KIỂM THỬ PHẦN MỀM

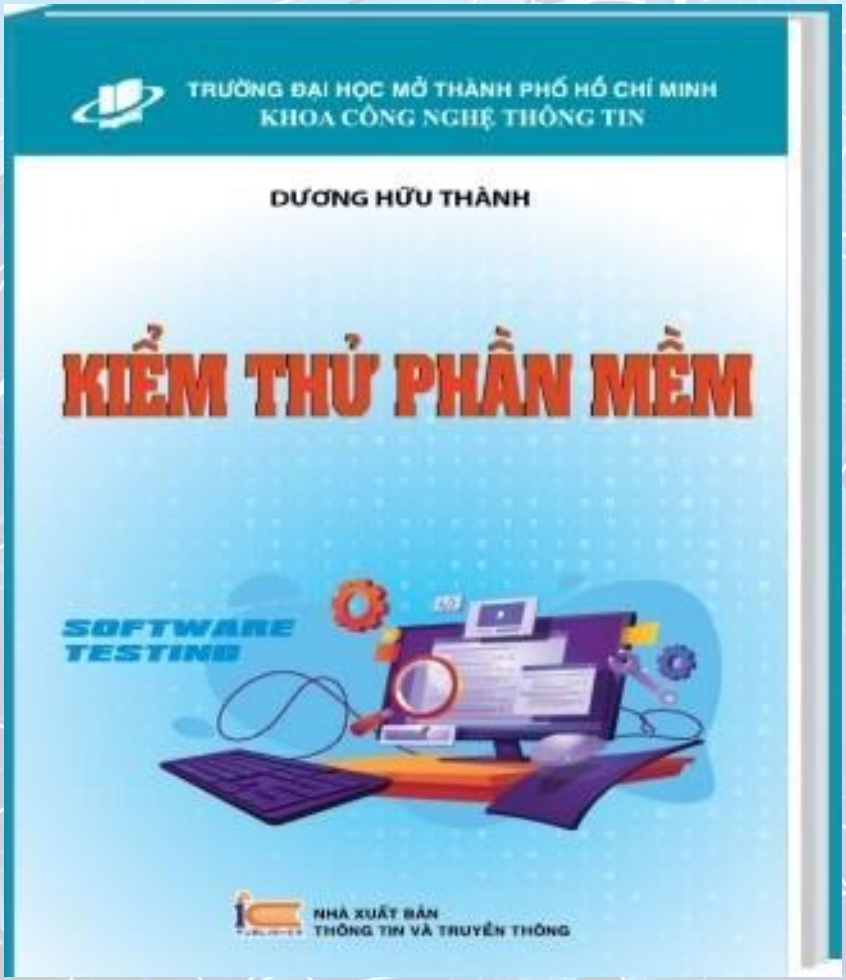
TỔNG QUAN KIỂM THỬ PHẦN MỀM

ThS. Nguyễn Thị Ngọc Thanh
Khoa CNTT, Đại học Mở Tp.HCM
thanh.ntn@ou.edu.vn

Nội dung chính

1. Quy trình phát triển phần mềm
2. Chất lượng phần mềm
3. Tổng quan kiểm thử phần mềm
4. Các vai trò kiểm thử phần mềm
5. Các cấp độ kiểm thử phần mềm
6. Kiểm thử tĩnh
7. Kiểm thử động
8. Các tài liệu kiểm thử phần mềm

Tài liệu tham khảo



http://sachin.ou.edu.vn/vi/shop/product/402-kiem-thu-phan-mem

Quy trình phát triển phần mềm

- Quy trình phát triển phần mềm là **các giai đoạn** mà quá trình phát triển phần mềm phải trải qua.
- Mỗi giai đoạn cần xác định rõ:
  - **Mục tiêu, kết quả** đạt được của giai đoạn trước.
  - **Kết quả chuyển giao** cho giai đoạn tiếp sau.

Quy trình phát triển phần mềm

- Tiếp cận quy trình phát triển phần mềm
  - Quy trình có kế hoạch (**plan-driven process**): là quy trình mà tất cả các hoạt động đã được **lên kế hoạch trước** và tiến độ được xác định thông qua kế hoạch này.
  - Quy trình phát triển nhanh (**agile process**): các kế hoạch **tăng trưởng dần** và quy trình này dễ dàng phản ứng với những thay đổi yêu cầu của khách hàng.
  - Trong thực tế, thường kết hợp hai cách tiếp cận này.

Quy trình thác nước

Quy trình lặp

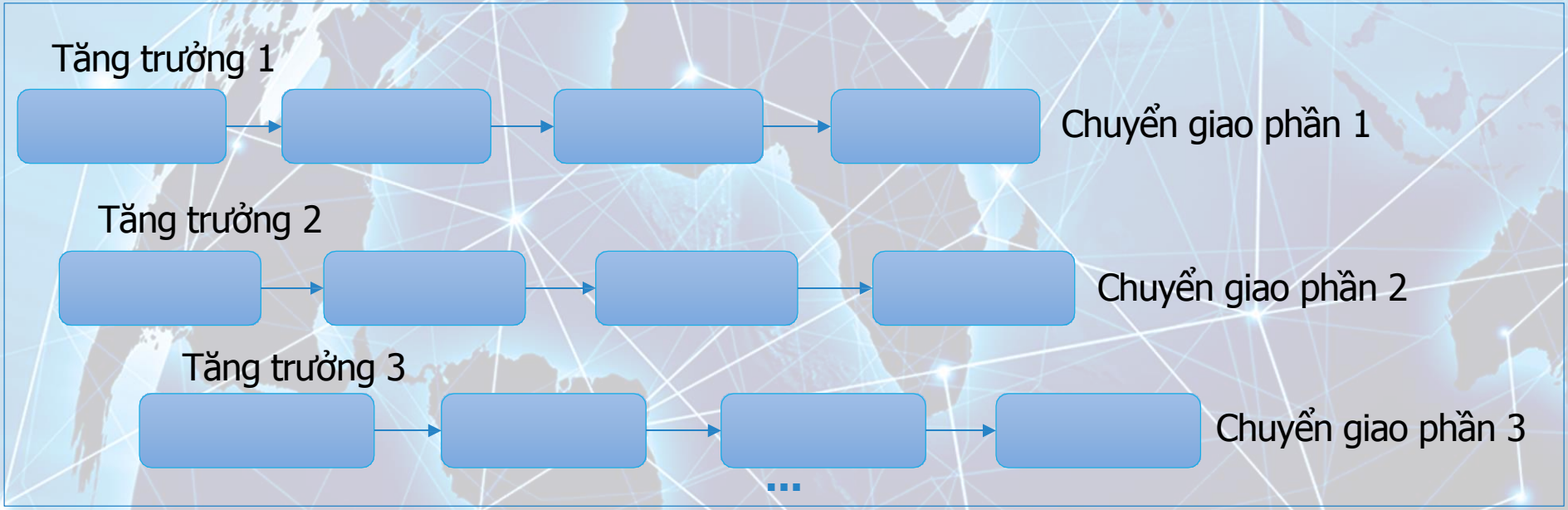
- Quy trình lặp (Iterative Process) phát triển phần mềm theo nhiều vòng lặp, mỗi vòng lặp sẽ cho ra một phiên bản của phần mềm.
- Kết thúc mỗi vòng lặp sẽ đánh giá, cải thiện để xác định yêu cầu cho phiên bản của vòng lặp tiếp theo. Quá trình này lặp lại cho đến khi sản phẩm phần mềm hoàn thành.

Quy trình lặp

- Đặc trưng quan trọng của quy trình này là có đại diện người dùng trong quá trình kiểm thử.
- Nhược điểm của quy trình này là thiếu các tài liệu chính thức khiến quá trình kiểm thử trở nên khó khăn. Theo quy trình này khi lập trình viên thực hiện các thay đổi theo yêu cầu khách hàng không cần ghi nhận lại một cách chính thức khiến khả năng truy vết (traceability) trong các tiến độ dự án bị giảm.

Quy trình tăng trưởng

- Chia hệ thống thành các thành phần nhỏ để phát triển, mỗi thành phần áp dụng quy trình thác nước hoặc một quy trình nào đó để tiến hành đến khi thành phần cuối cùng được hoàn tất thì quá trình phát triển toàn bộ hệ thống kết thúc.



Quy trình tăng trưởng

- Quy trình này linh hoạt, giúp phát triển phần mềm nhanh, giảm chi phí khi thay đổi phạm vi và yêu cầu phần mềm, dễ dàng kiểm thử và phát hiện vấn đề trong từng vòng lặp nhỏ hơn.
- Quy trình này có chi phí cao hơn quy trình thác nước, cần có kế hoạch và thiết kế đầy đủ trước khi chia nhỏ và phát triển tăng trưởng dần.

Quy trình Scrum

- Scrum là một quy trình phát triển phần mềm theo **mô hình linh hoạt** (agile).
- Scrum **chia dự án thì nhiều vòng lặp phát triển** gọi là **sprint**, một sprint thường kéo dài 2-4 tuần (30 ngày).
- Scrum phù hợp các dự án có **nhiều thay đổi** và **yêu cầu tốc độ cao**.

Quy trình Scrum

Khởi động dự án

Tạo ra các yêu cầu

Quyết định các chức năng, đánh giá độ ưu tiên từng chức năng (Product Backlog)

Thường là khách hàng hoặc đại diện khách hàng đảm nhận.

