ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

NGUYỄN THỊ TỰ

XÂY DỰNG CÔNG CỤ HỖ TRỢ SINH CA KIỂM THỬ CẶP

Ngành: Công nghệ thông tin
Chuyên ngành: Kỹ thuật phần mềm
Mã số: 60 48 01 03

LUẬN VĂN THẠC SĨ NGÀNH CÔNG NGHỆ THÔNG TIN

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS ĐẶNG ĐỨC HẠNH

Hà Nội – 2016

MỞ ĐẦU

Đặt vấn đề, định hướng nghiên cứu:

Trong những năm gần đây, chúng ta thấy rằng ngành công nghệ phần mềm phát triển ngày càng vượt bậc ở nhiều lĩnh vực. Đặc biệt tính ứng dụng cao bắt buộc cho phần mềm phải có một chất lượng nhất định. Việc phát triển phần mềm chỉ tập trung vào khâu thiết kế, lập trình là chưa đủ. Chúng ta cần tập chung cao vào cả khâu kiểm thử và đặc biệt hơn đó chính là kiểm thử chức năng (function). Nhưng kiểm thử như thế nào để có thể tiết kiệm chi phí, tối ưu nhất nguồn lực mà vẫn đảm bảo chất lượng.

Một giải pháp hợp lý cho các vấn đề đặt ra ở trên đó là áp dụng các kỹ thuật kiểm thử tối ưu và các công cụ kiểm thử tự động cho các phần mềm. Trong thực tế đã có rất nhiều công cụ kiểm thử tự động ví dụ như selenium IDE, QTP, nhưng nhìn trung lại chúng lại khá gò bó và mang nhiều nhược điểm.

Luận văn được thực hiện dựa trên ý tưởng từ nhu cầu thực tế và kiến thức được học. Cùng với đó là quá trình làm việc từ đó đưa ra cách thực hiện.

Luận văn được chia thành 3 chương, nội dung được phân bổ như sau:

Chương 1: Tổng quan về kiểm thử phần mềm.

Phần này nêu hệ thống cơ sở lý thuyết về kiểm thử như khái niệm cơ bản về kiểm thử, quy trình kiểm thử, các mức kiểm thử, các chiến lược kiểm thử và đặc biệt là các kỹ thuật trong kiểm thử chức năng.

Chương 2: Kỹ thuật kiểm thử cặp dữ liệu(Pairwise testing).

Phần này sẽ giới thiệu về kiểm thử cặp dữ liệu. Đây là một kỹ thuật trong kiểm thử chức năng. Trong đó luận văn sẽ nghiên cứu 2 kỹ thuật chính là mảng trực giao(OA) và thứ tự tham số(IPO). Ngoài ra phần này sẽ giới thiệu về công cụ sinh ra bộ dữ liệu kiểm thử theo phương pháp cặp dữ liệu là PICT.

Chương 3: Xây dựng công cụ sinh ca kiểm thử theo kỹ thuật cặp.

Phần này sẽ xây dựng một công cụ cho phép sinh ca kiểm thử dạng selenium IDE và kết hợp kỹ thuật cặp dữ liệu trong đó. Nó cho phép sinh một lúc nhiêu testcase

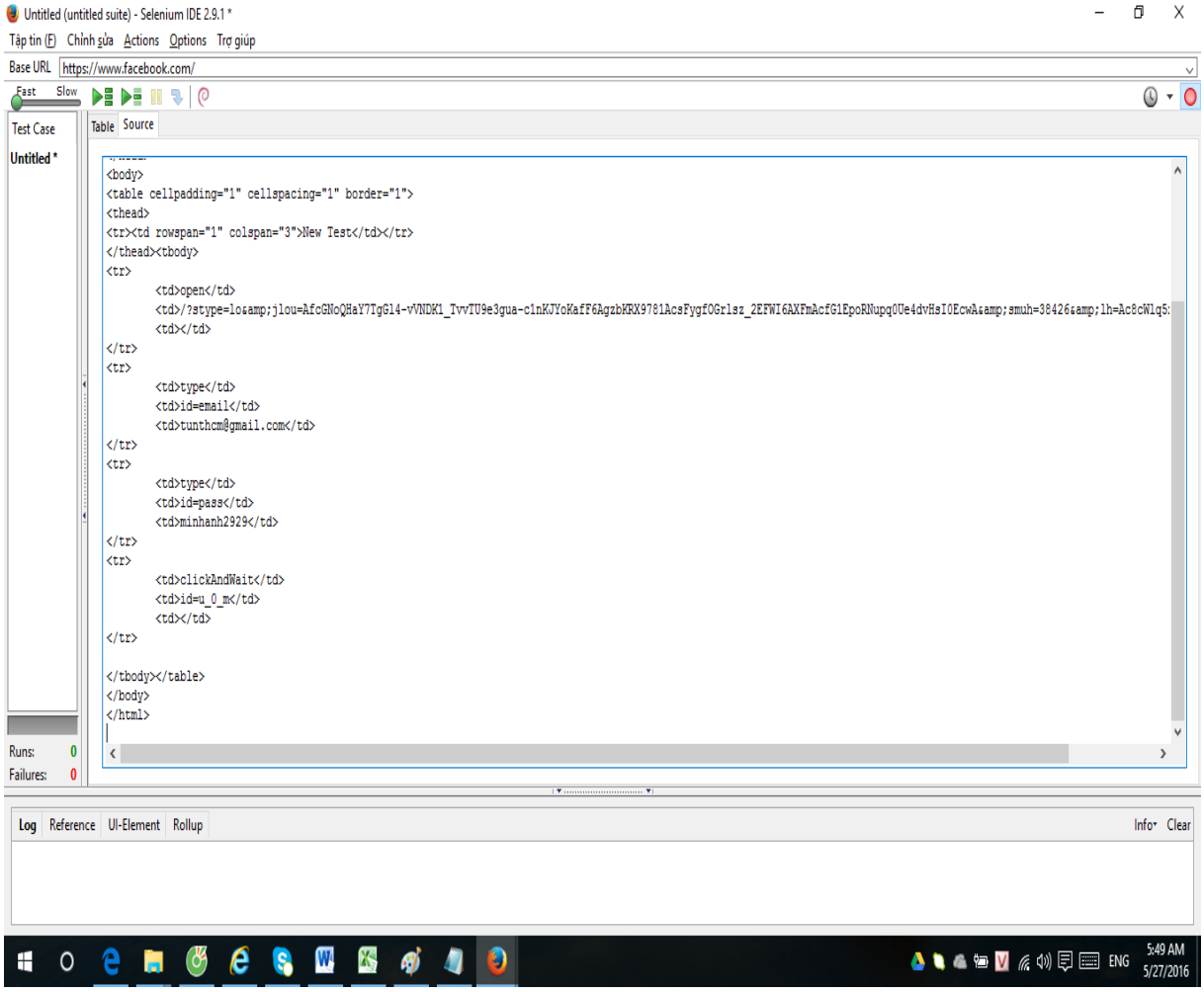
Chương 1: TỔNG QUAN VỀ KIỂM THỬ PHẦN MỀM

Tóm tắt chương 1: Trong chương này em trình bầy về một số vấn đề sau.

