# Teorema fundamental de la prueba

Se entiende por prueba la ejecución de un programa con el fin de encontrar errores.

El propósito de una prueba es determinar si un programa contiene errores.

Probar un programa no es demostrar que funciona; si no asumir que el programa contiene

errores y tratar de encontrarlos.

Con la prueba buscamos aumentar la calidad. Buscamos encontrar y eliminar fallas.

Si la meta es demostrar que un programa contiene errores, los datos de prueba deben tener una probabilidad alta de encontrar errores. Por eso es importante saber qué pruebas son buenas (significativas) para conseguir los problemas.

Una prueba de software bien diseñada y ejecutada tiene éxito cuando detecta errores que entonces pueden ser corregidos.

Un caso de prueba que encuentra un error difícilmente puede considerarse no exitoso. Un caso de prueba no exitoso es el que hace que un programa produzca un resultado correcto sin encontrar errores.

Sin embargo, una prueba ideal es exitosa solo cuando un programa no tiene errores. Buscamos definir las características de una prueba ideal de forma que nos ilumine sobre los problemas de prueba. Para esto usamos el teorema fundamental de la prueba.

En este teorema tenemos que:

F —> Programa

D —> Conjunto de entrada de F (Dominio de Entrada). Todas las posibles entradas

d —> Elemento del conjunto D: d ∈ D. Es una entrada de F

F(d) —> Resultado de ejecutar F con d como entrada, donde d ∈ D

OUT(d, F(d)) —> Es el valor esperado. El requerimiento de salida de F

OK(d) —> Forma abreviada de OUT(d, F(d)). Valor esperado

T —> Es un caso de prueba ideal y es un subconjunto de D: T ⊆ D. A veces nos referimos a T como “conjunto de datos de prueba”

C —> Predicado sobre D que define el subconjunto T. C define cuales son las propiedades a probar de un programa. C debe ser confiable y válido

OUT(d, F(d)) = verdadero *sii* F se ejecuta y termina “limpiamente” para d y F(d) es un resultado aceptable

T ⊆ D que satisface C (es decir, C(T) es true)

∀ t ∈ T, OK(t) ⇒ OK(d), ∀ d ∈ D

Ejecutar un programa una vez con cada t ∈ T es la ejecución de la prueba T

El conjunto T se selecciona generalmente de acuerdo con un criterio de selección de datos C, donde C denota un predicado sobre subconjuntos de D

El criterio C especifica condiciones que T debe satisfacer y permite seleccionar un conjunto de datos de prueba

Ejemplos de criterios definiendo cuales son las propiedades a probar de un programa (o sea, ejemplos de C), serían:

* Todas las instrucciones se ejecutan al menos una vez
* Todos los predicados se evalúen en true y false
* Todo camino de control se ejecute al menos una vez
* “El dominio de entrada se particiona en clases de equivalencia y se selecciona un representativo de cada clase”

Un requerimiento clave del enfoque de prueba es definir criterios de selección de datos de prueba de manera tal que T ⊆ D, que satisface un criterio dado, produce resultados consistentes y significativos. Para esto, C debe ser confiable y válido

Un conjunto de datos de prueba T que satisface un criterio confiable C se denomina conjunto de datos de prueba confiable y un conjunto de datos de prueba T que satisface un criterio válido C se denomina conjunto de datos de prueba válido

Enfoque de requerimiento de prueba:

*Definir criterios para escoger de D las pruebas ideales (T) y que estos produzcan resultados consistentes y significativos, donde el criterio escogido (C) debe ser confiable y valido*

**Confiable**: C es confiable *sii* toda prueba que satisface el criterio tiene el mismo resultado (todos son exitosos o son no exitosos). C es confiable *sii* todo elemento de T, donde T satisface C, se ejecuta con éxito o todo elemento de T no se ejecuta con éxito. Es confiable si es consistente.

La definición formal es:

Eureka (T) = (∀t ∈ T) (OK(t))

Confiable (C) = (∀T1, T2, ⊆D)

(C(T1) ∧ C(T2)) ⇒(Eureka (T1) ≡ Eureka (T2))

**Validez**: C es valido *sii* los resultados de las pruebas siempre son significativas. Entendiendo significativo por que detecta el error, o sea, encuentra datos que revelen el error. Específicamente, C es válido *sii* para todo error en un programa particular existe un conjunto de datos de prueba T que satisface C y que revela el error. Este criterio indica que es posible seleccionar datos que revelen el error, pero no garantiza que sean seleccionados.

Nótese que la validez no implica que toda prueba T que satisface C fallará si el programa es incorrecto. Esto será así sólo si C es confiable y válido.

La definición formal es:

Válido (C) = (∀d ∈ D) (¬ OK(d)) ⇒ (∃T ⊆ D)(C(T) ∧ ¬Eureka(T))

**Teorema Fundamental**

Sea un programa F con dominio D, con requerimiento de salida OK(d) = OUT(d, F(d)) y un criterio de selección de datos de prueba C:

Eureka (T) = (∀t ∈ T) (OK(t))

Confiable (C) = (∀T1, T2, ⊆D) ((C(T1) ∧ C(T2)) ⇒(Eureka (T1) ≡ Eureka (T2)))

Válido (C) = (∀d ∈ D) (¬ OK(d)) ⇒ (∃T ⊆ D)(C(T) ∧ ¬Eureka(T))

Teorema Fundamental:

(∃C) (Válido(C) ∧ Confiable(C) ∧ (∃T ⊆ D) (Eureka(T) ∧ C(T))) => Eureka(D)

Demostración:

Considere un criterio de selección de datos de prueba C válido y confiable.