Product Owner

Giải quyết trở ngại, xung đột trong team

Đảm bảo Sprint hoàn thành đúng mục tiêu

Bảo vệ team

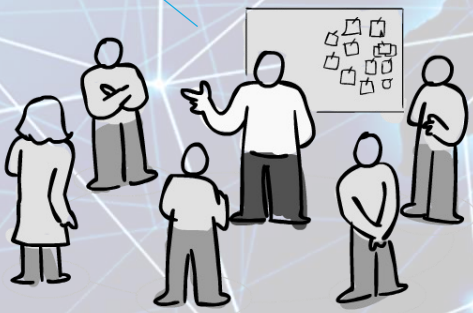
Scrum Master

Tự quản lý, tổ chức sao cho hiệu quả nhất

Cài đặt các chức năng theo bản yêu cầu

Team

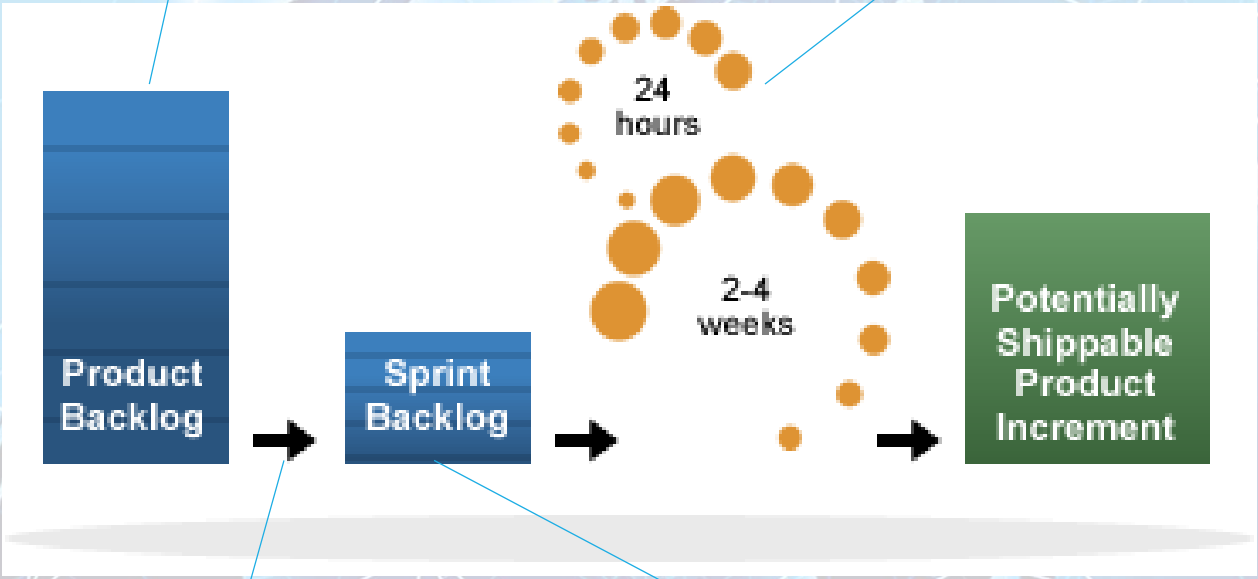
Các thành viên có vai trò như nhau



Quy trình Scrum

Danh sách các chức năng cần phát triển cho sản phẩm

Họp khoảng 15 phút (Daily meeting): hôm qua làm gì? Vấn đề phát sinh? Hôm nay sẽ làm gì?



Sprint Review: đánh giá kết quả Sprint, cái gì làm được, chưa làm được, tiếp tục phát triển? Và cập nhận trạng thái chức năng.

Họp (Sprint Planning) xác định các chức năng cho Sprint

Danh sách các chức năng (chọn từ Product Backlog) phát triển cho một Sprint

Câu hỏi 1

Quy trình phát triển phần mềm Scrum thích hợp với những dự án nào?

A. Có kế hoạch và yêu cầu rõ ràng.

B. Có nhiều thay đổi và yêu cầu tốc độ cao.

C. Dự án có quy mô vừa và nhỏ.

D. Dự án yêu cần hoàn tất trong thời gian ngắn.

Câu hỏi 2

Trong quy trình Scrum, ai là người quyết định thứ tự ưu tiên các nhiệm vụ trong product backlog?

A. Product Owner

B. Scrum

C. Project Manager

D. Technical Leader

Chất lượng phần mềm

- Chất lượng phần mềm chỉ định mức độ để phần mềm, một thành phần của phần mềm hoặc một quy trình đáp ứng được các yêu cầu chỉ định (IEEE).
- Để quyết định chúng có chất lượng cao hay không có thể đánh giá thông qua các thuộc tính chất lượng.

Chất lượng phần mềm

- Tính chính xác (Correctness)
- Tính tin cậy (Reliability)
- Tính khả dụng (Usabiliy)
- Tính toàn vẹn (Integrity)
- Tính khả chuyển (Portability)
- Tính bảo trì (Maintainability)
- Tính tương tác (Interoperability)

Tổng quan kiểm thử phần mềm

- Kiểm thử (testing) là quá trình đánh giá hệ thống hoặc các thành phần của hệ thống, nhằm kiểm tra nó **đúng với bản đặc tả yêu cầu** và **tìm ra lỗi** trước khi đưa vào sử dụng.
- Kiểm thử nghĩa là thực thi chương trình với **dữ liệu được tạo trước**.

Mục đích kiểm thử phần mềm

- Chứng minh nhóm phát triển phần mềm đã phát triển đúng với yêu cầu của khách hàng (**validation testing**).
- Phát hiện ra những trường hợp phần mềm còn lỗi, thiếu sót hoặc hoạt động không đúng (**defect testing**) với đặc tả của nó.

Tổng quan kiểm thử phần mềm

Ai là người thực hiện kiểm thử phần mềm?

Project leader/ manager

End User

...

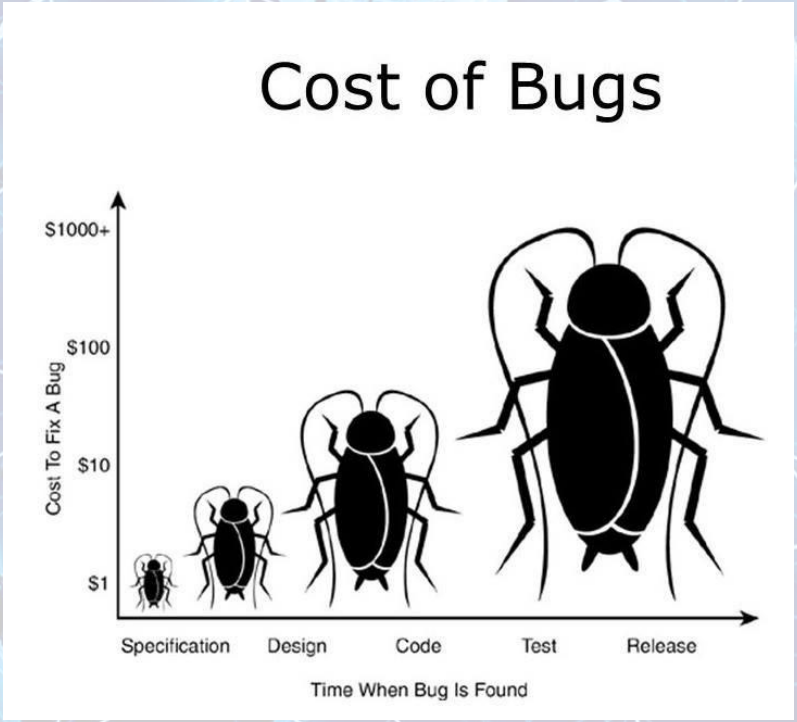
Developer

Tester

Tổng quan kiểm thử phần mềm

Khi nào cần bắt đầu kiểm thử phần mềm?

Kiểm thử nên bắt đầu càng sớm càng tốt trong quy trình phát triển phần mềm



Nguồn hình: Internet

Tổng quan kiểm thử phần mềm

Khi nào kiểm thử phần mềm kết thúc?

- Đến deadline
- Thực thi hết test case
- Tỷ lệ lỗi ở một mức cho phép
- ...

Kiểm thử là quy trình không bao giờ kết thúc.

Xem xét

Tổng quan kiểm thử phần mềm

Điều gì xảy ra nếu quá trình kiểm thử phần mềm không phát hiện lỗi?

Phần mềm quá tốt hoặc kiểm thử quá tồi

Tổng quan kiểm thử phần mềm

Tố chất cần thiết cho một người làm kiểm thử phần mềm (tester)?

Tò mò

Phán đoán

Cẩn thận

Tư duy logic

7 nguyên tắc kiểm thử phần mềm

Câu hỏi

Phát biểu nào sau đây thường không phải mục tiêu của kiểm thử?

A. Tìm lỗi phần mềm.

B. Đánh giá phần mềm có sẵn sàng triển khai không.

C. Chứng tỏ phần mềm không làm việc.

D. Chứng minh phần mềm phát triển đúng.

Một số thuật ngữ

- **Error (lỗi)**: lỗi do con người gây ra.
- **Fault (sai sót)**: những sai sót do dư hoặc thiếu hoặc không đúng các yêu cầu phần mềm cần thực hiện.

→ Error và fault gọi chung là **bug**.

- **Failure (hỏng)**: xảy ra khi các sai sót (fault) được thực thi.

Một số thuật ngữ

- Ví dụ

Yêu cầu:
- Nhập i
- Tính 2 * i^3
- Nếu nhập i = 3 thì kết quả là 54

Phần mềm:
- Nhập i
- i= i * 2
- i = power(i, 3)
- Xuất i

Kết quả:
- Input i = 3
- i= i * 2 = 6
- i = power(i, 3) = 216
- Kết quả xuất ra: 216

error

fault

fault + failure

Một số thuật ngữ

- **Incident (biến cố)**: là những trường hợp mà phần mềm có những **thực thi đáng nghi ngờ**, nguyên nhân của nó có thể là do cấu hình môi trường kiểm thử không đúng, dữ liệu dùng kiểm thử sai hoặc do lỗi của kiểm thử viên, v.v.
- Incident được xem là lỗi (defect hoặc bug) khi nguyên nhân của vấn đề thuộc chính thành phần đang kiểm thử.

Câu hỏi 1

Khi người dùng cuối (end-user) thấy sự khác nhau giữa kết quả mong muốn và kết quả thực thi của hệ thống, điều này được gọi là …

A. Error

B. Fault

C. Failure

D. Defect

Câu hỏi 2

Phát biểu nào sau đây sai?

A. Incident luôn được sửa (fixed).

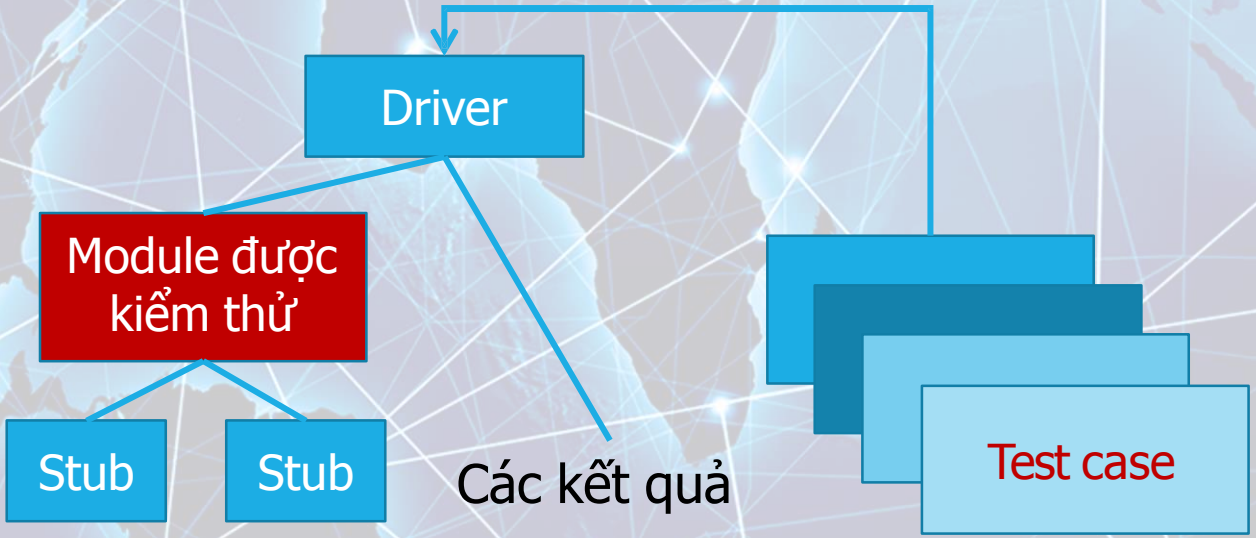
B. Incident xảy ra khi kết quả mong muốn và kết quả thực sự khác nhau.

