The Category-Partition Method for Specifying and Generating Functional Tests.

Thomas J. Ostrand and Marc J.Balcer

[CACM ,1988].

The goal of functional testing

- To find discrepancies between the **actual behavior** of the implemented system's function and the **desired behavior** as described in the system's functional specification.

How to achieve this goal ?

- Tests have to be execute for all the system functions.

- Tests have to be designed to maximize the chances of finding errors in the software.

Functional test can be derived <u>from 3 sources:</u>

1. The software specification.

2. Design information.

3. The code itself.

Partition - The standard approach

- The main idea is to **partition the input domain** of function being tested, and then **select test data for each class** of the partition.

- The problem of all the existing techniques is the **lack of systematic**.

The category partition method - <u>main characteristics:</u>

- **The test specification** :
  - is concise and uniform representation of the test information for a function.
  - it can be easily modified.
  - it gives the tester a logical way to control the volume of tests.

The category partition method - main characteristics (cont.):

- Using **generator tool** help us :
  - to provides an automated way to produce thorough tests.
  - to avoid impossible or undesirable tests.

- The method emphasizes both the specification **coverage** and the **error detection** aspects of testing.

A strategy for test case generation

- Il sistema è diviso in funzioni che possono essere testate indipendentemente
- Il metodo individua i *parametri* di ogni “funzione” e per ogni parametro individua *categorie* distinte
  - Oltre ai parametri, possono essere considerati anche *oggetti dell’ambiente*
  - Le categorie sono le principali proprietà o caratteristiche
- Le categorie sono ulteriormente suddivise in scelte allo stesso modo in cui si applica la partizione in classi di equivalenze (possibili valori)
  - normal values, boundary values, special values, error values

A strategy for test case generation

- Vengono individuati i vincoli che esistono tra le scelte, ossia in che modo l'occorrenza di una scelta può influenzare l'esistenza di un'altra scelta

- Vengono generati *Test frames* che consistono di combinazioni valide di scelte nelle categorie
- I Test frames sono quindi convertiti in *test data*

Vincoli

- *Poprietà e Selettori* associati alle scelte

Categoria A
    SceltaA1 [property X, Z]
    SceltaA2 [property Y, Z]
Categoria B
    SceltaB1
    SceltaB2 [if X and Z]

Annotazioni speciali: [Error], [Single]

Esempio: sorting program (parametri, categorie e scelte)

- Una funzione che prende in ingresso un array di lunghezza variabile di qualunque tipo e restituisce l’array ordinato (in accordo a qualche criterio) e i valori massimo e minimo
- *Categorie* per il parametro array: *dimensione dell’array*, *tipo degli elementi*, *massimo valore*, *minimo valore*, *posizioni dei valori massimo* e *minimo* nell’array
- *Scelte* per la dimensione dell’array: array di dimensione 0, di dimensione 1 e di dimensione da 2 a n (dove n è la dimensione massima) e un tentativo con n+1 (classe non valida)
- Scelte per posizione: il valore massimo in prima posizione, posizione centrale e ultima posizione dell’array
- Etc.

Example: sorting program

- While the categories are derived entirely from information in the specification, the **choices** can be based on the *specification*, the tester's past *experience* in selecting effective test cases, and *knowledge* of likely situations for errors to occur.
- To partition the category array size into ***choices***:

  size = 0, size = 1, 2 <= size <= 100, and size > 100.

- If the tester happens to know that memory for variable size arrays is allocated in blocks of size 256,

  include ***choices*** of size < 256, size = 256, and size > 256.

Esempio completo 1

Specifica

- Il programma chiede all'utente un intero positivo nell'intervallo 1-20 e quindi una stringa di caratteri di quella lunghezza. Il programma chiede all'utente un carattere e restituisce la prima posizione nella stringa in cui il carattere viene trovato o un messaggio che indica che il carattere non è presente nella stringa. L'utente ha l'opzione di cercare più caratteri.