Asuma que existe algún d∈D para el cual F tiene una falla (es decir, ¬OK(d))

Entonces Válido(C) implica que existe un conjunto de datos de prueba T, tal que C(T) y ¬Eureka(T).

Confiable(C) implica que si una prueba tiene falla, todas fallan.

Pero esto contradice la premisa del teorema ((∃T ⊆ D) (Eureka(T) ∧ C(T))

(es decir, que existe una prueba que satisface C que se ejecuta exitosamente)

Observaciones generales:

* Puede tener criterios confiables, pero no validos, por que no me permiten revelar el error. Por ejemplo, si Eureka(T1) = verdadero y Eureka(T2) = verdadero, los resultados son iguales (es confiable), pero ningún resultado revela el error (no es válido)
* Puedo tener criterios no confiables (sus Eurekas son distintos), pero si válidos. Por ejemplo, si Eureka(T1) = verdadero y Eureka(T2) = falso, los resultados son distintos (no es confiable), pero revela el error con T2 (es válido)
* Puedo tener criterios confiables y validos. Si Eureka(T1) = falso y Eureka(T2) = falso ⇒ C es valido (revela el error) y confiable (los resultados son iguales)
* No puedo tener criterios no confiables y no validos. Para ser no confiable tiene que tener resultados distintos (verdadero y falso), y para ser no valido debe tener resultados verdaderos siempre (verdad y verdad). Por lo tanto, al ser no confiable siempre será valido por que al menos uno de los resultados revela el error.
* ¿En qué caso OK(T) ≠ OK(D)? Cuando T no tiene los datos inválidos, los que harían a OK(T) falso. Dado que T ⊆D, T2 ⊆D distinto a T
* Siempre habrá al menos un d para el cual OK(d) es true, esto se puede comprobar con:

Dada la aserción p ⇒ q

la negación es ¬q ⇒ ¬p

Como la validez es:

(∀d ∈ D) (¬ OK(d)) ⇒ (∃T ⊆ D) (C(T) ∧ ¬Eureka(T))

la negación sería:

¬((∃T ⊆ D) (C(T) ∧ ¬Eureka(T))) ⇒ ¬ ((∀d ∈ D)(¬OK(d)))

por lo tanto, esto quiere decir que existe al menos un d para el cual OK(d)

Ejemplo

[sacado del libro John Goodenough, Susan Gerhart: “Chapter 3: Toward a Theory of Testing: Data Selection Criteria”. In R.T. Yeh, editor, Current trends in programming methodology. Prentice Hall, 1979, Pág. 46 ]

**Problema**:

Sean:

F(d) = d\*d ∀d ∈D donde D es el conjunto de enteros;

OK(d) = (d + d = F(d));

Es decir, se requiere que F genere el doble del entero d, pero en lugar de esto, genera el cuadrado de d.

Considere los siguientes criterios de selección de datos de prueba:

C1(T) = (T = {0} ∨ T = {2})

C2(T) = (T = {t} ∧ t ∈ {0, 1, 2, 3, 4})

C3(T) = (T = {t} ∧ t ∈ {1, 3, 4})

C4(T)=(T={t1,t2,t3} ∧T ⊆ D)

Determine la confiabilidad y validez de cada uno de los criterios anteriores

**Solución**:

**PARA C1: T = {0} ∨ T = {2}**

***¿ C1 es confiable?***

Por definición:

Eureka (T) = (∀t ∈ T) (OK(t))

Confiable (C) = (∀T1, T2, ⊆D) ((C(T1) ∧ C(T2)) ⇒(Eureka (T1) ≡ Eureka (T2)))

Entonces para contestar si es confiable, tenemos que determinar el valor de Eureka (T).

Para eso sabemos que

Eureka (T) = OK(t)

OK(t) = OUT(d, F(d))

Llevamos esto a cada valor de T:

**si T = {0}** aquí Eureka(T) = OK(0)

F(d) es el resultado de ejecutar F con el dato de entrada d= 0 es decir:

F(0) = 0\*0 = 0 ⇒ F(0) = 0

Ahora veamos el valor de OK(d) = (d + d = F(d)) con d = 0:

OK(0)=OUT(0,F(0))=(0+0=F(0)) ≡ (0=0) ≡ true

como OK(t) = true por definición de Eureka obtenemos

Eureka(T) = true

Por otra parte,

**si T = {2}** el valor de Eureka(T) = OK(2)

F(d) es el resultado de ejecutar F con el dato de entrada d= 2 es decir:

F(2) = 2\*2 = 4 ⇒ F(2) = 4

Veamos el valor de OK(d) = (d + d = F(d)) con d = 2:

OK(2)=OUT(2,F(2))=(2+2=F(2)) ≡ (4=4) ≡ true

como OK(t) = true, por definición de Eureka obtenemos

Eureka(T) = true

Por lo tanto:

(Eureka({0}) ≡ Eureka({2})) = true

Ahora consigamos si C1 es confiable, aplicando la definición:

Confiable (C1) = (∀T1, T2, ⊆D) ((C(T1) ∧ C(T2)) ⇒(Eureka (T1) ≡ Eureka (T2)))

Confiable (C1) = (Eureka({0}) ≡ Eureka({2})) = true

con T1 = {0} y T2 ={2} **C1 es confiable**. Todo el conjunto T que satisface C1 produce resultados consistentes.

***¿C1 es válido?***

Por definición:

Válido (C) = (∀d ∈ D) (¬ OK(d)) ⇒ (∃T ⊆ D)(C(T) ∧¬Eureka(T))

Recordemos que:

(Eureka({0}) ≡ Eureka({2})) = true

Por lo tanto**, C1 no es válido**, porque no existe T tal que ¬ OK(d) y por lo tanto no se satisface ¬Eureka(T) para T= {0} o T ={2}