Nhập môn Công nghệ phần mềm

Topic 8
Verification & Validation

12/2022

Các chủ đề

- Verification
- Validation
- Software testing

Verification vs validation

- **Verification**: kiểm tra xem chúng ta có làm ra sản phẩm đúng theo các đặc tả (specification) không?

- **Validation**: kiểm tra xem sản phẩm có đúng với những gì mà khách hàng thật sự mong muốn không?

Tiến trình verification và validation

- Trong quá trình phát triển phần mềm, chúng ta cần áp dụng tiến trình V&V tại mỗi giai đoạn của tiến trình phần mềm.
- Hai mục tiêu chính của V&V:
  - Phát hiện các lỗi trong hệ thống.
  - Đánh giá xem hệ thống có hợp lý và đáp ứng đúng yêu cầu sử dụng khi hoạt động không?

Mục tiêu của V&V

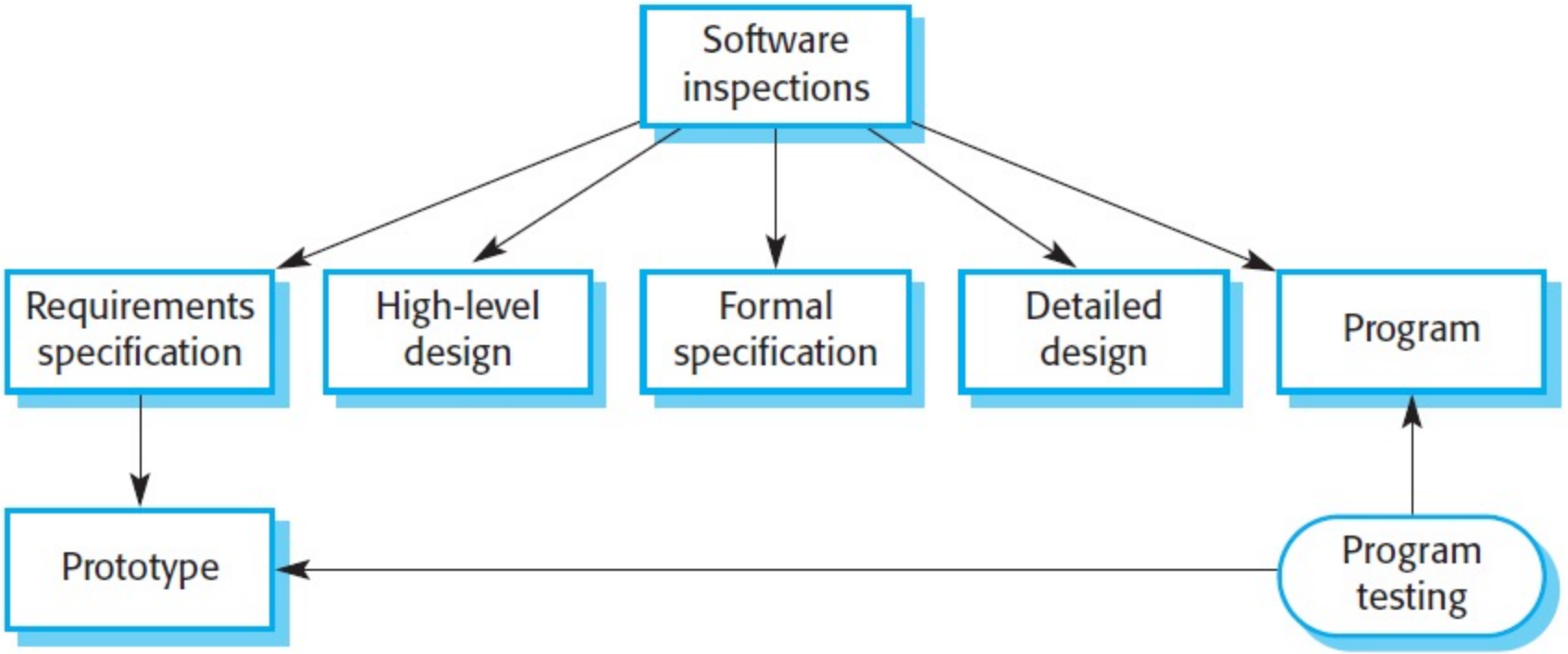
- V&V xác định được độ tin cậy của sản phẩm phần mềm so với các mục tiêu của sản phẩm.

- V&V không đảm bảo là hệ thống sẽ không còn lỗi mà nó cho thấy được mức độ đáp ứng của sản phẩm là đủ tốt cho việc sử dụng.

Verification tĩnh và động

- **Software inspections**: liên quan đến việc phân tích hệ thống để tìm ra các sự cố liên quan. Quá trình này có thể dựa trên 1 số công cụ phân tích và thanh tra mã nguồn.
- **Software testing**: liên quan đến việc thực thi hệ thống và xem xét sự hoạt động của hệ thống.
  - Chạy hệ thống với các dữ liệu kiểm tra (test data) và xem xét kết quả và cách hoạt động của hệ thống.