C. Incident có thể được phân tích để cải thiện quy trình kiểm thử.

D. Incident có thể xảy ra khác với tài liệu hệ thống.

Stub & Driver

- **Stub**: một mẫu code để mô phỏng hoạt động của thành phần đang thiếu.
- **Driver**: một mẫu code để truyền test case (dữ liệu kiểm thử) đến mẫu code khác (module cần kiểm thử), nhận kết quả và xuất ra.



Câu hỏi

Để kiểm thử một chức năng, lập trình viên có thể viết một ... Gọi chức năng cần kiểm thử và truyền test data cho nó.

A. Stub

B. Driver

C. Proxy

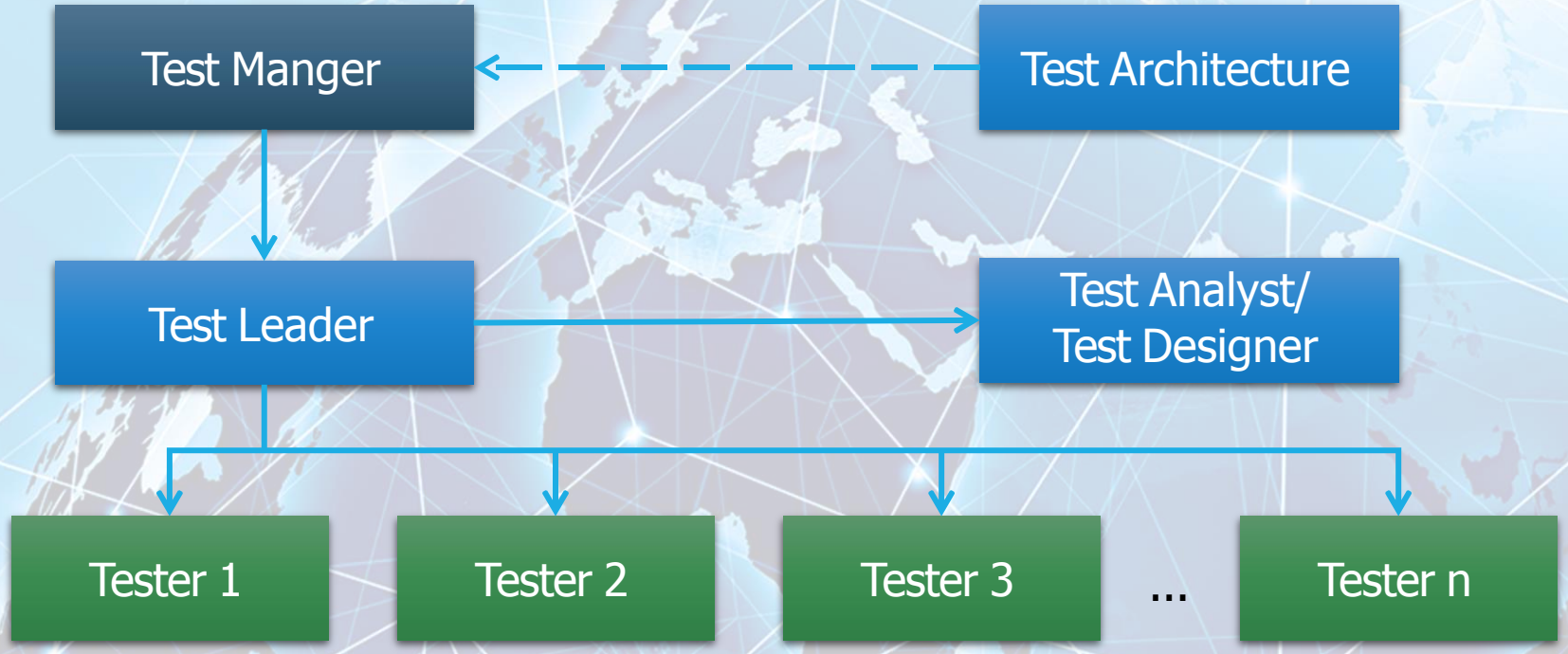
D. Tất cả đều sai

QA & QC

- QC (Quality Control): những **hoạt động, kỹ thuật** nhằm đảm bảo chất lượng sản phẩm.
- QA (Quality Assurance): những **kế hoạch, hoạt động mang tính hệ thống** nhằm đảm bảo quá trình sản xuất sẽ tạo những sản phẩm có chất lượng.
- So sánh QA & QC

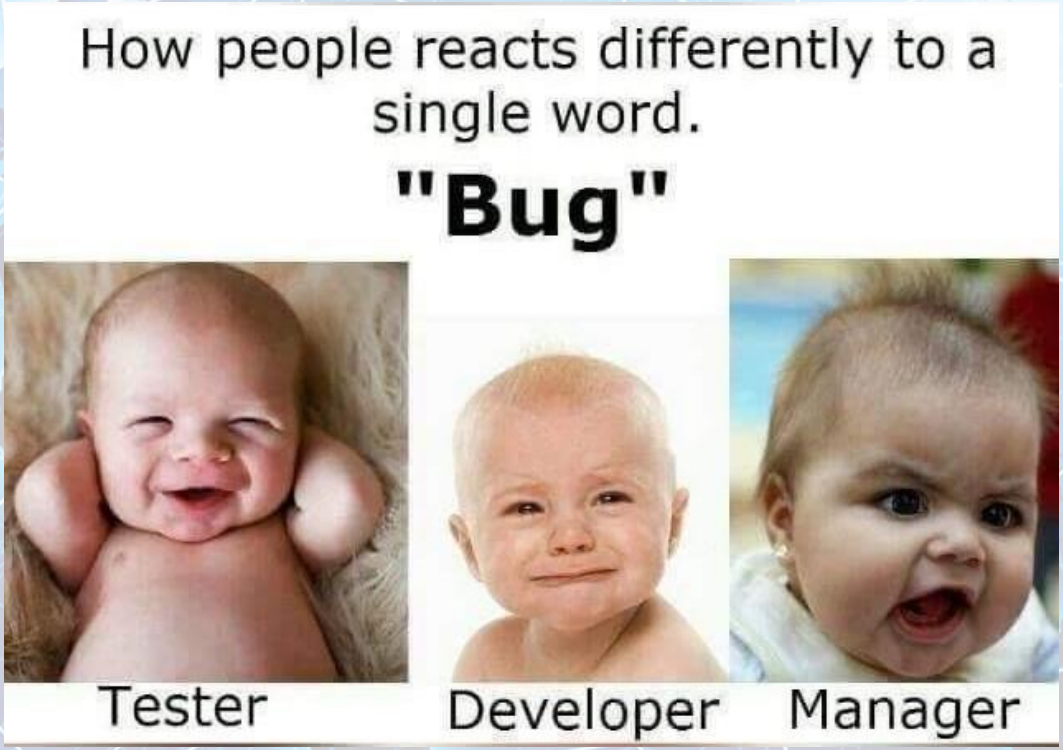
QA	QC
Quy trình	Sản phẩm
Ngừa lỗi	Tìm lỗi

