Chương 6

Kỹ thuật kiểm thử hộp đen (tt)

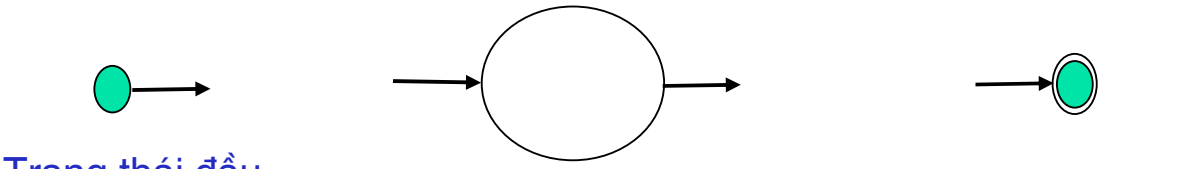
6.1 Kỹ thuật dùng lược đồ chuyển trạng thái

Cũng giống như bảng quyết định, lược đồ chuyển trạng thái là 1 công cụ rất hữu ích để đặc tả các yêu cầu phần mềm hoặc để đặc tả bảng thiết kế hệ thống phần mềm.

Thay vì miêu tả các qui tắc nghiệp vụ phức tạp mà phần mềm phải thực hiện dưới dạng dễ đọc và dễ kiểm soát như bảng quyết định, lược đồ chuyển trạng thái ghi nhận các sự kiện xảy ra, rồi được hệ thống xử lý cũng như những đáp ứng của hệ thống.

Khi hệ thống phải nhớ trạng thái trước đó của mình, hay phải biết trình tự các hoạt động nào là hợp lệ, trình tự nào là không hợp lệ thì lược đồ chuyển trạng thái là rất thích hợp.

Lược đồ chuyển trạng thái được cấu thành từ các thành phần cơ bản sau đây :



Trạng thái đầu

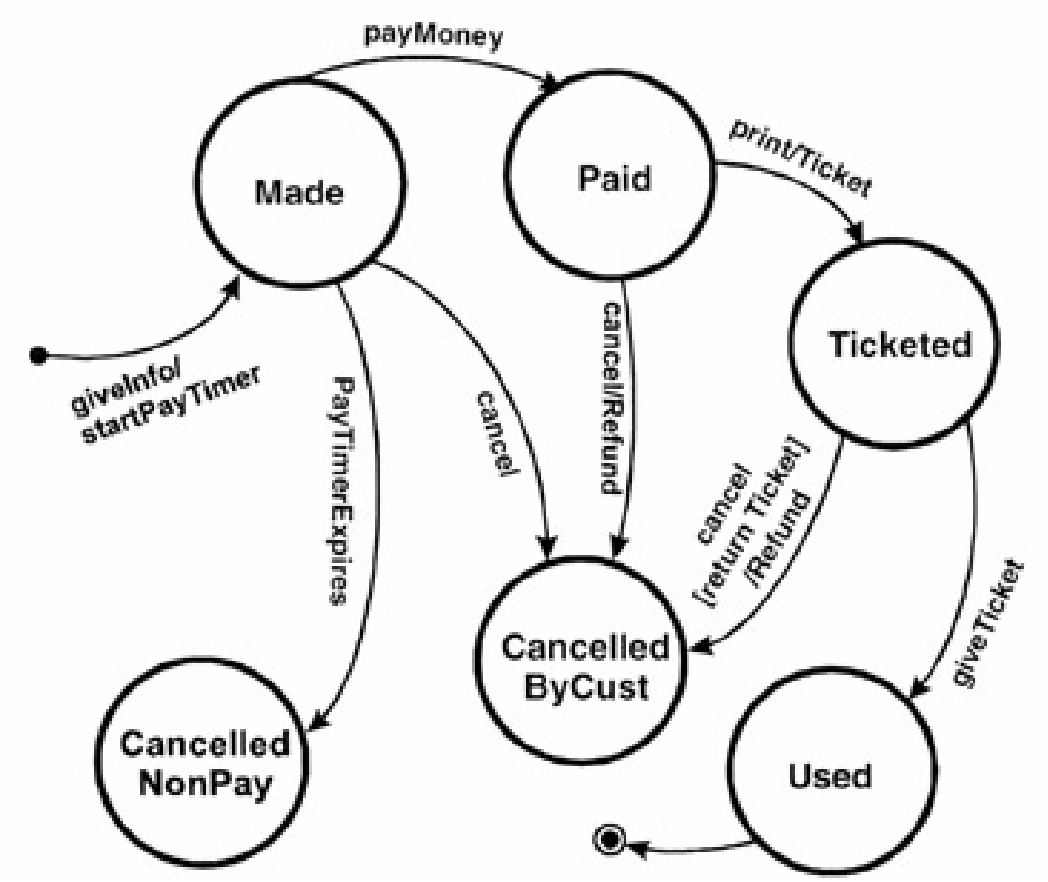
Trạng thái trung gian

Trạng thái cuối

Ta có thể đặt tên nhận dạng cho từng trạng thái trung gian, miêu tả điều kiện chuyển trạng thái kèm theo từng cung chuyển trạng thái.

Ta có thể miêu tả hành động cần thực hiện kết hợp với việc chuyển trạng thái.

Lược đồ chuyển trạng thái của TPPM đặt mua vé máy bay :



TPPM đặt mua vé máy bay có 6 trạng thái khác nhau :

1. Made :
   - điều kiện chuyển đến : sau khi người dùng đã nhập thông tin khách hàng.
   - Hành động cần thực hiện kèm theo : khởi động timer T0 đếm thời gian giữ trạng thái.
2. Cancelled (NonPay) :
   - điều kiện chuyển đến : sau khi timer T0 đã hết.
   - Hành động cần thực hiện kèm theo : null.
3. Paid :
   - điều kiện chuyển đến : sau khi người dùng đã thanh toán tiền.

- Hành động cần thực hiện kèm theo : null.

4. Cancelled (ByCustomer) :
   - điều kiện chuyển đến : sau khi người dùng đã cancel.
   - Hành động cần thực hiện kèm theo : null.
5. Ticketed :
   - điều kiện chuyển đến : sau khi in vé xong.
   - Kết quả kèm theo : vé máy bay.
6. Used :
   - điều kiện chuyển đến : sau khi người dùng đã dùng vé.
   - Hành động cần thực hiện kèm theo : null.

Trong khi lược đồ chuyển trạng thái là cách thức miêu tả hành vi của TPPM dễ hiểu và dễ đọc thì 1 dạng khác - bảng chuyển trạng thái – có thể miêu tả hành vi của TPPM hệ thống hơn và dễ xử lý tự động hơn.

Bảng chuyển trạng thái gồm 4 cột : trạng thái hiện hành, sự kiện xảy ra, hành động cần thực hiện/kết quả thu được, trạng thái kế tiếp.

Thí dụ lược đồ chuyển trạng thái ở slide trước có thể chuyển thành bảng chuyển trạng thái :

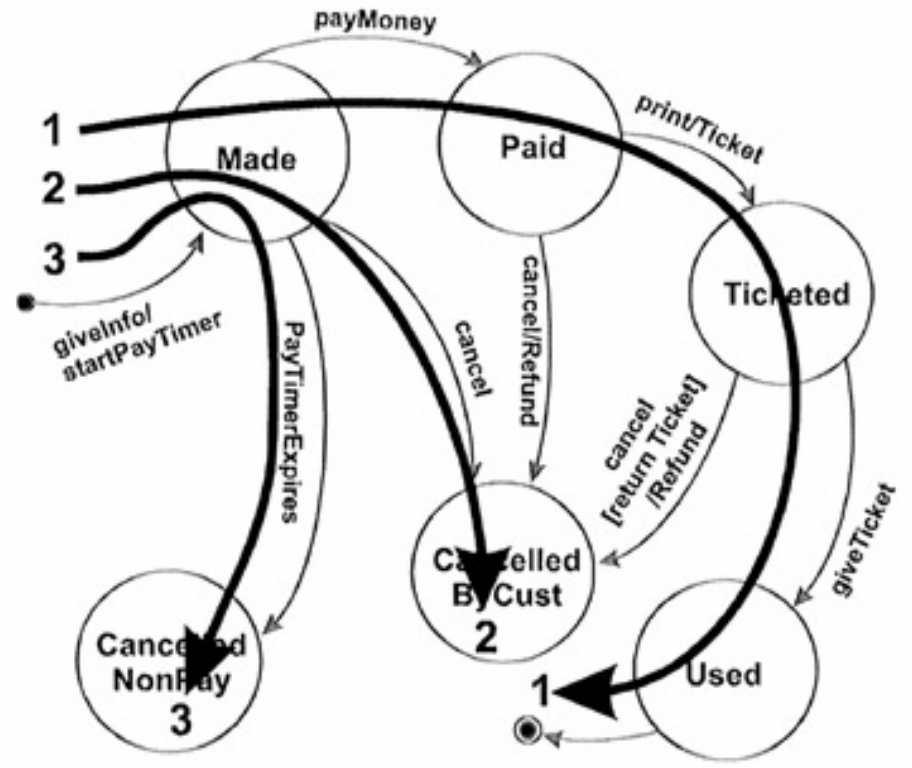
Current State	Event	Action	Next State
null	giveInfo	startPayTimer	Made
null	payMoney	--	null
null	print	--	null
null	giveTicket	--	null
null	cancel	--	null
null	PayTimerExpires	--	null
Made	giveInfo	--	Made

Current State	Event	Action	Next State
Made	payMoney	--	Paid
Made	print	--	Made
Made	giveTicket	--	Made
Made	cancel	--	Can-Cust
Made	PayTimerExpires	--	Can-NonPay
Paid	giveInfo	--	Paid
Paid	payMoney	--	Paid
Paid	print	Ticket	Ticketed
Paid	giveTicket	--	Paid
Paid	cancel	Refund	Can-Cust
Paid	PayTimerExpires	--	Paid
Ticketed	giveInfo	--	Ticketed
Ticketed	payMoney	--	Ticketed
Ticketed	print	--	Ticketed
Ticketed	giveTicket	--	Used
Ticketed	cancel	Refund	Can-Cust
Ticketed	PayTimerExpires	--	Ticketed
Used	giveInfo	--	Used
Used	payMoney	--	Used
Used	print	--	Used
Used	giveTicket	--	Used
Used	cancel	--	Used
Used	PayTimerExpires	--	Used
Can-NonPay	giveInfo	--	Can-NonPay
Can-NonPay	payMoney	--	Can-NonPay
Can-NonPay	print	--	Can-NonPay
Can-NonPay	giveTicket	--	Can-NonPay
Can-NonPay	cancel	--	Can-NonPay
Can-NonPay	PayTimerExpires	--	Can-NonPay

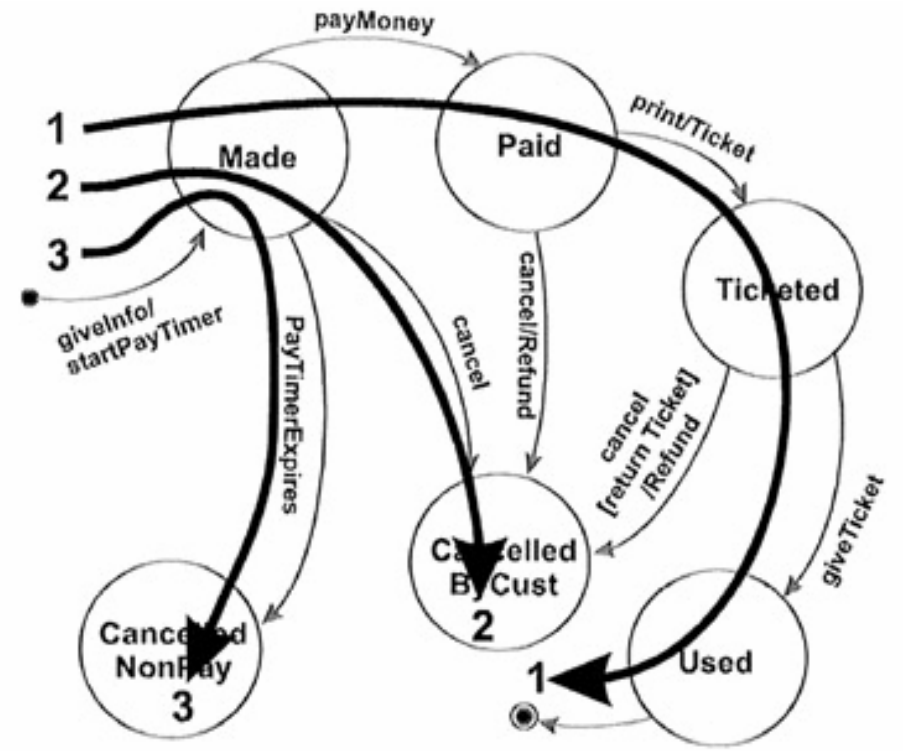
Current State	Event	Action	Next State
Can-Cust	giveInfo	--	Can-Cust
Can-Cust	payMoney	--	Can-Cust
Can-Cust	print	--	Can-Cust
Can-Cust	giveTicket	--	Can-Cust
Can-Cust	cancel	--	Can-Cust
Can-Cust	PayTimerExpires	--	Can-Cust

Dựa vào lược đồ chuyển trạng thái, ta có thể dễ dàng định nghĩa các testcase.

1. Phủ cấp 1 : tạo các testcase sao cho mỗi trạng thái đều xảy ra ít nhất 1 lần. Thí dụ 3 tescase sau sẽ kiểm thử được TPPM đạt phủ cấp 1 :



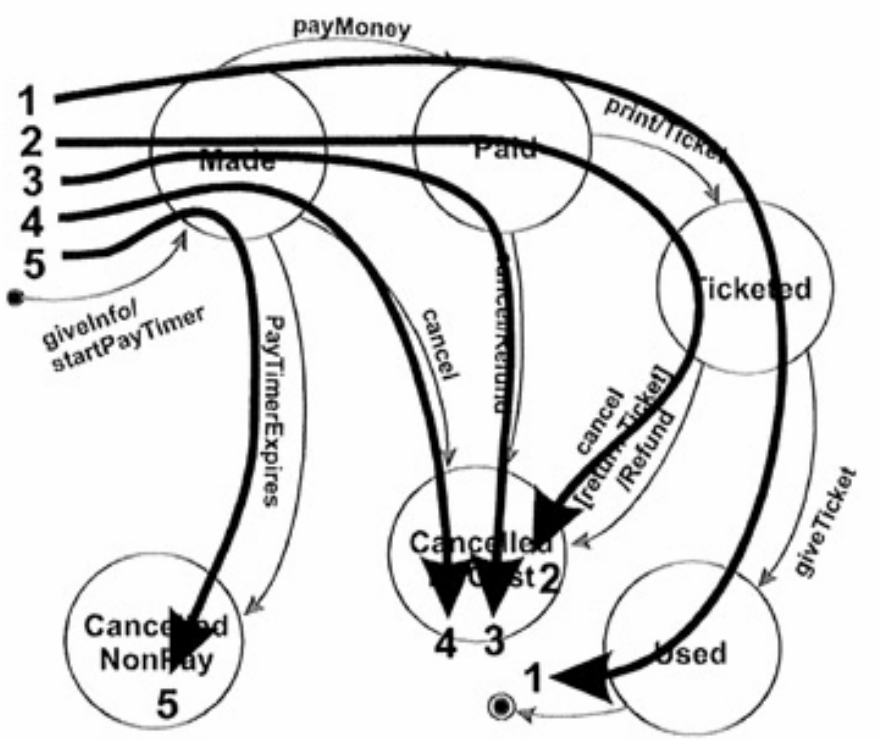
2. Phủ cấp 2 : tạo các testcase sao cho mỗi sự kiện đều xảy ra ít nhất 1 lần. Thí dụ 3 tescase sau sẽ kiểm thử được TPPM đạt phủ cấp 2 :



3. Phủ cấp 3 : tạo các testcase sao cho tất cả các path chuyển đều được kiểm thử. 1 path chuyển là 1 đường chuyển trạng thái xác định, bắt đầu từ trạng thái nhập và kết thúc ở trạng thái kết thúc.

   Đây là phủ tốt nhất vì đã vét cạn mọi khả năng hoạt động của TPPM, tuy nhiên không khả thi vì 1 path chuyển có thể lặp vòng.

4. Phủ cấp 4 : tạo các testcase sao cho mỗi path chuyển tuyến tính đều xảy ra ít nhất 1 lần. Thí dụ 5 tescase sau sẽ kiểm thử được TPPM đạt phủ cấp 4 :



Dựa vào bảng chuyển trạng thái, ta cũng có thể dễ dàng định nghĩa các testcase.

Current State	Event	Action	Next State
null	giveInfo	startPayTimer	Made
null	payMoney	--	null
null	print	--	null
null	giveTicket	--	null
null	cancel	--	null
null	PayTimerExpires	--	null
Made	giveInfo	--	Made
Made	payMoney	--	Paid
Made	print	--	Made
Made	giveTicket	--	Made
Made	cancel	--	Can-Cust
Made	PayTimerExpires	--	Can-NonPay
Paid	giveInfo	--	Paid

Current State	Event	Action	Next State
Paid	payMoney	--	Paid
Paid	print	Ticket	Ticketed
Paid	giveTicket	--	Paid
Paid	cancel	Refund	Can-Cust
Paid	PayTimerExpires	--	Paid
Ticketed	giveInfo	--	Ticketed
Ticketed	payMoney	--	Ticketed
Ticketed	print	--	Ticketed
Ticketed	giveTicket	--	Used
Ticketed	cancel	Refund	Can-Cust
Ticketed	PayTimerExpires	--	Ticketed
Used	giveInfo	--	Used
Used	payMoney	--	Used
Used	print	--	Used
Used	giveTicket	--	Used
Used	cancel	--	Used
Used	PayTimerExpires	--	Used
Can-NonPay	giveInfo	--	Can-NonPay
Can-NonPay	payMoney	--	Can-NonPay
Can-NonPay	print	--	Can-NonPay
Can-NonPay	giveTicket	--	Can-NonPay
Can-NonPay	cancel	--	Can-NonPay
Can-NonPay	PayTimerExpires	--	Can-NonPay
Can-Cust	giveInfo	--	Can-Cust
Can-Cust	payMoney	--	Can-Cust
Can-Cust	print	--	Can-Cust

Current State	Event	Action	Next State
Can-Cust	giveTicket	--	Can-Cust
Can-Cust	cancel	--	Can-Cust
Can-Cust	PayTimerExpires	--	Can-Cust

6.2 Kỹ thuật phân tích vùng (Domain Analysis)

Như ta đã biết, 2 kỹ thuật kiểm thử phân lớp tương đương và phân tích giá trị biên chủ yếu xử lý các biến dữ liệu độc lập, rời rạc. Tuy nhiên thường thì các biến dữ liệu có mối quan hệ với nhau, do đó cách tốt nhất là nên tổ hợp chúng để kiểm thử :

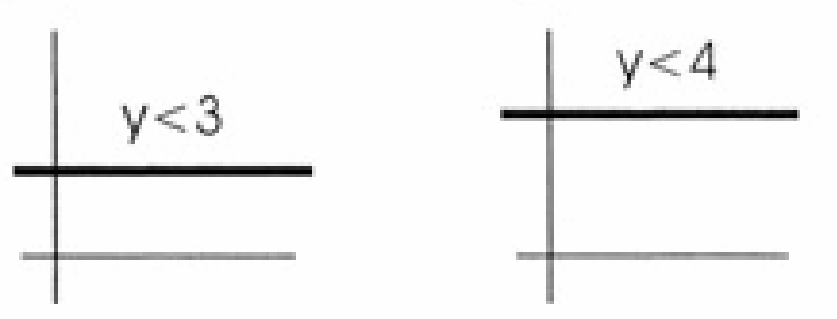
- Nếu tạo các testcase cho từng biến dữ liệu độc lập thì số lượng testcase sẽ quá nhiều.
- Các biến dữ liệu thường tương tác nhau, giá trị của biến này có thể ràng buộc giá trị của biến kia, do đó nếu kiểm thử chúng độc lập thì không thể phát hiện lỗi liên quan đến sự ràng buộc này.

Kỹ thuật phân tích vùng rất thích hợp trong việc xác định các testcase hiệu quả khi các biến dữ liệu có sự tương tác lẫn nhau. Nó được xây dựng trên 2 kỹ thuật kiểm thử được đề cập ở trên và tổng quát hóa chúng để kiểm thử đồng thời n biến dữ liệu.

Xét trường hợp 2 biến dữ liệu tương tác nhau, ta thấy có 4 loại lỗi sau :

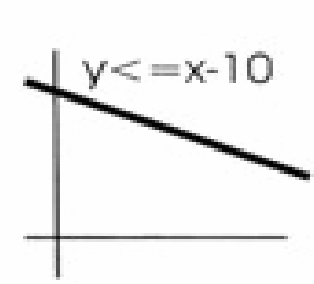
1. Biên ngang bị dịch lên hay xuống

Đúng Sai

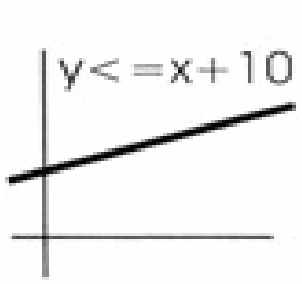


2. Biên nghiêng sai góc

Đúng

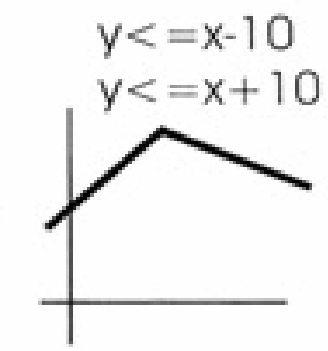


Sai

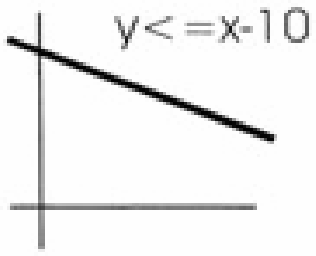


3. Thiếu biên

Đúng

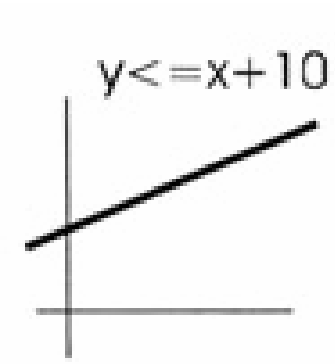


Sai

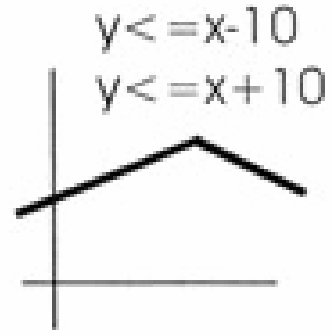


4. Thừa biên

Đúng



Sai



Ta định nghĩa 1 số thuật ngữ :

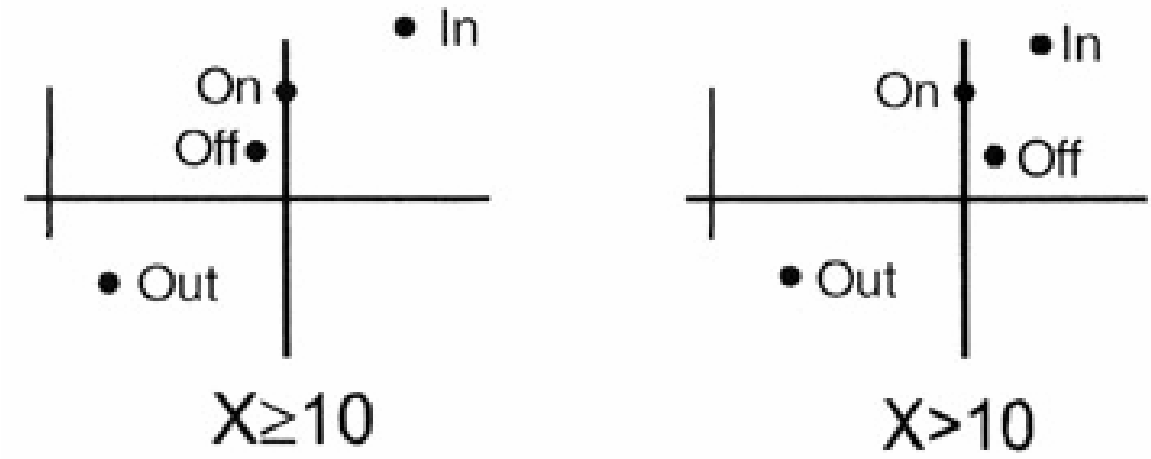
1. Điểm on : là điểm nằm trên biên

2. Điểm off : là điểm không nằm trên biên
3. Điểm in : là điểm thỏa mọi điều kiện biên nhưng không nằm trên biên.
4. Điểm out : là điểm không thỏa bất kỳ điều kiện biên.

Việc chọn điểm on và off thường phức tạp hơn chúng ta nghĩ :

- Nếu biên đóng (dùng toán tử so sánh có yếu tố =), thì điểm on nằm trên biên và thuộc vùng xử lý. Trong trường hợp này, ta chọn điểm off nằm ngoài vùng xử lý.
- Nếu biên mở (dùng toán tử so sánh không có yếu tố =), thì điểm on nằm trên biên nhưng không thuộc vùng xử lý. Trong trường hợp này ta chọn điểm off nằm trong vùng xử lý.

Thí dụ về các điểm on, off, in và out :



Kỹ thuật phân tích vùng yêu cầu chúng ta chọn các tescase theo cách thức sau :

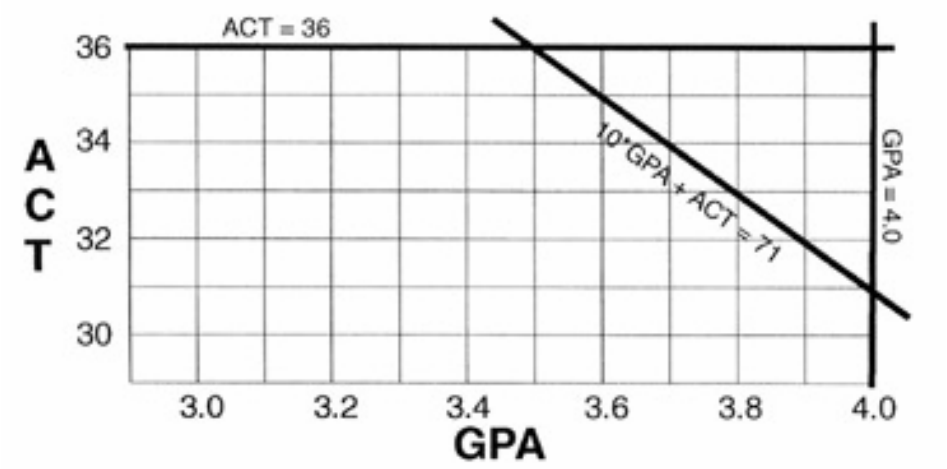
1. Ứng với mỗi điều kiện <, >, ≤, ≥, chọn 1 điểm on và 1 điểm off.
2. Ứng với mỗi điều kiện =, ≠, chọn 1 điểm on, 2 điểm off ngay trên và ngay dưới điểm on.

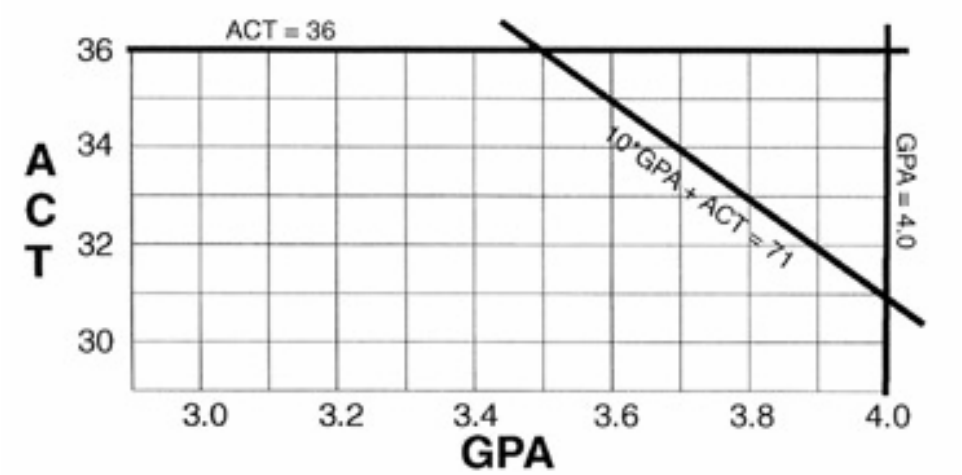
Binder đề nghị 1 bảng rất hữu ích – ma trận kiểm thử vùng :

Variable/ Condition Type			Test Cases															
			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X1	C11	On																
		Off																
	C12	On																
		Off																
	...	On																
		Off																
	C1m	On																
		Off																
	Typical	In																
X2	C21	On																
		Off																
	C22	On																
		Off																
	...	On																
		Off																
	C2m	On																
		Off																
	Typical	In																
Expected Result																		

Thí dụ, TPPM xét kết quả đậu đại học theo tiêu chuẩn sau :

- 10*GPA + ACT >= 71
- GPA : điểm trung bình tích lũy của lớp phổ thông (<=4.0)
- ACT : điểm thi tuyển vào đại học (<= 36).





		GPA						
		0.0 – 3.4	3.5	3.6	3.7	3.8	3.9	4.0
ACT Score	36							
	35							
	34							
	33							
	32							
	31							
	0 - 30							

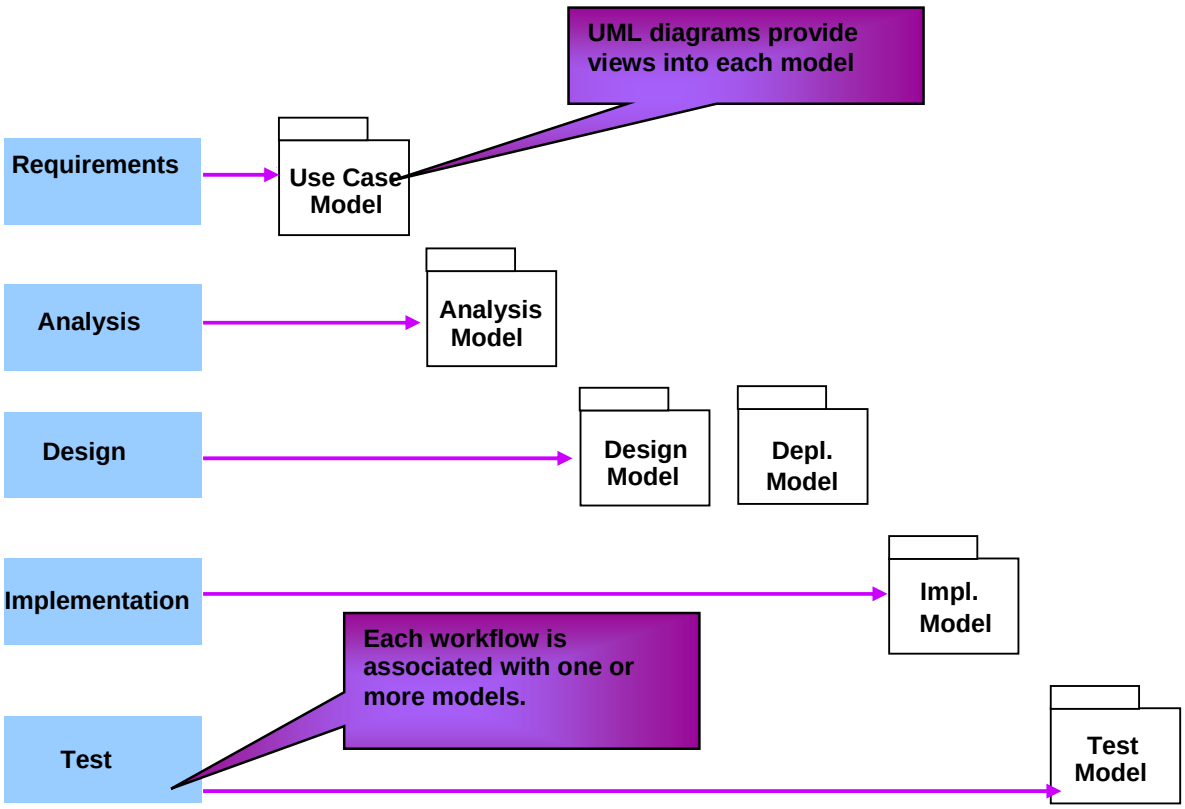
			1	2	3	4	5	6
GPA	GPA ≤ 4.0	On	4.0					
		Off		4.1				
	Typical	In			3.7	3.8	3.8	3.9
ACT	ACT ≤ 36	On			36			
		Off				37		
	Typical	In	34	33			32	35
GPA/ACT	10*GPA + ACT ≥ 71	On						
		Off						
	Typical	In	3.9/35	3.8/34	3.6/36	3.8/34	3.7/34	3.8/32
Expected Result			Admit	Reject	Admit	Reject	Admit	Reject

6.3 Kỹ thuật dùng thông tin trong use-case

Trong qui trình phát triển phần mềm hợp nhất, ta thực hiện nhiều workflows khác nhau : nắm bắt yêu cầu phần mềm, phân tích từng yêu cầu, thiết kế chi tiết để giải quyết từng yêu cầu, hiện thực từng phần bảng thiết kế, kiểm thử kết quả.

Mỗi workflows, thậm chí mỗi lần lập thực hiện 1 workflow, ta phải có kết quả, kết quả này phải được miêu tả ở dạng dễ đọc, dễ hiểu bởi nhiều người và phải cố gắng đơn nghĩa để tránh nhập nhằng.

Ta dùng khái niệm mô hình để miêu tả hệ thống phần mềm theo một góc nhìn nào đó.

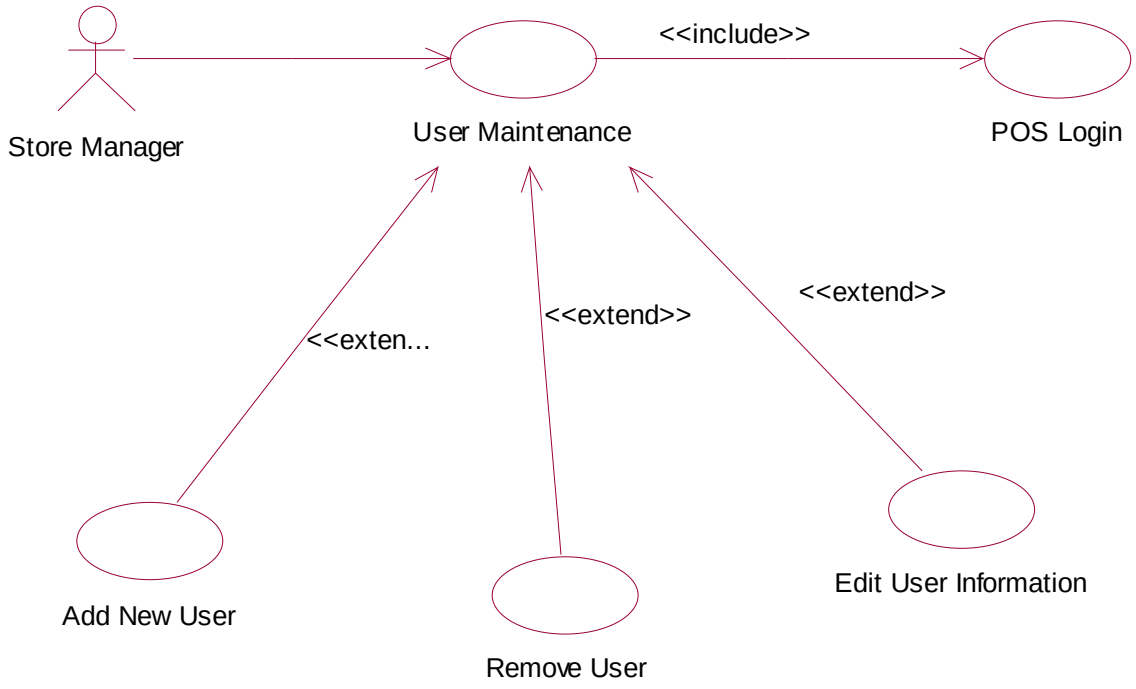


Về nguyên tắc, khi kiểm thử 1 thành phần phần mềm, ta có thể tận dụng bất kỳ thông tin nào trong bất kỳ mô hình nào. Kỹ thuật kiểm thử dùng thông tin trong use-case là kỹ thuật định nghĩa các testcase dựa vào các kịch bản thực hiện usecase.

Như chúng ta biết, mô hình usecase miêu tả hệ thống phần mềm theo góc nhìn bên ngoài : nó cung cấp các chức năng nào cho những “user” nào. Thành phần thiết yếu nhất của mô hình usecase là các lược đồ usecase.

Mỗi lược đồ usecsae thể hiện 1 bộ phận nhỏ của phần mềm : nó bao gồm nhiều chức năng và các chức năng này tương tác với các actor nào.

Thí dụ lược đồ usecase liên quan đến bộ phận chức năng quản lý khách hàng trong hệ thống thương mại điện tử.



Trong lược đồ usecase, mỗi usecase thể hiện 1 chức năng mà bên ngoài có thể truy xuất, tuy nhiên mỗi usecase chỉ được miêu tả ở dạng tối giản : gồm 1 hình ellipse và tên gợi nhớ sơ bộ về chức năng của usecase.

Để hiểu đầy đủ hơn về usecase, người ta cần đặc tả usecase ở 1 dạng chi tiết nào đó. Rất tiết là hiện nay, mỗi nơi mỗi khác, chưa có 1 chuẩn nào được mọi người chấp thuận.

Ở đây, ta hãy dùng khuôn mẫu chi tiết để đặc tả usecase do Alistair Cockburn đề nghị trong sách "Writing Effective Use Cases".

Use Case Component	**Description**
Use Case Number or Identifier	A unique identifier for this use case
Use Case Name	The name should be the goal stated as a short active verb phrase
Goal in Context	A more detailed statement of the goal if necessary
Scope	Corporate I System I Subsystem
Level	Summary I Primary task I Subfunction

Use Case Component	Description	
Primary Actor	Role name or description of the primary actor	
Preconditions	The required state of the system before the use case is triggered	
Success End Conditions	The state of the system upon successful completion of this use case	
Failed End Conditions	The state of the system if the use case cannot execute to completion	
Trigger	The action that initiates the execution of the use case	
Main Success Scenario	Step	Action
	1	
	2	
	...	
Extensions	Conditions under which the main success scenario will vary and a description of those variations	
Sub-Variations	Variations that do not affect the main flow but that must be considered	
Priority	Criticality	
Response Time	Time available to execute this use case	
Frequency	How often this use case is executed	
Channels to Primary Actor	Interactive I File I Database I ...	
Secondary Actors	Other actors needed to accomplish this use case	
Channels to Secondary Actors	Interactive I File I Database I ...	
Date Due	Schedule information	
Completeness Level	Use Case identified (0.1)I Main scenario defined (0.5) I All extensions defined (0.8) I All fields complete (1.0)	
Open Issues	Unresolved issues awaiting decisions	

Thí dụ bảng đặc tả usecase "đăng ký môn học" trong phần mềm quản lý học vụ có nội dung chi tiết như sau :

Use Case Component	Description	
Use Case Number or Identifier	SURS1138	
Use Case Name	Register for a course (a class taught by a faculty member)	
Goal in Context		
Scope	System	
Level	Primary task	
Primary Actor	Student	
Preconditions	None	
Success End Conditions	The student is registered for the course—the course has been added to the student's course list	
Failed End Conditions	The student's course list is unchanged	
Trigger	Student selects a course and "Registers"	
Main Success Scenario A: Actor S: System	Step	Action
	1	A: Selects "Register for a course"
	2	A: Selects course (e.g. Math 1060)
	3	S: Displays course description
	4	A: Selects section (Mon & Wed 9:00am)
	5	S: Displays section days and times
	6	A: Accepts
	7	S: Adds course/section to student's course list
Extensions	2a	Course does not exist S: Display message and exit
	4a	Section does not exist S: Display message and exit
	4b	Section is full S: Display message and exit
	6a	Student does not accept S: Display message and exit

Use Case Component	Description
Sub-Variations	Student may use • Web • Phone
Priority	Critical
Response Time	10 seconds or less
Frequency	Approximately 5 courses x 10,000 students over a 4-week period
Channels to Primary Actor	Interactive
Secondary Actors	None
Channels to Secondary Actors	N/A
Date Due	1 Feb
Completeness Level	0.5
Open Issues	None

Dựa vào đặc tả về kịch bản chính và về các nới rộng của kịch bản, ta sẽ thiết kế các testcase theo ý tưởng như sau :

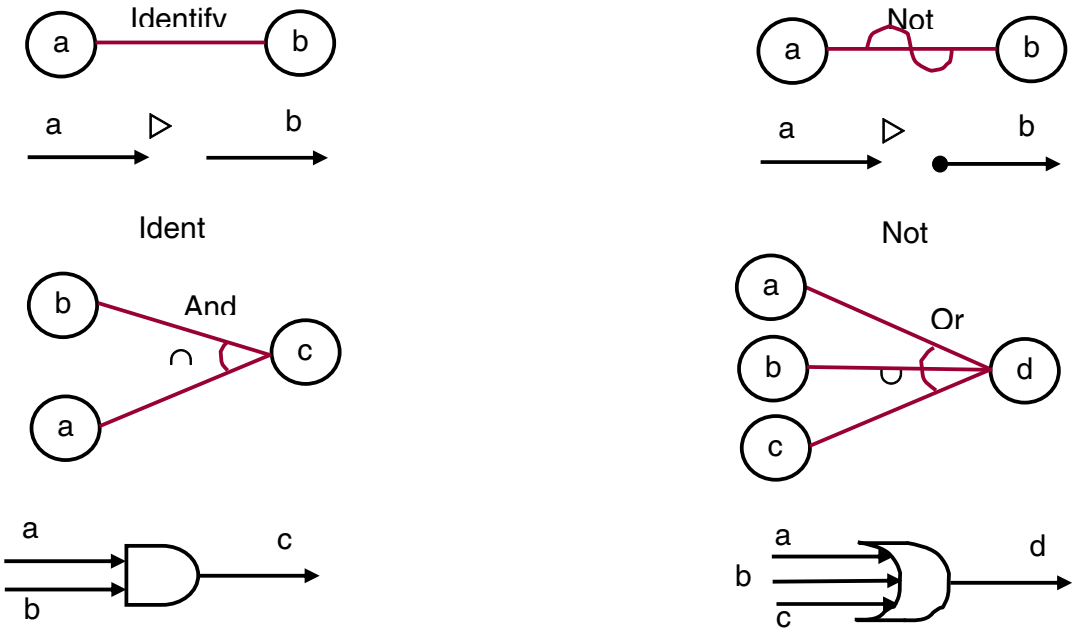
- Ít nhất 1 testcase để kiểm thử kịch bản chính.
- Ít nhất 1 testcase cho từng nới rộng có thể có.
- Nếu kịch bản chính hay 1 nới rộng nào đó bị loop thì không cần thiết phải kiểm thử phần loop lại.

6.4 Kỹ thuật dùng đồ thị nhân quả (Cause-Effect Diagram)

Đồ thị nhân quả là 1 dạng khác của mạng luận lý tổ hợp mà phần cứng thường dùng. Các phần tử cấu thành đồ thị nhân quả là :

- các nút : mỗi nút miêu tả 1 hậu quả (1 hay nhiều hoạt động + 1 hay nhiều kết quả).
- các đoạn thẳng : mỗi đoạn thẳng miêu tả 1 nguyên nhân (1 điều kiện dữ liệu nhập ở dạng nhị phân)

- các ký hiệu : mỗi ký hiệu miêu tả 1 phép toán luận lý.
- các phần tử ràng buộc, mỗi phần tử miêu tả 1 ràng buộc xác định nào đó.



Giả sử đặc tả 1 TPPM như sau : dữ liệu đầu vào là tên file gồm 2 ký tự, ký tự đầu là A hay B, ký tự còn lại là ký số, TPPM sẽ cập nhật file, nếu ký tự đầu không phải là A hay B thì TPPM báo lỗi X1, nếu ký tự thứ 2 không phải là số thì báo lỗi X2.

Duyệt đọc đặc tả và phân tích đặc tả, ta tìm được các điều kiện đầu vào là :

1 : Ký tự đầu là A.

2 : Ký tự đầu là B.

3 : Ký tự thứ hai là ký số.

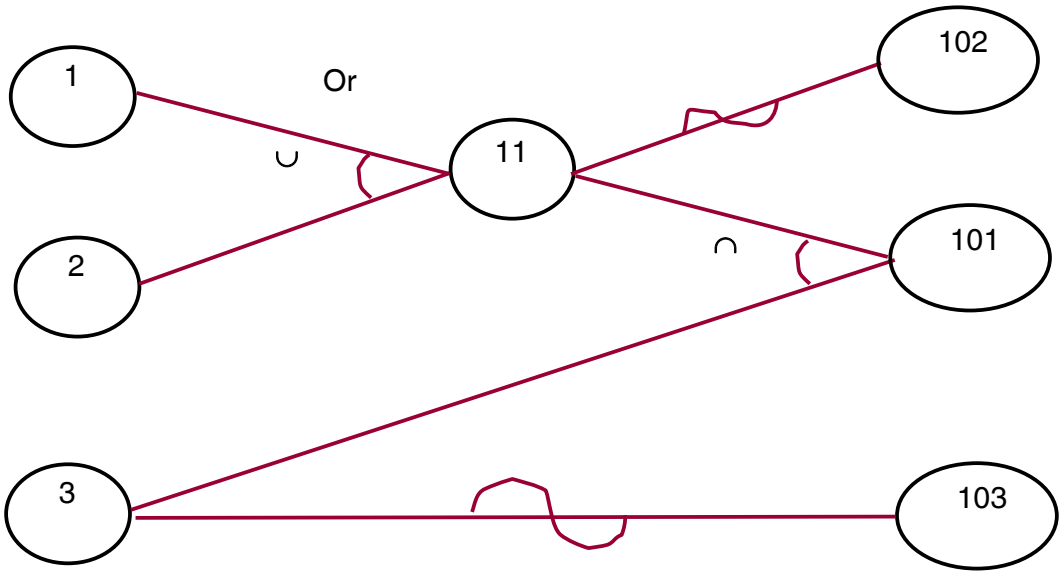
Và các hậu quả ở đầu ra là :

101 : cập nhật file.

102 : báo lỗi X1.

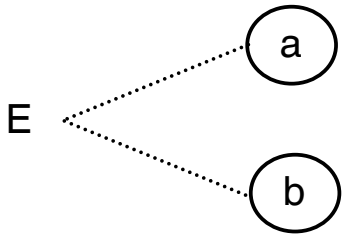
103 : báo lỗi X2.

Đồ thị nhân quả của TPPM ở silde trước là :

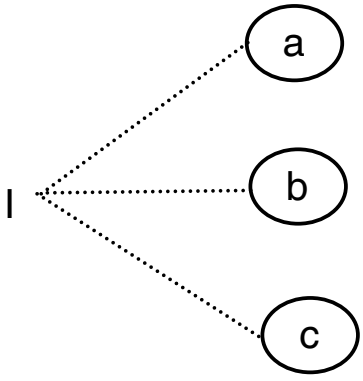


Trong nhiều trường hợp, tồn tại tổ hợp điều kiện nhập không thể xảy ra, thí dụ ở slide trước, điều kiện 1 và 2 không thể xảy ra đồng thời vì ký tự đầu không thể vừa là A vừa là B. Để miêu tả các ràng buộc này, ta dùng các ký hiệu ràng buộc sau :

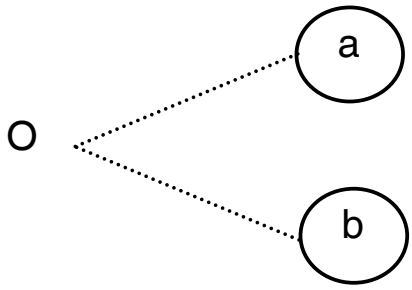
1. E : không thể đồng thời xảy ra.



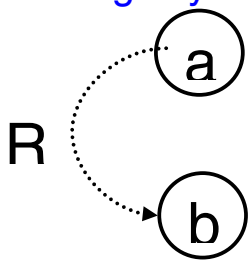
2. I : phải ít nhất 1 điều kiện xảy ra.



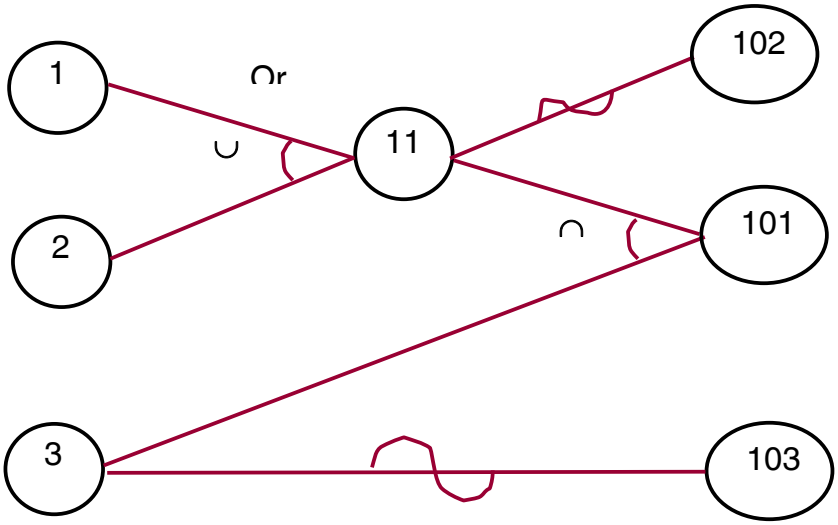
3. O : 1 và chỉ 1 điều kiện xảy ra.



4. R : nếu a xảy ra thì b cũng xảy ra.



Đồ thị nhân quả của TPPM ở slide 34 được hoàn chỉnh là :



Qui trình định nghĩa các testcase dùng kỹ thuật đồ thị nhân quả gồm các bước :

1. Đặc tả của TPPM được chia nhỏ ra nhiều phần nhỏ để có thể làm việc dễ dàng (nếu không thì đồ thị nhân quả sẽ rất phức tạp).
2. Nhận dạng các nguyên nhân và hậu quả của phần nhỏ đang xử lý.

3. Tìm mối quan hệ giữa các nguyên nhân và hậu quả, mỗi mối quan hệ được vẽ thành 1 đường nối.
4. Xác định các ràng buộc giữa các nguyên nhân và chú thích chúng vào đồ thị.
5. Chuyển đồ thị nhân quả về bảng quyết định.
6. Chuyển bảng quyết định thành bảng các testcase.

6.5 Kết chương

Chương này đã tiếp tục giới thiệu chi tiết cụ thể về 4 kỹ thuật kiểm thử hộp đen được dùng phổ biến khác là kỹ thuật dùng lược đồ chuyển trạng thái, kỹ thuật phân tích vùng, kỹ thuật dùng thông tin trong use-case, và kỹ thuật dùng đồ thị nhân quả.

Ứng với mỗi kỹ thuật kiểm thử, chúng ta cũng đã giới thiệu 1 thí dụ cụ thể để demo qui trình thực hiện kỹ thuật kiểm thử tương ứng.