Parametri e Categorie

- Tre parametri: l’intero x (lunghezza), la stringa a e il carattere c
- Categorie per x: “in-range” (1-20) o “out-of-range”
- Categorie per a: lunghezza minima, massima, intermedia
- Categorie per c: il carattere appare all’inizio, al centro o alla fine della stringa, oppure non appare nella stringa

Scelte

- Intero x, out-of-range: 0, 21
- Intero x, in-range: 1, 2-19, 20
- Stringa a: 1, 2-19, 20
- Carattere c: prima posizione, ultima posizione, posizione centrale, non in stringa
- Note: a volte è possibile che ci sia solo una scelta per una categoria

Specifiche Formali dei Test

x:	
1) 0	[error]
2) 1	[property stringok, length1]
3) 2-19	[property stringok, midlength]
4) 20	[property stringok, length20]
5) 21	[error]
a:	
1) Lunghezza 1	[if stringok and length1]
2) Lunghezza 2-19	[if stringok and midlength]
3) Lunghezza 20	[if stringok and length20]
c:	
1) Prima posizione	[if stringok]
2) Ultima posizione	[if stringok and not length1]
3) Posizione centrale	[if stringok and not length1]
4) Non in stringa	[if stringok]

Test Frames and Casi di test

x1	x = 0
x2a1c1	x = 1, a = ‘A’, c = ‘A’
x2a1c4	x = 1, a = ‘A’, c = ‘B’
x3a2c1	x = 7, a = ‘ABCDEFG’, c = ‘A’
x3a2c2	x = 7, a = ‘ABCDEFG’, c = ‘G’
x3a2c3	x = 7, a = ‘ABCDEFG’, c = ‘D’
x3a2c4	x = 7, a = ‘ABCDEFG’, c = ‘X’
x4a3c1	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘A’
x4a3c2	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘T’
x4a3c3	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘J’
x4a3c4	x = 20, a = ‘ABCDEFGHIJKLMNOPQRST’, c = ‘X’
x5	x = 21

Conclusioni

- L'individuazione di parametri, condizioni di ambiente e categorie dipende fortemente dall'esperienza del tester
- Rende le decisioni di testing esplicite (e.g., vincoli) e aperte a revisioni
- Una volta completati i primo passi, la tecnica è semplice e può essere automatizzata
- Riducendo i casi di test, la tecnica è utile e rende il testing sistematico più praticabile

Example 2

Command: find

Syntax: find <pattern> <file>

Function: The find command is used to locate one or more instance of a given pattern in a text file. All lines in the file that contain the pattern are written to standard output. A line containing the pattern is written only once, regardless of the number of times the pattern occurs in it.

The pattern is any sequence of characters whose length does not exceed the maximum length of a line in the file. To include a blank in the pattern, the entire pattern must be enclosed in quotes (“). To include quotation mark in the pattern, two quotes in a row (“ “) must be used.

Example:

find john myfile

display lines in the file **myfile** which contain **john**

find “john smith” in myfile

display lines in the file **myfile** which contain **john smith**

find “john”” smith” in myfile

display lines in the file **myfile** which contain **john” smith**

Test specification: categories and choices

Parameters:

Pattern size:

empty

single character

many character

longer than any line in the file

Quoting:

pattern is quoted

pattern is not quoted

pattern is improperly quoted

Embedded blanks:

no embedded blank

one embedded blank

several embedded blanks

Un-restricted test specification

Embedded quotes:

- no embedded quotes
- one embedded quotes
- several embedded quotes

File name:

- good file name
- no file with this name

Environments:

Number of occurrence of pattern in file:

- none
- exactly one
- more than one

Pattern occurrences on target line:

- one
- more than one

Total Tests frames:
1944

Test Frame - Example:

Pattern size : empty

Quoting : pattern is quoted

Embedded blanks : several embedded blanks

Embedded quotes : no embedded quote

File name : good file name

Number of occurrence of pattern in file : none

Pattern occurrence on target line : one

Property and selector expression

Parameters:

Pattern size:

empty	[property Empty]
single character	[property NonEmpty]
many character	[property NonEmpty]
longer than any line in the file	[property NonEmpty]

Quoting:

pattern is quoted	[property quoted]
pattern is not quoted	[if NonEmpty]
pattern is improperly quoted	[if NonEmpty]

Embedded blanks:

no embedded blank	[if NonEmpty]
one embedded blank	[if NonEmpty and Quoted]
several embedded blanks	[if NonEmpty and Quoted]

To eliminate contradictory frame: *constraints* are indicated with ***properties*** that can be assigned to certain choices, and tested for by other choices.