1.1 Khái niệm kiểm thử phần mềm

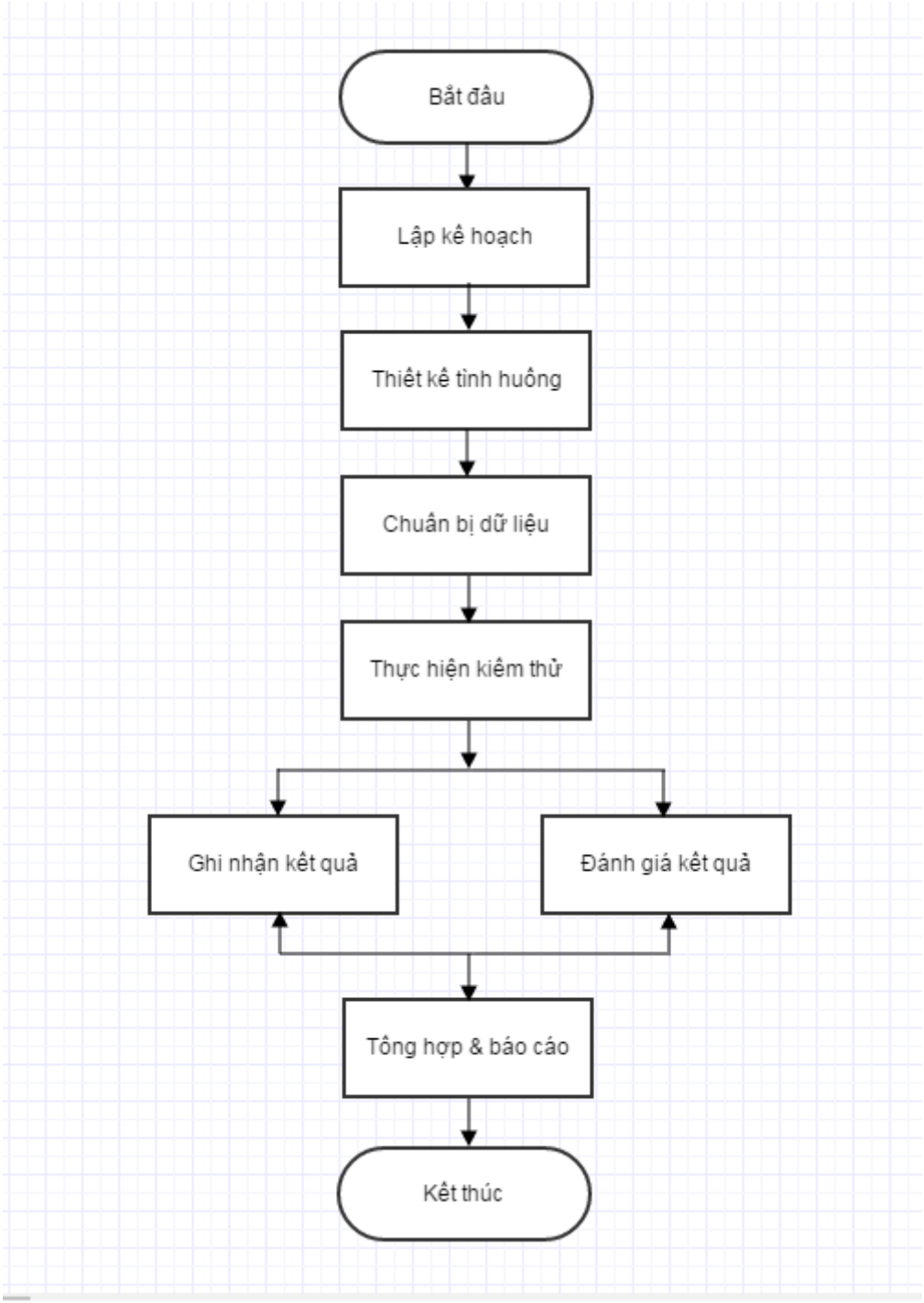
2.2 Một số thuật ngữ được sử dụng phổ biến trong kiểm thử như Bug, test case, Build, realease version.

Đặc biệt phần này em có nêu ra một mẫu ca kiểm thử mà công cụ của em sẽ phát triển ra. Đó là testcase selenium IDE.

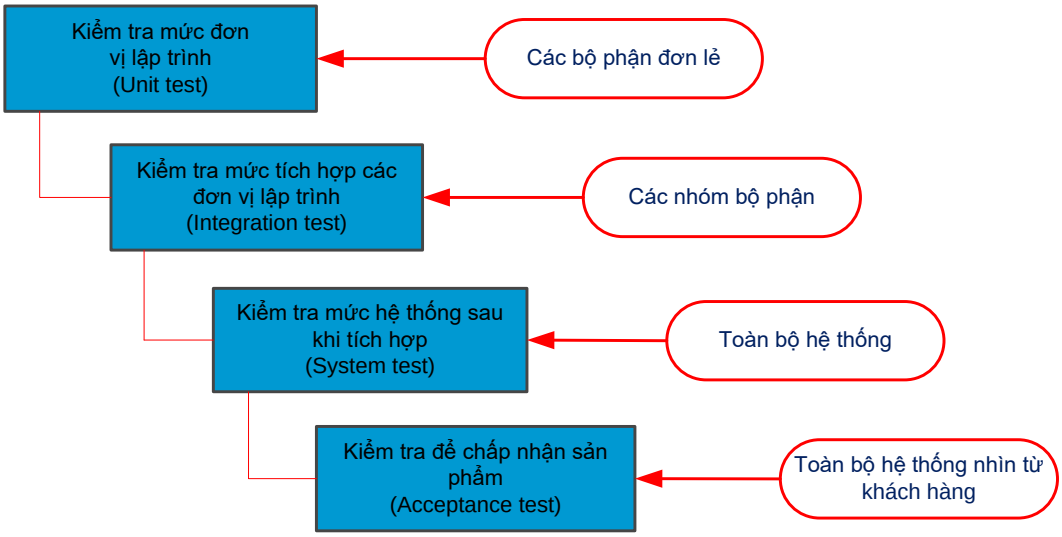


1.3 Trình bầy về quy trình kiểm thử phần mềm

Tại đây em sẽ trình bầy về quy trình kiểm thử phần mềm tại fpt software, nơi em làm việc



1.4 Trình bầy về các mức kiểm thử phần mềm.



1.5 Các chiến lược kiểm thử phần mềm.

+ Kiểm thử hộp trắng.

+ Kiểm thử hộp đen

1.6 Trình bầy về kiểm thử chức năng.

Tại đây có một số kỹ thuật em trình bầy như phân vùng tương đương(, phân tích giá trị biên, bảng quyết định, kiểm thử ngẫu nhiên, đoán lỗi, CPM

Đây là một trong những nội dung mà em muốn nhấn mạnh trong nội dung của mình. Những kỹ thuật này thường xuyên được em sử dụng trong thực tế tronng môi trường làm việc. Tuy đã được tìm hiểu và giới thiệu nhiều ở nhiều luận văn nhưng em xin tìm hiểu lại vì nó khá hữu ích.

a. Phân lớp tương đương (Equivalence class partioning)
b. Phân tích giá trị biên Boundary value analysis
c. Bảng quyết định Bảng quyết định (Decision tables
d. Kiểm thử ngẫu nhiên(Random testing)
e. Đoán lỗi (Error guesing)
f. Category partition (CPM

Chương 2: KIỂM THỬ CẶP DỮ LIỆU

2.1 Trình bầy về kiểm thử pairwise testing.

Pairwise testing là kỹ thuật kiểm thử thuộc phạm vi của kiểm thử chức năng.

