3. Test Generation Strategies I

Based on requirements

Andrea Polini

Software Engineering II – Software Testing
MSc in Computer Science
University of Camerino

October 28th, 2014

Software Requirements

Requirements Specification

- informal
- semi-formal
- formal

Depending on the degree of formality more or less automated strategies can be applied

The test selection problem

Challenge

Construct a test set $\mathscr{T} \subseteq \mathscr{D}$ that will reveal as many errors in p as possible (where $\mathscr{D}$ is the input domain and $\mathscr{T}$ is the set of tests)

To give an idea…

Consider a procedure that has to manage data of an employee defined as follows:

- ID:`int` – three digit long from 001 to 999
- name:`string` – name is a 20 character long. Each characters belong to the set of 26 letters and space
- rate:`float` – rate varies from \$5 to \$10 per hour and in multiple of a quarter
- hoursWorked:`int` – hoursWorked varies from 0 to 60

Therefore:

$$999 \times 27^{20} \times 21 \times 61 \approx 5.42 \times 10^{34}$$

The test selection problem

Challenge

Construct a test set $\mathscr{T} \subseteq \mathscr{D}$ that will reveal as many errors in *p* as possible (where $\mathscr{D}$ is the input domain and $\mathscr{T}$ is the set of tests)

To give an idea...

Consider a procedure that has to manage data of an employee defined as follows:

- ID:`int` – three digit long from 001 to 999
- name:`string` – name is a 20 character long. Each characters belong to the set of 26 letters and space
- rate:`float` – rate varies from \$5 to \$10 per hour and in multiple of a quarter
- hoursWorked:`int` – hoursWorked varies from 0 to 60

Therefore:

$$999 \text{ x } 27^{20} \text{ x } 21 \text{ x } 61 \approx 5.42 \text{ x } 10^{34}$$

Equivalence partitioning

How to . . .

using the equivalnce partitioning strategy a tester should subdivide the input domain into “small numbers” of subdomains, which can be disjoint

Assumption

Equivalence classes are built assuming that the program under test exhibits the same behaviour on all elements of the same subset. One element for each subset is selected to form $\mathscr{T}$

Results?

Quality of $\mathscr{T}$ depends from experience, familiarity with requirements, access and familiarity with the code

Equivalence partitioning

How to . . .

using the equivalnce partitioning strategy a tester should subdivide the input domain into “small numbers” of subdomains, which can be disjoint

Assumption

Equivalence classes are built assuming that the program under test exhibits the same behaviour on all elements of the same subset. One element for each subset is selected to form $\mathscr{T}$

Results?

Quality of $\mathscr{T}$ depends from experience, familiarity with requirements, access and familiarity with the code

Equivalence partitioning

How to . . .

using the equivalnce partitioning strategy a tester should subdivide the input domain into “small numbers” of subdomains, which can be disjoint

Assumption

Equivalence classes are built assuming that the program under test exhibits the same behaviour on all elements of the same subset. One element for each subset is selected to form $\mathscr{T}$

Results?

Quality of $\mathscr{T}$ depends from experience, familiarity with requirements, access and familiarity with the code

Faults targeted

Simple partitioning:

- set of legal and not legal input
- requirements explicitely referring to different sets (Req1:$i \in [1, .., 60]$ and Req2:$i \in [60, .., 120]$)
- above and below

Formalizing the approach

Relations helping a tester in partitioning are of the form:

$$R : \mathscr{I} \to \mathscr{I}$$

where $\mathscr{I}$ represents the input domain.

The grocery (simple one)

Consider a method `getPrice` that takes the name of a grocery item consults a database of prices and return the unit price for the item. How would you partion the input?

$$pFound : \mathscr{I} \to \mathscr{I}$$

Elements in the database pF and elements non in the database pNF. They constitute a partion of $\mathscr{I}$

Formalizing the approach

Relations helping a tester in partitioning are of the form:

$$R : \mathscr{I} \to \mathscr{I}$$

where $\mathscr{I}$ represents the input domain.

The grocery (simple one)

Consider a method `getPrice` that takes the name of a grocery item consults a database of prices and return the unit price for the item. How would you partion the input?

$$pFound : \mathscr{I} \to \mathscr{I}$$

Elements in the database *pF* and elements non in the database *pNF*. They constitute a partion of $\mathscr{I}$

Examples

Printers

Consider an automatic printer testing application named `pTest`. The application takes the manufacturer name and the model of a printer as input and selects a test script from a list. The script is then executed to test the printer. Our goal is to test if the script selection part of the application is implemented correctly. Different types of printers available (B/W Inkjet, Color Inkjet, Color laserjet, Color multifunction).
How would you partion the input?

Words Count I

The `wordCount` method takes a word *w* and a file name *f* and returns the number of occurrences of *w* in the text contained in the file. An exception is raised if there is no file with name *f*.
How would you partion the input?

Examples

Printers

Consider an automatic printer testing application named `pTest`. The application takes the manufacturer name and the model of a printer as input and selects a test script from a list. The script is then executed to test the printer. Our goal is to test if the script selection part of the application is implemented correctly. Different types of printers available (B/W Inkjet, Color Inkjet, Color laserjet, Color multifunction).
How would you partion the input?

Words Count I

The `wordCount` method takes a word *w* and a file name *f* and returns the number of occurrences of *w* in the text contained in the file. An exception is raised if there is no file with name *f*.
How would you partion the input?

Examples

Words Count II

Now suppose to have access to the code of wordCount:

```
begin
  string w,f;
  input (w,f);
  if (^exists(f)) {raise exception; return(0)};
  if (length(w)==0) return (0);
  if (empty(f)) return (0);
  return (getCount(w,f));
end
```

How would you partion the input, now?

Combination of *w*: null/non-null, *f*: esists/does not exist, empty/non empty

In some cases the equivalence classes are based on the output generated by the program

Examples

Words Count II

Now suppose to have access to the code of wordCount:

```
begin
  string w,f;
  input (w,f);
  if (^exists(f)) {raise exception; return(0)};
  if (length(w)==0) return (0);
  if (empty(f)) return (0);
  return (getCount(w,f));
end
```

How would you partion the input, now?

Combination of *w*: null/non-null, *f*: esists/does not exist, empty/non empty

In some cases the equivalence classes are based on the output generated by the program

Examples

Words Count II

Now suppose to have access to the code of wordCount:

```
begin
  string w,f;
  input (w,f);
  if (^exists(f)) {raise exception; return(0)};
  if (length(w)==0) return (0);
  if (empty(f)) return (0);
  return (getCount(w,f));
end
```

How would you partion the input, now?
Combination of *w*: null/non-null, *f*: esists/does not exist, empty/non empty

In some cases the equivalence classes are based on the output generated by the program