When the generator encounters a choice with ***a selector expression***, it omits that choice from any partial test frame that does not already have the properties specified in the selector expression.

Embedded quotes:

no embedded quotes	[if NonEmpty]
one embedded quotes	[if NonEmpty]
several embedded quotes	[if NonEmpty]

Restricted test specification

File name:

good file name

no file with this name

Total Tests frames: 678

Environments:

Number of occurrence of pattern in file:

none	[if NonEmpty]
exactly one	[if NonEmpty] [property Match]
more than one	[if NonEmpty] [property Match]

Pattern occurrences on target line:

one	[if Match]
more than one	[if Match]

- Many test frame are **redundant**; they test the same essential situation.
- They differ only in varying parameters that have no effect on the command's outcome.
- This occurs frequently when *some parameter or environment condition causes an error*.

- **[error]** tests are designed to test a particular feature which will cause an exception or other error state.
- A choice marked with **[error]** is not combined with choices in the other categories to create test frames.

Parameters:

Pattern size:

empty	
single characte	
many characte	
longer than any line in the file	[property NonEmpty]

Quoting:

pattern is quoted	[property quoted]
pattern is not quoted	[if NonEmpty]
pattern is improperly quoted	[if NonEmpty]

Embedded blanks:

no embedded blank	[if NonEmpty]
one embedded blank	[if NonEmpty and Quoted]
several embedded blanks	[if NonEmpty and Quoted]

Parameters:

Pattern size:

empty	[property Empty]
single character	[property NonEmpty]
many character	[property NonEmpty]
longer than any line in the file	[error]

Quoting:

pattern is quoted	[property quoted]
pattern is not quoted	[if NonEmpty]
pattern is improperly quoted	[error]

Embedded blanks:

no embedded blank	[if NonEmpty]
one embedded blank	[if NonEmpty and Quoted]
several embedded blanks	[if NonEmpty and Quoted]

Embedded quotes:

no embedded quotes	[if NonEmpty]
one embedded quotes	[if NonEmpty]
several embedded quotes	[if NonEmpty]

File name:

good file name	
no file with this name	[error]

Total Tests frames: 125

Environments:

Number of occurrence of pattern in file:

none	[if NonEmpty]
exactly one	[if NonEmpty] [property Match]
more than one	[if NonEmpty] [property Match]

Pattern occurrences on target line:

one	[if Match]
more than one	[if Match]

Embedded quotes:

no embedded quotes	[if NonEmpty]
one embedded quotes	[if NonEmpty]
several embedded quotes	[i

File name:

good file name	
no file with this name	[

- The annotation **[single]** is intended to describe special, unusual, or redundant conditions that do not have to be combined with all possible choices.
- As with [error] choices, the generator does not combine a **[single]** choice with any choices from other categories.

Environments:

Number of occurrence of pattern

none	[if NonEmpty]
exactly one	[if NonEmpty] [property Match]
more than one	[if NonEmpty] [property Match]

Pattern occurrences on target line:

one	[if Match]
more than one	[if Match]

Embedded quotes:

no embedded quotes	[if NonEmpty]	
one embedded quotes	[if NonEmpty]	
several embedded quotes	[if NonEmpty]	[single]

File name:

good file name	
no file with this name	[error]

Total Tests frames: 40

Environments:

Number of occurrence of pattern in file:

none	[if NonEmpty]	[single]
exactly one	[if NonEmpty]	[property Match]
more than one	[if NonEmpty]	[property Match]

Pattern occurrences on target line:

one	[if Match]	
more than one	[if Match]	[single]

Test Frame :

Test case 28 : (Key = 3.1.3.2.1.2.1.)

Pattern size : many character

Quoting : pattern is quoted

Embedded blanks : several embedded blanks

Embedded quotes : one embedded quote

File name : good file name

Number of occurrence of pattern in file : exactly none

Pattern occurrence on target line : one

Command to set up the test:

```
copy/testing/sources/case_28 testfile
```

find command to perform the test:

find “has” “one quote” testfile

Instruction for checking the test :

the following line should be display:

```
This line has “ one quote on it
```