Lý thuyết
Kiểm Tra Phần Mềm

Bài 05: ***Các kỹ thuật thiết kế kịch bản kiểm tra***

GV: Nguyễn Ngọc Tú

Email: nntu@hoasen.edu.vn

Bộ môn: Kỹ thuật Phần mềm

Nội dung

- Kỹ thuật thiết kế kịch bản kiểm thử
- Xác định thiết kế và Giai đoạn phát triển kiểm thử
- Phân loại kỹ thuật thiết kế
- Các kỹ thuật dựa trên đặc tả / Hộp đen (*Black box*)
  - Phân hoạch tương đương
  - Phân tích giá trị biên
  - Kiểm tra bảng quyết định
  - Kiểm tra chuyển trạng thái
- Các kỹ thuật dựa trên cấu trúc / Hộp trắng (*White box*)
  - Kiểm tra các câu lệnh
  - Kiểm tra các câu lệnh rẽ nhánh
  - Các kỹ thuật kiểm tra khác
- Các kỹ thuật dựa trên kinh nghiệm
- Chọn lựa kỹ thuật kiểm tra

Kỹ thuật thiết kế Kiểm thử

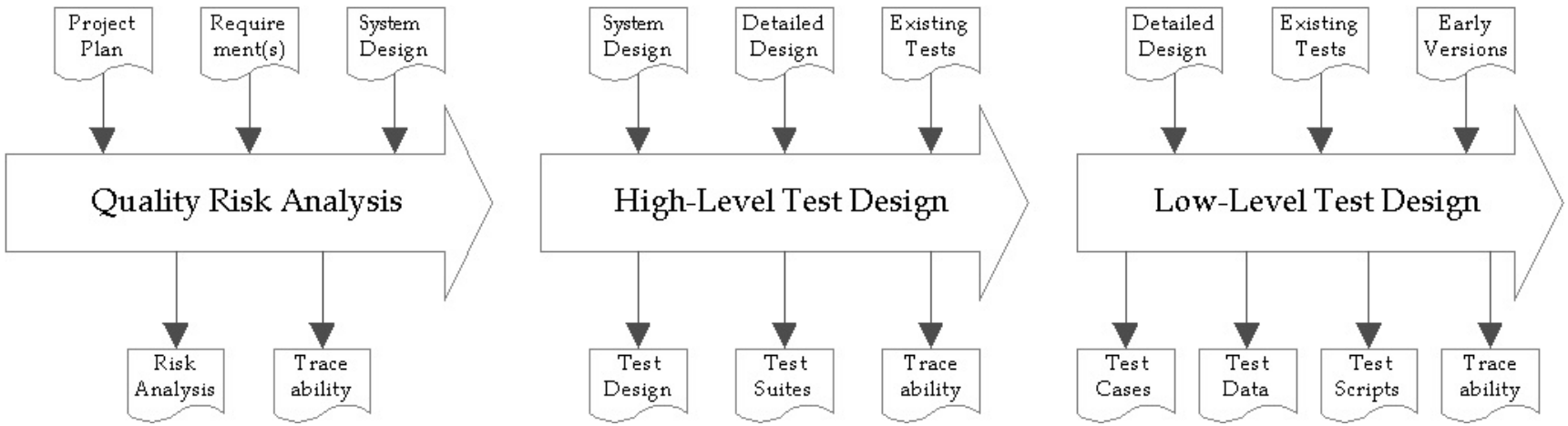
- Xác định và thiết kế test cases
- Phân loại kỹ thuật thiết kế Kiểm thử
- Kỹ thuật dựa trên đặc tả - hộp đen
- Kỹ thuật dựa trên cấu trúc - hộp trắng
- Kỹ thuật dựa trên kinh nghiệm
- Chọn kỹ thuật kiểm thử

Xác định và thiết kế Kịch bản

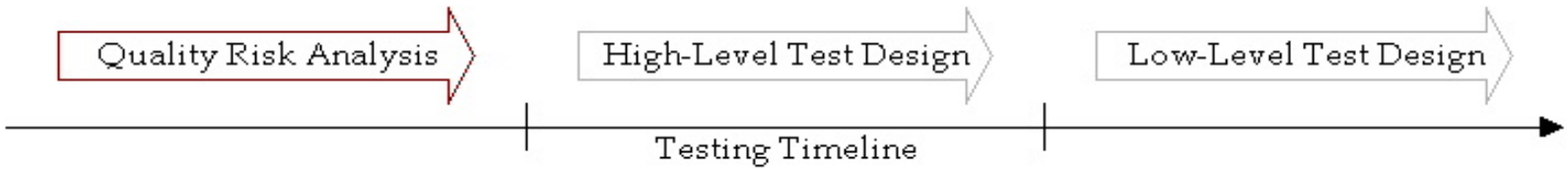
- Khái niệm chính
  - Cơ sở kiểm thử trong phân tích rủi ro
  - Xác định mức rủi ro: khả năng xảy ra và ảnh hưởng
  - Định rõ thiết kế kiểm thử, kịch bản, thủ tục
  - Viết kịch bản
  - Quan hệ Kịch bản và thủ tục
  - Lập lịch thực hiện kiểm thử

Giai đoạn phát triển kiểm thử

- Thực hiện trong qua giai đoạn
  - Phân tích rủi ro chất lượng
  - Thiết kế kiểm thử mức cao
  - Thiết kế kiểm thử mức thấp



Phân tích

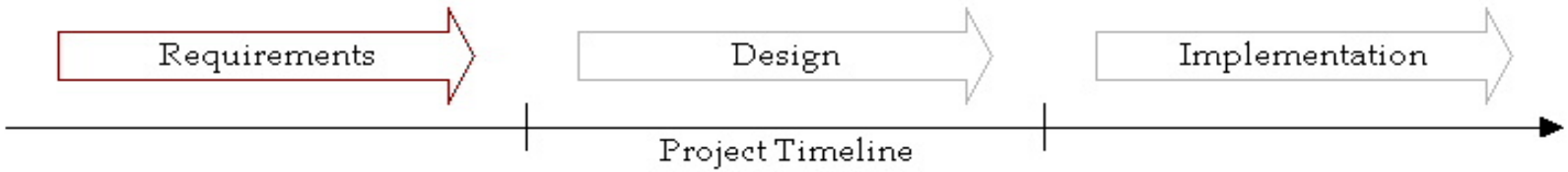


1.0 Quality Risk Category 1
1.1 Quality Risk
1.2 Quality Risk
1.3 Quality Risk

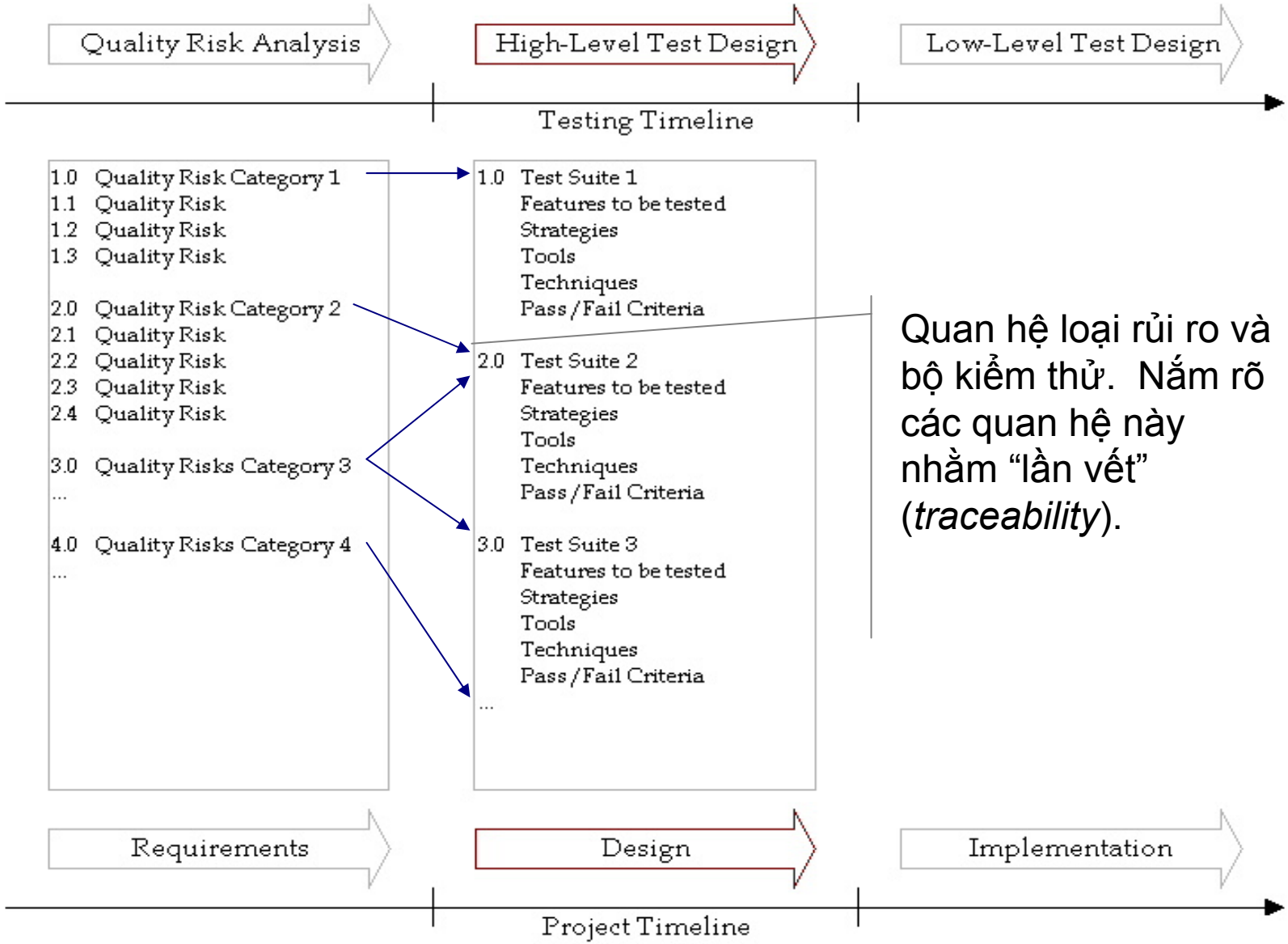
2.0 Quality Risk Category 2
2.1 Quality Risk
2.2 Quality Risk
2.3 Quality Risk
2.4 Quality Risk

3.0 Quality Risks Category 3
...

4.0 Quality Risks Category 4
...

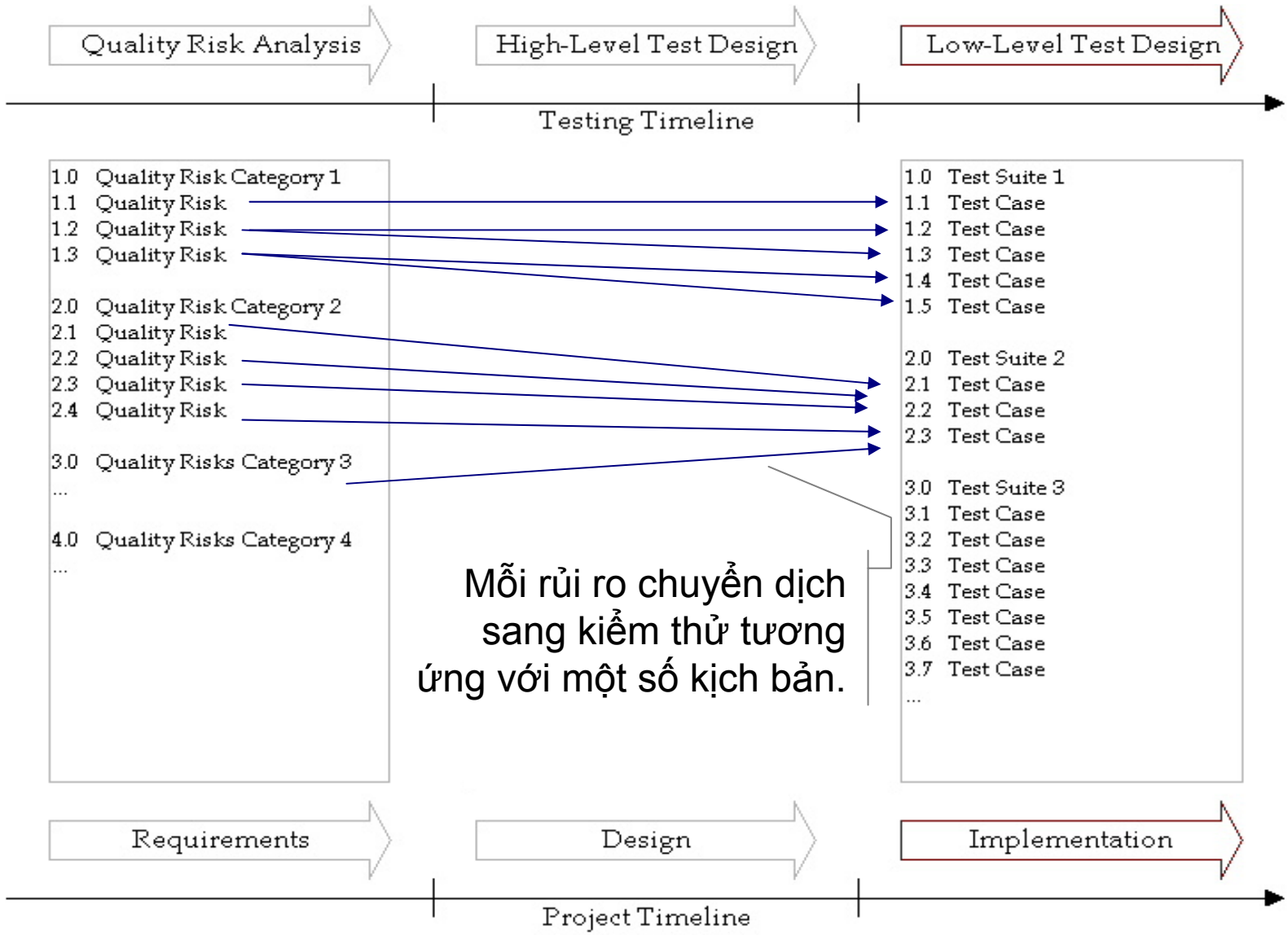


Thiết kế mức cao



Quan hệ loại rủi ro và bộ kiểm thử. Nắm rõ các quan hệ này nhằm "lần vết" (*traceability*).

