

**UNIVERSIDAD PRIVADA DE TACNA**

**FACULTAD DE INGENIERÍA**

**Escuela Profesional de Ingeniería de Sistemas**

**INFORME**

**“Técnicas de Prueba de Diseño”**

Curso: Calidad y Pruebas de Software

Docente: Ing. Rafael Poma Laura

**Aguirre Ramos, Andrei (2014049278)**

**Catalán Vargas, Enzo (2014049170)**

**Chambilla Lanchipa, Luis Alfredo (2014049087)**

**Canchari Roque, Diego André (2014049086)**

**Tacna – Perú**

**2017**

### INDICE

[**INDICE**](#_jvjtd2pqov4d) **2**

[**Partición de equivalencia**](#_ckq58ao5jx67) **3**

[**Valores Límites**](#_ts0uctwrqb7w) **5**

[**Tablas De Decisión**](#_e7c0edcqbj9l) **5**

[**Cobertura de Caminos (Pruebas Ciclomáticas)**](#_5b337jrpvlxj) **8**

[**Web-Grafía**](#_g3zyk8yylcvd) **10**

### 

### 

### Partición de equivalencia

Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Una **partición equivalente** es una técnica de prueba de **Caja Negra** que divide el dominio de entrada de un programa en clases de datos de los que se pueden **derivar casos de prueba**. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

**Las clases de equivalencia** se pueden definir de acuerdo con las siguientes directrices:

Si un parámetro de entrada debe estar comprendido en un cierto rango, **aparecen 3 clases de equivalencia**:

* por debajo
* en
* por encima del rango

Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.

Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.

Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

Para aplicar esta técnica de prueba se tienen en cuenta los siguientes pasos:

Primero se deben identificar las clases de equivalencia lo cual se hace tomando

cada condición de entrada y aplicándole las directrices antes expuestas.

Para definir las clases de equivalencia hace falta tener en cuenta un conjunto de reglas: Si una condición de entrada especifica un rango, entonces se confeccionan una clase de equivalencia válida y 2 inválidas. Si una condición de entrada especifica la cantidad de valores, identificar una clase de equivalencia válida y dos inválidas. Si una condición de entrada especifica un conjunto de valores de entrada y existen razones para creer que el programa trata en forma diferente a cada uno de ellos, identificar una clase válida para cada uno de ellos y una clase inválida. Si una condición de entrada especifica una situación de tipo “debe ser”, identificar una clase válida y una inválida. Si existe una razón para creer que el programa no trata de forma idéntica ciertos elementos pertenecientes a una clase, dividirla en clases de equivalencia menores.

Luego de tener las clases válidas e inválidas definidas, se procede a definir los casos de pruebas, pero para ello antes se debe haber asignado un identificador único a cada clase de equivalencia.

Luego entonces se pueden definir los casos teniendo en cuenta lo siguiente:

Escribir un nuevo caso de cubra tantas clases de equivalencia válidas no cubiertas como sea posible hasta que todas las clases de equivalencia hayan sido cubiertas por casos de prueba.

Escribir un nuevo caso de prueba que cubra una y solo una clase de equivalencia inválida hasta que todas las clases de equivalencias inválidas hayan sido cubiertas por casos de pruebas.

Con la aplicación de esa técnica se obtiene un conjunto de pruebas que reduce el número de casos de pruebas y nos dicen algo sobre la presencia o ausencia de errores. A menudo se plantea que las pruebas a los software nunca terminan, simplemente se transfiere del desarrollador al cliente. Cada vez que el cliente usa el programa está llevando a cabo una prueba.

Aplicando el diseño de casos de pruebas al software en cuestión se puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que comiencen las “pruebas del cliente”.

### Valores Límites

Pruebas de valores de límite" es un método de caja negra que pone a prueba el límite si se trata de una entrada, una salida o un límite de rendimiento. Las pruebas se centran en los valores límites y no en toda la gama de datos. Se utiliza cuando tenemos un campo que puede contener una serie de valores como una entrada, una salida o como requisito.

Se basa en la evidencia experimental de que los errores suelen aparecer con mayor probabilidad en los extremos de los campos de entrada. Este método utiliza valores por encima y por debajo de los márgenes de la clase de equivalencia, y los analiza en contexto con el programa.

**Obtención de los casos de prueba**

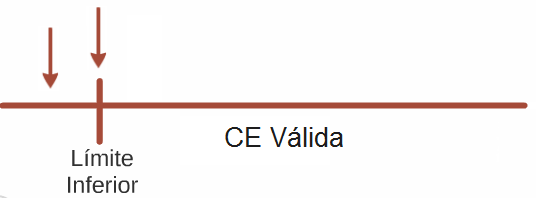
Generar tantos casos de prueba como sean necesarios para ejercitar las condiciones límites de las clases de equivalencia. La siguiente tabla muestra una guía de casos de prueba necesarios por cada tipo de condición que posea la variable a evaluar.

|  |  |
| --- | --- |
| **Condiciones de la especificación** | **Obtención de los casos de prueba** |
| 1. Rango de valores como condición de entrada | 1 caso que ejercite el valor máximo del rango |
| 1 caso que ejercite el valor mínimo del rango |
| 1 caso que ejercite el valor justo por encima del máximo del rango |
| 1 caso que ejercite el valor justo por debajo del mínimo del rango |
| 2. Valor numérico específico como condición de entrada | 1 caso que ejercite el valor numérico específico |
| 1 caso que ejercite el valor justo por encima del valor numérico específico |
| 1 caso que ejercite el valor justo por debajo de valor numérico específico |
| 3. Rango de valores como condición de salida | Generar casos de prueba según el criterio 1 que ejerciten dichas condiciones de salida |
| 4. Valor numérico específico como condición de salida | Generar casos de prueba según el criterio 2 que ejerciten dichas condiciones de salida |
| 5. Estructura de datos como condición de salida o de entrada | 1 caso que ejercite el primer elemento de la estructura |
| 1 caso que ejercite el último elemento de la estructura |

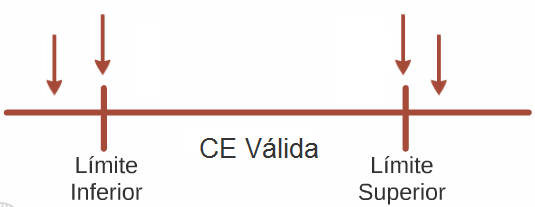
**Proceso para el análisis de valores límites**

Como una guía para el proceso de análisis se propone el siguiente proceso, que involucra la identificación y evaluación de una variable dentro del criterio 1 de las condiciones de especificación.

1. Identificar la variable.
2. Identificar la clase o clases válidas. Para cada clase válida identificar:
   1. Valor límite inferior, valor menor al límite inferior.



* 1. Valor límite superior, valor mayor al límite superior.



1. Crear un caso de prueba por cada valor
2. Identificar el representante de cada caso.
3. Evaluar el caso de prueba.
4. Emitir el resultado de la prueba.

### Tablas De Decisión

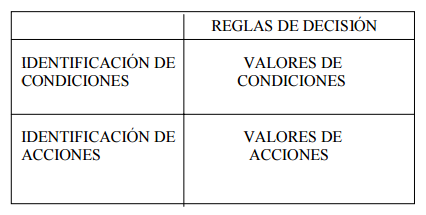
La técnica llamada Tablas de Decisión es muy aplicable cuando la lógica a probar está basada en decisiones, principalmente si se puede expresar la lógica en forma de reglas tales como:  
  
• Si A es mayor que X entonces…  
• Si el cliente C tiene deuda entonces…  
• etc.

O, dicho de otra forma, aplica a programas donde la lógica predominante es del tipo if-then-else. También tiene sentido aplicar cuando existen relaciones lógicas entre variables de entrada o ciertos cálculos involucrando subconjuntos de las variables de entrada. En estas situaciones se apunta a encontrar errores lógicos o de cálculo, principalmente en las combinaciones de distintas condiciones.

Esta técnica se utiliza en:

1. **La etapa de análisis de sistemas:** para efectuar una representación gráfica simplificada de los procesos lógicos que hayan sido relevados durante la investigación detallada, a efectos de analizar si se adecuan o no a los requerimientos del sistema.
2. **La etapa de diseño de sistemas:** para representar gráficamente procesos lógicos creados para satisfacer las necesidades del sistema bajo estudio.
3. **Aisladamente**, es decir, en tareas que no tengan que ver con el estudio de sistemas, para la representación simplificada de procedimientos específicos que sirvan de apoyo para una interpretación correcta del mismo y su posterior ejecución (procedimientos legales, impositivos, laborales, previsionales, aplicación de normas técnicas, etc.)

La tabla de decisión está compuesta por 4 secciones



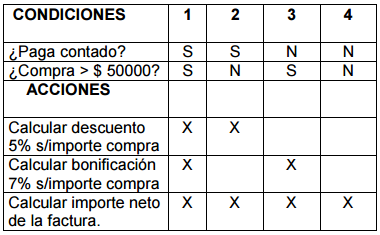
1. **Sección Identificación de condiciones:** se detalla una condición por renglón. Se llaman condiciones a situaciones variables que pueden ocurrir (p.ej: tipo de cliente, monto de ventas, antigüedad, etc.).
2. **Sección Identificación de acciones:** se describen todos los pasos que se deben realizar. Se llaman acciones a los distintos comportamientos que se asumirán en función de los valores que tomen las condiciones. Se escriben en el orden en que deben ser ejecutadas (p.ej: calcular descuento, calcular retención, pedir materiales, etc.).
3. **Sección Valores de condiciones:** se indican valores de las condiciones indicadas en la primer sección, dependiendo del tipo de tabla de decisión (de entrada limitada o extendida) que se construya para representar el Proceso.
4. **Sección Valores de acciones:** se indican valores de las acciones descritas en la segunda sección, dependiendo del tipo de tabla de decisión (de entrada limitada o extendida) que se confeccione.

Una vez confeccionada la tabla, quedarán determinadas las reglas de decisión. Estas son proposiciones que se leerán verticalmente, partiendo desde la sección Valores de Condiciones y descendiendo por la sección Valores de Acciones.

**Tablas de Entrada Limitada**

Sus principales características son:

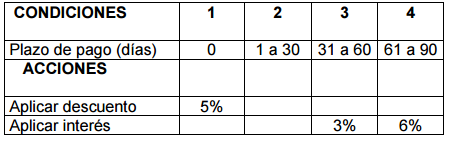
1. Las condiciones y las acciones figuran expresadas en forma completa.
2. Los valores asignados a las condiciones sólo pueden ser SI o NO. Por tal motivo las condiciones se las expresa en forma de preguntas, las cuales sólo podrán ser respondidas afirmativa o negativamente.
3. Los valores asignados a las acciones pueden ser: X (la acción debe ser ejecutada) o “blanco” (la acción no debe ser ejecutada).
4. La cantidad de reglas surge de calcular “2 a la n”, donde n es la cantidad de condiciones. Luego, estas reglas de decisión se depuran:
   1. Eliminando las reglas inconsistentes, es decir, combinaciones de condiciones que no se pueden dar en la realidad
   2. Eliminando las reglas redundantes. Ello se logra fusionando reglas a las cuales les corresponden las mismas acciones y los valores de las condiciones son todos iguales, excepto uno; a la regla que queda se le coloca una **I** (indiferente o indistinto) como valor de la única condición en que difieren.



**Tablas de Entrada Extendida**

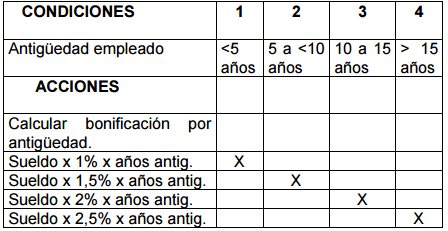
Sus principales características son:

1. Se utilizan cuando hay variables que pueden asumir más de dos valores. En este caso deberá considerarse que cada variable se desdobla en tantas condiciones como valores diferentes pueda asumir la misma. En consecuencia, en lugar de indicar SI o NO, van a escribirse todos los valores que pueda tener cada condición.
2. En las acciones en vez de colocar una X se describen las acciones específicas a ejecutar.
3. La cantidad de reglas surge de multiplicar la cantidad de valores que pueden tener las condiciones que se detallan. Es decir, si hay tres condiciones: una con tres valores, otra con dos valores y la restante con tres valores, se multiplica 3 x 2 x 3 lo cual arroja un total de 18 combinaciones o reglas posibles. Luego estas reglas se depuran eliminando las combinaciones inconsistentes y redundantes.



**Tablas de Entrada Mixta**

Se combinan características de las dos anteriores, considerando los valores de las condiciones en forma de entrada extendida e identificando las acciones en forma de entrada limitada, o viceversa



Pasos para elaborar una Tabla de Decisión:

* Listar todas las variables y las distintas condiciones a incluir en la tabla de decisión.
* Calcular la cantidad de combinaciones posibles, y con eso definimos cuántas columnas debemos designar.
* Agregar las acciones a la tabla: Las reglas de negocio indican qué acciones se disparan para cada combinación de datos de entrada, con lo cual para cada posible combinación deberemos analizar las reglas manualmente y determinar la salida esperada.
* Verificar la cobertura de las combinaciones, asegurando que con las combinaciones que vamos a seleccionar estaremos cubriendo el total del espacio de valores.

### 

### 

### 

### 

### 

### 

### 

### Cobertura de Caminos (Pruebas Ciclomáticas)

La aplicación de este criterio de cobertura asegura que los casos de prueba diseñados permiten que todas las sentencias del programa sean ejecutadas al menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Una de las técnicas empleadas para aplicar este criterio de cobertura es la Prueba del Camino Básico. Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa.

Los pasos a realizar para aplicar esta técnica son:

- Representar el programa en un grafo de flujo

- Calcular la complejidad ciclomática

- Determinar el conjunto básico de caminos independientes

- Derivar los casos de prueba que fuerzan la ejecución de cada camino.

Notación de Grafo de Flujo

Para aplicar la técnica del camino básico se debe introducir una sencilla notación para la representación del flujo de control, el cual puede representarse por un Grafo de Flujo. Cada nodo del grafo corresponde a una o más sentencias de código fuente. Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo. Para construir el grafo se debe tener en cuenta la notación para las instrucciones. Un Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son:

**Nodo**

Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

**Aristas**

Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

**Regiones**

Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran. La cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

Cualquier representación del diseño procedimental se puede traducir a un grafo de flujo. Cuando en un diseño se encuentran condiciones compuestas (uno o más operadores AND, NAND, NOR lógicos en una sentencia condicional), la generación del grafo de flujo se hace un poco más complicada.

**Complejidad Ciclomática**

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Un camino independiente es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. El camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente

### 

### 

### Web-Grafía

* Valores límite, obtención de casos de prueba <https://www2.infor.uva.es/~jvalvarez/docencia/tema7.pdf>
* Diseño de casos de prueba en ANGELFIRE.COM <http://www.angelfire.com/empire2/ivansanes/bywbox.htm>
* Prueba funcional o Prueba de Caja Negra en estingfuncional.wordpress.com <http://testingfuncional.wordpress.com/2011/03/12/%C2%BFtesting-funcional-o-pruebas-de-caja-negra/>
* Prueba general y técnicas de prueba de caja negra <http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf>