que verificar que los eventos, restricciones temporales, paralelismo entre tareas, etc. nos permitan seguir secuencialmente dichos pasos.

93. ¿Cómo puede la planificación de un proyecto afectar a la prueba de integración?
    Solución:
    una mala planificación de un proyecto software puede ocasionar que falten pruebas unitarias de componentes cuando se llegue a la fase de *pruebas de integración*.

94. ¿Son posibles o incluso deseables las pruebas unitarias en cualquier circunstancia? Proporcionar ejemplos para justificar la respuesta.
    Solución:
    No, las pruebas unitarias no son indicadas cuando los componentes a probar realizan más de una función, es decir, cuando estamos en el caso de componentes afectados por una *baja cohesión*.

95. ¿Quién debería realizar la prueba de validación: el desarrollador o el usuario del software? Justificar la respuesta.
    Solución: Normalmente estas pruebas suelen ser las denominadas *alpha, beta–testing*, en ese caso, debería llevarlas a cabo el usuario del software.

96. Discutir las diferencias entre las pruebas de un sistema crítico para el negocio, un sistema de seguridad crítica y un sistema cuya falla no afecta seriamente a las vidas, la salud o los negocios.

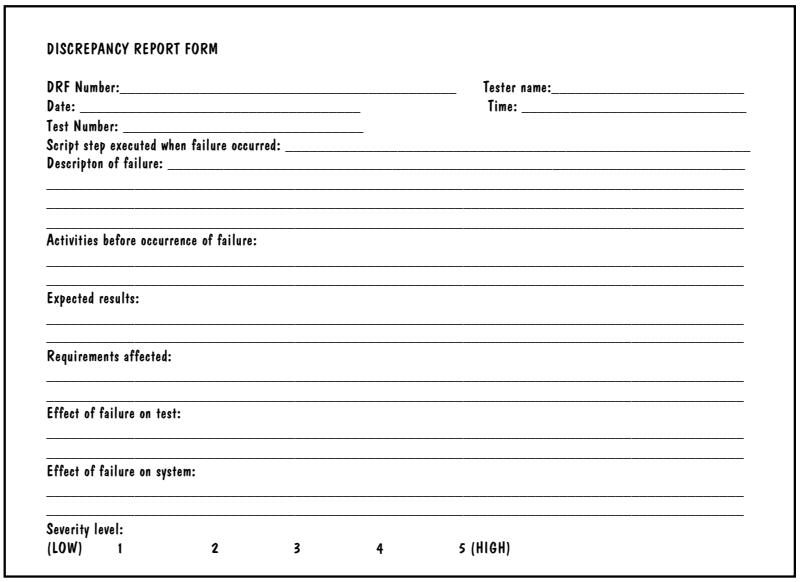
97. Dar un ejemplo de un sistema orientado a objetos donde los problemas de sincronización requieran una prueba cuidadosa.

98. Si un equipo de prueba independiente realiza una prueba de integración, y un defecto crítico permanece en el código después de completar la prueba ¿quién es legal y éticamente responsable por el daño que ha causado el defecto?

99. Suponer que se está construyendo un sistema para la preparación de impuestos que tiene tres componentes:
    (1) Crea formularios sobre pantalla que permiten que el usuario teclee su nombre, dirección, número de identificación impositiva e información financiera
    (2) Utiliza tablas de impuestos y la información ingresada por el contribuyente para calcular el monto a pagar para el año actual
    (3) Utiliza información del domicilio del contribuyente para imprimir formularios para el pago de las diferentes contribuciones (nacional,provincial,municipal), incluyendo el monto a pagar.

    Presentar la estrategia que podría utilizarse para probar este sistema y delinear los casos de prueba en un plan de pruebas.
100. Considerar el desarrollo de un ensamblador en dos pasos. Perfilar sus funciones y describir cómo se podría probar para que cada función se probara completamente antes de pasar a examinar la próxima función. Sugerir un plan de construcción para el desarrollo y explicar cómo se pueden diseñar juntos el plan de construcción y el de pruebas.
101. La certificación es una aprobación otorgada por una fuente externa que garantiza la exactitud de un sistema. Se concede comparando el sistema con un estándar predefinido de rendimiento. Por ejemplo, el Departamento de Defensa (DoD) Norteamericano certifica un compilador del lenguaje Ada después de probarlo contra una extensa lista de especificaciones funcionales. En la terminología introducida en este tema, tal prueba, de qué tipo sería:
    - Rendimiento
    - Aceptación
    - Instalación

    Justificar en cada uno de los casos anteriores.
102. Cuando se desarrolla un plan de construcción, es necesario tener en cuenta los recursos disponibles tanto para los desarrolladores y los clientes, incluidos tiempo, personal y dinero. Presentar ejemplos de restricciones de recursos que pueden afectar el número de construcciones de prueba (*builds*) definidos para el desarrollo del sistema. Explicar cómo estas restricciones afectan el plan de construcción.
103. Suponer que la calculadora de un matemático posee una función que calcula la pendiente y la intersección de una línea. El requerimiento en el documento de definición dice: "'La calculadora aceptará como entrada una ecuación de la forma Ax+By+C = 0 e imprimirá como salida la pendiente y la intersección". La implementación del sistema para este requerimiento es la función LINE(A,B,C), donde A y B son los coeficientes de x y de y. La constante de la ecuación es C. El resultado es una salida impresa de D y E, donde D es la pendiente y E la intersección. Escribir este requerimiento como un conjunto de causas–efectos y dibujar el gráfico correspondiente.
104. Los requerimientos han de ser comprobables, es decir, que se puedan *probar*. Explicar por qué la facilidad de prueba es esencial para la prueba de rendimiento. Utilizar ejemplos para justificar la explicación.
105. ¿Qué tipos de pruebas de rendimiento podrían requerirse para un sistema procesador de palabras?¿Un sistema de nómina de pagos?¿Un sistema de cajero bancario automatizado?¿Un sistema de supervisión de calidad del agua?¿Un sistema de control para una planta de energía?
106. Un sistema de control de tráfico aéreo puede diseñarse para que sirva a un único usuario o a muchos. Explicar de qué manera un sistema como éste puede tener una variedad de configuraciones y describir cómo se podría diseñar un conjunto de pruebas de configuración?
107. Un sistema de navegación está a punto de ser instalado en un avión. ¿Qué problemas deben ser considerados al diseñar la prueba de instalación?
108. Dar un ejemplo para demostrar que, sin la utilización de un simulador del dispositivo, la prueba a veces es imposible de realizar. Dar otro ejemplo para demostrar la necesidad de un simulador del sistema.



DISCREPANCY REPORT FORM

DRF Number:_____ Tester name:_____
Date: _____ Time: _____
Test Number: _____
Script step executed when failure occurred: _____
Descripton of failure: _____
_____
_____
_____
Activities before occurrence of failure:
_____
_____
Expected results:
_____
_____
Requirements affected:
_____
_____
Effect of failure on test:
_____
_____
Effect of failure on system:
_____
_____
Severity level:
(LOW) 1 2 3 4 5 (HIGH)

Figura 7: Formulario para informar de discrepancia

109. Hacer un comentario sobre el formulario de informe de discrepancia presentado en la figura 7, a la luz de las preguntas que es necesario contestar acerca de la falla a partir de la lectura del formulario.
110. Un sistema de salarios está diseñado de manera que hay un registro de información de empleado por cada persona que trabaja para la compañia. Una vez por semana, el registro del empleado se pone al día con el número de horas trabajadas por el empleado durante la semana. Cada dos semanas se imprimen los resúmenes, para desplegar el número de horas trabajadas desde el comienzo del año fiscal. Una vez al mes, la retribución mensual de cada empleado durante el mes se transfiere electrónicamente a su cuenta del banco. Para cada uno de los tipos de pruebas de rendimiento descritos en este capítulo, indicar si corresponde aplicarse a este sistema.
111. “Segarra Alpargatas" ha comisionado a un equipo de la ETSIIT de Granada para que desarrolle un sistema basado en computadora para probar la resistencia de su línea completa de calzado de goma. Segarra tiene nueve fábricas en varias ubicaciones alrededor del mundo y cada sistema se configurará según el tamaño de la fábrica. Explique por qué Segarra y el equipo de la ETSIIT deben dirigir la prueba de instalación aun cuando la prueba de aceptación se haya completado satisfactoriamente.
112. Escribir un guión de prueba para probar la función LINE descrita en el ejercicio 103.
113. Se ha propuesto una medida de la confiabilidad en términos del tiempo medio hasta la falla, de la disponibilidad en términos del tiempo medio entre fallas y de la facilidad de mantenimiento en términos de tiempo medio para reparar ¿Esta medidas son consistentes con las definiciones presentadas? Es decir, si se definen *confiabilidad*, *disponibilidad* y *facilidad* de mantenimiento como probabilidades, ¿se obtentrán los mismos números que si se usan las métricas? Si no es así, ¿un método puede transformarse en el otro, o existen diferencias básicas irreconciliables?
114. Un sistema de seguridad crítica falla y se pierden varias vidas. Cuando se investiga la causa de la falla, la comisión a cargo descubre que el plan de prueba desestimó la consideración del caso de prueba que ha causado la falla del sistema ¿Quién es el responsable de las muertes: los verificadores por no notar el caso omitido?¿Los planificadores de la prueba por no escribir un plan de prueba completo?¿Los gerentes por no haber verificado el plan de la prueba?¿El cliente por no haber hecho una prueba de aceptación completa?
115. Si los requerimientos de un sistema de confiabilidad muy elevada significan que la confiabilidad nunca puede verificarse, a pesar de eso, ¿el sistema debe usarse?

116. A veces, los clientes contratan una organización independiente (separada de la organización de desarrollo) para realizar una verificación y validación independiente (V&V). El personal de V&V examina todos los aspectos del desarrollo, incluso el proceso y producto para asegurar la calidad del producto final. Si se emplea un equipo de V&V independiente y el sistema todavía experimenta una falla catastrófica, ¿quién debe considerarse responsable?: ¿los gerentes, el equipo de V&V, los diseñadores, los codificadores o los verificadores?

117. Se definen dos funciones:

    (a) Función de distribución:$F(t) = \oint_{t_1}^{t_2} f(t)\,\mathrm{d}t$

    (b) Función de confiabilidad: $C(t) = 1 - F_i(t)$

    - Función de densidad de probabilidad: $f(t)$, describe la comprensión de cuándo es probable que el software falle
    - La función de confiabilidad $C(t)$ se define como la probabilidad de que el software funcione correctamente hasta el instante t.

    Si la confiabilidad de un sistema mejora cuando se lo prueba y arregla, ¿qué les sucede a los gráficos de dichas funciones?

118. Se describen dos versiones de software VxWorks, uno para un chip PowerPC y otro para un chip R6000. Explicar los problemas de gestión de la configuración relacionados con la construccón de un sistema para dos chips diferentes ¿Podría la estrategia de gestión de configuración haber ayudado a que el vendedor “portara" la versión del PowerPC al R6000?

    Había varios sistemas operativos disponibles para el microprocesador R6000 que llevaría el dispositivo de aterrizaje Pathfinder que la NASA colocó en el Sojourner para Marte. La NASA seleccionó VxWorks de WindRiver Systems. Cuando llegó, el VxWorks estaba probado y listo para PowerPC, pero la versión para el R6000 todavía no estaba lista. Wind River Systems convirtió la versión para PowerPC del sistema VxWorks al R6000, en lugar de utilizar un desarrollo nativo, sacando ventaja de la supuesta portabilidad del lenguaje C. Por lo tanto, el software del Pathfinder estaba construido con una versión de prueba *beta* de su sistema operativo, en lugar de estar montado sobre un sistema robusto y completamente probado.
    En Julio de 1997, debido a las fallas relacionadas con la administración de pilas y punteros durante los procesos de conmutación, el Pathfinder se reinicializó e interrumpió su trabajo durante algunos periodos de tiempo en su misión a Marte.

119. Un oráculo de prueba es una persona o máquina hipotética que puede decir cuándo los resultados de la prueba reales son iguales que los resultados esperados. Explicar la necesidad de inclusión de un oráculo de pruebas en el desarrollo de la teoría de la prueba.