Thiết kế mức thấp – hiện thực



Chất lượng – rủi ro

- Rủi ro
  - Khả năng xảy ra không mong đợi
- Chất lượng và rủi ro sản phẩm
  - Khả năng hệ thống sẽ sai với yêu cầu khách hàng, người sử dụng, giao kèo
  - Họ các lỗi có thể ẩn dấu sau rủi ro chất lượng

Phân tích rủi ro chất lượng !

Không chính thức	Chuẩn ISO 9126	Lỗi không mong đợi và phân tích ảnh hưởng
Bắt đầu với phân loại rủi ro cổ điển	Bắt đầu với 6 đặc tính chất lượng chính	Bắt đầu với đặc tính, phân loại, hệ thống con
Chức năng, trạng thái-giao dịch, công suất-khối lượng, chất lượng dữ liệu, xử lý lỗi-phục hồi, hiệu năng, chuẩn-bản địa hóa, khả dụng, …	Chức năng, độ tin cậy, khả dụng, bảo trì, hiệu quả, khả chuyển (*FRUEMP*) → phân rã theo từng đặc điểm nhỏ cho hệ thống	Liệt kê các cách có thể xảy ra không mong đợi, dự đoán ảnh hưởng lên hệ thống, NSD, … Phân công nghiêm ngặt, lập sự ưu tiên, khả năng xảy ra → tính số ưu tiên rủi ro (*risk priority number - RPN*)
Lập ưu tiên kiểm thử cho mỗi rủi ro chất lượng	Lập ưu tiên kiểm thử cho kiểu đặc tính	Sử dụng RPN để dướng dẫn mức độ kiểm tra sâu-rộng

Rủi ro → tiềm tàng sự suy giảm mức độ hài lòng

Risk priority number: đánh giá tổng thể của rủi ro

Rủi ro nghiệp vụ: ảnh hưởng của vấn đề

Rủi ro kỹ thuật: khả năng xảy ra vấn đề

Lần vết thông tin ngược lại các yêu cầu, khía cạnh rủi ro, …

Quality Risk	Tech. Risk	Bus. Risk	Risk Pri. #	Extent of Testing	Tracing
Risk Category 1					
Risk 1					
Risk 2					
Risk *n*					

Phân cấp các loại rủi ro giúp dễ liệt kê và gợi nhớ

1 = Very high
2 = High
3 = Medium
4 = Low
5 = Very low

Đánh giá rủi ro từ tổ hợp rủi ro KT – NV
1-25

1-5 = Extensive
6-10 = Broad
11-15 = Cursory
16-20 = Opportunity
21-25 = Report bugs

Gợi ý cho phân tích rủi ro

- Nhóm não công chéo chức năng
- Xác định và gán mức rủi ro cho mục xác định
- Chỉ tách rời các mục khi cần thiết để phân biệt giữa các mức khác nhau
- Cân nhắc rủi ro kỹ thuật và nghiệp vụ
  - Rủi ro kỹ thuật:
    - khả năng xảy ra, ảnh hưởng của “vấn đề” lên hệ thống
  - Rủi ro nghiệp vụ:
    - khả năng sử dụng, ảnh hưởng tới NSD
- Tiếp tục và sắp xếp lại phân tích rủi ro, dự án tại những mốc thời gian quan trọng

IEEE 829: Đặc tả thiết kế

- Mô tả tập hợp các kịch bản kiểm thử ở mức cao, bao gồm các phần sau
  - Xác định đặc tả thiết kế kiểm thử
  - Đặc trưng được kiểm thử (*trong bộ kiểm thử - test suite*)
  - Làm rõ cách tiếp cận (*kỹ thuật, công cụ đặc biệt, ...*)
  - Nhận dạng kiểm thử (*lần vết tới kịch bản kiểm thử trong bộ*)
  - Các đặc trưng thỏa/không thỏa tiêu chuẩn (*test oracle, cơ sở kiểm thử, hệ thống kế thừa, ...*)
- Tập hợp các kịch bản gọi là bộ kiểm thử - test suite
- Thứ tự (*test suites , test cases*) thường
  - hướng theo độ ưu tiên rủi ro, nghiệp vụ
  - chịu tác động bởi hợp đồng, tài nguyên, tiến triển

IEEE 829: Đặc tả kịch bản

- Đặc tả kịch bản kiểm thử bao gồm
  - Xác định đặc tả kịch bản kiểm thử
  - Các Mục kiểm thử
    - *phần nào được chuyển giao / kiểm thử*
  - Đặc tả thông tin đầu vào
    - *nhập liệu từ NSD, tập tin, ...*
  - Đặc tả thông tin đầu ra
    - *kết quả kỳ vọng: màn hình kết quả, tập tin, thời gian,…*
  - Môi trường cần thiết
    - *phần cứng, phần mềm, con người, phụ trợ, …*
  - Yêu cầu thủ tục đặc biệt
    - *tác vụ can thiệp, quyền, …*
  - Phụ thuộc liên kịch bản
    - *nếu cần cho tiền điều kiện*
- Thực tế, kịch bản thay đổi đáng kể trong việc cố gắng, quá trình, bảo đảm điều kiện kiểm thử

IEEE 829: Đặc tả thủ tục

- Mô tả làm sao để thực thi một hay nhiều kịch bản, bao gồm các phần sau
  - Xác định đặc tả thủ tục kiểm thử
  - Mục đích
    - *kiểm thử nào được thực thi*
  - Yêu cầu đặc biệt
    - *kỹ năng, quyền, môi trường, …*
  - Các bước thủ tục
    - *ghi nhận, thiết lập, khởi động, thực hiện, đánh giá kết quả, shutdown/suspension, khởi động lại, dừng, wrap up/tear down, sự kiện bất ngờ*
- Các thủ tục kiểm thử thường được nhúng trong các kịch bản
- Kịch bản kiểm thử đôi khi được nhắc đến như là mã kịch bản kiểm thử (*script*) và có thể thực hiện tay hay tự động

Bao phủ kịch bản Kiểm thử
(Lần vết - Traceability)

- Đo lường, đánh giá tương quan các miền nội dung kiểm thử
- Sử dụng như cách đánh giá và làm nổi bật kiểm thử
- Kiểu thực tế
  - Đặc tả yêu cầu
  - Đặc tả thiết kế
  - Miền chức năng
  - Rủi ro chất lượng
  - Cấu hình
- Kỹ thuật thường sử dụng để đảm bảo bao phủ kiểm thử hoàn toàn

	Test Case													
	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	2.1	2.2	2.3	2.4	2.5	Total
Spec														
1.1	1	1		1		1		2						**6**
1.2	2			2	1				1		2			**8**
1.3														**0**
1.4		2									2			**4**
1.5	1			1				1	1					**4**
1.6						2								**2**
1.7			2		1		2			2				**7**
1.8					2							1		**3**
1.9	1			1							2			**4**
1.10		1					1	1			1			**4**
Total	**5**	**4**	**2**	**5**	**4**	**3**	**3**	**4**	**2**	**2**	**7**	**1**	**0**	

- Kỹ thuật
  - Sử dụng bảng tính
  - Liệt kê kiểm thử và miền bao phủ để đánh giá
  - 0: none; 1: indirect; 2: direct

Phân loại Kỹ thuật thiết kế

- Khái niệm chính
  - Lý do thực hiện kiểm thử
    - hộp đen (*specification-based - black box*),
    - hộp trắng (*structure-based - white box*),
    - thực nghiệm (*experience-based tests*)
  - Các kỹ thuật thông thường
  - Đặc điểm và sự khác biệt

3 kiểu thiết kế Kiểm thử

- Hộp đen (*Specification-based - black box*), chức năng, phi chức năng (*functional , non-functional*)
  - Tạo kiểm thử chủ yếu nhờ vào phân tích các "cơ sở" kiểm thử
  - Tìm kiếm lỗi theo cách của hệ thống thực hiện
- Hộp trắng (*Structure-based - white box*)
  - Tạo kiểm thử chủ yếu nhờ vào phân tích cấu trúc các thành phần hay hệ thống
  - Tìm kiếm lỗi theo cách hệ thống được xây dựng
- Dựa trên kinh nghiệm (*Experience-based : tấn công, thăm dò, danh sách ý kiểm tra*)
  - Tạo kiểm thử chủ yếu nhờ vào sự hiểu biết về hệ thống, kinh nghiệm quá khứ, phương pháp phỏng đoán về lỗi
  - Tìm lỗi nơi mà hệ thống có lỗi

Hộp đen

- Các thành phần thông thường:
  - Các mô hình chính thức, không chính thức sử dụng định rõ vấn đề được giải quyết, phần mềm hoặc thành phần
  - Kịch bản kiểm thử hệ thống hóa từ các mô hình này

- Bao gồm các phương pháp:
  - Phân hoạch tương đương (*Equivalence partitioning*)
  - Phân tích giá trị biên (*boundary value analysis*)
  - Sơ đồ chuyển trạng thái (*State transition diagrams*)
  - Bảng quyết định (*Decision tables*)
  - Đồ thị Nhân quả (*Cause-Effect Graph*)

Hộp trắng

- Bao gồm các phần:
  - Cấu trúc hệ thống (*mã, thiết kế,* ...) thường dẫn tới kịch bản kiểm thử
  - Qui mô bao phủ kiến trúc có thể được đánh giá cho kịch bản kiểm thử khác
  - Kiểm thử xa hơn có thể được dẫn ra có hệ thống hóa để gia tăng khả năng bao phủ

- Bao gồm:
  - Bao phủ phát biểu
  - Bao phủ nhánh

Dựa trên Thực nghiệm

- Bao gồm các phần:
  - Tri thức, kinh nghiệm dùng để dẫn ra các kịch bản
  - Có thể cân nhắc tri thức và kinh nghiệm phần mềm
    - việc sử dụng, môi trường, …
  - tri thức về lịch sử, khả năng có khiếm khuyết và phân bố lỗi

- Bao gồm:
  - Tấn công
  - Danh sách ý kiểm tra (*Checklists*)
  - Thăm dò (*Exploratory*)

Kỹ thuật dựa trên đặc tả

- Khái niệm chính
  - Sử dụng các phương pháp để viết kịch bản kiểm thử từ mô hình
  - Mục tiêu chính của mỗi kỹ thuật
  - Làm sao có thể đánh giá độ bao phủ
  - Kiểm thử trường hợp sử dụng (*Use case*)

Phân lớp/hoạch tương đương

- Phân Chia thông tin đầu vào, ra, hành vi và môi trường thành các phân lớp được "xử lý" tương đương
- Định nghĩa ít nhất 1 kịch bản cho mỗi phần hoặc các vùng giá trị
  - Kiểm thử một giá trị đại diện của lớp.
  - Nếu giá trị đại diện bị lỗi thì các thành viên trong lớp đó cũng sẽ bị lỗi như thế.
- Có thể sử dụng thông tin tiếp thị vào các phân lớp đặc biệt
- VD: kiểm tra hỗ trợ của máy in
  - Giao tiếp vật lý:
    - Parallel, serial, USB 1.1, USB 2.0, infrared, Firewire, SCSI, …?
  - Giao tiếp luận lý:
    - Postscript, HPPL, ASCII, …
  - Ứng dụng ảnh:
    - Laser jet, ink jets, bubble jets, dot matrix, line printers, letter quality, pen plotters, …

Phân lớp/hoạch tương đương

- Nhận dạng các lớp tương đương đầu vào:
  - Dựa vào các điều kiện vào/ra trong đặc tính kỹ thuật/mô tả kỹ thuật.
  - Cả hai lớp tương đương đầu vào: 'valid' và 'invalid'.
  - Dựa vào heuristics và chuyên gia.
    - "input x in [1..10]" → classes: $x<1$, $1 \le x \le 10$, $x>10$
    - "Loại liệt kê A, B, C" → classes: A, B, C, not{A,B,C}
- Định nghĩa mẫu thử của các trường hợp cho mỗi lớp.
- Kiểm thử các trường hợp thuộc lớp tương đương 'valid'
- Kiểm thử các trường hợp thuộc lớp tương đương 'invalid'

Phân lớp/hoạch tương đương

- Ví dụ
  - Nhập tọa độ 1 điểm trong không gian hai chiều với
    - $3 \leq x \leq 7$
    - $5 \leq y \leq 9$
  - Phân hoạch không gian tương đương ?

Phân tích giá trị biên *(BVA)*

- Ràm rõ phân hoạch tương đương tại các "góc cạnh", điểm trị cuối của mỗi miền
  - Phân lớp tương đương tìm lỗi trong mã xử lý mỗi miền như nhau
  - Giá trị biên cũng tìm lỗi tại các vùng biên-viền
- Chỉ có thể sử dụng khi các vùng tương đương có thứ tự
- Biên phi chức năng (*công suất , khối lượng , ...*) có thể được dùng để kiểm tra phi chức năng

Phân tích giá trị biên

- Chọn các giá trị biên đầu vào để kiểm tra các lớp đầu vào thay vì thêm vào những giá trị tùy ý.

- Cũng chọn những giá trị đầu vào như thế để cho ra những giá trị biên đầu ra.

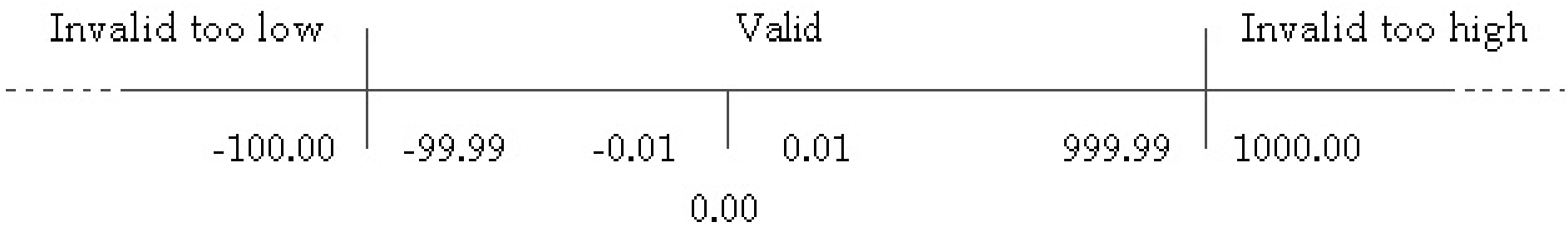
- Ví dụ *về chiến lược mở rộng việc phân lớp:*
  - *Chọn một giá trị tùy ý cho mỗi lớp.*
  - *Chọn các giá trị chính xác ở biên trên và biên dưới của mỗi lớp.*
  - *Chọn các giá trị ngay lập tức ở dưới và trên mỗi biên (nếu có thể).*

Ví dụ: Số nguyên

- "Muốn phân bao nhiêu phần ?"
  - Nhập số nguyên dương nhỏ hơn 100

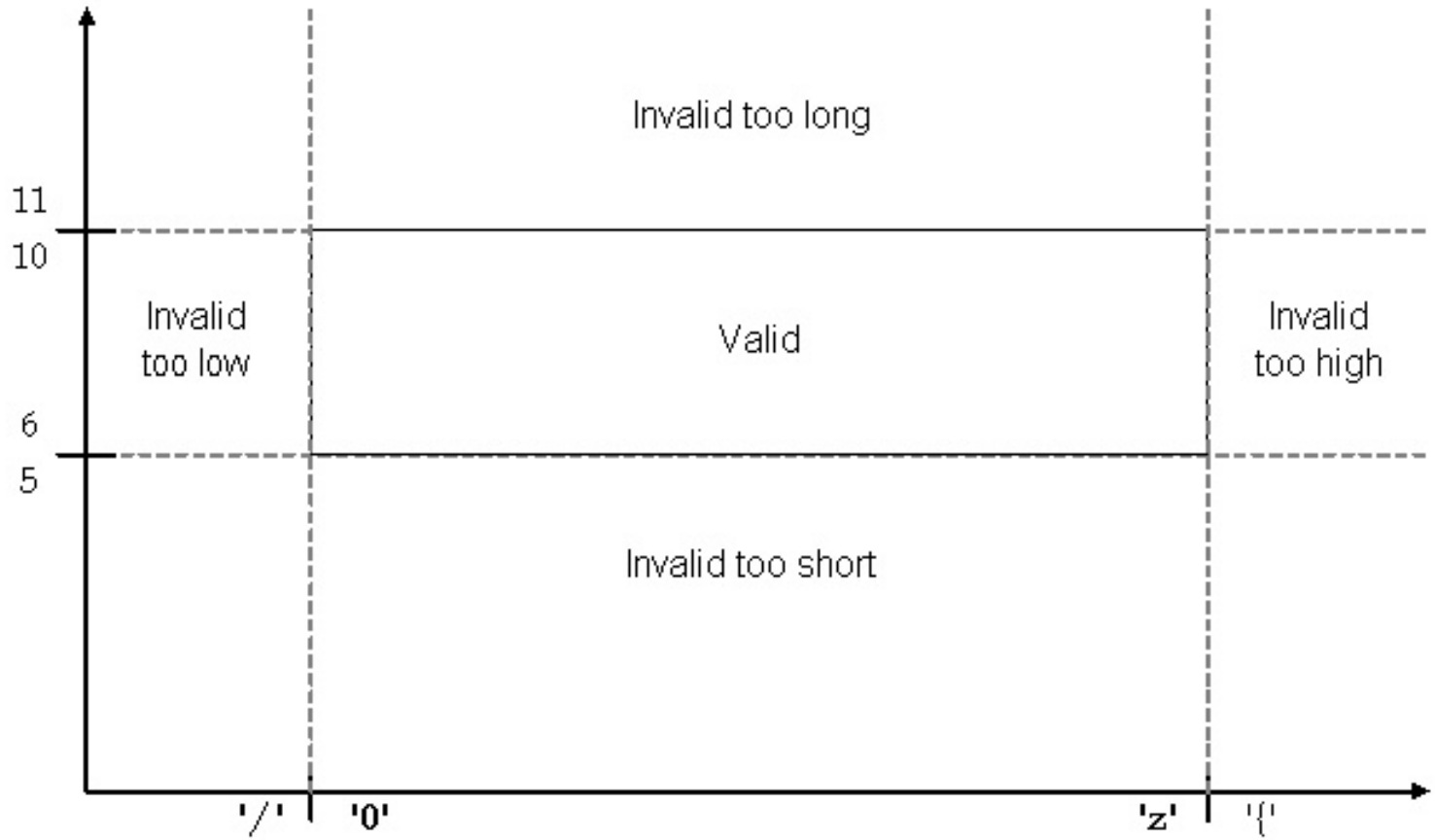
Ví dụ: Số thực

- "Nhiệt độ trung bình là bao nhiêu ?"



Ví dụ: Ký tự - Chuỗi

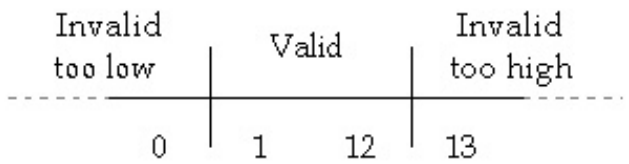
- "Mật khẩu (6-10 ký tự chữ)"

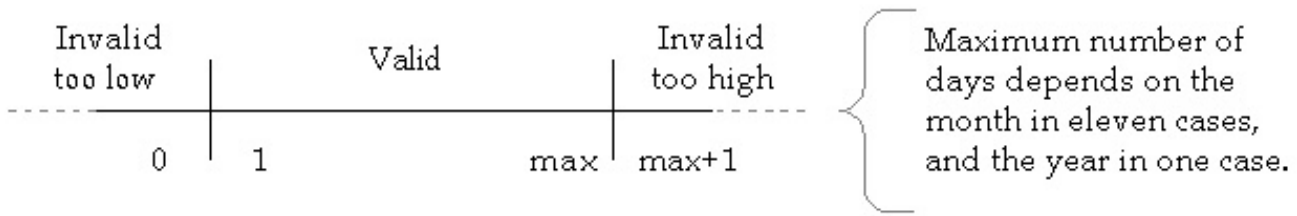


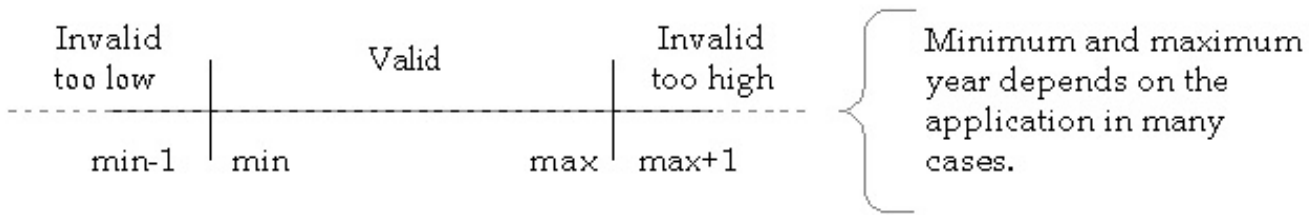
Ví dụ: Ngày tháng

- "Nhập ngày khởi hành cho chuyến bay ?"

MM/DD/YY

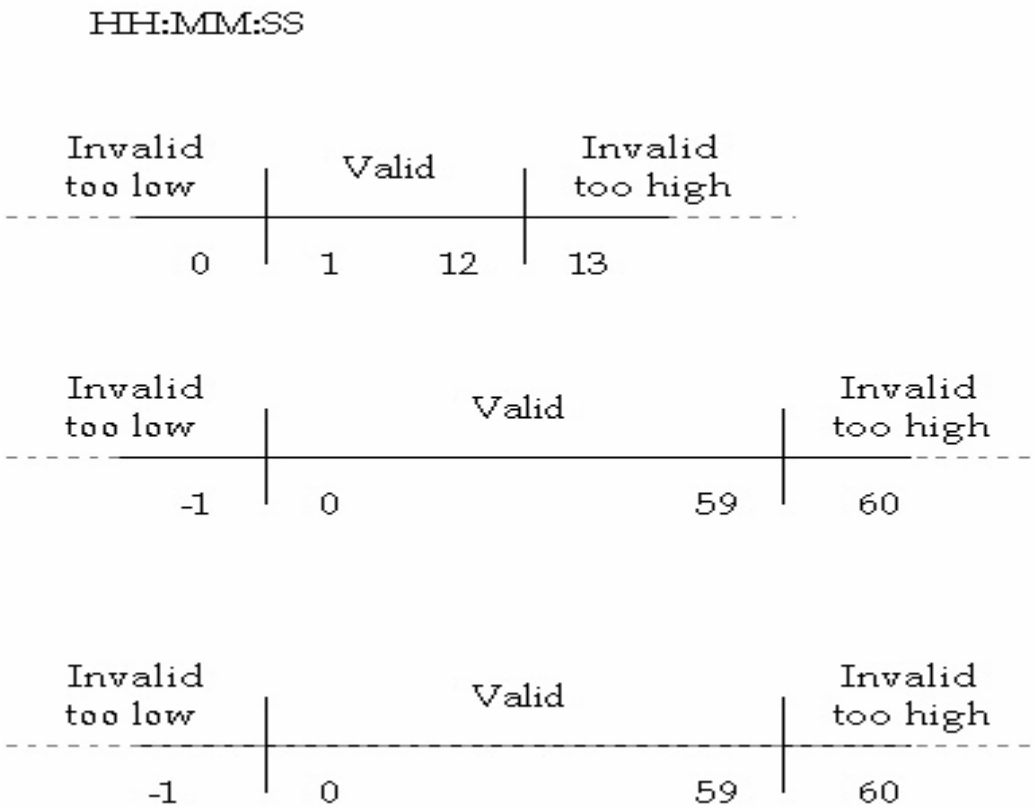






Ví dụ: Thời gian

- "Nhập thời gian khởi hành cho chuyến bay ?"



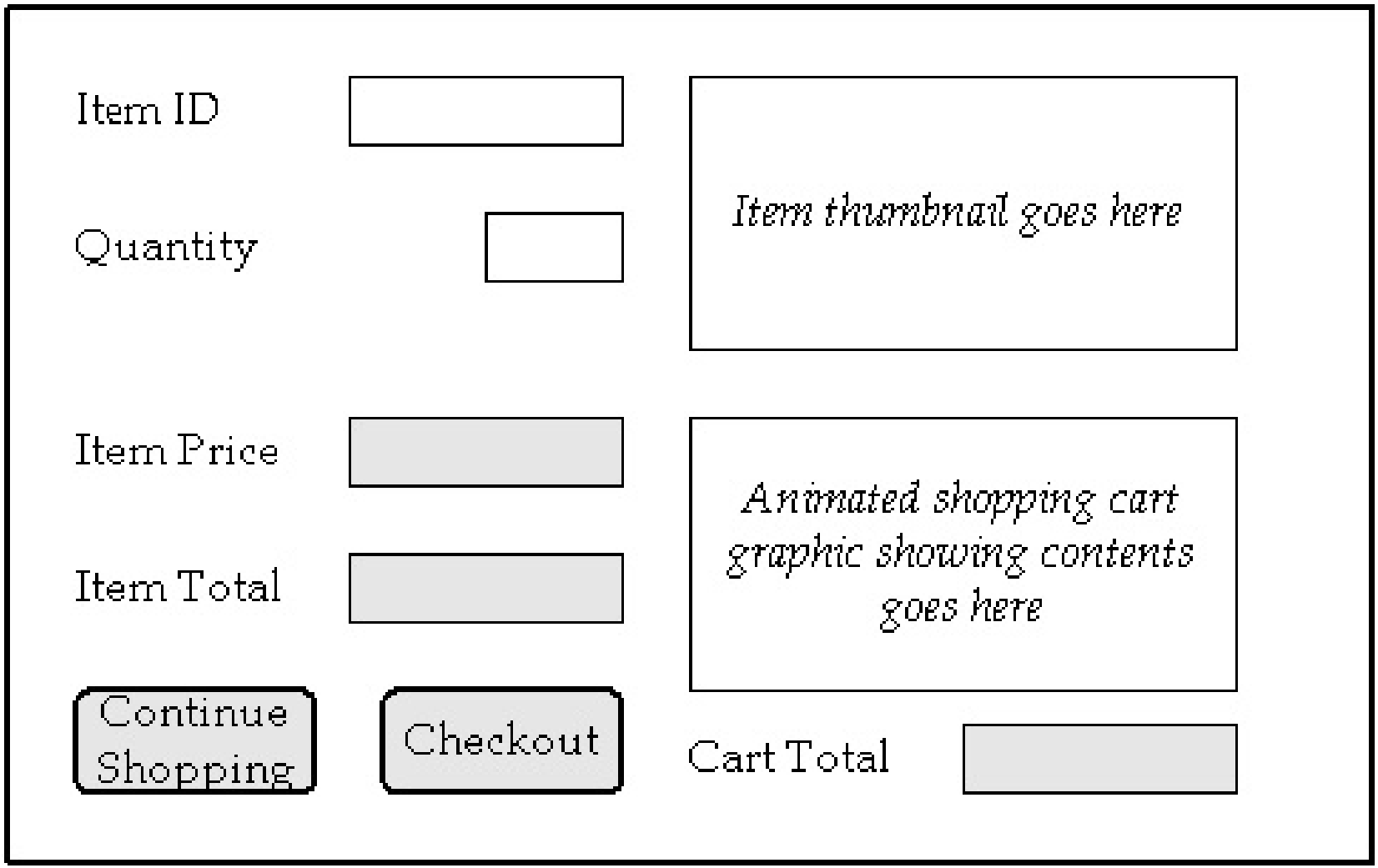
Ví dụ: Tiền tệ

- "Nhập giá mua (dưới $1000):"

Bài tập: Phân lớp - biên

- Kiểm tra web thương mại bán đồ lặt vặt: banh, áo, …
- Tạo kiểm cha chức năng cho việc hấp nhận đơn hàng:
  - Hệ thống chấp nhận mã số ID của hàng hóa là số 5 chữ số từ 00000 tới 99999
  - ID được sắp xếp theo giá, từ rẻ nhất có mã gần 00000 tới mắc nhất có mã 99999
  - Hệ thống chấp nhận số lượng đặt hàng từ 1 tới 99
  - Nếu người sử dụng nhập mã trước , nhập số lượng là 0, sẽ bị loại sản phẩm ra khỏi giỏ hàng
  - Tổng giá trị đơn hàng tối đa $999.99
- Sử dụng phương pháp để tạo kiểm thử theo mẫu sau

Bản mẫu màn hình



Mẫu kiểm thử

Test Number	1	2	…
Inputs, Actions			
Item ID			
Quantity			
?			
?			
Expected Results			
Item Price			
Cart Contents			
Cart Total			
?			
?			
?			

Kiểm thử theo kịch bản
(Use Cases/Scenario)

- Thiết kế các trường hợp khác nhau để phản ánh
- VD:

  đơn cho vay cầm cố nhà
  - "John - Jenny Stevens có 3 đứa trẻ…"
  - "…Ngôi nhà đáng giá $400K , họ sở hữu $350K…"
  - "… hai xe trị giá $25K , họ sở hữu $17K…"
  - "…Thu nhập $45K và 75K tính riêng…"
  - "… chậm trễ chi trả Visa card và cho xe đã 17 và 35 tháng …"
  - "…họ đệ đơn xin vay 15K."

Use Cases và giới hạn biên

- Xem lại các điều kiện biên, các điều kiện thích hợp ảnh hưởng điều kiện biên ?
- Nếu có biến đếm 4-bytes, có muốn cho phép đặt 32K sản phẩm vì phần mềm hỗ trợ được
- Cho phép ngày khởi hành sau ngày đến ?
- Chấp nhận các dữ liệu “xàm” là lỗi ?

Bảng quyết định

- Luật về nghiệp vụ thường đặc tả chính xác trong bảng quyết định
  - Quyết định xử lý đơn hàng như thế nào trên cơ sở kích cỡ, kho, trường hợp vận chuyển, … luôn có trong **luật** nghiệp vụ
  - Có thể thể hiện dạng lưu đồ hoặc bảng
  - Bảng quyết định có thể tạo ra kịch bản kiểm thử tức thì

Bảng quyết định ATM

Condition	1	2	3	4	5
Valid Card	N	Y	Y	Y	Y
Valid PIN	-	N	N	Y	Y
Invalid PIN=3	-	N	Y	N	N
Balance OK	-	-	-	N	Y
Action					
Reject Card	**Y**	N	N	N	N
Reenter PIN	N	**Y**	N	N	N
Keep Card	N	N	**Y**	N	N
Reenter Request	N	N	N	**Y**	N
Dispense Cash	N	N	N	N	**Y**

Mô tả nghiệp vụ cho một máy ATM. Chú ý "-" mô tả điều kiện không thể thực hiện. Các luật loại trừ tương hỗ, trong đó, một luật chỉ được áp dụng tại một thời điểm.

Chú ý, mức nghiệp vụ thường ở bên dưới lớp giao tiếp NSD do đó việc kiểm tra đúng đắn cần thực hiện

Bảng quyết định phức tạp hơn

- Một vài phòng cảnh sát có máy tính cầm tay đọc thẻ căn cước của người vi phạm tốc độ
- Bảng quyết định có thể thực hiện để kiểm tra hệ thống này
- Cũng có thể sử dụng phân tích giá trị biên để bao phủ tốt các luật

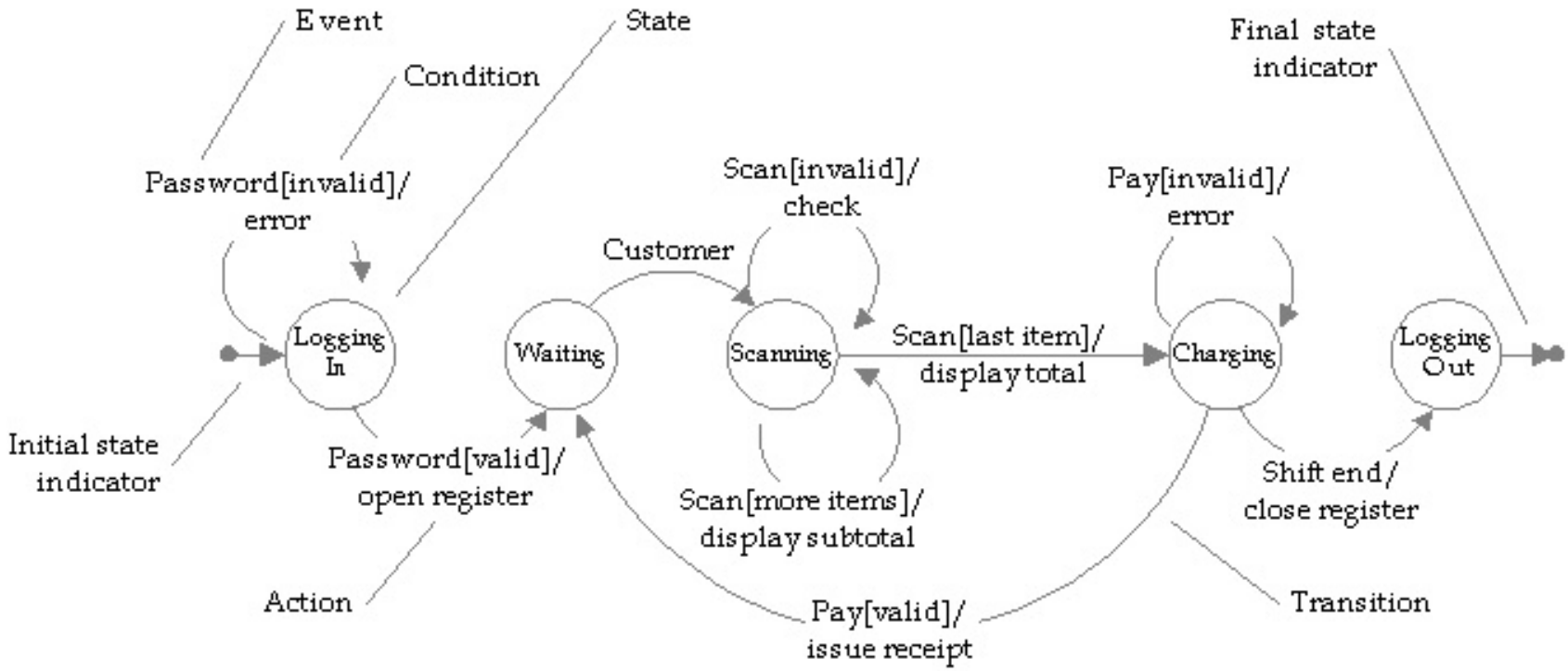
Bảng quyết định hệ thống CS

Condition	1	2	3	4	5	6	7	8
License OK	No	-	-	-	-	-	-	-
Warrant	-	Yes	-	-	-	-	-	-
Registration OK	-	-	No	-	-	-	-	-
Vehicle OK	-	-	-	No	-	-	-	-
Excess Speed	-	-	-	-	1-10	11-20	21-25	>25
Action								
Arrest	Yes	Yes	-	-	-	-	-	Yes
Fix-It Ticket	-	-	Yes	Yes	-	-	-	-
Warning	-	-	-	-	Yes	-	-	-
Fine	-250	-250	-25	-25	+0	+75	+150	+250

Các Mô hình trạng thái hữu hạn

- Hiểu được các trạng thái khác nhau của hệ thống bao gồm cả việc khởi tạo và kết thúc
- Nhận diện các giao dịch, sự kiện, điều kiện, hoạt động trong mỗi trạng thái
- Sử dụng đồ thị hay bảng để mô hình hệ thống
- Cho mỗi sự kiện và điều kiện, kiểm chứng lại hoạt động và trạng thái sau đó

Sơ đồ chuyển trạng thái



Sơ đồ trạng thái hệ thống điểm bán hàng từ quan điểm một thâu ngân. Hãy xét theo quan điểm của khách hàng, hệ thống, tìm lỗi ?

Bảng chuyển trạng thái

Current State	Event[Condition]	Action	New State
Logging In	Password[invalid]	Error	Logging In
Logging In	Password[valid]	Open register	Waiting
Logging In	Customer	[Undefined]	[Undefined]
Logging In	Scan[any]	[Undefined]	[Undefined]
Logging In	Pay[any]	[Undefined]	[Undefined]
Logging In	Shift end	[Undefined]	[Undefined]

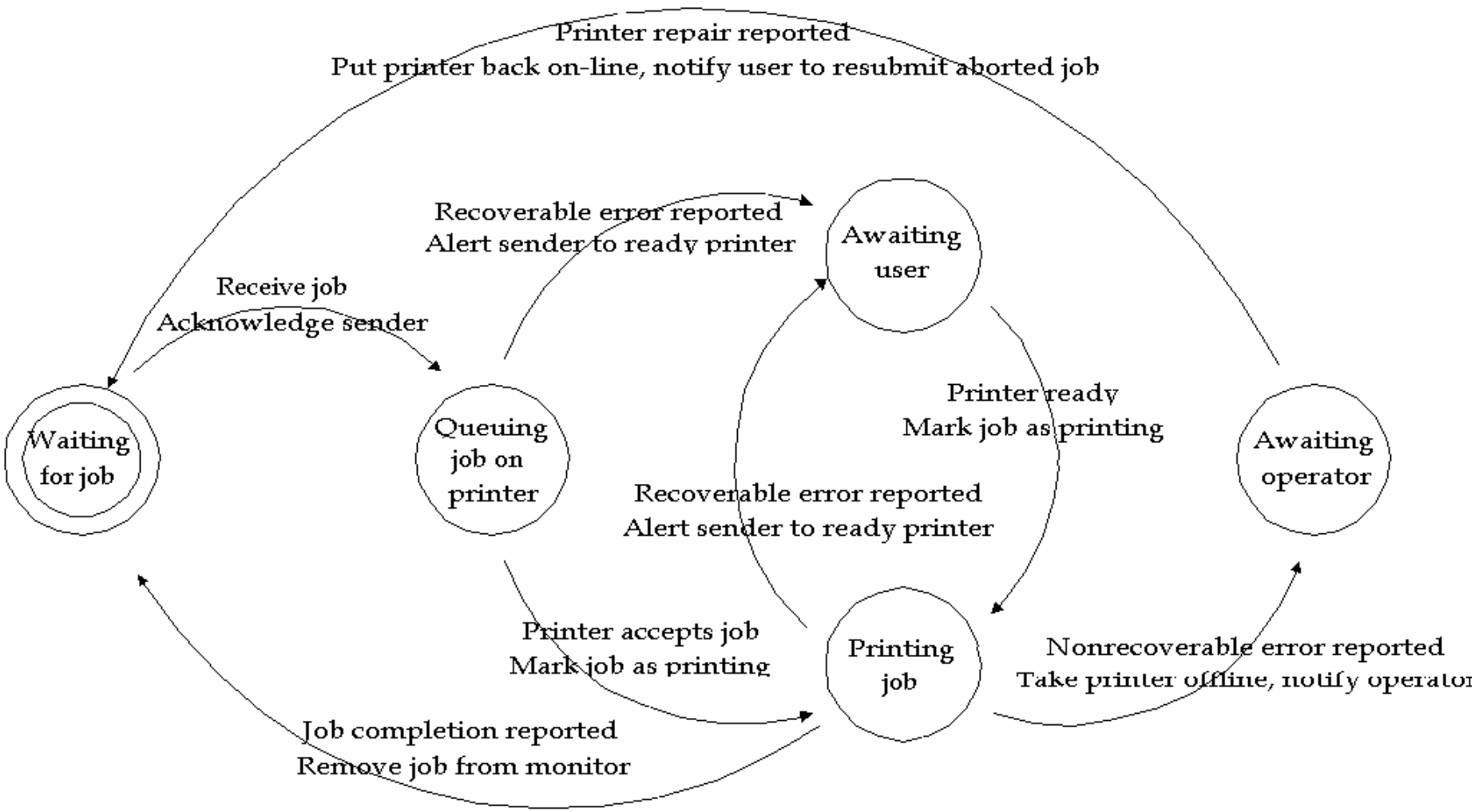
Bảng trạng thái có thể mô tả lại chuyển trạng thái phức tạp. Nó có thể dẫn tới những trường hợp không xác định …

Bài tập: Mô tả hoạt động của máy ATM dạng chuyển trạng thái

Các Mô hình trạng thái hữu hạn

- Xét thêm ví dụ sau
- Máy in chủ có các trường hợp sau:
  - Chờ
  - Lập hàng đợi
  - In
  - Chờ đợi NSD can thiệp
  - Chờ đợi bộ điều khiển can thiệp
- Máy chủ đáp ứng các sự kiện

Đồ thị máy trạng thái Máy in



Đồ thị Nhân quả
(Cause – Effect Graph)

- Đặc tính

Cause và *Effect*

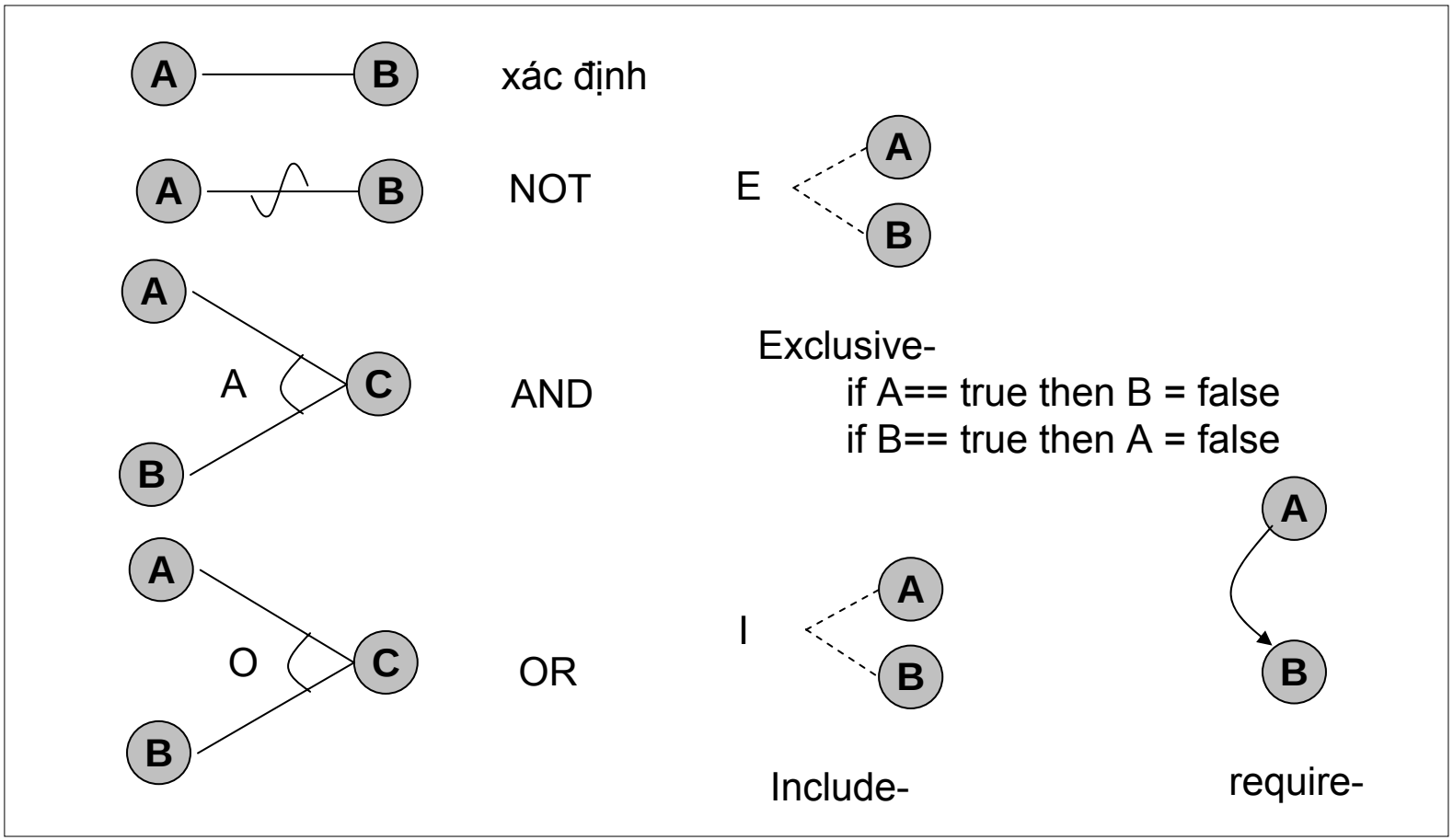
↓ ↓

inputs outputs

Trạng thái hiện tại Trạng thái mới

- Kỹ thuật này gồm có 4 bước như sau :
  - Xác định Cause (*điều kiện nhập vào*) và effect (*hành động – kết quả*) cho mỗi một module cần kiểm định.
  - Xây dựng đồ thị cause-effect:
  - Đồ thị được chuyển thành bảng quyết định
  - Những phần/luật trong bảng quyết định được chuyển thành các trường hợp kiểm thử.

Đồ thị Nhân quả
(Cause – Effect Graph)

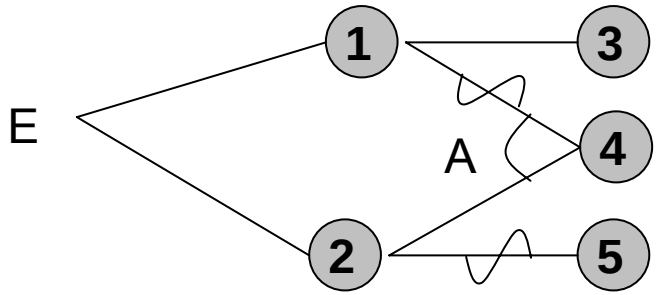


Đồ thị Nhân quả: ví dụ
(Cause – Effect Graph)

- Mô tả của một module:
  - Sồ lượng vấn đề cần kiểm tra là 5.
  - *Kết quả*
    - "Pass" - nếu kết quả của 4 hay nhiều hơn 4 chủ đề là thành công.
    - "Temporary pass" - nếu kết quả của 2 hay 3 chủ đề là thành công
    - "Failure" - nếu kết quả có duy nhất 1 chủ đề là thành công

Cause(Dữ Liệu Nhập)		Result (Dữ Liệu Xuất)
1. Nếu kết quả của 4 hay nhiều hơn 4 chủ để là thành công 2. Nếu kết quả của 2 hay 3 chủ để là thành công		4. Pass 5. Temporary Pass 6. Failure

Đồ thị Nhân quả: ví dụ
(Cause – Effect Graph)



Cause & Result		T1	T2	T3
Cause	1. Pass 4 hay nhiều hơn 4 chủ đề	Y	N	N
	2. "Pass" 2 hay nhiêu hơn 2 chủ đề	-	Y	N
Result	3. "Pass" là kết quả xuất ra	X	-	-
	4. "Temporary Pass" là kết quả xuất ra	-	X	-
	5. "Failure" là kết quả xuất ra	-	-	X

Kỹ thuật dựa trên cấu trúc

- Khái niệm chính
  - Bao phủ mã (*Code coverage*)
  - Bao phủ phát biểu-điều kiện (*Statement and decision coverage*)
  - Kỹ thuật thiết kế kiểm thử luồng điều khiển

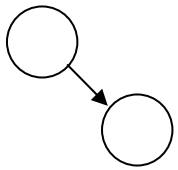
Cơ sở hộp trắng

- Hệ thống làm việc bên trong ra sao ?
  - Xác định và đạt được mức bao phủ các luồng điều khiển trên cơ sở phân tích mã
  - Xác định và đạt được mức bao phủ các luồng dữ liệu trên cơ sở phân tích mã và dữ liệu
  - Xác định và đạt được mức bao phủ các luồng giao tiếp, lớp luồng gọi thực thi trên cơ sở phân tích API, thết kế hệ thống.
- Bao phủ cấu trúc là cách kiểm tra "khoảng cách" giữa kiểm thử đặc tả và theo kinh nghiệm
  - kiểm nghiệm một chương trình (*một phần chương trình, hay một hệ thống, một phần của hệ thống*) đáp ứng tốt tất cả các giá trị input bao gồm cả các giá trị không đúng hay không theo dự định của chương trình

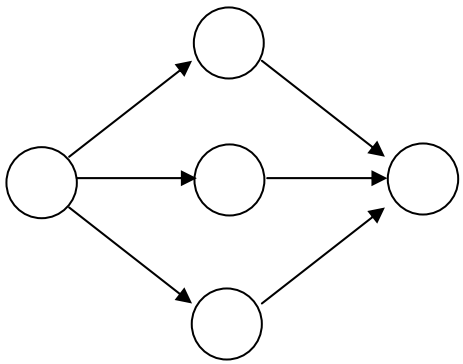
Bao phủ mã

- Mức bao phủ
  - Bao phủ phát biểu
    - Mỗi phát biểu đều được thực thi
  - Bao phủ nhánh
    - Mỗi nhánh thực hiện theo mỗi cách (T/F)
  - Bao phủ điều kiện
    - Mỗi tổ hợp các điều kiện được đánh giá – bảng sư thật
  - Bao phủ quyết định đa điều kiện
    - Chỉ các điều kiện tổ hợp ảnh hưởng quyết định
  - Bao phủ lặp (*Loop coverage*):
    - Tất cả các đường dẫn lặp 0,1, nhiều lần (*lý tưởng, cực đại*)

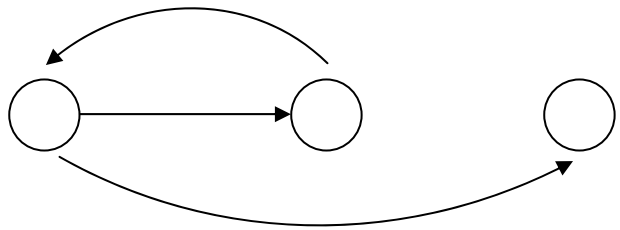
Bao phủ mã: lược đồ mô tả



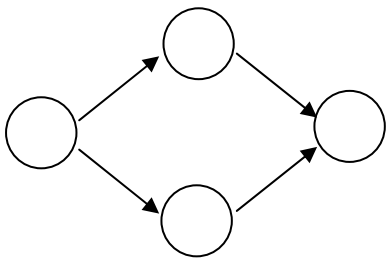
Tuần tự
(sequence)



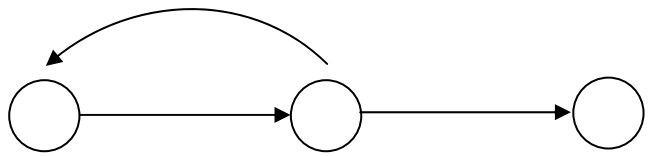
CASE



WHILE

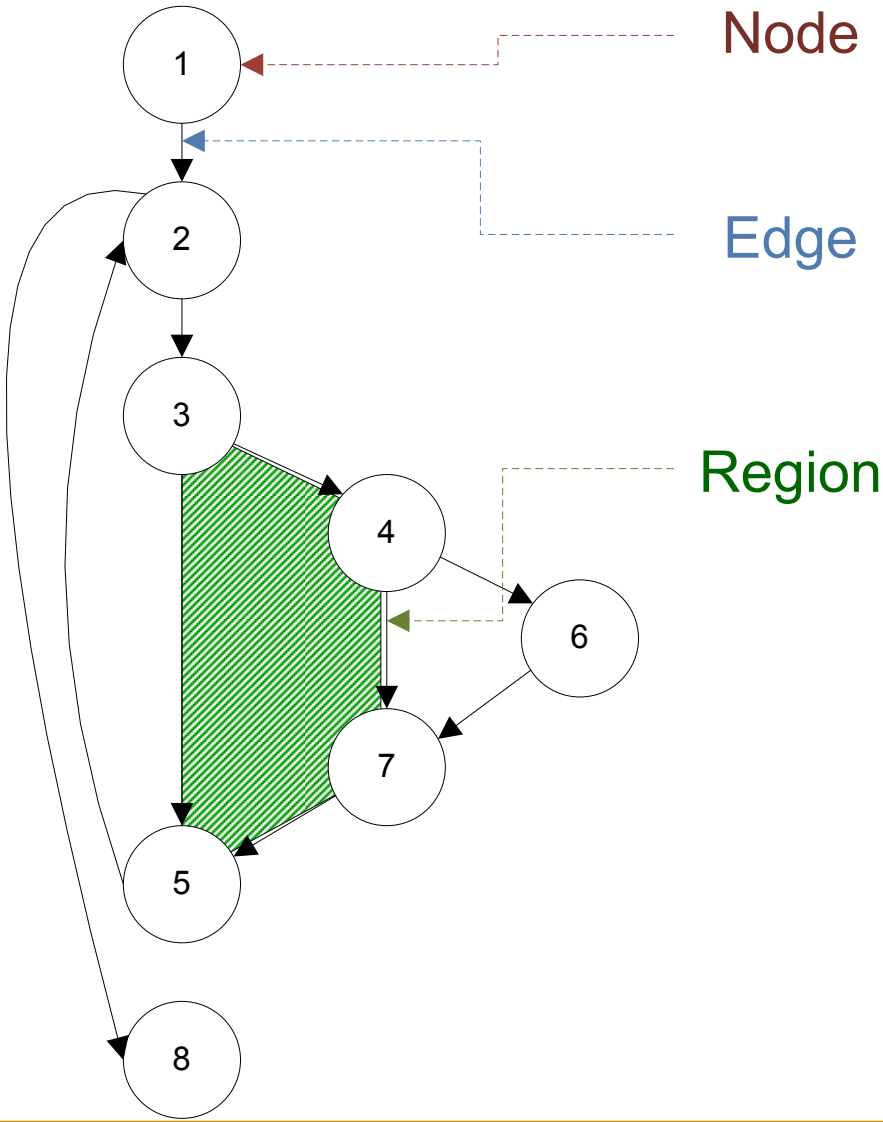


IF



UNTIL

Bao phủ mã: lược đồ mô tả



Ví dụ: bao phủ mã

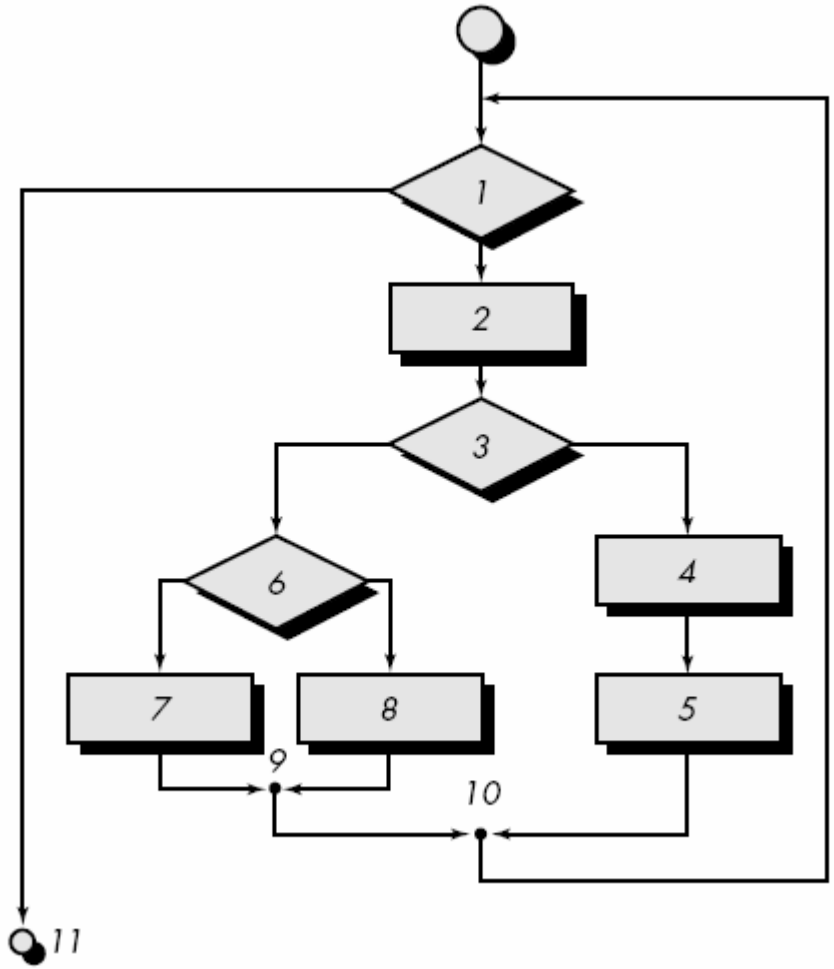
- Giá trị nào của n để thực thi hết các phát biểu ?
  - n < 0, n > 0
- Bao phủ nhánh ?
  - n=0
- Bao phủ điều kiện ?
  - Không có điều kiện tổ hợp
- Bao phủ đường ?
  - Cần xác định n = 1 và n = max

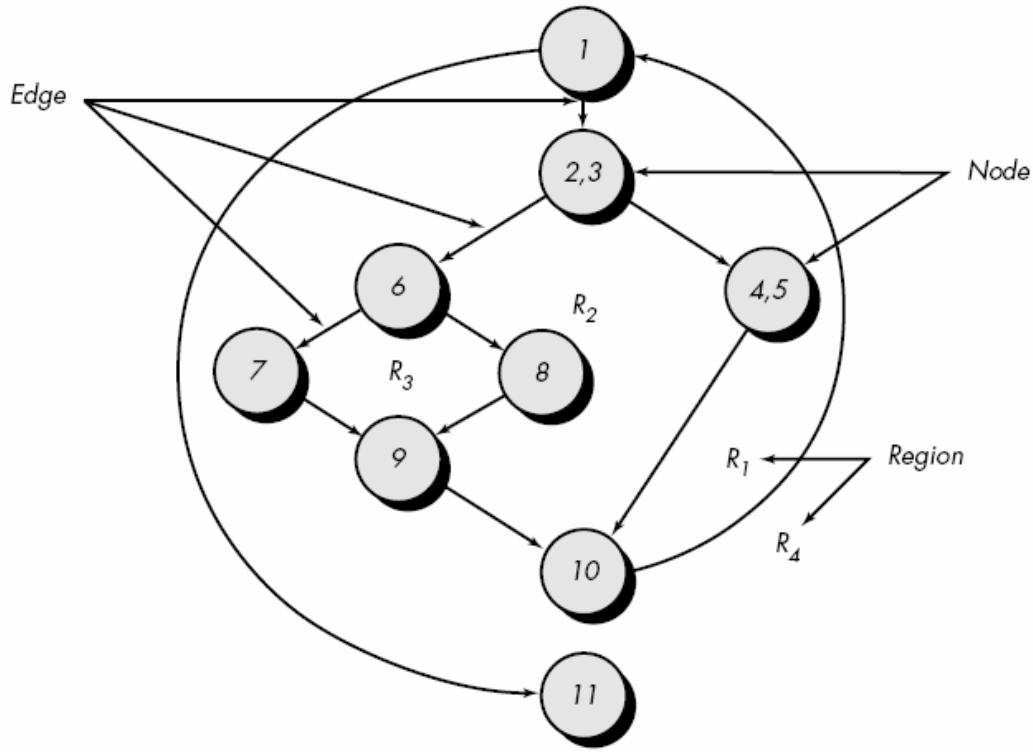
```
 1 #include <stdio.h>
 2 main()
 3 {
 4   int i, n, f;
 5   printf("n = ");
 6   scanf("%d", &n);
 7   if (n < 0) {
 8     printf("Invalid: %d\n", n);
 9     n = -1;
10   } else {
11     f = 1;
12     for (i = 1; i <= n; i++) {
13       f *= i;
14     }
15     printf("%d! = %d\n", n, f);
16   }
17   return n;
18 }
```

Bao phủ mã –
công cụ thiết kế kiểm thử

- Kỹ thuật hộp đen có thể bỏ qua 75% hay hơn nữa các phát biểu không bao phủ
  - Vấn đề ?
  - Phụ thuộc vào không bao phủ cái gì !
- Các công cụ bao phủ mã cung cấp chương trình theo dõi bao phủ mã thông qua kiểm thử
- Khoảng cách trong bao phủ mã có thể dẫn tới nhiều kịch bản kiểm thử để đạt được mức bao phủ cao hơn

Đường cơ sở





Đường cơ sở

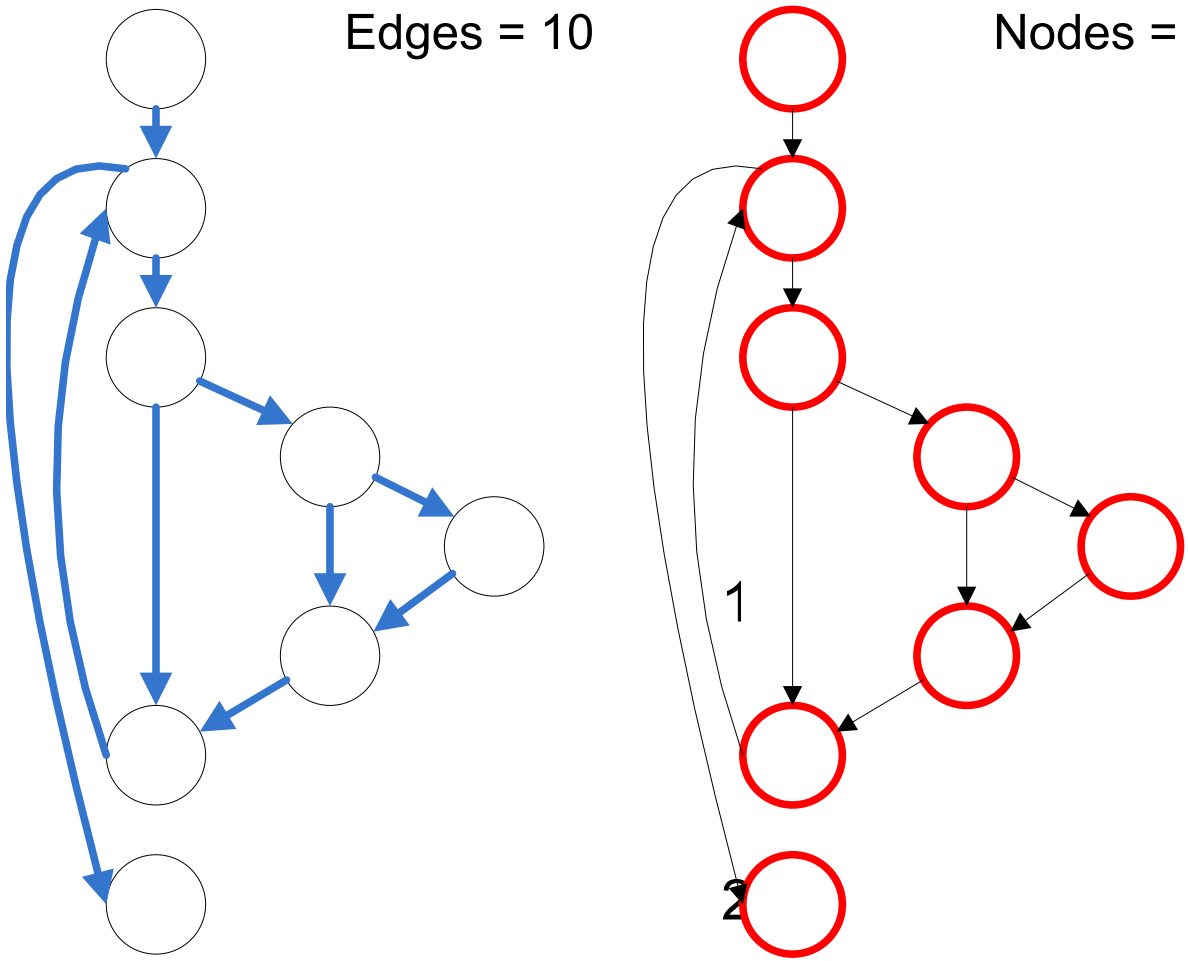
- Các đường cơ sở
  - path 1: 1-11
  - path 2: 1-2-3-4-5-10-1-11
  - path 3: 1-2-3-6-8-9-10-1-11
  - path 4: 1-2-3-6-7-9-10-1-11
- Số đường cơ sở

V(G) = E -N + 2 *V(G) = 11 edges 9 nodes + 2 = 4*

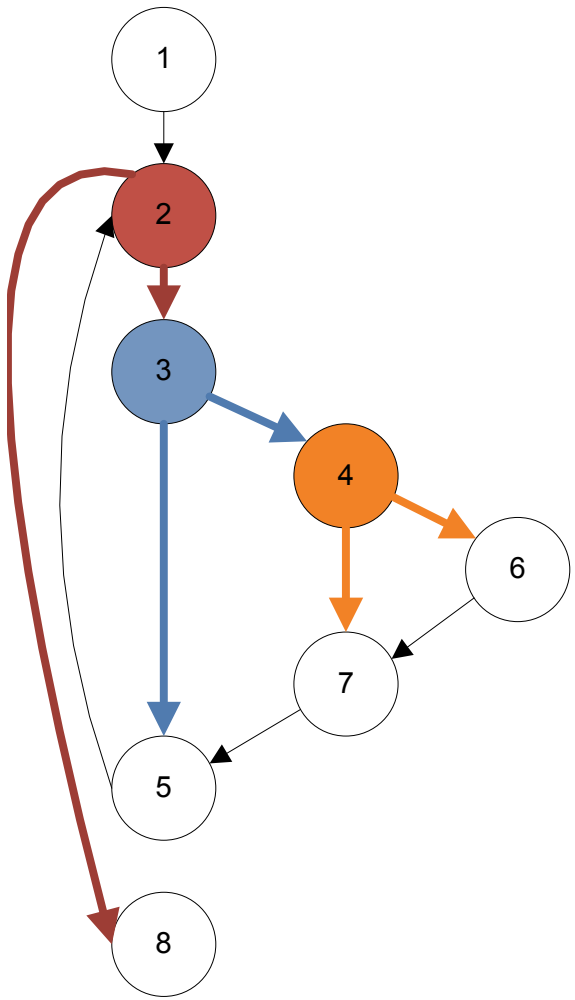
V(G) = P + 1 *V(G) = 3 predicate nodes + 1 = 4*

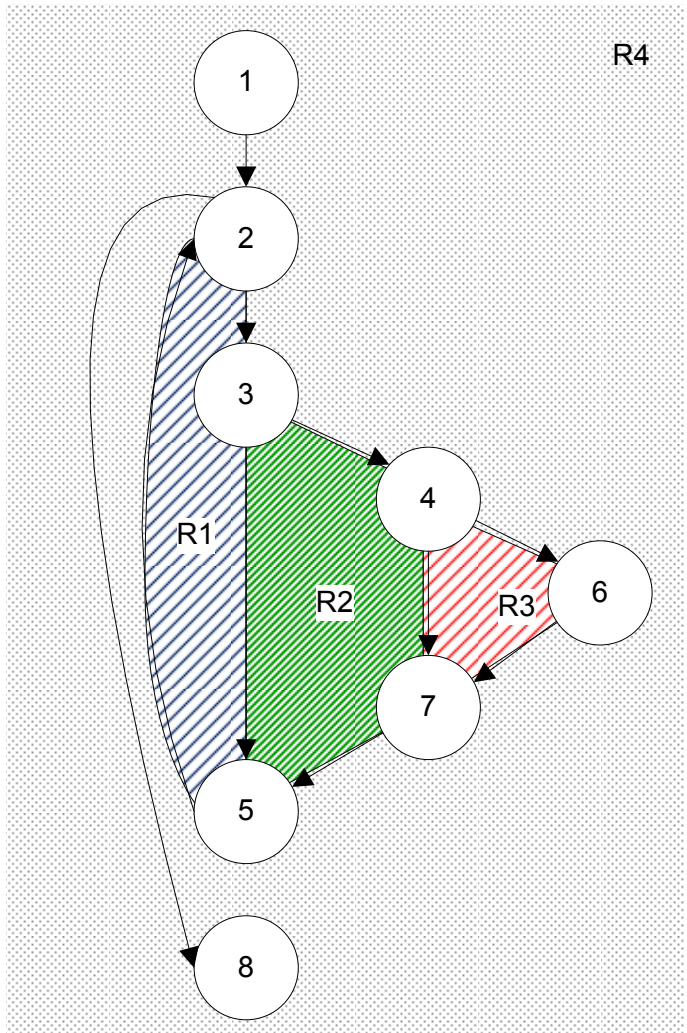
V(G) = R + 1(o)

Đường cơ sở



Đường cơ sở





Phức tạp vòng McCabe Cyclomatic

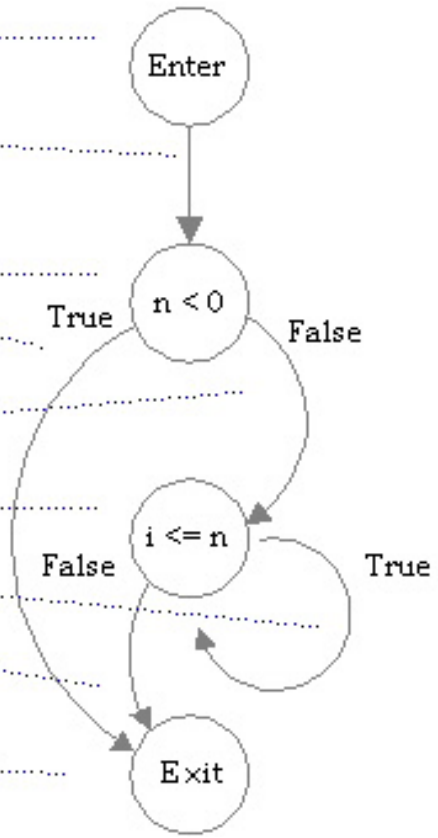
- Đánh giá độ phức tạp luồng điều khiển
  - Đánh giá bởi phác họa đồ thị trực tiếp
  - Nút mô tả vào, ra và điều kiện
  - Cạnh mô tả các phát biểu không nhánh
- Hữu ích trong kiểm thử
  - Các module có độ phức tạp cao vốn có nhiều "lỗi" và dễ hồi quy
  - Số đường dẫn cơ bản tương ứng số kiểm thử căn bản

Phức tạp Cyclomatic với Giai thừa

Program

```
main()
{
 int i, n, f;
 printf("n = ");
 scanf("%d", &n);
 if (n < 0) {
  printf("Invalid: %d\n", n);
  n = -1;
 } else {
  f = 1;
  for (i = 1; i <= n; i++) {
   f *= i;
  }
  printf("%d! = %d.\n", n, f);
 }
 return n;
}
```

Flow Diagram



Cyclomatic Complexity

C = #R +1 =
2 + 1 = 3

or

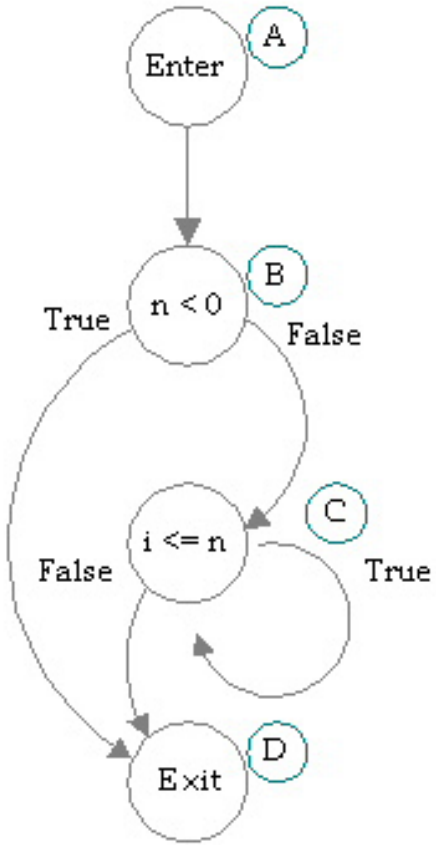
C = #E - #N +2 =
5 - 4 + 2 = 3

Đường cơ bản/cơ sở - kiểm thử

Program

```
main()
{
  int i, n, f;
  printf("n = ");
  scanf("%d", &n);
  if (n < 0) {
    printf("Invalid: %d\n", n);
    n = -1;
  } else {
    f = 1;
    for (i = 1; i <= n; i++) {
      f *= i;
    }
    printf("%d! = %d.\n", n, f);
  }
   return n;
}
```

Flow Diagram



Basis Paths

1. ABD
2. ABCD
3. ABCCD

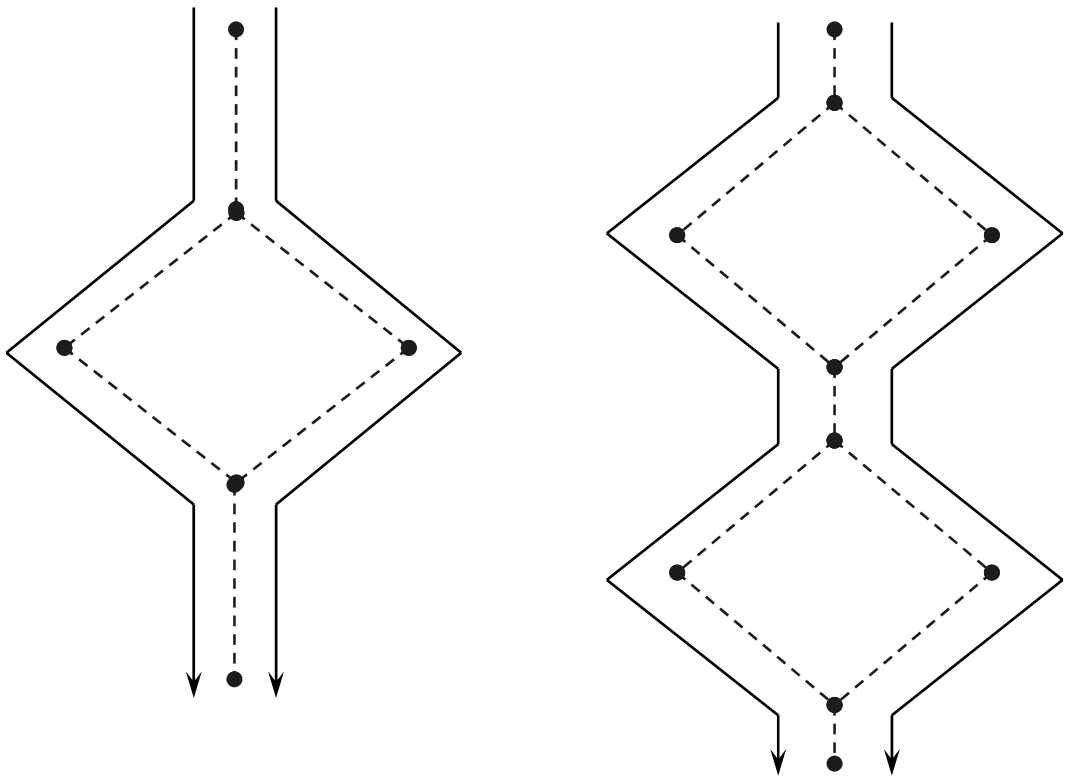
Basis Tests

	Input	Expect
1.	-1	Invalid: -1
2.	0	0! = 1.
3.	1	1! = 1.

Bao phủ câu lệnh

- Sao cho mỗi câu lệnh của chương trình được thực hiện ít nhất một lần.
  - không thể biết được có lỗi xảy ra trong câu lệnh đó hay không.
  - kiểm tra với một giá trị đầu vào không đảm bảo là sẽ đúng cho mọi trường hợp.

Bao phủ câu lệnh



Bao phủ câu lệnh

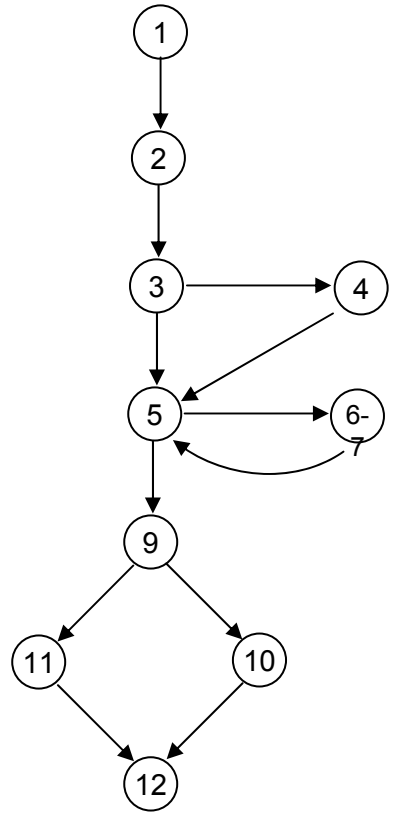
Ví dụ:

Đoạn chương trình thực hiện tính:

result = 0+1+…+|value|,

nếu result <= maxint, báo lỗi trong trường hợp ngược lại.

```
1    PROGRAM maxsum ( maxint, value : INT )
2            INT result := 0 ; i := 0 ;
3            IF value < 0
4            THEN value := - value ;
5            WHILE ( i < value ) AND ( result <= maxint )
6            DO      i := i + 1 ;
7                    result := result + i ;
8            OD;
9            IF result <= maxint
10           THEN OUTPUT ( result )
11           ELSE OUTPUT ( “too large” )
12   END.
```



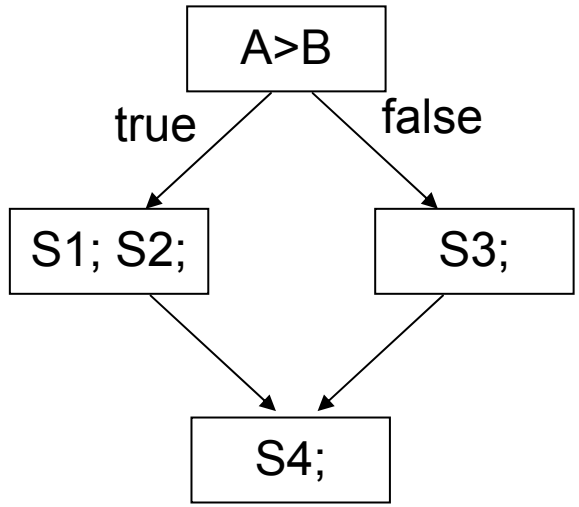
maxint = 10, value = -1

Hay

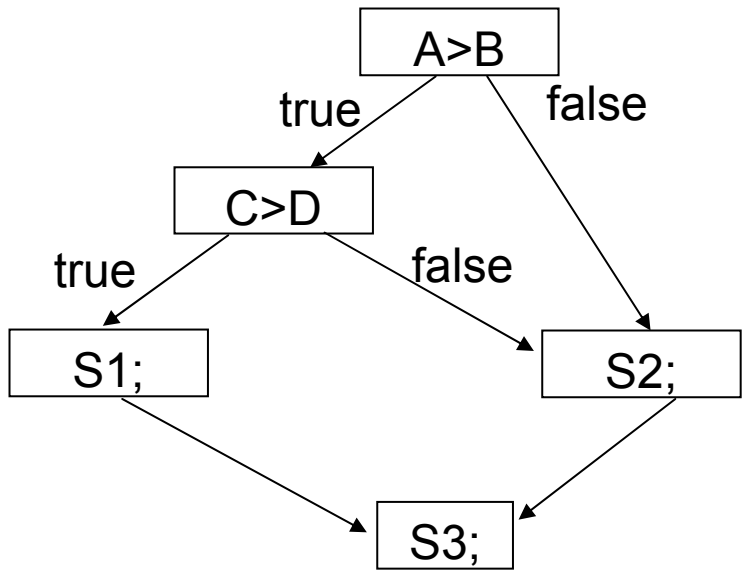
maxint = 0, value = -1

Bao phủ đường

```
if ( A > B)
        S1;
        S2;
else
        S3;
        S4;
```




```
if (A<B && C<D)
        S1;
else
        S2;
        S3;
```



Bao phủ điều kiện

- phương pháp kiểm tra các biểu thức điều kiện trên 2 giá trị true và false

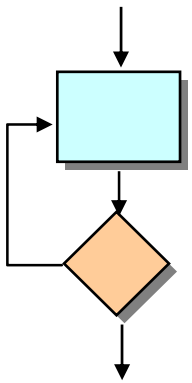
```
if (x > 0 && y > 0)
        x = 1;
else
        x = 2;
```

{ (x>0, y>0), (x <=0, y>0) } !!

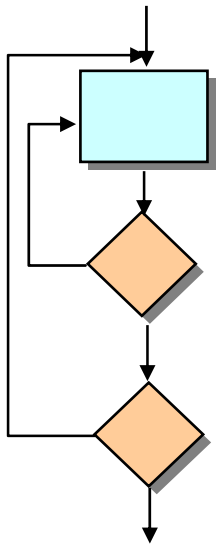
```
while (x > 0 || y > 0)
{
        x--; y--;
        z += x*y;
}
```

{ (x>0) } !!!

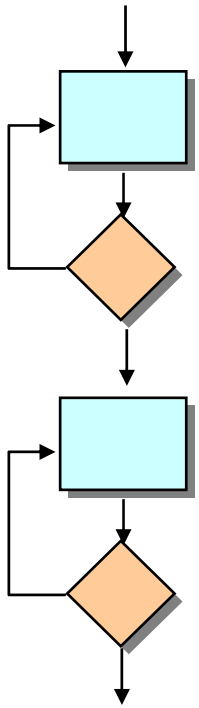
Bao phủ vòng lặp



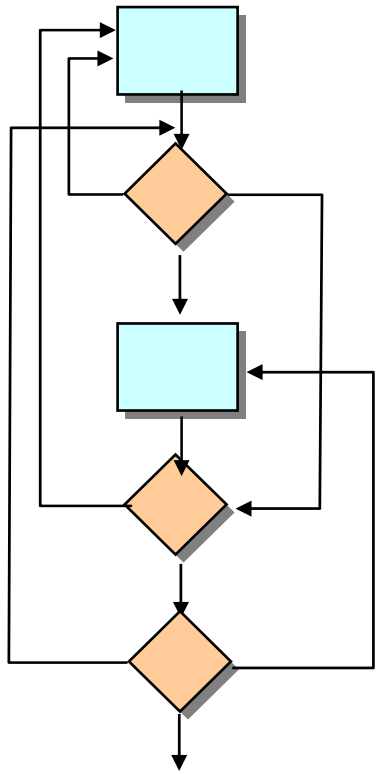
Vòng lặp
đơn giản



Vòng lặp
lồng nhau



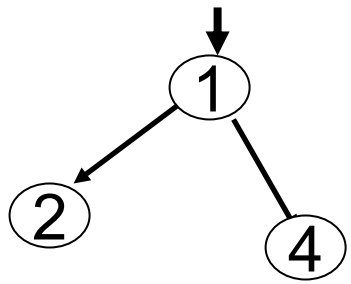
Vòng lặp
nối tiếp nhau



Bao phủ nhánh – điều kiện

- Ví dụ

```
if (x<1 && y>1)
      x = x + y;
else
      y = y - x;
```

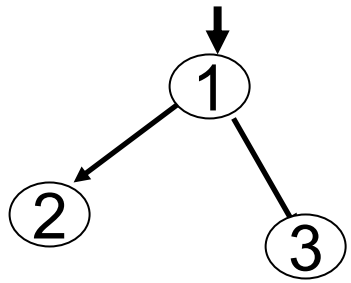


- 100% bao phủ nhánh
  - (x=0,y=2), (x=1,y=2) [TT,FT]

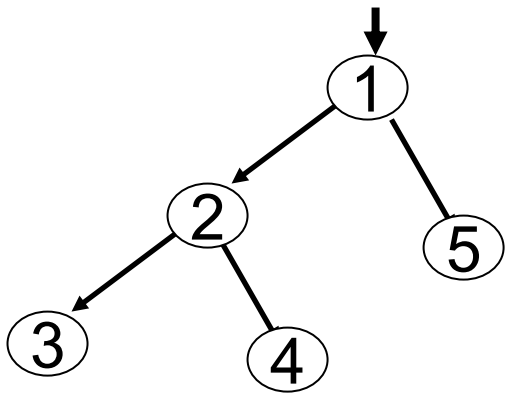
- Nhưng không bao phủ hết điều kiện
  - Cần trường hợp TF (x=0,y=1)

Vấn đề trong tổ hợp điều kiện

- Tạo ra điều kiện phức tạp hơn là nó biểu hiện
- Có thể bỏ xót trường hợp
- Ví dụ
  - Thực chất là 3 đcs, thay vì 2



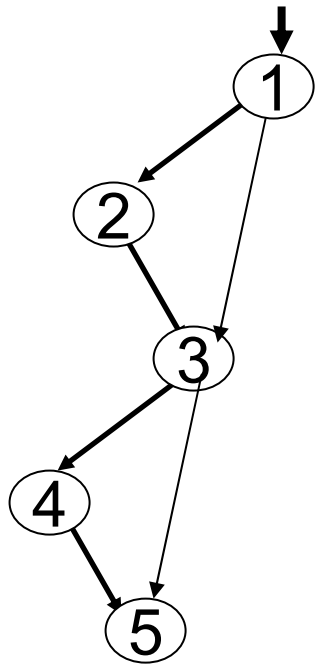

```
if (x<1){
  if (y>1)
      x=x+y;
  else
      y=y-x;
} else
  y=y-x;
```



Vấn đề trong bao phủ đường

- Không phải tất cả các đường có thể đi qua
- Các điều kiện có thể loại trừ nhau
- Ví dụ:
  - Không thể đi qua [1,2,3,4,5]

```
if (x<1)
   y=2;
if (x >= 1)
   y=3;
z=y;
```



Ví dụ:

```
if (Age>80 ||
          Weight>300)
   return 0;
if (Age <= 12)
   return 1;
if (Age > 65)
   return 2;
if (Weight < 120)
   return 2
else
   return 2+(Weight-120)/50;
```

Age>80	Y	N	N	N	N	N
Weight>300	-	Y	N	N	N	N
Age<=12			Y	N	N	N
Age>65				Y	N	N
Weight<120					Y	N
Pills =	0	0	1	2	2	C

Note: C: 2+(Weight/120)/50

McCabe = 6

Ví dụ:

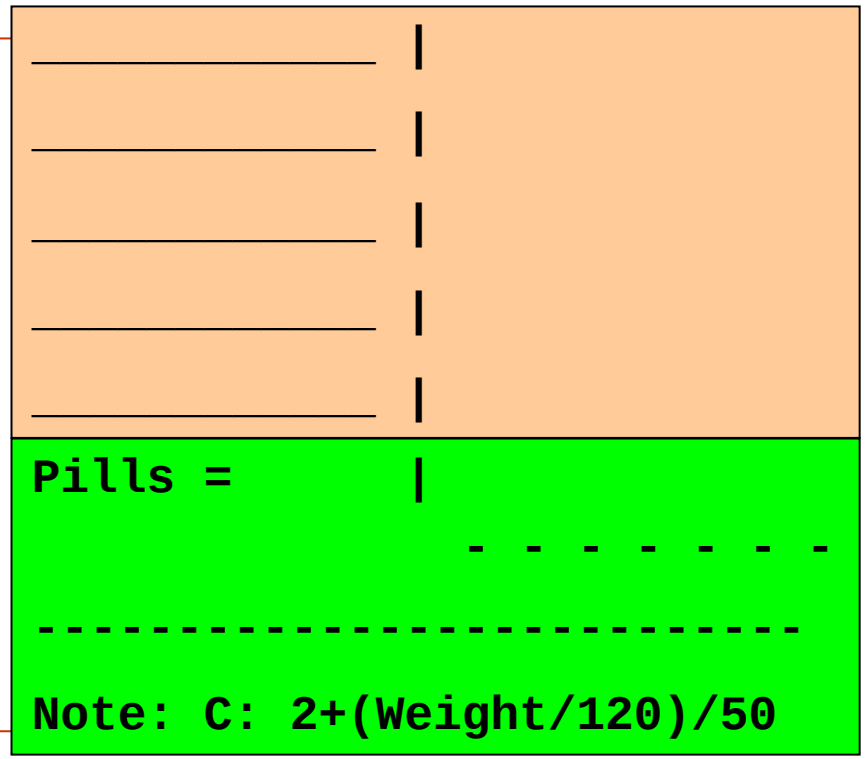
```
if (Age>80 ||
    Weight>300)
return 0;
if (Age <= 12)
   return 1;
if (Age > 65 ||
    (Age<=65 && Weight<120))
   return 2;
return 2+(Weight-120)/50;
```

Age>80	Y	N	N	N	N	N	N
Weight>300	-	Y	N	N	N	N	N
Age<=12			Y	N	N	N	
Age>65				Y	N	N	
Weight<120				-	Y	N	
Pills =	0	0	0	1	2	2	C

McCabe = 7

Ví dụ:

```
pills=0;
if (Age < 80 && Weight <300)
{
  pills=1;
  if (Age >= 65)
    pills=2;
  else if (Age > 12)
    pills=2+(Weight-120)/50;
}
return pills;
```



Bài tập: chuyển đổi số Hex

- Chấp nhận chuỗi ký tự thập lục phân (*giữa các ký tự không mong muốn*) . Nó bỏ qua các ký tự khác, chuyển Hex sang số
- Kiểm tra chuỗi dạng “059”, “ace”, “ACD” mức bao phủ đạt được là gì ?

Bài tập: chuyển đổi số Hex

```
main()
{
  int c;
  unsigned long int hexnum, nhex;
  hexnum = nhex = 0;
  while ((c = getchar()) != EOF) {
    switch (c) {
      case '0': case '1': case '2':
      case '3': case '4': case '5':
      case '6': case '7': case '8':
      case '9':
      /* Convert a decimal digit */
          nhex++;
          hexnum *= 0x10;
          hexnum += (c - '0');
          break;
      case 'a': case 'b': case 'c':
      case 'd': case 'e': case 'f':
      /* lower case hex digit */
          nhex++;
          hexnum *= 0x10;
          hexnum += (c - 'a' + 0xa);
          break;
    case 'A': case 'B': case 'C':
    case 'D': case 'E': case 'F':
    /*upper case hex digit */
          nhex++;
          hexnum *= 0x10;
          hexnum += (c - 'A' + 0xA);
          break;
    default:
    /* non-hex characters */
          break;
    }
}
printf("Got %d hex digits: %x\n",
                    nhex, hexnum);
return 0;
}
```

Kỹ thuật dựa trên kinh nghiệm

- Khái niệm chính
  - Lý do viết kịch bản kiểm thử dựa trên trực giác, kinh nghiệm và tri thức
  - So sánh với kỹ thuật dựa trên cấu trúc

Kỹ thuật dựa trên kinh nghiệm

- Dựa vào người kiểm thử với …
  - …kỹ năng – trực giác
  - …kinh nghiệm với ứng dụng tương tự
  - …kinh nghiệm với kỹ thuật tương tự
- Kiểm tử tiền thiết kế, thực nghiệm thường tạo ra trong suốt quá trình thực thi kiểm thử (*chiến lược kiểm thử động*)
- Kiểm thử thường giới hạn thời gian (*time-boxed*) (*nhấn mạnh vào các điều kiện đặc biệt*)
- Ví dụ: phán đóan, lùng kiếm lỗi, phá vỡ phần mềm (*breaking software*) trên cơ sở danh sách ý, phân loại lỗi, kiểm thử thăm dò

Cách tiếp cận thông thường

- Không tạo ra toàn bộ kiểm thử trong quá trình thực thi
- Có thể được hướng dẫn bởi:
  - Danh sách ý kiểm tra (*Checklist*)
  - Phân loại lỗi (*Bug taxonomy*)
  - Liệt kê các kiểu tấn công (*List of attacks*)
  - Cách tiếp cận "săn" lỗi (*Bug hunt approach*)
  - Tập các lý do kiểm thử
- Được chuẩn bị cải tiến, một mức chi tiết
- Thường là kiểm thử ad hoc (*on-the-fly testing - ad hoc testing*), kiểm tra ngẫu nhiên (*random testing*)

Chiến lược kiểm tra động

Thuận lợi

- Tìm lỗi hiệu quả
- Không mắc phải nghịch lý "thuốc trừ sâu" → biến đổi lớn
- Có hiệu quả
- Kiểm soát tốt việc chuẩn bị kiểm thử
- Sáng tạo

Bất lợi

- Lỗ hổng bao phủ, đặc biệt khi có áp lực
- Khó đánh giá
- Không ngăn ngừa lỗi
- Không thích hợp nhóm lớn
- Khác biệt về kỹ năng và kinh nghiệm

Trường hợp kiểm thử thăm dò

Staff	7 Technicians	3 Engineers + 1 Mgr.
Experience	<10 years total	> 20 years total
Test Type	Precise scripts	Chartered exploratory
Test Hrs/Day	42	6
Bugs Found	928 (78%)	261 (22%)
Effectiveness	22	44

Bài tập: Kiểm tra động – tiền thiết kế

Major factors or concerns	Dynamic	Pre-designed
Do thorough regression testing		
Maximize bug finding efficiency		
Use testers with limited experience		
Automate half of the test cases		
Deliver precise, accurate estimates		
Test without test preparation time		
Test a rapidly changing UI		
Use a distributed test effort		

Đánh dấu

+: phương pháp thích hợp

-: phương pháp không thích hợp

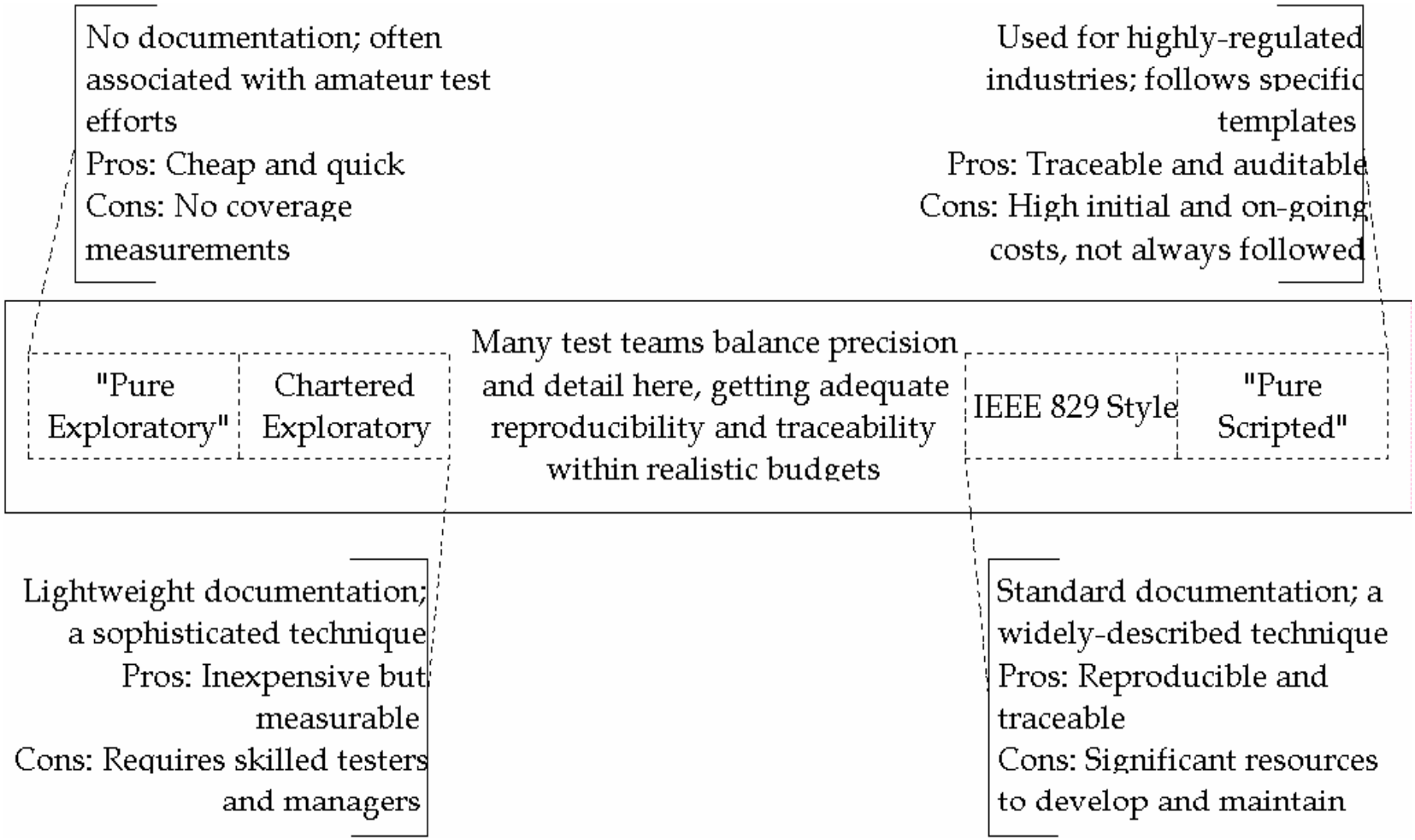
Chọn kỹ thuật kiểm thử

- Khái niệm chính
  - Các nhân tố ảnh hưởng tới việc chọn kỹ thuật thiết kế phù hợp

Chọn kỹ thuật kiểm thử

- Kiểu hệ thống
- Chuẩn quy tắc
- Yêu cầu khách hàng – hợp đồng
- Mức và kiểu rủi ro
- Mục tiêu kiểm thử
- Tài liệu sẵn sàng
- Kiến thức người kiểm thử
- Thời gian/ngân sách
- Vòng đời phát triển
- Kinh nghiệm – kiểu khiếm khuyết phát hiện
- …

Động – Tiền thiết kế



Kiểm thử Chi tiết và chính xác

- Việc cân bằng kiểm thử
  - Kiểm thử chính xác cho phép giảm người lành nghề
  - Kiểm thử không chính xác có thể bao phủ nhiều điều kiện
  - Kiểm thử chính xác cung cấp tiêu chuẩn minh bạch nhưng khó và bảo trì tốn kém
  - Kiểm thử không chính xác viết nhanh nhưng bao phủ khó định nghĩa và đánh giá

Bài tập - đọc hiểu

Đọc thêm

- [1]. Chapter 04
- [2]. Chapter 18
- [4]. Chapter 06, 05
- [4]. Chapter 07, 08
- [4]. Chapter 09, 10, 11

Q/A