Các vai trò kiểm thử phần mềm



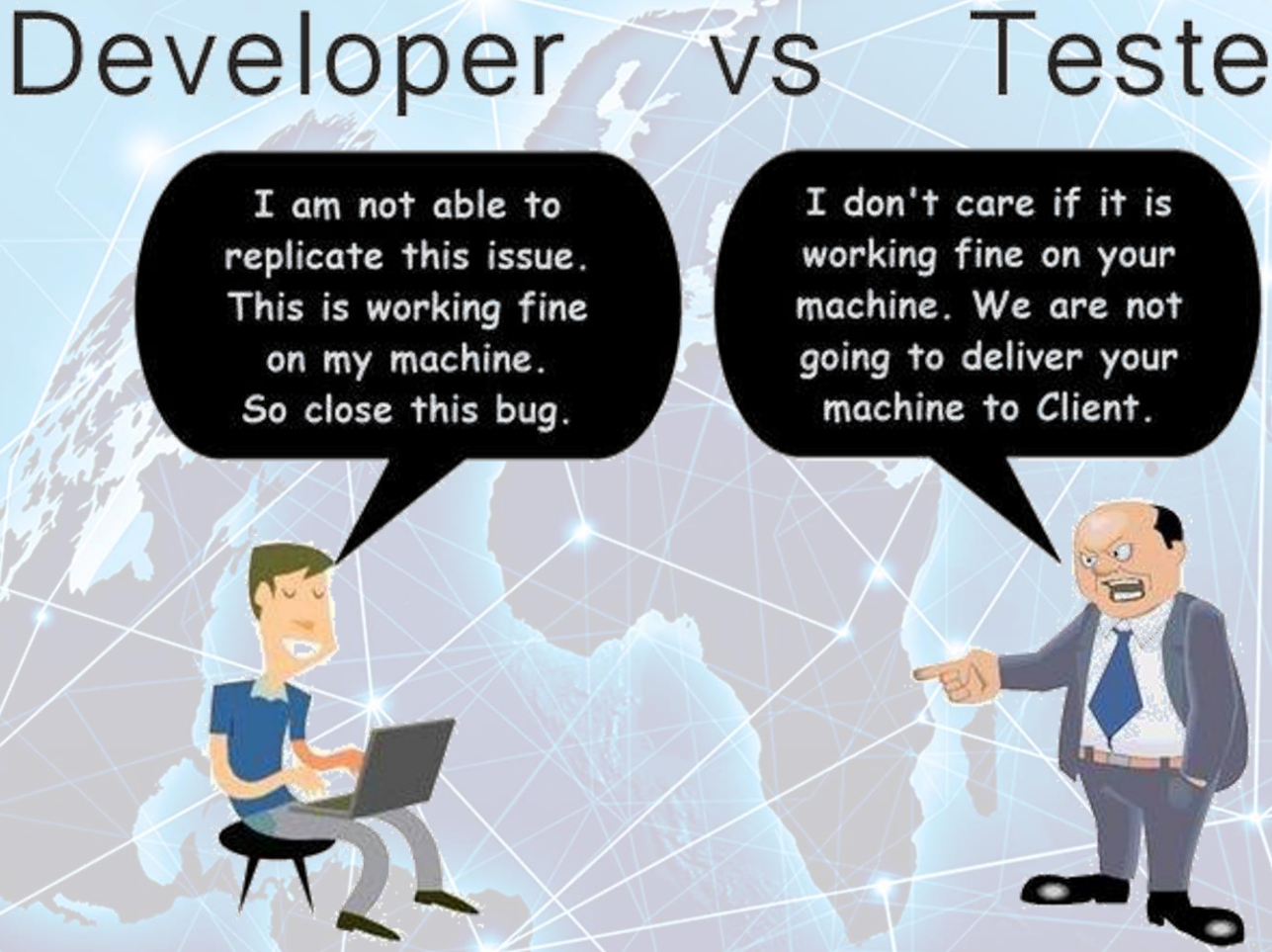
Các vai trò kiểm thử phần mềm

Khi bug được tìm tìm thấy



Nguồn: internet

Các vai trò kiểm thử phần mềm

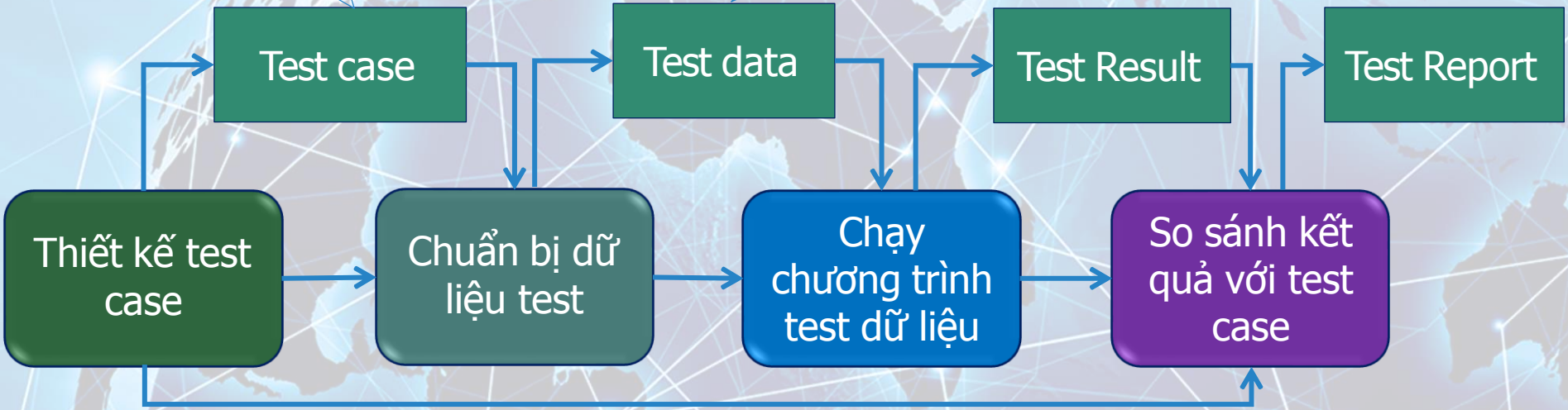


Nguồn: internet

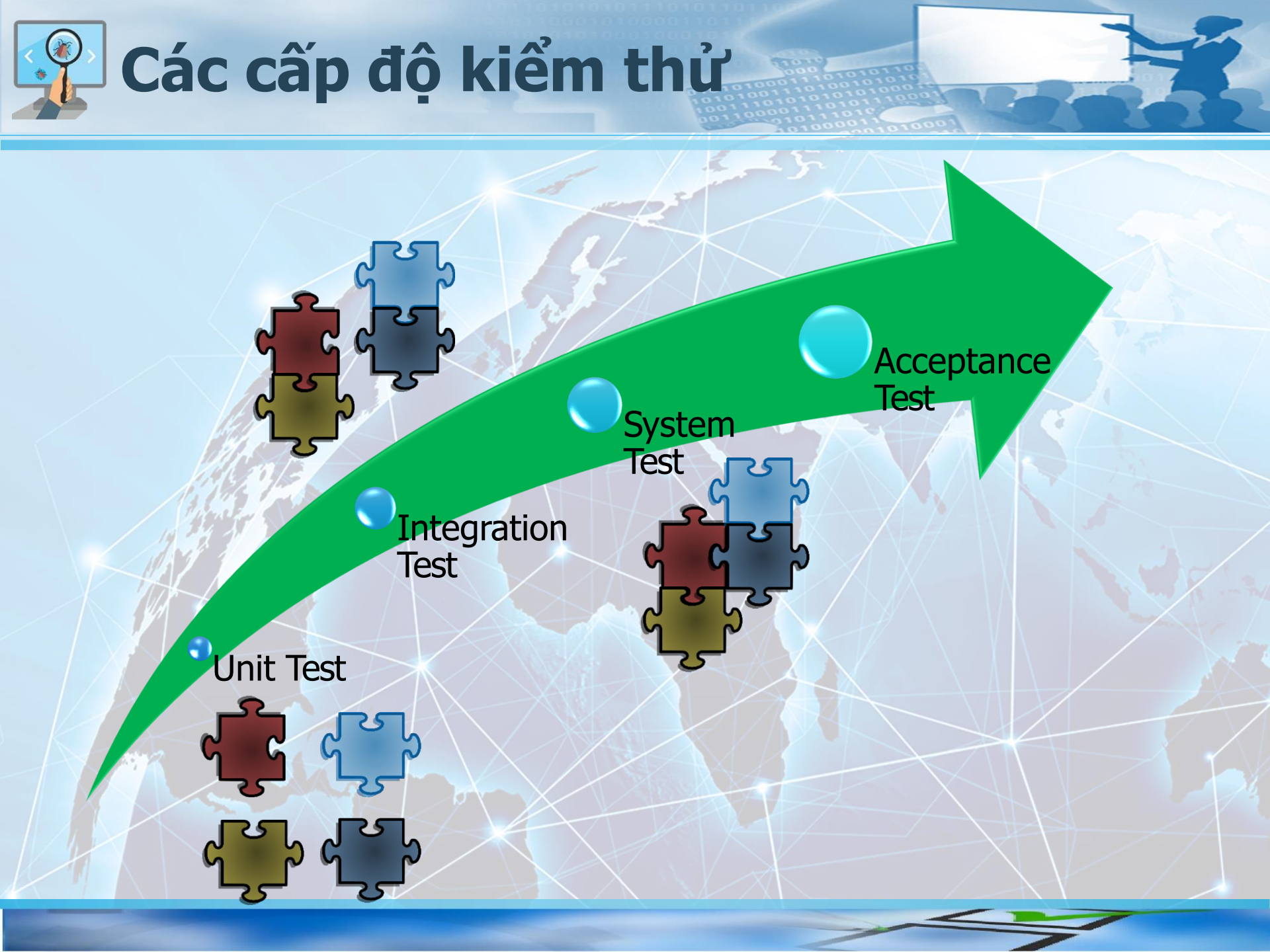
Quy trình kiểm thử phần mềm

Test case là đặc tả dữ liệu đầu vào, dữ liệu ra mong muốn, và bản mô tả những bước để kiểm thử

Test data là những dữ liệu vào có thể lấy để kiểm thử.



Các cấp độ kiểm thử



Unit test

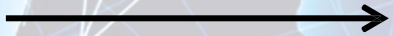
- Kiểm thử đơn vị (Unit test) dùng kiểm tra trên từng đơn vị riêng rẽ, thường được thực hiện bởi chính developer phát triển đơn vị đó nhằm đảm bảo mã nguồn viết cho đơn vị nào đó đúng với đặc tả yêu cầu, xác nhận mã nguồn trong đơn vị đó có thể thực thi.
- Một đơn vị là thành phần nhỏ nhất có thể kiểm thử. Nó có thể là một hàm, một thủ tục, một lớp, hoặc một phương thức.
- Các lỗi tìm thấy và sửa trong giai đoạn kiểm thử đơn vị thường không cần ghi nhận trong tài liệu.

Unit test

Program.java

```
16  public class Program {
17      public static boolean ktNguyenTo(int n) {
18          if (n >= 2) {
19              for (int i = 2; i < Math.sqrt(n); i++)
20                  if (n % i == 0)
21                      return false;
22
23              return true;
24          }
25
26          return false;
27      }
28  }
```

Unit test



TestCase.java

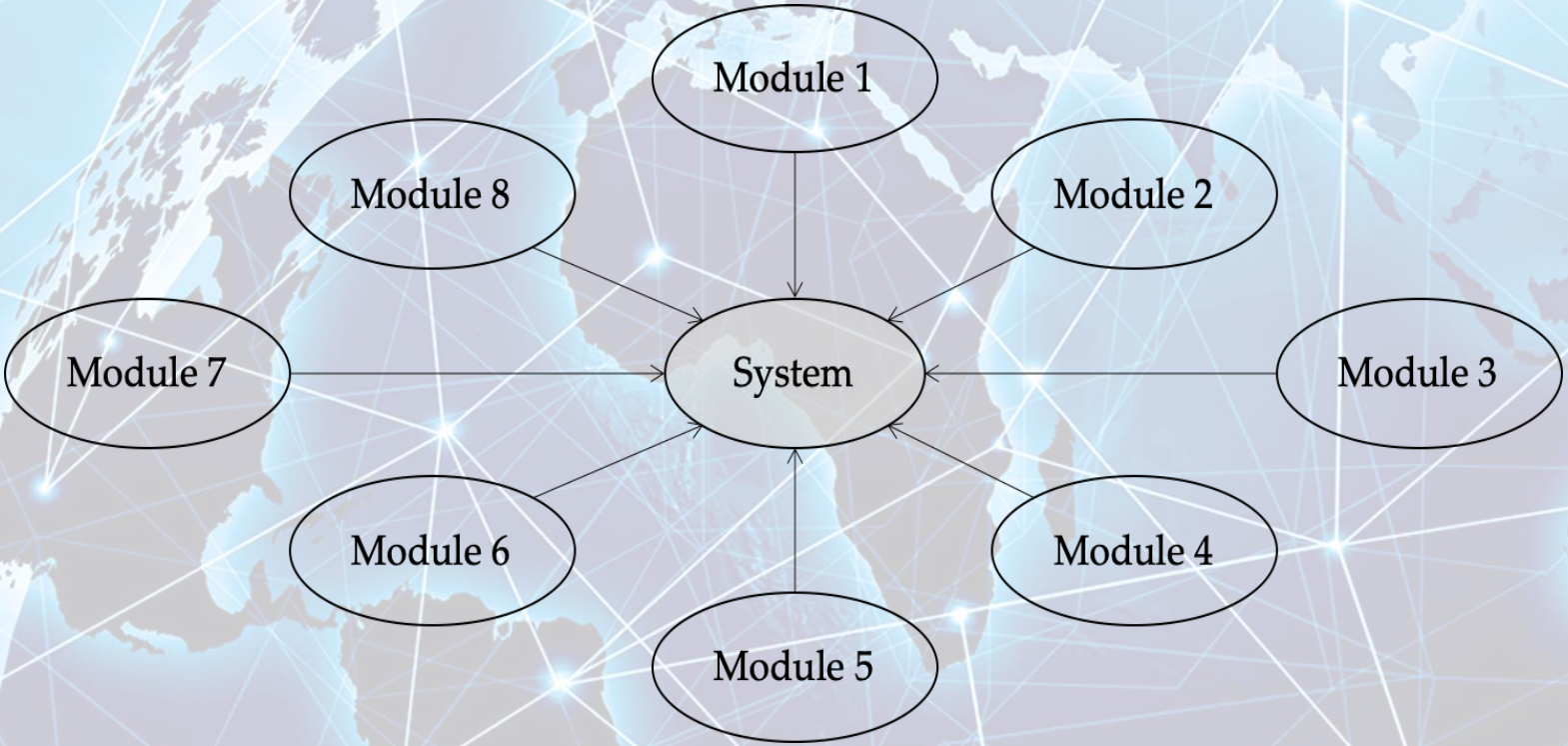
```
23  public class TestCase {
24      @ParameterizedTest
25      @ValueSource(ints = {2, 3, 5, 7, 11, 13})
26      public void testNguyenTo(int n) {
27          assertTrue(Program.ktNguyenTo(n));
28      }
29
30      @ParameterizedTest
31      @ValueSource(ints = {1, 4, 6, 10, 15})
32      public void testKhongNguyenTo(int n) {
33          assertFalse(Program.ktNguyenTo(n));
34      }
35
36      @ParameterizedTest
37      @CsvSource({"2,true", "4,false", "7,true"})
38      public void testCSV(int n, boolean expected) {
39          assertEquals(Program.ktNguyenTo(n), expected);
40      }
```

Integration test

- Kiểm thử tích hợp (Integration test) sẽ kết hợp các thành phần riêng rẽ lại với nhau như một hệ thống đã hoàn tất và kiểm tra **sự tương tác** giữa chúng.

Các chiến lược tích hợp

- **Chiến lược Big-bang**: tất cả các đơn vị, thành phần được tích hợp cùng một lúc để có hệ thống đầy đủ.



Các chiến lược tích hợp

- Ưu điểm: quá trình kiểm thử bắt đầu chỉ khi mọi thứ của hệ thống đã hoàn tất
- Khuyết điểm: khi có lỗi xảy ra rất khó cô lập lỗi để tìm ra nguyên nhân.
- Big-bang này thường là lựa chọn không tốt cho chiến lược tích hợp, nó có thể dẫn đến nhiều rủi ro sau đó của dự án, thời điểm tốn nhiều chi phí để giải quyết các vấn đề tìm thấy.

Các chiến lược tích hợp

- **Chiến lược Incremental**: chiến lược này kiểm thử từng phần riêng biệt.
- Ưu điểm của chiến lược này là lỗi được phát hiện sớm và dễ dàng tìm ra nguyên nhân.
- Khuyết điểm tốn nhiều thời gian vì phải phát triển các stub&driver để thực hiện kiểm thử. Hai chiến lược incremental thường được sử dụng là top-down và bottom-up.

Các chiến lược tích hợp

- Có hai tiếp cận incremental:
  - Chiến lược top-down: hệ thống được xây dựng theo từng giai đoạn, bắt đầu từ thành phần cao nhất (top) gọi các thành phần khác ở cấp thấp hơn. Kiểm thử theo chiến lược này yêu cầu sự tương tác giữa các thành phần phải được kiểm thử trước khi xây dựng các thành phần. Các thành phần ở cấp thấp hơn chưa xây dựng sẽ được thay thế bằng stub.
  - Chiến lược bottom-up: chiến lược này ngược lại với chiến lược top-down, các thành phần được tích hợp theo thứ tự từ dưới lên.

System test

- Kiểm thử hệ thống (System test) kiểm thử trên **hệ thống đầy đủ** sau khi các thành phần đã được tích hợp.
- Mục đích của kiểm thử hệ thống đảm bảo phần mềm **tuân thủ các đặc tả yêu cầu** của người dùng.
- Nhóm kiểm thử chỉ dựa trên đặc tả yêu cầu mà không được xem mã nguồn hệ thống. Đặc tả yêu cầu thường chứa các yêu cầu chức năng và các yêu cầu phi chức năng cần được kiểm thử.

System test

- Yêu cầu chức năng là các yêu cầu chỉ định các chức năng của hệ thống hoặc các thành phần cần được thực hiện, nó chỉ định chi tiết những gì (WHAT) hệ thống cần làm.
- Trong suốt quá trình kiểm thử chức năng, ta cần thực hiện các kiểm tra chức năng như cài đặt, chạy thử ứng dụng trên máy cục bộ, hệ điều hành cục bộ, kiểm tra các chức năng của ứng dụng, các xử lý chuỗi, văn bản như sao chép (copy), dán (paste) cho các ký tự mở rộng, ...

System test

- Yêu cầu phi chức năng là những yêu cầu không liên quan đến dịch vụ được chỉ định, mà liên quan đến những thuộc tính ràng buộc hệ thống như tính tin cậy, tính hiệu quả, tính tiện dụng:
  - Kiểm thử tính tin cậy (Reliabilty Tesing).
  - Kiểm thử tính tiện dụng (Usablilty Testing).
  - Kiểm thử tính hiệu quả (Efficiency Testing).
  - Kiểm thử khả năng bảo trì (Maintainablity Testing).
  - Kiểm thử tính khả chuyển (Portable Testing).
  - Kiểm thử hiệu năng (Performance Testing).

System test

- **Performance testing** nhằm xác định một số vấn đề về thắt cổ chai (bottleneck) hoặc hiệu năng của hệ thống.

Kiểm thử hiệu năng

- Kiểm thử tính bền vững (Endurance Testing) kiểm tra hành vi của hệ thống với lượng tải đáng kể được thực hiện trong một khoảng thời gian dài.
  - Trong quá trình thực hiện loại kiểm thử này, sự tiêu thụ bộ nhớ thường là nguyên nhân dẫn đến failure của hệ thống.
  - Ví dụ: một chức năng vẫn hoạt động tốt khi thực hiện kiểm thử trong 1 giờ, nhưng cũng cùng yêu cầu đó nếu thực hiện trong 3 giờ thì hệ thống phát sinh vấn đề do rò rỉ bộ nhớ (Leak Memory).

Kiểm thử hiệu năng

- Scalability Testing kiểm thử hệ thống để đo khả năng mở rộng cho các yêu cầu phi chức năng.
- Volume Testing kiểm thử hệ thống với một lượng dữ liệu nhất định.

Kiểm thử hiệu năng

- Để kiểm tra **khả năng chịu tải** của phần mềm thực hiện bằng cách chạy hàng loạt test, tăng dần sức tải cho đến khi hệ thống không thể thực thi được nữa, gọi là **load test**.
- Với cách tiếp cận này, test case thường được thiết kế xung quanh giá trị biên.

Kiểm thử hiệu năng

- Ví dụ: bản yêu cầu nêu rằng hệ thống **có khả năng chịu tải 300 giao tác** trên một giây thì ban đầu kiểm tra dưới 300 giao tác trên giây, và tăng dần cho đến khi hệ thống không thể thực thi được nữa.
- Với cách này, ta kiểm tra được 2 thứ
  - Cách xử lý của hệ thống khi nó quá tải.
  - Phát hiện những lỗi mà không thể phát hiện ở các trường hợp thông thường.

Kiểm thử hiệu năng

- Việc kiểm tra hệ thống với những giá trị ngoài phạm vi (abnormal conditions) đã được thiết kế gọi là **stress test**.
- Mục đích để **xác định những điểm gãy** (breaking point) của phần mềm.

Câu hỏi 1

Mục đích phân chia kiểm thử thành nhiều giai đoạn khác nhau là gì?

A. Dễ dàng quản lý kiểm thử ở các giai đoạn.

B. Mỗi giai đoạn kiểm thử có mục đích khác nhau.

C. Có thể chạy test khác nhau trên các giai đoạn khác nhau.

D. Càng nhiều giai đoạn thì quá trình kiểm thử càng tốt hơn.

Câu hỏi 2

Phát biểu nào sau đây KHÔNG đúng về kiểm tra hệ thống (System Testing)?

A. Kiểm tra hệ thống được thực hiện bởi một nhóm độc lập.

B. Kiểm tra chức năng (functional testing) được ưu tiên thực hiện hơn so với kiểm tra cấu trúc (structural testing)

C. Lỗi tìm trong kiểm tra hệ thống tốn rất nhiều chi phí để sửa.

D. Người dùng cuối (end-user) nên tham gia vào kiểm tra hệ thống.

Acceptance test

- Kiểm thử chấp nhận (acceptance test) được thực hiện bởi **khách hàng** để đảm bảo sản phẩm phần mềm hoạt động đúng những gì khách hàng mong đợi và khách hàng chấp nhận sản phẩm – thanh toán hợp đồng.

Acceptance test

- Với những sản phẩm có người dùng đa dạng trên thị trường thường có hai loại kiểm thử:
  - **Alpha testing**: người dùng kiểm tra phần mềm ngay môi trường phát triển phần mềm.
  - **Beta testing**: người dùng kiểm tra phần mềm trong môi trường thực của người dùng.

Acceptance test

- **Alpha testing**
  - Thực hiện bởi một nhóm người trong tổ chức phát triển (**in-house**) phần mềm độc lập với nhóm phát triển (developer).
  - Developer quan sát người dùng sử dụng phần mềm và ghi nhận lỗi phát sinh để sửa chữa.
  - Alpha testing giúp phát hiện ra những vấn đề thực tế khi sử dụng phần mềm mà nhóm phát triển chưa biết.

Acceptance test

- **Beta testing**
  - Tester là nhóm người từ khách hàng (**external**).
  - Phần mềm cũng **có thể công khai** cho những người dùng quan tâm dùng thử.
  - Khách hàng ghi nhận lỗi và báo lại cho các developer.
  - Beta testing giúp phát hiện những vấn đề tương tác giữa phần mềm và những đặc trưng trong môi trường nó được sử dụng.
  - Beta testing được thực hiện sau alpha testing.

Câu hỏi

Phát biểu nào sau đây đúng về Beta Testing?

A. Thực hiện bởi khách hàng trong môi trường làm việc của họ.

B. Thực hiện bởi khách hàng trong môi trường làm việc của developer.

C. Thực hiện bởi một nhóm Tester độc lập.

D. Thực hiện sớm nhất có thể trong vòng đời phát triển phần mềm.

Kiểm thử tĩnh

- Kiểm thử tĩnh là các kỹ thuật kiểm thử không cần thực thi mã nguồn chương trình.
- Hai kỹ thuật được sử dụng thường xuyên trong kiểm thử tĩnh là đánh giá (Review) và phân tích tĩnh (Static Analysis).
- Review thường được thực hiện thủ công, static analysis thường được thực hiện tự động bằng các công cụ.

Kỹ thuật review

- Kỹ thuật review để tìm và gỡ bỏ các lỗi, những điều chưa rõ trong các tài liệu trước khi chúng được sử dụng trong quy trình phát triển.
- Kỹ thuật review kiểm thử bất cứ thứ gì được tài liệu hoá như đặc tả yêu cầu, thiết kế hệ thống, mã nguồn, kế hoạch kiểm thử, các test case.
- Có hai loại review
  - Formal review được thực hiện theo quy trình chính thức, có kế hoạch.
  - Informal review khi thực hiện không xác định trước các bước, quy trình thực hiện.

Kỹ thuật review

Kỹ thuật review

Phân tích tĩnh

- Kỹ thuật phân tích tĩnh (Static Analysis) cho phép mã nguồn được phân tích các vấn đề cấu trúc chương trình hoặc những điểm yếu của chương trình (Source Code), các mô hình phần mềm (Model) một cách hệ thống mà không cần thực thi chương trình.

Phân tích tĩnh

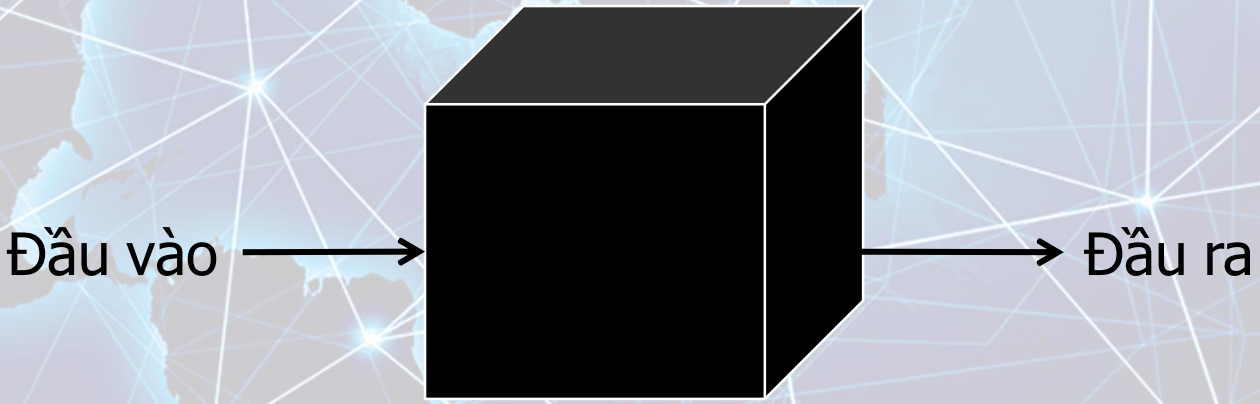
- Phân tích tĩnh giúp phát hiện các vấn đề sớm, đưa ra những cảnh báo những điểm đáng nghi ngờ trong mã nguồn chương trình hoặc trong thiết kế bằng cách tính toán một số độ đo (Metrics) như độ phức tạp chương trình.
- Nó cũng giúp cải thiện khả năng bảo trì mã nguồn và các thiết kế của phần mềm.

Phân tích tĩnh

- Các vấn đề thường được phát hiện:
  - Tham chiếu đến biến nhưng giá trị của nó chưa được chỉ định (Undefined).
  - Biến khai báo nhưng không được sử dụng.
  - Phát hiện các hàm và thủ tục không được gọi.
  - Vi phạm chuẩn lập trình.
  - Chương trình thiếu tính an toàn (Security).
  - Sử dụng các cú pháp không chính xác trong ngôn ngữ lập trình hay một ngôn ngữ thiết kế phần mềm đang sử dụng.

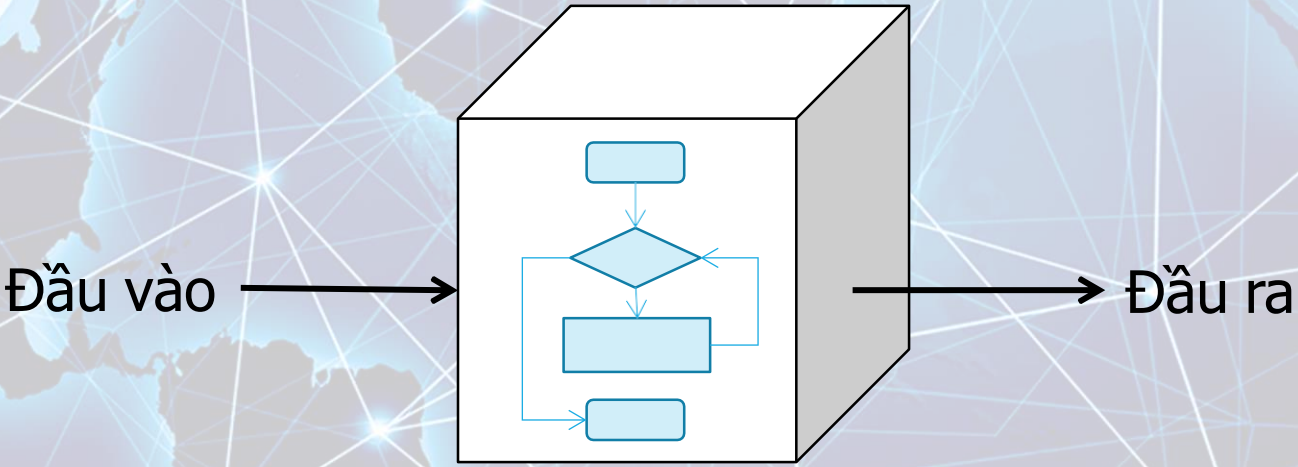
Kiểm thử động

- Kiểm thử hộp đen (Black-box):
  - Kỹ thuật kiểm thử mà **không cần biết** những công việc bên trong của ứng dụng gọi là black-box testing.
  - Tester sẽ **tương tác với giao diện người dùng**, cung cấp dữ liệu vào và kiểm tra dữ liệu ra, không cần biết cách thức và nơi nào xử lý dữ liệu vào đó.



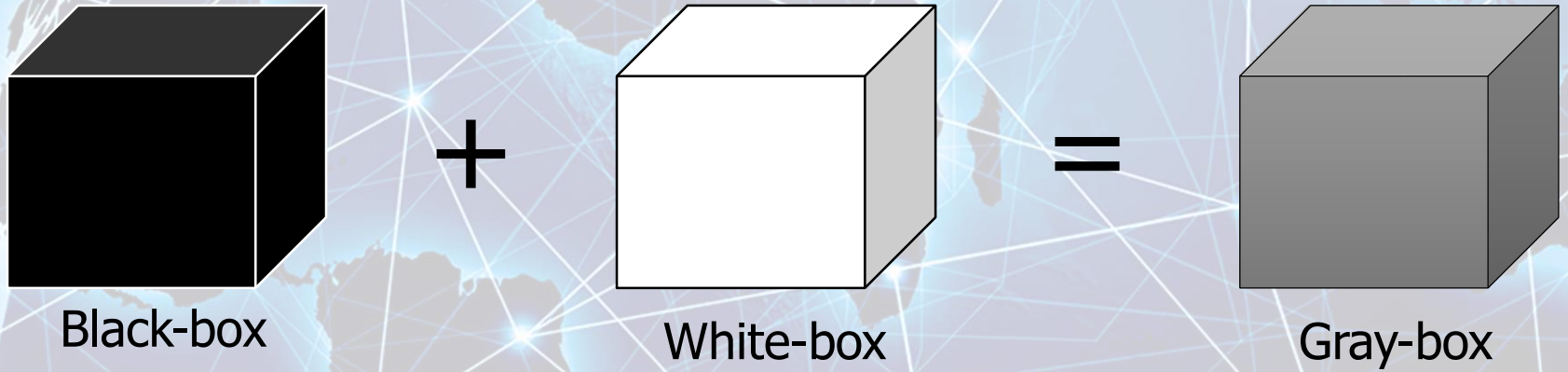
Kiểm thử động

- Kiểm thử hộp trắng (White-box):
  - Kỹ thuật kiểm thử white-box yêu cầu **biết logic và cấu trúc mã nguồn** bên trong chương trình.
  - Tester cần biết rõ những công việc bên trong mã nguồn, làm rõ từng đơn vị hoặc đoạn mã nguồn không phù hợp.



Kiểm thử động

- Kiểm thử hộp xám (Gray-box)
  - Gray-box testing yêu cầu tester phải có **chút ít hiểu biết** công việc bên trong của ứng dụng.
  - Tester có thể truy cập vào tài liệu thiết kế và cơ sở dữ liệu. Với các tài liệu này, tester có thể chuẩn bị dữ liệu và kịch bản kiểm thử tốt hơn.



Kiểm thử động

	Black-box	Gray-box	White-box
Cấu trúc bên trong chương trình	Không biết	Biết một ít	Biết đầy đủ
Người thực hiện	End-user	End-user Tester developer	Tester Developer
Cơ sở thực hiện	Dựa trên đầu vào, đầu ra mong muốn	Dựa trên lược đồ cơ sở dữ liệu, sơ đồ luồng dữ liệu	Cấu trúc bên trong chương trình
Chi phí	Thấp	Trung bình	Cao
Kiểm tra thuật toán	Không phù hợp	Không phù hợp	Phù hợp

Câu hỏi

Khác nhau chính kiểm thử hộp trắng và hộp đen?

A. Kiểm thử hộp đen chỉ được thực hiện ngay sau kiểm thử hộp trắng.

B. Kiểm thử hộp đen là kiểm thức chức năng (functional), còn kiểm thử hộp trắng là kiểm thử phi chức năng (non-functional)

C. Kiểm thử hộp đen là kiểm thử chức năng (functional), kiểm thử hộp trắng là kiểm thử cấu trúc (structural).

D. Không khác nhau giữa hai kỹ thuật.

Kiểm thử tĩnh & động

- **Kiểm thử tĩnh**:
  - Xem mã nguồn, **không thực thi**.
  - Thực hiện trong quy trình Verfication.
  - Code inspection, walk through, code reviews, desk checking.
- **Kiểm thử động**:
  - **Thực thi** mã nguồn.
  - Thực hiện trong quy trình Validation.
  - Black-box testing, white-box testing

Câu hỏi

- Lỗi hay thiếu sót (defect) có thể được tìm thấy bằng chiến lược kiểm thử tĩnh (static) là:
  i. Biến khai báo nhưng không được sử dụng.
  ii. Vi phạm chuẩn lập trình.
  iii. Phát hiện các hàm và thủ tục không được gọi.
  iv. Chương trình thiếu tính an toàn (security).

A. i,ii, iii đúng; iv sai

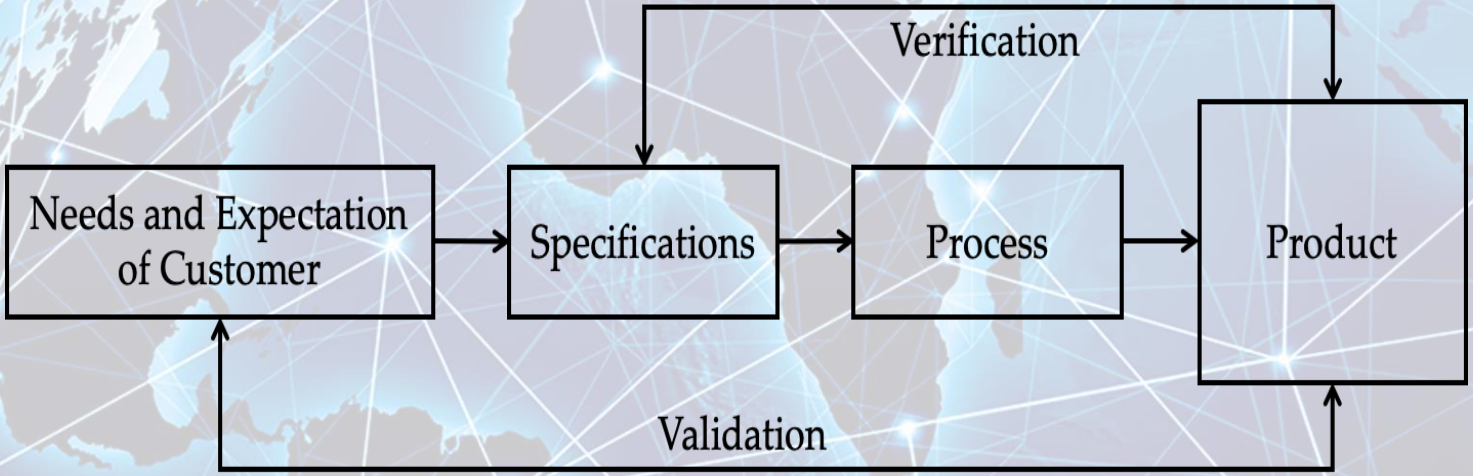
B. i,ii, iii, iv đều đúng

C. i,iii đúng; ii, iv sai

D. i, ii sai; iii, iv đúng

Mô hình V&V

- V&V bao gồm **một dãy rộng các hoạt động trong đảm bảm chất lượng** phần mềm như technial reviews, documentation review, database review, algorithm analysis. Trong đó kiểm thử (testing) đóng vai trò cực kỳ quan trọng.



Mô hình V&V

- **Verficiation** đảm bảo sản phẩm phần mềm đúng với đặc tả yêu cầu của nó (right way) được thực hiện khi bắt đầu quy trình phát triển, bao gồm các buổi review để đánh giá các tài liệu, kế hoạch, mã nguồn, các tài liệu về yêu cầu và đặc tả yêu cầu.
- **Validation** là quá trình đảm bảo sản phẩm phần mềm đáp ứng được yêu cầu của người dùng (user needs). Validation được thực hiện vào cuối của quy trình phát triển và sau khi verification được hoàn thành.

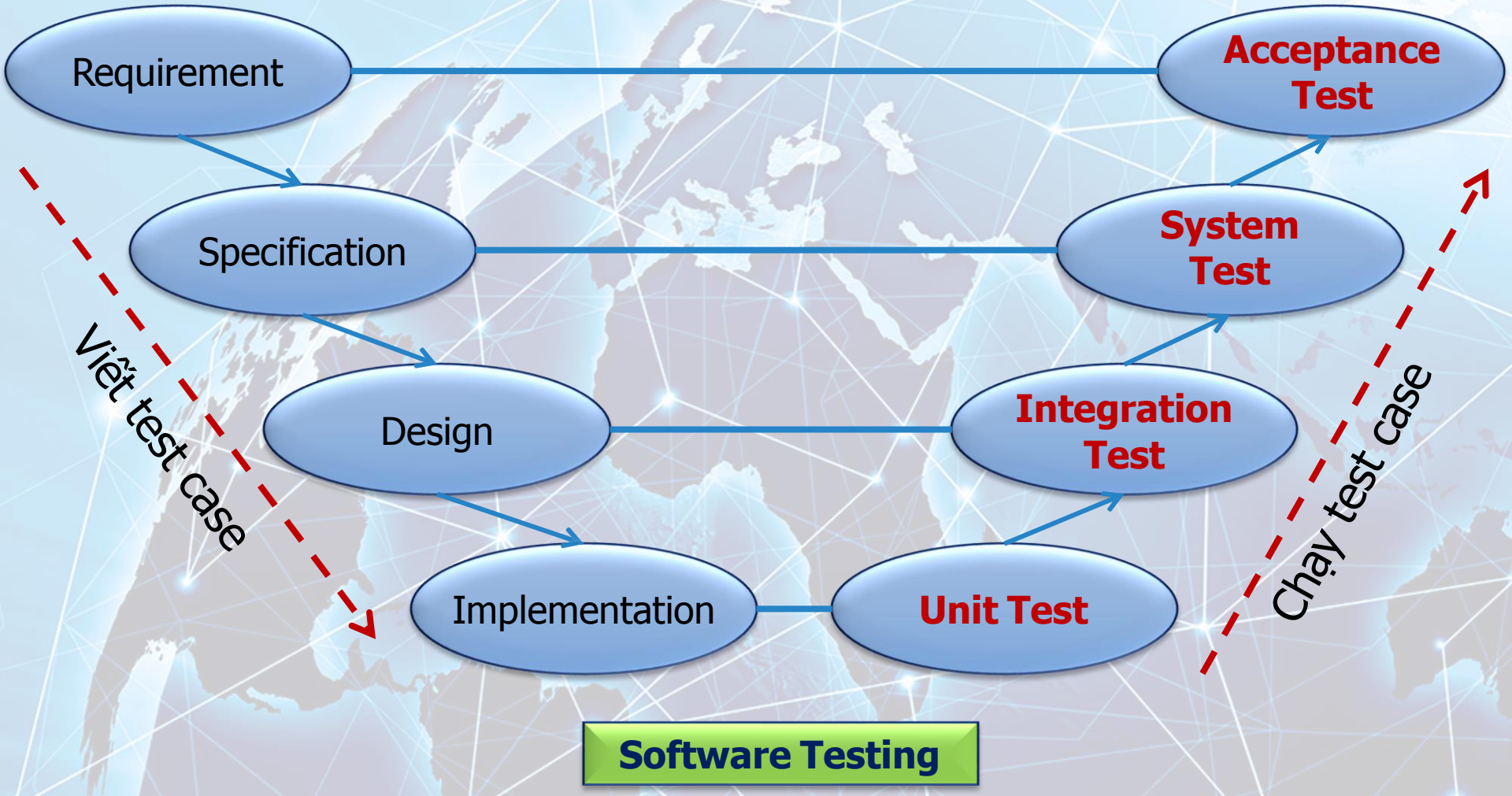
Mô hình V&V

- Verification
  - Là quá trình đảm bảo sản phẩm phần mềm đúng với đặc tả của nó.
  - Thực hiện bởi developer
  - "**Are we building the product right?**"
- Validation
  - Là quá trình đảm bảo sản phẩm phần mềm đáp ứng được yêu cầu của người dùng.
  - Thực hiện bởi tester
  - "**Are we building the right product?**"

V-Model

- V-Model cũng là một mô hình quy trình phát triển phần mềm dựa trên sự kết hợp các giai đoạn kiểm thử cho mỗi giai đoạn tương ứng trong quy trình phát triển.
- Quá trình kiểm thử được thực hiện sau mỗi kết quả đạt được ở từng bước, điều này giúp phát hiện ra những vấn đề sớm nhất trong vòng đời phát triển của phần mềm.

V-Model



Software Testing

Câu hỏi

- Phát biểu nào sau đây đúng về V-Model?

A. Các bước giống mô hình thác nước trong phát triển phần mềm.

B. Nó là mô hình theo chu kỳ (cyclical model) trong phát triển phần mềm.

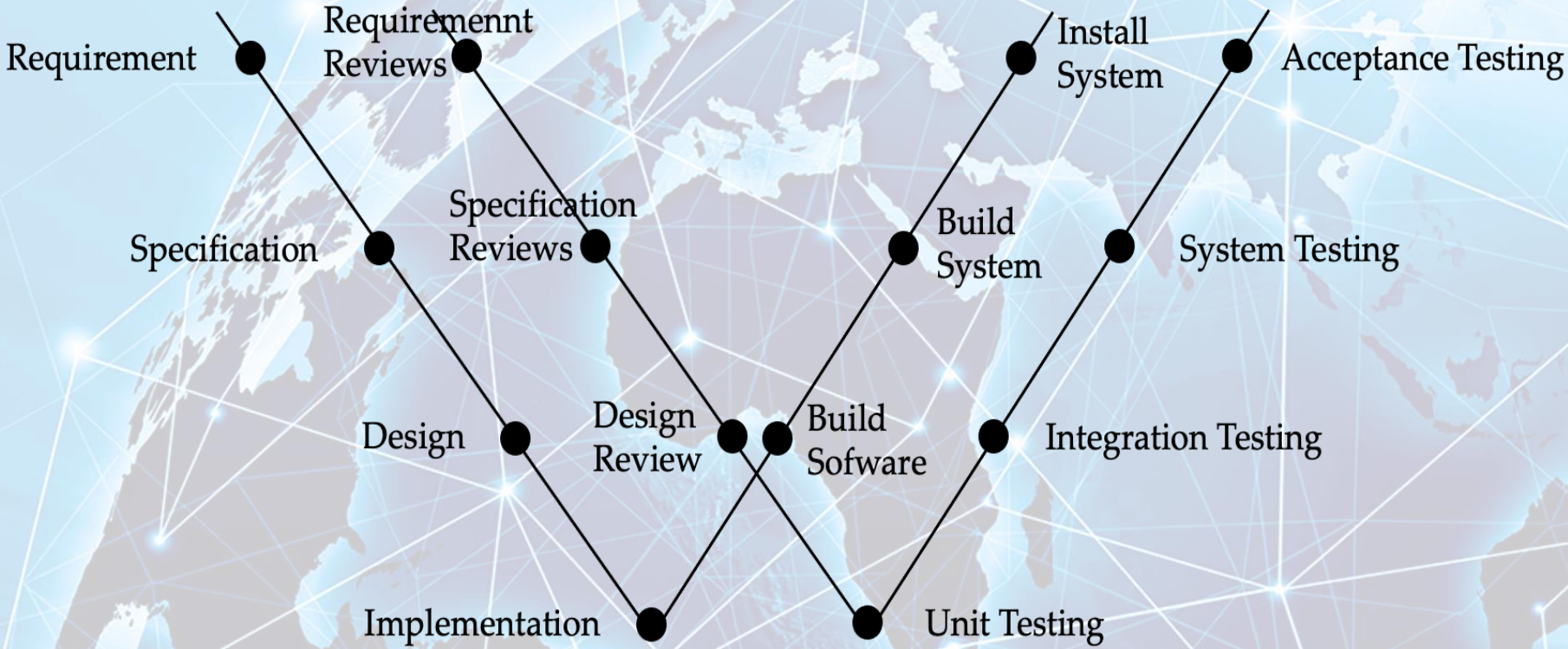
C. Nó cho phép ra đời các phiên bản làm việc được của hệ thống sớm nhất có thể.

D. Nó cho phép kế hoạch kiểm thử bắt đầu sớm nhất có thể.

W-Model

- V-Model chỉ tập trung vào kiểm thử động, mà bỏ qua hiệu quả của kiểm thử tĩnh.
- W-Model khắc phục được điều có bằng cách cho phép thực hiện quá trình kiểm thử song song với quá trình phát triển phần mềm, các kỹ thuật kiểm thử tĩnh được áp dụng trong giai đoạn đầu của quy trình phát triển phần mềm, mỗi hoạt động trong quá trình phát triển đều có một hoạt động kiểm thử tương ứng.

W-Model


Requirement
Requirement Reviews
Specification
Specification Reviews
Design
Design Review
Implementation
Build Sofware
Unit Testing
Integration Testing
Build System
Install System
System Testing
Acceptance Testing

Các tài liệu kiểm thử phần mềm

- Test case
- Checklist
- Test plan
- Test report
- Test scenario
- Traceability matrix

Test case

- Test case là một tài liệu trong quá trình kiểm thử phần mềm được phát triển cho một kịch bản cụ thể để xác minh tính đúng đắn của một yêu cầu cụ thể.
- Test case thể hiện cho một hành vi nào đó trong chương trình.
- Một test case gồm dữ liệu vào, các bước thực hiện và dữ liệu ra mong muốn.

Test case

- Số định danh test case (Test case ID)
- Kịch bản kiểm thử (Test scenario)
- Mô tả test case
- Các bước thực hiện kiểm thử
- Điều kiện tiên quyết (Prerequisite)
- Dữ liệu kiểm thử
- Kết quả mong muốn (expected result)
- Các tham số kiểm thử
- Kết quả thực sự (actual result)
- Thông tin môi trường kiểm thử
- Ghi chú (comment)

Test case

TEST CASE

System Name:	Sample Project		
Module Code :	CR100 - Export to excel		
Test requirement:	CR1 -		
Pass	25	Pending	0
Fail	1	Number of test cases:	37

ID	Test Case Description	Test Case Procedure	Expected Output	Test date	Result	Note
1. Check add role "CanExportAllCarrierChoises": This will be typically set for System users only						
TC1	Checking new role is added	1: Go to the system TestProEngine with Classic or Current or Expert Mode 2: Go to maintenance 3: Click Maintenance Users 4: Click Role Tab	Role Tab is added a new role, it's name **"CanExportAllCarrierChoices"**		Pass	
Check set Role CanExportAllCarrierChoices = True						
TC2	Checking new function is added in **Classic or Current Mode**	1: Go to the system TestProEngine with Classic or Current Mode that has to set role "CanExportAllCarrierChoices" 2: Submit for the quote 3: Open the quote at the home	**See** a new selector: **"Export all Carriers"**		Pass	

Nguồn: https://strongqa.com/

Checklist

- Checklist là một danh sách các test case được thực thi trong một thủ tục xác định, nó giúp ta nhanh chóng nắm được các test case đã được thực thi hết hay chưa và có bao nhiêu test bị fail

Application Testing Checklist			
Tested By	Tester	Date	
Application Name			
Procedure	Expected Result	Pass/Fail (P/F)	Actual Results/Comments
Application Functionality			
Performs primary functionality and maintains stability	Yes	P	Opens and allows students to practice keyboarding without program errors or hangs.
Windows Fundamentals			
Installs under a user account	No	P	Does not install under student user account
Installs under a power user account	Yes	P	Installs correctly
Installs under an administrator account	Yes	P	Installs correctly
Completes a minimal installation	NA	NA	NA

Nguồn: https://strongqa.com/

Test plan

- Tài liệu phác thảo **chiến lược kiểm thử phần mềm** như tài nguyên sử dụng, môi trường kiểm thử, phạm vi kiểm thử, lịch trình cho các hoạt động kiểm thử.
- Tài liệu này thường được phát triển trong giai đoạn đầu phát triển phần mềm.

Test plan

- Các giả thiết khi kiểm thử phần mềm.
- Giới thiệu tài liệu test plan
- Danh sách các test case để kiểm thử phần mềm
- Danh sách các đặc trưng (feature) sẽ được kiểm thử.
- Danh sách các sản phẩm (deliverabled) sẽ được kiểm thử.
- Xác định ưu tiên cách tiệp cận cho việc kiểm thử.
- Tài nguyên sử dụng cho việc kiểm thử.
- Nêu các rủi ro có thể xảy ra trong quá trình kiểm thử
- Lịch trình các task, milestone cần đạt được

Test plan

Table of Contents

Nguồn: https://strongq

Test report

- Test report thể hiện kết quả kiểm thử chính thức.
- Tài liệu này lưu lại kết quả kiểm thử phần mềm một cách có tổ chức, mô tả điều kiện và môi trường kiểm thử, so sánh kết quả kiểm thử với mục tiêu kiểm thử đề ra.

Test report

Overall progress of the QA cycle(Ontime, delayed, Stopped)	On time
Total number of test cases	100
Number of testers	5
Test cycle duration	5 days

Status for	
Number of test cases planned	20
Number of test cases executed	18
Number of test cases executed overall	78
Number of defects encountered today	2
Number of defect encountered so far	10
Number of critical defects- still open	3

Overall status	
Number of test cases planned	100
Number of test cases executed	78
Pass Percentage of the defects	98%
Defects density	2.5 per day
Critical defects percentage	20%

Nguồn: https://strongqa.com/

Test scenario

- Test scenario đảm bảo các luồng xử lý từ đầu đến cuối. Một test scenario bao gồm nhiều bước được thực thi.

System and Integration Test Scenario					
System and Integration Test Scenario # 1					
Version #:	<Version number of the application being tested>	**Build #:**	<Tracking number associated with the module, or set of modules, packaged together that perform the function tested by this test scenario>	**Retry #:**	<A sequential number representing the number of times the test case has been executed.>
Test Scenario ID:	<A sequential number assigned to this test scenario for tracking purposes>		**Process ID:**	<Identify the Process ID that this Test Scenario relates with>	
Environment:	<Enter the environment this test scenario is using, e.g. mainframe, client/server>		**Machine tested:**	<Enter the machine that will be used for this test, e.g. PC, server>	
Test Scenario Description:	<Describe briefly what this test scenario is testing.>				
Objective:	<Describe the desired objective of this test scenario.>				
Assumptions/ Constraints:	<List any assumptions or constraints that the tester should be aware of.>				
Test Files/Test Data:	<List the test files and or test data in order to successful run this test scenario.>				
Author:	<Identify the author of this Test Scenario>		**Last Modified:**	<Update the date that the Test Scenario was last updated.>	
Reviewed by:	<Identify the person who reviewed this Test Scenario>		**Reviewed Date:**	<List the date that the Test Scenario was reviewed on.>	
Executed by:	<Identify the person who executed this Test Scenario>		**Execution Date:**	<List the date that the Test Scenario was last executed on.>	

Test Steps:			
Step #	**Description**	**Expected Result**	**Actual Result**
<Use sequential counting numbers>	<Enter the description for each step of the test scenario. Fully describe what the tester should be doing on the application.>	<Enter the expected result of a successful execution of each step.>	<Enter the actual result for each step. If the actual result matches the expected result, then enter the same information.>

Traceability Matrix

- Traceability Matrix là một bảng để lưu vết (trace) các yêu cầu trong vòng đời phát triển phần mềm.
- Mỗi yêu cầu bảng này được liên kết đến một test case để quá trình kiểm thử được thực hiện. Ngoài ra, Bug ID cũng được liên kết đến yêu cầu này.

Traceability Matrix

- Mục đích chính của bảng này
  - Đảm bảo phần mềm phát triển như yêu cầu đề cập.
  - Thuận tiện tìm nguyên nhân chính (root cause) gây ra bug.
  - Lưu vết các tài liệu được phát triển trong vòng đời phát triển phần mềm.

Traceability Matrix

Project Name:
Project Number:
Project Manager:

Purpose: To manage requirements throughout the software development lifecycle. The Traceabilty Matrix ensures that requirements are captured in the design, implemented in the code, and verified by testing.

Use Case	Requirement Number	Requirement Description	Status	Release	Test Case	Design Component / Module	Comments

Câu hỏi 1

Nhiệm vụ chính của test plan là gì?

A. Quyết định cách tiếp cận để kiểm thử.

B. Chuẩn bị đặc tả kiểm thử.

C. Xác định tiêu chuẩn kiểm thử hoàn tất.

D. Xác định độ đo và phân tích kết quả.

Câu hỏi 2

Nếu kết quả mong muốn không được chỉ định trong test case thì ...

A. Không thể chạy các test case.

B. Khó lặp lại việc kiểm thử.

C. Khó quyết định test case là pass hay failed.

D. Không thể tự động nhập liệu cho kiểm thử được.

Q&A