V&V tĩnh và động



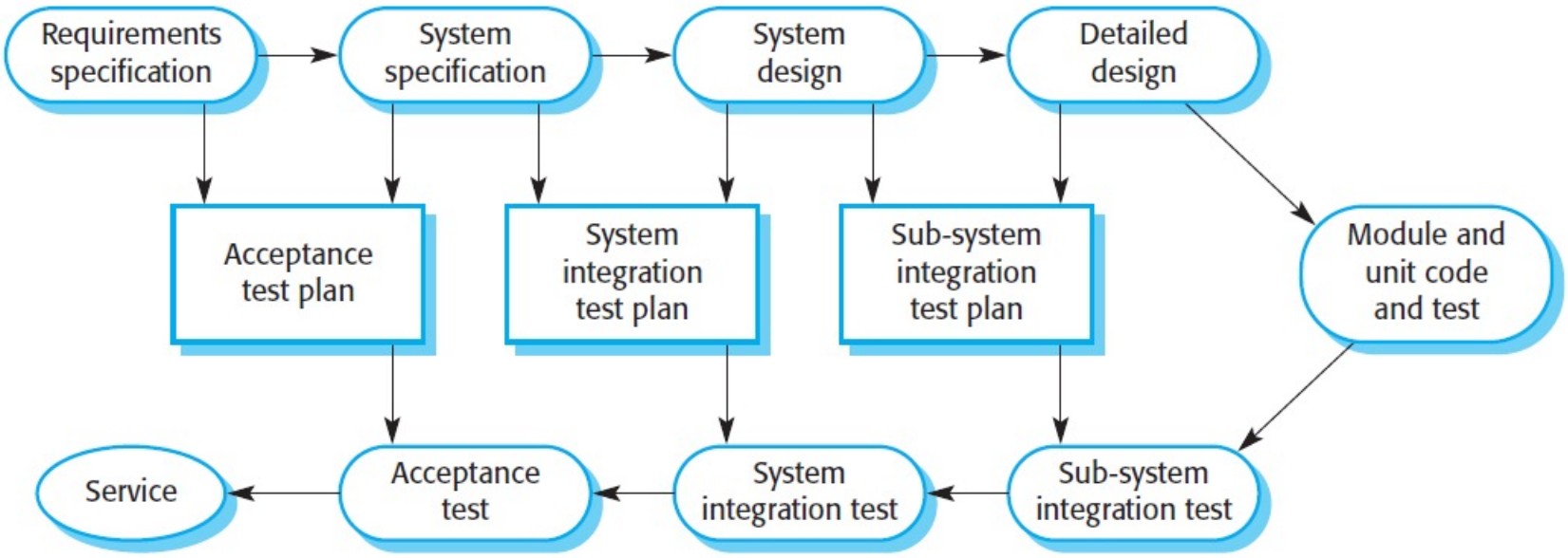
Program/software testing

- Chỉ ra các lỗi còn tồn tại trong hệ thống (lưu ý: không đảm bảo sẽ chỉ hết)
- Là kỹ thuật duy nhất để kiểm tra các yêu cầu phi chức năng vì phải cho hệ thống thực hiện mới thấy được chúng hoạt động ra sao.
- Thông thường cần kết hợp với việc verification tĩnh để đảm bảo đã kiểm tra đầy đủ V&V.

Các loại testing

- **Tìm lỗi**
  - Các test được thiết kế để chỉ ra các lỗi.
  - Hệ thống kiểm tra lỗi là thành công khi nó chỉ ra được các lỗi đang có trong hệ thống.
- **Validation testing**
  - Mục tiêu cho thấy sản phẩm phù hợp với các yêu cầu của khách hàng.

V-model


Requirements specification
System specification
System design
Detailed design
Acceptance test plan
System integration test plan
Sub-system integration test plan
Module and unit code and test
Service
Acceptance test
System integration test
Sub-system integration test

Software inspections

- Thanh tra, xem xét và tìm ra các lỗi, các tình huống bất thường.
- Thanh tra không cần phải thực thi/chạy sản phẩm. Do vậy, việc thanh tra có thể thực hiện trước khi cài đặt.
- Việc thanh tra có thể được áp dụng cho nhiều nơi trong hệ thống: yêu cầu phần mềm, thiết kế, dữ liệu cấu hình, dữ liệu test v.v...
- Thanh tra phần mềm là một kỹ thuật hiệu quả để phát hiện ra các lỗi của chương trình.

Ưu điểm của việc thanh tra

- Trong 1 lần thanh tra, có thể có nhiều lỗi khác nhau được phát hiện. Trong testing, lỗi này có thể che đi lỗi khác → cần thực hiện nhiều lần.
- Việc áp dụng lại các kiến thức về lập trình và kiến thức của domain tương ứng, khi đó người thanh tra có thể thấy được các loại lỗi thường xảy ra.

Inspection và testing

- Thanh tra và kiểm tra là 2 kỹ thuật hỗ trợ lẫn nhau. Kỹ thuật này không thể thay thế kỹ thuật kia.
- Cả hai cần được áp dụng trong tiến trình V&V
- Thanh tra có thể kiểm tra sự phù hợp và đáp ứng của hệ thống so với đặc tả chứ không kiểm tra được sự đáp ứng với yêu cầu thật sự của khách hàng.
- Thanh tra không thể kiểm tra được các tính chất phi chức năng, chẳng hạn: performance hay usability v.v...

Thanh tra chương trình

- Là cách tiếp cận hình thức cho việc review
- Mục tiêu: tìm ra các lỗi (không phải sửa lỗi).
- