

PROCEDIMIENTO PARA PRUEBA DE SOFTWARE

MARIA EUGENIA VALENCIA DE ABADIA

Ingeniero Electricista de la Universidad del Valle. Magister en Ingeniería Industrial y de Sistemas, Sistemas de la Universidad del Valle. Especialista en Diseño de Software de la Universidad Carolina del Sur U.S.A. Profesora del Departamento de Información y Sistemas de la Universidad del Valle.

INTRODUCCION

Este artículo presenta los procedimientos para realizar la prueba de aplicaciones o programas grandes de computador y define los términos asociados con la misma.

Una de las metas de la Ingeniería de Software es aumentar el nivel de corrección del software de computador. El propósito de la prueba es dar una medida de la corrección de un programa.

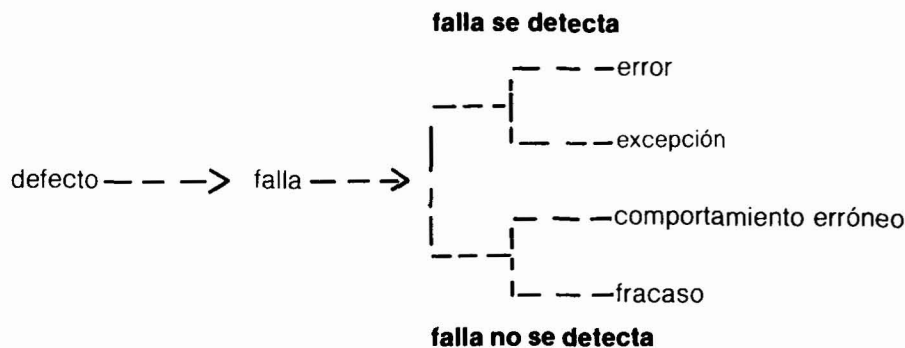
La prueba es parte integral del ciclo de diseño y por tanto debe chequearse la corrección del programa cuando éste se está desarrollando.

DEFINICIONES

Si la prueba se diseña para descubrir defectos, debemos definir el significado de "defecto" en el campo de la programación y lo que ocurre cuando un defecto se encuentra durante la ejecución de un programa.

Existe un **defecto** en un ambiente, algoritmo o dato, cuando esa entidad no reúne sus especificaciones.

Una **falla** puede resultar cuando persiste en mantener una entidad con un defecto. Cuando el sistema procesa una falla se da lugar a la aparición de comportamientos anormales que pueden identificarse como un **error**, una **excepción**, un **comportamiento erróneo** o un **fracaso**.



El esquema anterior muestra la relación entre estos comportamientos anormales y el hecho de detectar o no la falla.

- Cuando el sistema encuentra una falla, la reconoce, y la maneja de tal forma que el procesamiento normal puede continuar, se dice que ocurre un **error**.
- Se tiene una **excepción** cuando el sistema encuentra una falla, la reconoce pero no la puede manejar de forma que el procesamiento normal pueda continuar.
- Existe un **comportamiento erróneo** cuando el sistema no detecta la falla y ésta no causa una violación observable de sus especificaciones.
- Se tiene un **fracaso** cuando el sistema no detecta una falla y ésta causa una violación de las especificaciones.

Un intento de encontrar defectos ejecutando un programa en un ambiente de prueba (simulado) se denomina **verificación**.

Un intento de encontrar defectos ejecutando un programa en un ambiente real se denomina **validación**.

INFORMACION FUNDAMENTAL

Debido a la gran complejidad de los programas de computador, ha sido imposible hasta el momento presentar una técnica estandarizada y razonable que permita dar a las aplicaciones un buen nivel de corrección. Por tal razón es necesario utilizar una técnica "negativa" de prueba, la cual no asegura que el programa sea correcto sino

que su propósito es localizar sus defectos.

Así, si un programa pasa todas las pruebas, eso no implica que esté libre de defectos, sólo que no se localizó ninguno. Es entonces esencial el que se diseñen cuidadosamente las pruebas para que sean lo más completas posibles.

Condiciones

Los procedimientos de prueba que aquí se presentan asumen ciertas condiciones sobre los programas que van a probarse. Estas son:

- a. Se asume que el programa tiene una estructura jerárquica con módulos particionados funcionalmente y los módulos que no estén incluidos en ella deben aparecer claramente como primitivos.
- b. El programa debe diseñarse, implementarse y probarse en la forma TOP-DOWN. La implementación de cada módulo debe seguir las técnicas estructuradas en el uso de estructuras de control. Los módulos no deben exceder en longitud, una página de listado de codificación.
- c. El programa debe diseñarse de tal forma que los parámetros explícitos se usen para pasar información entre módulos donde quiera sea posible, las variables globales sean modificadas solamente por un módulo (típicamente un primitivo) y la estructura de los datos debe ser presentada con el mayor nivel de detalle posible.

Niveles de prueba

Los principales niveles son:

- **Prueba de Módulo (o unidad)**
Prueba la lógica del módulo como unidad solamente, sin importar su función dentro del programa.
- **Prueba de integración**
Prueba la estructura del programa y el algoritmo que el programa implementa.
- **Prueba funcional**
Efectúa la verificación de una función dada del programa.
- **Prueba del sistema y prueba de aceptación.**
Ambas involucran pruebas sobre el programa total. La primera intenta verificar que el programa cumple con las especificaciones y se hace con datos de prueba. La segunda se hace con datos reales y prueba el programa contra los requerimientos. Es prácticamente una validación.
- **Prueba de instalación**
Es la validación de una instalación particular del programa.
- **Prueba de vida**
Es la prueba de las necesidades que satisface el programa, en ese momento, contra las necesidades originales.

Tipos de pruebas

Existen dos tipos de pruebas: las estáticas y las dinámicas.

Las pruebas estáticas son muy específicas y similares a las de revisión de la codificación y pueden hacerse sin necesidad de ejecutar el programa.

Las pruebas dinámicas toman como base los resultados de la última ejecución y deben realizarse después de las pruebas estáticas.

Procedimientos de prueba

Debe seguirse la técnica TOP-DOWN, es decir que cuando cada nuevo módulo se agregue al programa, éste se ejecuta utilizando "etiquetas" en reemplazo de la verdadera codificación de cada uno de los demás módulos de los niveles inferiores. Las "etiquetas" pueden ser por ejemplo,

instrucciones de impresión con el nombre del módulo que la contiene. Es muy importante que el programa se ejecute con un solo módulo cada vez. Esto permite simplificar enormemente el proceso de detección de defectos pues ellos estarían necesariamente en el módulo nuevo que se está probando.

Procedimientos y técnicas generales

Los siguientes procedimientos y técnicas se aplican a todos los programas y pruebas:

- a. Para probar los programas debe usarse siempre datos de entrada bien definidos para los cuales se conozca de antemano los resultados correctos que deben obtenerse.
- b. Debe tratar de detectarse primero los defectos "obvios" y para ello se utilizan datos de prueba muy simples y luego sí, realizar las pruebas más complejas.
- c. Si el programa se modifica mientras se aprueba, CAMBIE UNA SOLA COSA CADA VEZ y utilice los mismos datos de prueba con que detectó el defecto. El defecto es corregido cuando al repetirse la prueba, el defecto ya no se detecta.
- d. Debe probar su programa para verificar si es capaz de detectar entradas incorrectas.

Procedimientos de pruebas estáticas

Los procedimientos de pruebas estáticas son importantísimos, pues permiten determinar múltiples defectos y ubicarlos al mismo tiempo con pruebas sencillas tanto en la parte de declaraciones como en las porciones ejecutables del programa.

Los procedimientos son:

- a. Chequee el alcance de las variables para ver si es factible reducirlo y evitar así posibles problemas. Esto facilitará la lectura del programa, pues la declaración de variables estaría lo más cerca posible del punto donde se usan.
- b. Chequee el uso de las variables globales y si una variable global puede ser referenciada por más de un módulo,

trátase en lo posible, que ella no sea modificada por más de uno.

- c. Téngase especial cuidado con aquellas variables que son usadas en varios niveles. Asegúrese que estén definidas en cada nivel para que no se originen problemas. Una situación típica es aquella donde existen ciclos a más de un nivel (ciclos anidados) y se usa la misma variable como control de ciclo. Es buena práctica usar nombres únicos para cada ciclo.
- d. Verifique si todas las variables tienen asignado algún valor inicial antes de ser referenciadas.
- e. Chequee si todos los argumentos de los subprogramas o procedimientos coinciden en número y tipo con los de las respectivas invocaciones.
- f. Verifique que todas las estructuras de repetición tengan un mecanismo de terminación bien definido.

Procedimientos de pruebas dinámicas

El concepto básico de estas pruebas está en empezar desde arriba trabajando hacia abajo y asumiendo que todos los módulos de los niveles inferiores trabajan correctamente. Así en cualquier momento se estará probando la codificación de UN SOLO módulo y las interfases con los otros. El punto de partida para las pruebas dinámicas es el programa principal con todos los primitivos conocidos. Todos los módulos que sean invocados en el programa principal se reemplazan con "la etiqueta" correspondiente para después ir reemplazándolos uno a uno. Después de que cada módulo se reemplaza y se prueba sin que se detecten defectos, se va añadiendo otro, preferiblemente en el orden en que son llamados por el programa y se repite el procedimiento para los niveles siguientes.

Como cada módulo en la jerarquía se prueba, se puede decir que el programa tiene un buen nivel de corrección, cuando ya no se presente ningún defecto. Debe tenerse en cuenta lo siguiente:

- a. Deben probarse todos los primitivos individualmente antes de usarse en el programa y una vez probados insertar-

los en él y proceder a probar el programa principal.

- b. Cuando ya estén integrados al programa todos los módulos del último nivel de la jerarquía, deben conservarse las etiquetas que saquen por pantalla algún mensaje indicador de que ese módulo ya se alcanzó y además, a los argumentos de salida de cada módulo debe asignárseles algún valor para verificar si trabajan bien.
- c. Deben hacerse las pruebas necesarias para chequear todas las posibles alternativas que se presenten en cada módulo antes de probar el siguiente.
- d. Trate de diseñar pruebas que en el caso de presentarse un fracaso generen información sobre la naturaleza y localización del defecto.
- e. En la búsqueda de defectos es recomendable usar "instrucciones de depuración" dentro del programa para imprimir valores de variables o simplemente para verificar que se ha llegado hasta cierto punto del programa y utilice por ejemplo alguna marca de comentario antes de cada una de ellas para hacer más fácil su localización.
- f. Y por último, si después de detectar un defecto usted considera que todo lo que ha estado haciendo está correcto y sin embargo su programa aún no trabaja, necesariamente UNA DE LAS COSAS QUE USTED CREE ESTA BIEN, FALLA EN ALGO!

BIBLIOGRAFIA

1. Davis-Hoffman, Fortran 77. Mc Graw Hill - 1984. Pág. 88-91.
2. Grogono, Programación en Pascal. Fondo Educativo Interamericano -1984. Pág 288-302.
3. I Sommerville, Software Engineering. International Computer Science Series - 1982. Pág 152 - 180.
4. Zelkowitz-Shaw-Gannon, Principles of Software Engineering and Design. Prentice Hall - 1977. Pág. 7-9, 27-30, 89-99.