한국통신학회 1986년도 추계학술발표회 논문집 (1986.11.22)

Decision Table을 이용한 Software설계 및 Test

최 영 철*, 박 용 진**
*육군사관학교 전산실 ** 한양대학교 전자과

Software Design and Test By Using Decision Table

Choe Young-Cheol, Park, Yong Jin

Korea Military Computing Center,
Dept. of ElectronicEng., Hanyang Univ.

Abstract

This paper presents a methodology for the design and test of Software by using Decision Table. We believe that this methodology can be used in improving Software Environment in the future.

1. 서론

Software 설계 및 Testing 이 두 과제는 Software Life Cycle중 적지않은 비중을 차지하고있다. Software 를 잘 설계한다는 것은 처리해야할 데이터의 속성을 알고 이것을 설계시에 적절히 이용할 필요가 종종 발생한다. 여기서 어떤 데이터를 생성해야하며 이 데이터를 설계시에 어떻게 이용하며 구성된 Software 를 어떻게 Testing 해야하는 문제가 중요한 과제이다.

본 논문에서는 Software의 설계와 Testing을 별개 보기보다는 하나의 과정으로 다룰수있는 방법론을 제시하고자한다. 2장에서는 Decision Table 의 간단한 소개를하고 3장에서는 설계방법의 개요를,4장에서는 구체적 예를 제시하고 5장에서는 검토및 결론을 다룬다.

2. Decision Table

사용자의 요구를 올바르게 파악하고 효율적인 Software 설계를 위해서 많은 사람들이 부단한 노력을 경주해 왔다. 따라서 이것을 지원하기위한 여러가지 Tool, 예를들면 Data Flow Diagram, Structured Chart, HIPO Diagram, Pseudo Code, Structured Flowchart, Structured English, Decision Table 등이 그것이다. 이중에서 특히 Decision Table은 복잡한 Logic 에 대한 내용을 formal하게 기술할 수 있고 Testing 할 수 있음으로해서 유용성을 상당하게 지니고 있다.더욱 사용자와 설계자간의 대화의 매개가 될수있는 공통언어가 필요한 이 시점에서 사용자와 설계자간에 서로 쉽게 이해할수 있는 tabular 언어로는 좋은 Tool이 될수있다. 한편 이미 작성한 Decision Table은Source Code로 바로 변환이 가능하므로 Software 설계시에 유용한 Tool로 이용할수가 있다. 또한 Test Case를 쉽게 파악이 가능하여 Test를 하면서 Software 설계를 할수있는 장점도 보유하고있다.

Decision Table은 다음과같은 구조를 지닌다.

Condition Stub	Condition Entry
Action Stub	Action Entry

그림 1. Decision Table 구조

Decision Table은 Condition Entry에 Y혹은 N(1혹은 0)만 허용하는 LEDT (Limited Entry Decision Table) 과 MEDT(Mixed Entry Decision Table)로 크게 구분할수 있는데 전자의 경우 Y,N을 1과 0로 보면 Boolean Algebra 적용하여 minimization할수있어 optimal code생성이 가능하므로 본 논문에서는 LEDT를 사용하기로 한다.

3. 설계방법의 개요

LEDT 를 Software 설계 및 Testing 에 이용하기 위하여 다음과같은 Step 을 거쳐야 한다.

Step 1.
Condition set 을 파악하여 Condition Stub를 구성한다

Step 2.
Condition set의 원소의 갯수로부터 Condition Entry를 구성한다.

Step 3.
Action set을 파악하여 Action Stub을 구성한다.

Step 4.
Test 데이터 set을 설정하므로서 Condintion Entry 에 대응되는 Action Entry를 확인한다.

Step 5.
Step 1 - 4 로부터 Decision Table 을 완성시킨다.

Step 6.
같은 Action 끼리 Grouping하므로서 Decision Table를 새로구성한다. 그리고 필요하면minimization을 수행한다.

Step 7.
Step 6 의 Decision Table 로부터 Decision Tree 를 구성한다.