Mục đích của nó là tạo ra bộ dữ liệu kiểm thử có kích thước nhỏ nhưng có thể cover được nhiều lỗi nhất có thể. Kỹ thuật này được biết đến gần 20 năm nay, nhưng nó chỉ phổ biến và gia tăng trong vòng 5 năm nay và hiện nay đã trở thành một kỹ thuật không thể thiếu trong kiểm thử phần mềm.

Trong chương này em sẽ tìm hiểu về kiểm thử cặp dữ liệu với 2 kỹ thuật cơ bản là mảng trực giao và IPO. [1]

Ngoài ra em sẽ trình bầy về bộ công cụ sinh ra bộ dữ liệu kiểm thử theo kỹ thuật pairwise đó là PICT[4]

2.3 Kiểm thử cặp dữ liệu (Parirwise testing)

Đầu tiên chúng ta hãy xem xét khái niệm kiểm thử kết hợp tất cả "all-combination testing". Nó được hiểu đơn giản là kiểm thử tất cả các kết hợp có thể có của các giá trị của một tập các biến.

Chúng ta xét n biến đầu vào là :

$V = \{v_1, v_2, v_3, \ldots v_{n-1}, v_n\}$

Với mỗi biến đầu vào ta chọn k giá trị quan tâm. Vậy theo như " all combination testing " ta phải xem xét k^n vectors kiểm thử. Như vậy thì số lượng test case sẽ rất lớn. Khi mà số lượng biến lớn và giá trị nhiều.

Thay vì như vậy chúng ta có thể xem xét và áp dụng kiểm thử cặp dữ liệu (pairwise testing). Pairwise được hiểu là tât cả các kết hợp đôi một (cặp) có thể có của các giá trị của tập biến đầu vào. Mỗi cặp giá trị đó sẽ được xuất hiện ít nhất một lần trong một trường hợp kiểm thử. Nó là một trường hợp đặc biệt của " all combination testing".

Nó thường được gọi là "all-pair/two-way testing"

Vídụ: Ta xét 3 biến X,Y,Z là 3 biến đầu vào của hệ thống S.

X = {true,false}

Y = {0;5}

Z={Q;R}

Theo " all combination testing » , tổng số trường hợp kiểm thử là = 2x2x2=8 vector kiểm thử :

Testcase ID	Input x	Input y	Input z
TC1	True	0	Q
TC2	True	0	R
TC3	True	5	Q
TC4	True	5	R

TC5	False	0	Q
TC6	False	0	R
TC7	False	5	Q
TC8	False	5	R

Nhưng với pairwise testing ta sẽ chỉ cố 4 vector:

Testcase ID	Input x	Input y	Input z
TC1	True	0	Q
TC2	True	5	R
TC3	False	0	Q
TC4	False	5	R

Tích kiệm các trường hợp kiểm thử như vậy, nhưng liệu chúng có hiệu quả trong việc cover lỗi không.

Kiểm thử được tất cả các kết hợp có thể có của giá trị của tập các biến đương nhiên sẽ tốt hơn. Hiệu quả được đưa ra theo thống kê được đưa ra tại [5] thì pairwise có thể phát hiện ra được 70% các lỗi. Còn four way testing thì có thể phát hiện ra 100% các lỗi. Và đương nhiên thì all combination cũng cover được 100% các lỗi.

Sau đây chúng ta sẽ đi nghiên cứu một số phương pháp kỹ thuật để phối hợp bộ dữ liệu kiểm thử.

2.3.1 Mảng trực giao (Orthogonal array ($L_{run}(Lever^{factors})$))

Phương pháp được nghiên cứu bởi nhà thống kê CR.Raoo va sau năm 1940 Genichi Tagumi là người đầu tiên sử dụng ý tưởng mảng trực giao trong những thiết kế thí nghiệm về quản lý chất lượng toàn diện(total quality management). Vì vậy mà phương pháp này được biết đến là phương pháp Tagumi, đã được sử dụng trong những kỹ thuật thiết kế thử nghiệm trong lĩnh vực sản xuất và cung cấp một cách có hiệu quả, hệ thống để tối ưu hóa thiết kế đảm bảo hiệu xuất, chất lượng và chi phí.

Phương pháp được sử dụng thành công tại nhật và mỹ, với mục tiêu là thiết kế có độ tin cậy cao, chất lượng sản phẩm cao với chi phí thấp trong ngành công nghiệp điện tử ô tô và tiêu dùng.

Mandl là ngươi đầu tiên sử dụng khái niệm của bảng trức giao trong việc thiết kế các testcase của pairwise.

Ưu điểm của phương pháp

- Đảm bảo sự kết hợp của tất cả các biến được lựa chọn.
- Tạo ra một bộ testcase hiệu quả và ngắn gọn
- Tạo ra một tập các testcase có sự phân bố (đồng đều)của tất cả các sự kết hợp trong kỹ thuật pairwise.
- Đơn giản để tạo ra và ít lỗi so với được tạo bằng tay(phương pháp khác).

Nhược điểm của phương pháp.

- Không phải cái gì cũng có thể sử dụng.

- Không phải tất cả đều có thể áp dụng kỹ thuật này. Ví dụ như kỹ thuật này chỉ áp dụng đối với các biến rời rạc.

Các bước của phương pháp mảng trực giao:

Bước 1: Xác định số lớn nhất(max) của biến độc lập của hệ thống. Số này sẽ được gán làm factors . 1 input variables → sẽ là 1 factor.

Bước 2: Xác định số lớn nhất giá trị của mỗi biến đầu vào. Số này được gán là Levels của mảng trực giao.

Bước 3: Tìm mảng trưc giao phù hợp với số run nhỏ nhất.

Ta có $L_{run}(x^y)$ trong đó x: Levers, y Factors;($L_{run}(Lever^{factors})$)

Trong bảng này ta sẽ có:

Runs: Số lượng của rows trong mảng, cũng chính là số test cases được tạo ra bởi phương pháp OA này.

Factors: Số Cột của mảng trực giao.

Levers: Số lớn nhất của values, được mang bởi một bất kỳ một factor đơn nào đó.

Bước 4: Ánh xạ mỗi biến vào 1 factors và mỗi giá trị vào 1 levers trên bảng

Bước 5: Check for any "left-over" levers in the aray that have not been mapped. Choose arbitrary valid value..

Bước 6: Chuyển đổi run thành testcase.

Sau đây là bảng giúp cho việc lựa chọn mảng trực giao phù hợp:

Orthogonal Array	Number of Runs	Maximum Number of Factors	Maximum Number of Columns at These Levels			
			2	3	4	5
L_4	4	3	3			
L_8	8	7	7			
L_9	9	4	—	4		
L_{12}	12	11	11			
L_{16}	16	15	15			
L'_{16}	16	5	—	—	5	
L_{18}	18	8	1	7		
L_{25}	25	6	—	—	—	6
L_{27}	27	13	—	13		
L_{32}	32	31	31			
L'_{32}	32	10	1	—	9	
L_{36}	36	23	11	12		
L'_{36}	36	16	3	13		
L_{50}	50	12	1	—	—	11
L_{54}	54	26	1	25		
L_{64}	64	63	63			
L'_{64}	64	21	—	—	21	
L_{81}	81	40	—	40		

Hình 2.1 Bảng lựa chọn mảng trực giao tùy theo số lượng lever và factors.

Một số ví dụ về chi tiết của mảng trực giao:

$L_4(2^3)$

	Factors		
Runs	1	2	3
1	1	1	1
2	1	2	2
3	2	1	2
4	2	2	1

Hình 2.2 Mảng trực giao **$L_4(2^3)$**

Hình 2.3 Mảng trực giao **$L_9(3^4)$.**

Một số mẫu bảng khác có thể tham khảo tại : :

http://www.york.ac.uk/depts/maths/tables/orthogonal.htm

http://www.freequality.org/documents/tools/Tagarray_files/tamatrix.htm

Ví dụ: Hãy xem xét một trang web, được xem trên một số trình duyệt và với một vài plugin và một số hệ điều hành, thông qua một số kết nối khác nhau như sau:

Browser	Netscape, IE, FF
Plug in	Real player, media player
Os	Window, linux,macintosh.
Connection	Lan,PPP, Isps

Sau khi chạy thuật toán ta sẽ có mảng trực giao sau đây:

	Factors			
Runs	**Browsers**	**Phug-in**	**OS**	**Connection**
1	Netscape	Realplayer	Window	Lan

2	Netscape	Readplayer	Linux	PPP
3	Netscape	Mediaplayer	Macintosh	Isdn
4	IE	Readplayer	Linux	Isdn
5	IE	Mediaplayer	Macintosh	Lan
6	IE	Mediaplayer	Window	PPP
7	FF	Readplayer	Macintosh	PPP
8	FF	Readplayer	Window	Isdn
9	FF	Mediaplayer	Linux	Lan

Bước 6: tạo ra 9 testcase từ mỗi run.

2.3.2 Thứ tự tham số (In parameter order)

Tai và Lei đã đưa ra một thuật toán được gọi là IP0, để tạo ra các testsuite cho pairwise testing của các biến đầu vào.

Thuật toán:

Input: Tham số $p_1, p_2, p_3, \ldots, p_i, \ldots, p_n$ với i =1,2,3..n;

và $D(p_i) = \{v_1, v_2, v_3, v_4, \ldots v_q\}$

Out put: Một bộ test suite T thỏa mãn coverage pairwise

Tóm tắt thuận toán:

```
Strategy In-Parameter-Order
begin
   /* for the first two parameters p1 and p2 */
   T := {(v1, v2) | v1 and v2 are values of p1 and p2, respectively}
   if n = 2 then stop;
   /* for the remaining parameters */
   for parameter pi, i = 3, 4, …, n do
   begin
      /* horizontal growth */
      for each test (v1, v2, …, vi-1) in T do
         replace it with (v1, v2, …, vi-1, vi), where vi is a value of pi
       /* vertical growth */
       while T does not cover all pairs between pi and
            each of p1, p2, …, pi-1 do
          add a new test for p1, p2, …, pi to T;
    end
end
```

Hình 2.4 Thuật toán IPO.

```
Algorithm IPO_H(T, p_i)
// T is a test set. But T is also treated as a list with elements in arbitrary order.
{ assume that the domain of p_i contains values v_1, v_2, ..., and v_q;
  π = { pairs between values of p_i and values of p_1, p_2, ..., and p_{i-1} };
  if (|T| ≤ q)
  { for 1 ≤ j ≤ |T|, extend the jth test in T by adding value v_j and
    remove from π pairs covered by the extended test;
  }
  else
  { for 1 ≤ j ≤ q, extend the jth test in T by adding value v_j and
    remove from π pairs covered by the extended test;
    for q < j ≤ |T|, extend the jth test in T by adding one value of p_i
    such that the resulting test covers the most number of pairs in π, and
    remove from π pairs covered by the extended test;
  }
}
```

Hình 2.5 Thuật toán Horizontal growth

```
Algorithm IPO_V(T, π)
{ let T' be an empty set;
  for each pair in π
  { assume that the pair contains value w of p_k, 1 ≤ k < i, and value u of p_i;
    if (T' contains a test with "−" as the value of p_k and u as the value of p_i)
      modify this test by replacing the "−" with w;
    else
      add a new test to T' that has w as the value of p_k, u as the value of p_i,
      and "−" as the value of every other parameter;
  };
  T = T ∪ T';
};
```

Hình 2.6 Thuật toán vertical

Các bước cụ thể của thuật toán:

Bước 1: Với 2 tham số đầu vào p_1 và p_2 tạo ra test suite

T={(v_1,v_2)|v_1 và v_2 theo thứ tự là những giá trị của của p_1 và p_2}

Bước 2: Nếu i =2, ngừng. Còn không với i = 3,4,,, n sẽ lặp lại bước 3 và bước 4

Bước 3: Cho $D(p_i)$ ={v_1,v_2,v_3,v_4,…v_q}

Tạo cặp π_i ={cặp giữa các giá trị của p_i và tất cả các giá trị của p_1,p_2 ,p_3 …p_{i-1}}

Nếu |T| <=q thì

Xét j chạy từ 1→|T|(1<=j<=|T|), mở rộng kiểm thử thứ j trong T bằng cách thêm vào giá trị v_j và di chuyển từ π_i cặp đôi đã phủ bởi kiểm thử đã mở rộng.

Còn nếu không (|T| > q)

- Xét với $j \in [1;q]$, mở rộng phần tử thứ j trong T bằng cách thêm thêm vào giá trị v_j và remove từ π_i cặp đã được cover bởi kiểm thử mở rộng này.

- Còn với $j \in (q; |T|)$ mở rộng kiểm thử thứ j trong T bằng cách thêm vào một giá trị(v_j) của p_i, giả sử rằng kết quả kiểm thử covers được hầu hết số lượng của cặp trong π_i, và remove từ π_i cặp đã được cover bởi phần tử mở rộng.

Bước 4:

Hãy gán cho T'= Φ (tập rỗng) và $|\pi_i| > 0$;

Với mỗi cặp trong π_i cặp chứa giá trị w của p_k, $1 \le k < i$, và những giá trị u của p_i) hãy làm:

- Nếu T' chứa một kiểm tra với – như là gí trị của P_k và u như là giá trị của p_i) , thay đổi kiểm thử này và thay thế - bằng w

- Còn nếu không

+ Thêm vào một kiểm thử mới trong T', cái có w như là giá trị của p_k, u như là giá trị của P_i, và – như là giá trị của tất cả các tham số khác.

- T:= T ∪ T'.

Minh họa thuật toán IPO:

Áp dụng với hệ thống S với 3 biến đầu vào ở trên là x,y,z và

D(x) = {True,False}

D(y) = {0;5}

D(z) = {Q;R}

Sau khi chạy thuật toán ta có kết quả như gồn 6 bộ dữ liệu kiểm thử như sau:

$$T = \begin{bmatrix} (\text{True}, & 0, & P) \\ (\text{True}, & 5, & Q) \\ (\text{False}, & 0, & R) \\ (\text{False}, & 5, & P) \\ (\text{False}, & 0, & Q) \\ (\text{True}, & 5, & R) \end{bmatrix}$$

2.4 Công cụ PICT.(Pairwise Independent Combinatorial Testing)

2.4.1 Nguyên tắc thiết kết của pict:

2.4.2 File đầu vào của pict:

Đầu vào cho pict là một tập tin đơn giản. Tập tin này có ít nhất 1 thành phần và nhiều nhất 3 thành phần như sau:

parameter definitions [sub-model definitions] [constraint definitions]

Tập tin luôn được định nghĩa theo thứ tự ở trên và không được trùng lặp. Viết chú thích bằng ký tự # ở đầu dòng. Nó có thành phần mở rộng là .txt
Ví dụ1 về nội dung file đầu vào của PICT:

```
TYPE:        Primary (10), Logical, Single, Span, Stripe, Mirror, RAID-5
SIZE:        10, 100, 500, 1000, 5000, 10000, 40000
FORMAT:      quick, slow
FSYSTEM:     FAT, FAT32, NTFS (10)
CLUSTER:     512, 1024, 2048, 4096, 8192, 16384, 32768, 65536
COMPRESSION: on, off
```

a.Cách viết phần: [parameter definitions]

Mỗi tham số và giá trị được viết trên một dòng, và các giá trị được phân cách bằng dấu phẩy [,]

```
<ParamName>:  <Value1>, <Value2>, <Value3>, ...
```

Dấu [:] ngăn cách giữa biến và giá trị.

b.Cách viết phần: [sub-model definitions]

```
{<ParamName1>, <ParamName2>, <ParamName3>, ... } @ <Order>
```

c. Cách viết phần: [Constraint definitions]

Constraint gồm có 2 loại là conditional (IF-THEN-ELSE) and unconditional.

Conditional Constraints:

Mối liên hệ giữa tham số và giá trị là một phần nguyên tử của một predicate. Các mối quan hệ sau đây được sử dụng như là: =,<>, >, <=,<,<=, và like. Like trong đó có *(khớp toàn bộ) và ?(một ký tự)

```
[Size] < 10000
[Compression] = "OFF"
[File system] like "FAT*"
```

Toán tử In được sử dụng để xác định một tập các giá trị mà thỏa mãn quan hệ xác định nào đó.

```
IF [Cluster size] in {512, 1024, 2048} THEN [Compression] = "Off";
IF [File system] in {"FAT", "FAT32"} THEN [Compression] = "Off";
```

IF, Then, ELSE có thể chứa nhiều term được joined bởi các toán tử OR, AND, NOT. Dấu (được sử dụng để đặt lại độ ưu tiên.

```
IF [File system] <> "NTFS" OR
 ( [File system] =  "NTFS" AND [Cluster size] > 4096 )
THEN [Compression] = "Off";
IF NOT ( [File system] = "NTFS" OR
      ( [File system] = "NTFS" AND NOT [Cluster size] <= 4096 ))
THEN [Compression] = "Off";
```

2.4.3 Cách thức sinh test case của Pict.

Quá trình xử lý trong pict được thể hiện qua hai giai đoạn chính: preparation và generation.

Trong giai đoạn 1, pict sẽ tính toán tất cả các thông tin cần thiết cho giai đoạn sau. Điều này bao gồm thiết lập tập P của tất cả các tham số tương tác để cover. Mỗi một sự kết hợp của giá trị có thể có của cặp dữ liệu của tập biến đầu vào được cover được phản ánh trong cấu trúc tương tác (in a parameter-interaction structure.)

Ví dụ, có 3 tham số A, B và C. A, B có 2 giá trị. Và c có 3 giá trị. Và pairwise sẽ tạo ra 3 cấu trúc tương tac tham số (parameter-interaction structures) là AB, AC và BC. Mỗi cấu trúc sẽ có một số slots, mỗi slots tương ứng với kết hợp giá trị có thể có. 4 Slots cho AB, 6 slót cho AC và BC.

```
                                    AB  AC  BC
                                    00  00  00
A: 0, 1                             01  01  01
B: 0, 1          translates to      10  02  02
C: 0, 1, 2                          11  10  10
                                        11  11
                                        12  12
```

Mỗi slot có thể được đánh dậu là uncovered, covered or là excluded(loại trừ). Tất cả các slot uncovered trong tất cả các tham số tương tác tạo thành tập của sự kết hợp được covered. Nếu bất kỳ ràng buộc được định nghĩa trong một mô hình chúng được chuyển đổi thành tập exclusions tập mà giá trị kết hợp ở đó phải không được xuất hiện trong output cuối cùng. Slot trở thành covered khi thuật toán sinh ra các test case thỏa mãn sự kết hợp riêng(đặc biệt). Thuật toán kết thúc khi không có một slots nào không được cover.

Trong giai đoạn 2, sinh test case sẽ sử dụng thuật toán sinh ca kiểm thử là greedy heuristic. Nó xây dựng một ca kiểm thử ở một thời điểm và với giải pháp tối ưu hóa.

2.4.4 Sự ưu việt của PICT

2.4.5 Cài đặt và chạy pict

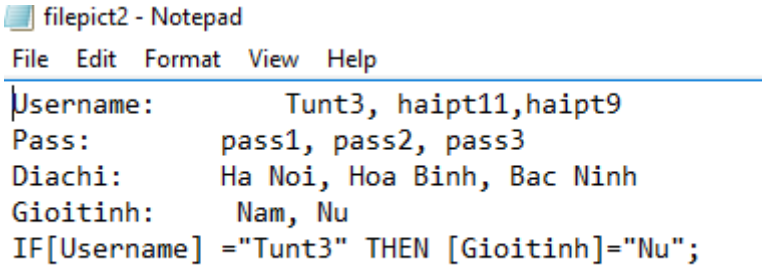
1. Download tịa link: http://download.microsoft.com/download/f/5/5/f55484df-8494-48fa-8dbd-8c6f76cc014b/pict33.msi

2.Thực hiện cài đặt

3.Từ Run ->CMD.

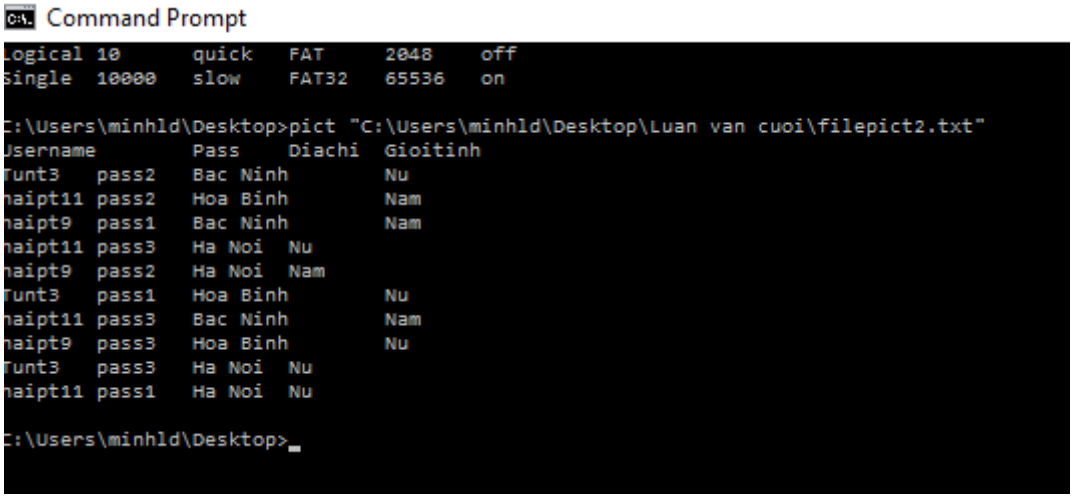
4.Trỏ đường dẫn đến thư mục cài đặt của PICT

Nội dung file đầu vào :