Step 8.
Target Language로 변환한다.

4. Example

설명의 편의를 위하여 어떤 key의 집합을 sorting 되어있다 가정하고 중복key 와 not numeric key를 구분하여 중복 key 와 not numeric key 를 error file 에 그렇지않으면 good file에 write 하는 처리과정을 생각하자.

Step 1.

C_1를 임의의 Condition Expression 라할때

C_1 : key is numeric
C_2 : key is duplicate

C_2 에서 중복임을 알기위해서 먼저 하나의 key를 읽고 temp 에 옮긴후 다음 key 와 비교하여 같으면 중복이라 할수있으므로 C 를 다시

C_2 : temp = key

또 C_2 가 false 이더라도 다음 key를 읽기 전까지는 temp를 good-file 에 write 할수없다. 여기서 flag를 사용할 필요성을 느낀다. 따라서 C 는

C_3 : flag = 'on'

이상의 Condition set를 SC $\{C_1, C_2, C_3\}$ 라 하자.

Step 2.

$|SC| = 3$ 이므로 모든 가능한 경우의 수는 $2^3 = 8$이라 생각할수있다. 따라서 Condition Entry는 다음 그림 2 와 같이 구성된다.

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8
C_1 : key is numeric	Y	Y	Y	Y	N	N	N	N
C_2 : temp = key	Y	Y	N	N	Y	Y	N	N
C_3 : flag = 'on'	Y	N	Y	N	Y	N	Y	N

그림 2. Decision Entry 구성

Step 3.

다음으로 생각할것은 Action 을 파악하여 Action Stub 를 구성하는 일이다. 다음과 같은 Action 이 존재한다.

A_1 : write temp on error-file
A_2 : write temp on good-file
A_3 : write key on error-file
A_4 : flag = 'on'
A_5 : flag = 'off'
A_6 : temp = key
A_7 : exit

$SA = \{A_1\ A_2\ A_3\ A_4\ A_5\ A_6\ A_7\}$

Step 4.

여기의 경우 rule이 8 개가 되다는 것은 test 해야할 경우가 8 개 라고 볼수 있다. 따라서 Step 2의 결과를 고려하여 다음과 같은 data set 를 생각할수 있다.

	set 1	set 2	set 3	set 4	set 5
record 1	111	111	111	111	ab$
record 2	111	111	111	222	
record 3	111	222	222	333	
record 4	222	222	333	444	
record 5	eof	222	444	555	
		eof	eof	eof	

그림 3. data set 생성

data set 1는 중복 key가 세번 이상 반복되는 경우이고 data set 2는 서로 다른 중복 key가 존재하는 경우이고 data set 3는 중복key가 존재하고 그다음에는 존재하지 않는 경우이고 data set 4는 전혀 중복 key가 존재하지 않는 경우 이며 data set 5는 not numeric인 경우이다.

처음에 flag를 'off'로 initialize시키고 다음과 같이 Condition Entry 에 대응되는 Action의 Entry를 data set 1-4 을 참조로 기입할수있다. 이 일을 도와주기 위하여 다음과 같은 operation table과 여기에 따른 몇가지 notation을 정의하자.

notation:

(1) R : Reading Operation
(2) M : Moving Operation(Assign Statement)
(3) W(t,e) : write temp on error-file
(4) W(k,e) : write key on error-file
(5) W(t,g) : write temp on good file

그림 3에서 eof는 end of file 로서 그때의 Action 은 exit이다. 즉 exit하여 flag = 'on'이면 error-file 에 write를하고 아니면 good-file에 write를 한다.

그림 4 는 일련의 operation table로서 Action Entry에 기입할수있는 정보를 제공한다. 특히 숫자는 operation 순서를 나타내며 적용되어지는 rule 을 표기 하므로서 rule에 대한 test 도 겸하고있다.

data set 1

flag 'off'

key	R	M	W(t,e)	W(t,g)	W(k,e)	flag	rule
111	1	2	4			on	2
111	3	5	7			on	1
111	6	8	10			off	3
222	9	11					9
eof	12						

data set 2(3)

flag='off'

key	R	M	W(t,e)	W(t,g)	W(k,e)	flag	rule
111	1	2	4			on	2
111	3	5	7			off	3
222	6	8	10			on	2
222	9	11	13			on	1
222	12	14					9
eof							

data set 4

flag='off'

key	R	M	W(t,e)	W(t,g)	W(k,e)	flag	rule
111	1	2		4		off	4
222	3	5		7		off	4
333	6	8		10		off	4
444	9	11		13		off	4
555	12	14					9
eof							

data set 5

flag='off'

key	R	M	W(t,e)	W(t,g)	W(k,e)	flag	rule
ab$	1				2	off	5 8

그림 4. test 데이터를 사용한 operation

Step 5.

Decision Table을 완성시킨다.

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9
C_1 :key is numeric	Y	Y	Y	Y	N	N	N	N	–
C_2 :temp=key	Y	Y	N	N	Y	Y	N	N	–
C_3 :flag='on'	Y	N	Y	N	Y	N	Y	N	
C_4 :eof	N	N	N	N	N	N	N	N	Y
A_1 :write temp on error-file	X	X	X						
A_2 :write temp on good-file				X					
A_3 :write key on error-file					X	X	X	X	
A_4 :flag='on'	X	X		X					
A_5 :flag='off'					X	X	X	X	
A_6 :temp=key	X	X	X	X					
A_7 :exit									X

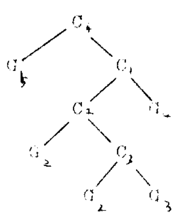
Step 6.

같은 Action끼리 grouping하여 Decision Table을 재구성한다. minimization 과정은 여기서는 간단하므로 생략한다.

C_1 :key is numeric	Y	Y	Y	N	–
C_2 :temp=key	Y	N	N		
C_3 :flag='on'	–	Y	N	–	–
C_4 :eof	N	N	N	N	Y
G_1 :$A_1 A_4 A_6$	X				
G_2 :$A_3 A_5 A_6$		X			
G_3 :$A_2 A_4 A_6$			X		
G_4 :$A_3 A_5$				Y	
G_5 :A_7					X

Step 7.

rule을 cover하는 순서대로 Condition을 선택하여 Decision Tree를 구성한다.



Step 8.

이상을 Implementation 하면 다음과 같은 Code가 생성된다.

```
READ-RTN.
      READ FILE AT END GO TO FIN-RTN.
      IF KEY IS NUMERIC
      THEN IF KEY = TEMP
            THEN PERFORM G1
            ELSE IF FLAG = 'ON'
                  THEN PERFORM G2
                  ELSE PERFORM G3
      ELSE PERFORM G4
      GO TO READ-RTN.
```

5. 검토 및 결론

이상으로 decision table을 이용하여 Software 설계 및 test 에관한 일련의 과정을 고찰하였다. Software 설계 및 test는 별개의 과정으로 보기 보다는 하나의 과정으로 간주해야 보다 효율적인 설계가 가능함을 Decision Table의 특성을 이용하여 나타내고 이것을 발전시켜 효율적인Software Environment를 구현할수있는 방법론을 제시하였다.

References

1. Art Lew." Optimal Conversion of Extented Entry Decision Tables with General Cost Criteria" CACM 21, 4(April 1978) 269-279
2. Chris Gane and Trish Saron. Structured Systems Anaiysis: Tools and Techniques (1979)
3. Reinwald, L.T. and Soland, R.M. "Conversion of Limited entry Decision Tables to Optimal Computer Program I:Minimum Average Requirement" JACM 13(July 1966) 339-358
4. Udo W. Pooch."Translation of Decision Tables "Computing Surveys Vol 6, No. 2 (June 1974) 125-151
5. 최영진 "정보처리에 관한 기법의 연구" (1980) 석사학위논문