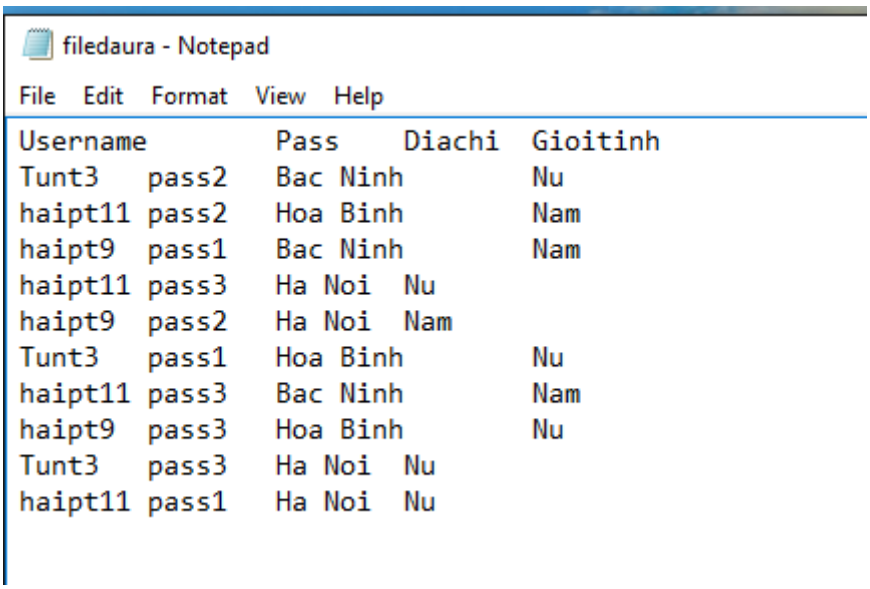


Kết quả hiển thị trên command

:



5. Pict vidu.txt > filedaura.txt



2.4.6 Ứng dụng của pict .

Pict được sử dụng trong 2 trường hợp sau.

a.Áp dụng viết itc khi viết bằng tay.

Với kỹ thuật mảng trực giao và IPO, chúng ta hoàn toàn có thể tạo ra bộ dữ liệu kiểm thử tốt. Tuy nhiên phải thực hiện chạy bằng tay. Giờ đây chúng ta ko phải nghĩ ngợi, tính toán gì nhiều nữa để đưa ra được các trường hợp kiểm thử. Chỉ việc áp dụng lấy các value và viết các ca kiểm thử. Tiết kiệm thời gian xuy nghĩ và tính toán.

b. Sử dụng làm data đầu vào cho selenium webdriver.

Ví dụ cụ thể như chúng ta hoàn toàn viết được testcase selenium drier để lấy dữ liệu từ file excell và chạy lần lượt các giá trị trong từng row đó.

Lấy dữ liệu từ file excell lên:

File excell có kết quả được xuất từ pairwise

STT	USER NAME	PASSWORD	Result ITC
1	tunthcm@gmail.com	minhanh2929	pass
2	luuminhvu5114@gmail.com	minhanh2929	pass
3	luuminhquyet20102015	minhanh2929	pass
4	..	..	..
5			
6			
7			
8			

Chạy selenium webdriver + apache poi

```
1.  import java.io.File;
2.  import java.io.FileInputStream;
3.  import java.io.IOException;
4.  import java.util.concurrent.TimeUnit;
5.  import org.apache.poi.xssf.usermodel.XSSFCell;
```

```
6.  import org.apache.poi.xssf.usermodel.XSSFRow;
7.  import org.apache.poi.xssf.usermodel.XSSFSheet;
8.  import org.apache.poi.xssf.usermodel.XSSFWorkbook;
9.  import org.openqa.selenium.By;
10. import org.openqa.selenium.WebDriver;
11. import org.openqa.selenium.firefox.FirefoxDriver;
12.
13.
14.    public class ReadExcellfile {
15.        private static WebDriver driver = null;
16.        public static XSSFWorkbook wordbook = null;
17.        public static String filepath=
"D:\\DRIVERSELENIUM\\Selenium_java poi_excell\\src\\testData\\Testdata.xlsx";
18.        public static String sheetname ="sheet1";
19.        public static void main(String[] args) throws IOException,
InterruptedException {
20.              //Tạo thể hiện của một lớp
21.        //    ReadExcellfile read = new ReadExcellfile();
22.              // Tạo một thể hiện của driver file fox.
23.         driver = new FirefoxDriver();
24.         driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
25.          //Khởi động  facebook
26.         driver.get("https://www.facebook.com/");
27.        // Doc file
28.         int rowcount=0;
29.         File file = new File(filepath);
30.              FileInputStream stream = new FileInputStream(file);
31.              wordbook = new XSSFWorkbook(stream);
32.              XSSFSheet sheet = wordbook.getSheet(sheetname);
33.         rowcount = sheet.getLastRowNum();
34.         for (int i =1; i<=rowcount; i++ )
35.         {
36.         XSSFRow  row = sheet.getRow(i);
37.         XSSFCell cell1 = row.getCell(1);
38.         XSSFCell cell2 = row.getCell(2);
39.         String datacell1 = cell1.getStringCellValue();
40.         String datacell2 = cell2.getStringCellValue();
41.       driver.findElement(By.id("email")).sendKeys(datacell1);
42.        // tìm phần tử có id"pass" và input value vào
43.          driver.findElement(By.id("pass")).sendKeys(datacell2);
44.          // supmit lên form.
45.
driver.findElement(By.xpath("//form[@id='login_form']/table/tbody/tr[2]/td[3]/labe
l/input")).click();
46.              //Đóng trình duyệt
47.         System.out.println("Dang den phan row thư "+i);
48.         driver.quit();
49.         driver = new FirefoxDriver();
50.         driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);
51.          //Khởi động  facebook
```

```
52.             driver.get("https://www.facebook.com/");
53.            //driver.findElement(By.id("userNavigationLabel")).click();
54.          //  Thread.sleep(5000);
55.            //
driver.findElement(By.xpath("//div[@id='u_q_1']/div/div/div/div/div/ul/li[12]/a/sp
an/span")).click();
56.             //driver.findElement(By.id("u_f_2")).click();
57.                  System.out.println("Ra day chưa");
58.
59.                    }
60.           }
61.    }
62.
```

Chương 3. XÂY DỰNG CÔNG CỤ SINH CA KIỂM THỬ TỰ ĐỘNG.

3.1 Ý tưởng của bài toán:

Hiện nay đã có một số công cụ hỗ trợ cho việc kiểm thử tự động ứng dụng Web như selenium ide, QTP. Tuy nhiên chúng mang nhiều nhược điểm.

Như Selenium ide chỉ có thể tạo ra các ca kiểm thử khi web đã hình thành. Và sau khi tạo bởi chính nó thì khi chạy lại nó cũng bị dừng ở nhiều chỗ. Nhưng đây là công cụ mà tôi thấy tâm đắc. Tôi mong muốn có thể ứng dụng được nó vào trong công việc thiết kế ca kiểm thử và thực hiện kiểm thử của mình.

Tuy nhiên, selenium ide chỉ có thể tạo ra test case khi trang web đó đã hình thành. Việc tạo ra testcase trước đó là không thể. Hơn nữa trong quá trình đưa ra kiểm thử thì nó cũng thay đổi liên tục. Và việc tạo ra các ca kiểm thử trên selenium ide mất khá nhiều thời gian. Vì vậy tôi mong muốn có thể tạo ra một công cụ có khả năng tự sinh ca kiểm thử dạng selenium ide và ngoài ra còn có được những đặc điểm sau :

+ Được sử dụng lại trong selenium ide.

+ Vẫn đảm bảo được khi webpage thay đổi.

+ Tạo trước khi webpage hình thành.

+ Kết hợp được kỹ thuật pairwise vào trong đó.

Trong luận văn này, tôi xin giới thiệu công cụ kiểm thử tự động được phát triển theo ý tưởng đã đặt đề ra.

3.2 Phân tích bài toán:

Để thực hiện ý tưởng trên, tôi đưa ra một số vấn đề sau:

• Xác định trang web và những thành phần trên trang cần kiểm tra tương tác.

+ Địa chỉ URL của trang cần kiểm thử.

+ Các thành phần trên trang cần kiểm thử như là: Textbox, combobox, button, link, radio button, verify text.

+ Các thành phần trên trang có thể tương tác qua thuộc tính nào chung. Tôi nhận thấy rằng các input hay những thành phần, sự kiện khác thường được định danh duy nhất bởi thuộc tính id.

• Xây dựng các thuật toán ra sao, sử dụng thuật toán nào để có thể sinh ra được nhiều ca kiểm thủ cùng một lúc. Làm sao tôi có thể ưng dụng được pairwise vào trong thuật toán này.

Và tôi thấy rằng IPO mà tôi đã nghiên cứu ở trên khá hay. Tôi đã quyết định sử dụng ý tưởng của nó để làm demo cho lần này.

Việc đưa một biến nhiều giá trị sẽ xử lý như thế nào.

• File selenium ide xuất ra có dạng như thế nào. Nhiều file hay ít file. Đưa ra một testscript cho tất cả hay đưa ra nhiều ca kiểm thử.

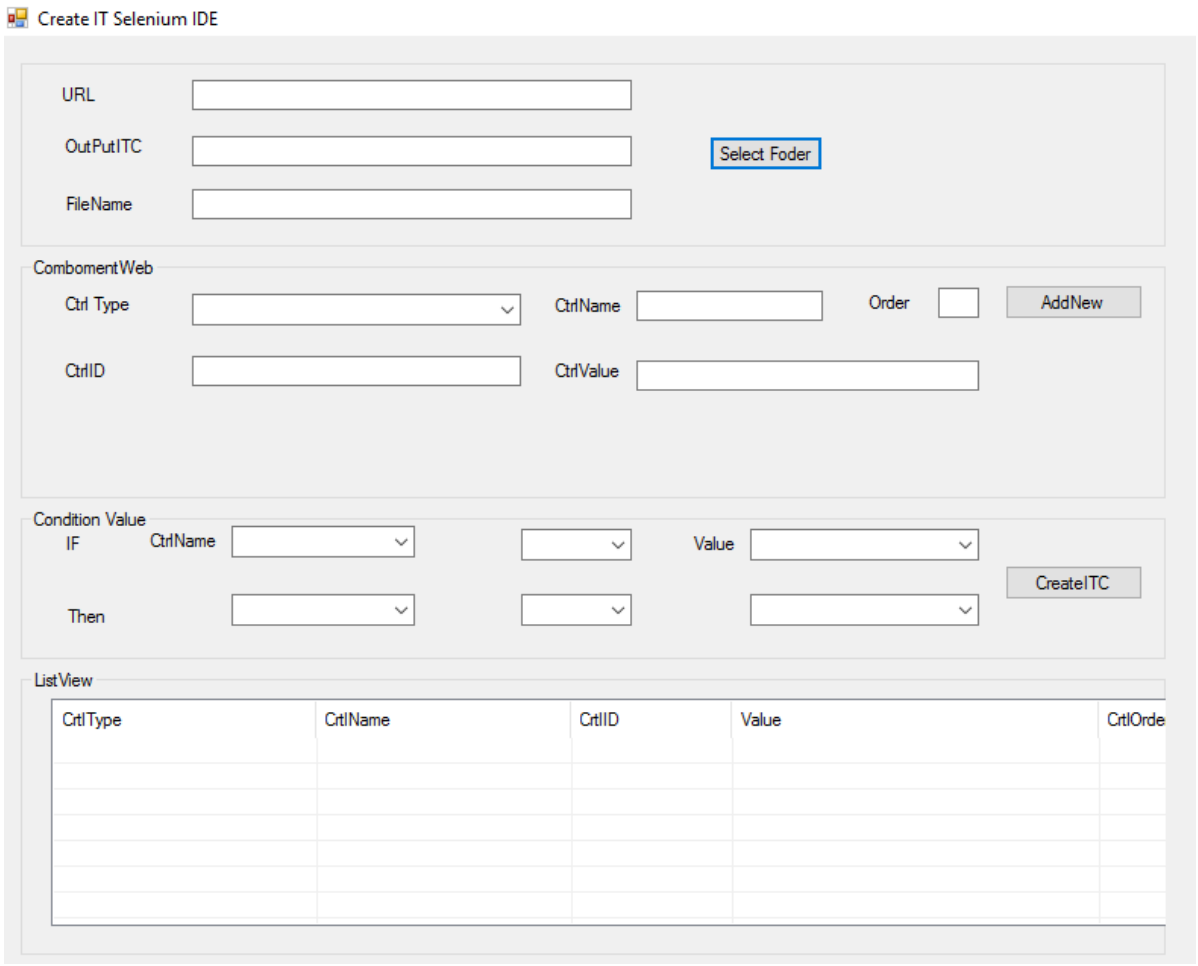
Tôi đã thấy rằng trong selenium ide thì mỗi một sự kiện sẽ dược định nghĩa là một dòng gồm có 3 column. Column thứ nhất luôn luôn thể hiện loại sự kiện input, Column thứ 2 chính là các id của sự kiện input đó. Và column thứ 3 chính là value của input đó.

Dòng nào ở trên sẽ được thực hiện trươc, dòng nào ở dưới sẽ thưc hiện sau.

Tôi cũng thấy rằng nên đưa ra nhiều ca kiểm thử thay vì 1 script.

3.3 Giải quyêt bài toán.

Từ những phân tích đó tôi tạo ra một form như sau:



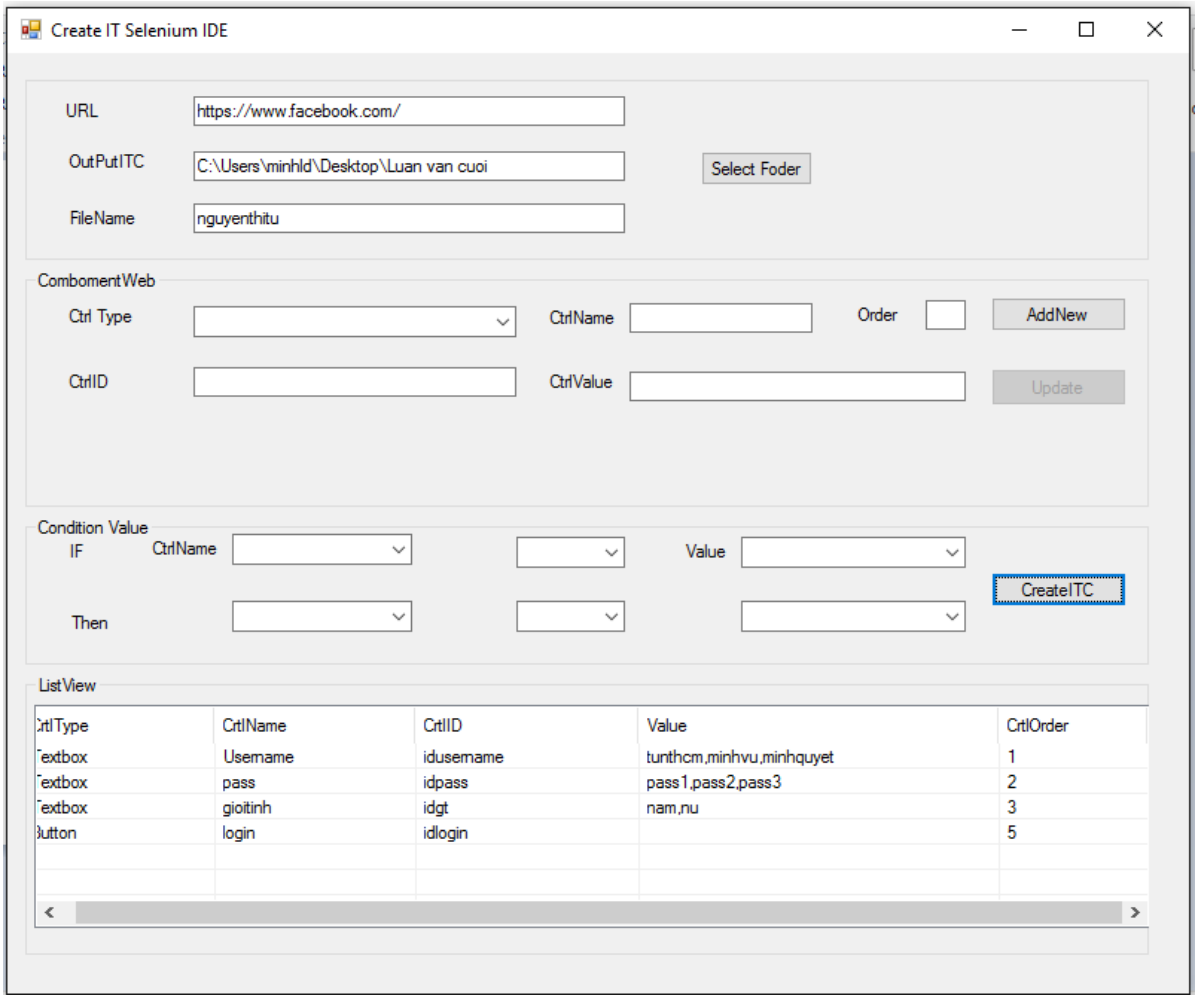
Hình 3.1 Form nhập liệu

STT	Tên item trên form	Ý nghĩa	Định dạng nhập
Information web			
1	URL	Nhập URL trang web	Item link web
2	Output ITC	Nhập đường link lưu	
3	Select path	Lựa chọn foder lưu test case	
4	File name	Nhập tên file testcase	Chữ và số tiếng anh
Phần comboment web			

5	Crtl Type	Loại control	Tiếng anh
6	Crtl Name	Tên control	Tiếng anh
7	Crtl ID	ID của control	Tiếng anh
8	Crtl Value	Value của control	Tiếng anh, ngăn cách dấu [,]
9	Crtl Order	Thứ tự của control	Số
10	Button Add new	Thêm một control vào listview	
Phần Contrains			
11	IF		
12	Crtl name	Tên của control	
13	Crtl Value	Value của control	
14	Condition	Conditon của control	
15	Create ITC	Tạo testcase selenium ide	
16	Then		
Phần List view			
Hiển thị danh sách các control sẽ tạo test case			Hiển thị

Bảng 3.1 Bảng mô tả các item trên form

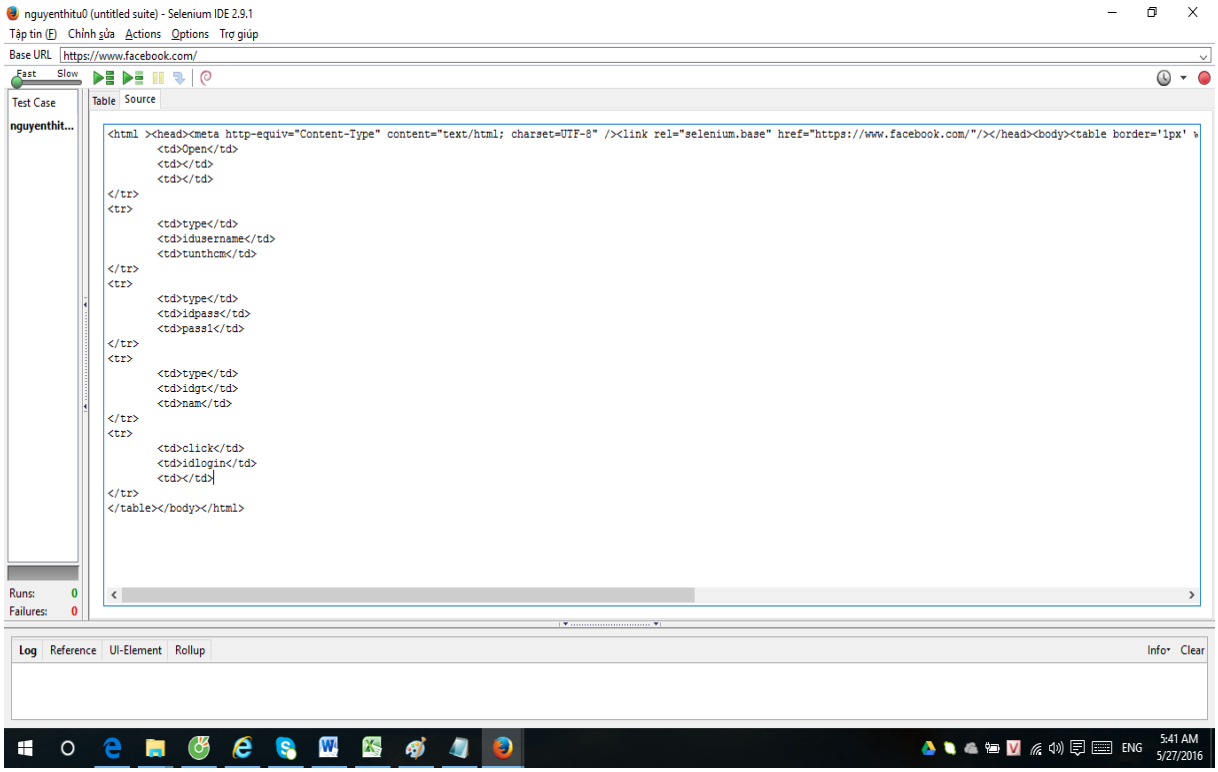
3.4 Kết quả của tool



Hình 3.2 Kết quả xuất ra trên list view

nguyenthitu0	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu1	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu2	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu3	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu4	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu5	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu6	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu7	5/17/2016 2:48 AM	HTML File	1 KB
nguyenthitu8	5/17/2016 2:48 AM	HTML File	1 KB

Hình 3. 3 Kết quả xuất file trên forder lựa chọn



Hình 3.4 Kết quả file được tạo khi mở bằng selenium ide

3.5 Ứng dụng công cụ vào thực tế:

Áp dụng vào trong trang facebook.com

3.6 Đánh giá ưu nhược điểm của công cụ

Ưu điểm:

Có thể tự sinh nhiều ca kiểm thử dựa trên điều kiện input.

Chương trình gọn nhẹ, dễ sử dụng.

Tiết kiệm thời gian, nhân lực và chi phí thực hiện.

Sử dụng lại trên selenium ide.

Tạo được ngay cả khi trang web chưa hình thành

Có thể ứng dụng cho nhiều trang web khác nhau.

Nhược điểm:

- Khi chạy nhiều trang vẫn bị những dừng do không có time out.
- Các thành phần còn chưa đầy đủ, thiếu verify text.
- Thiếu time out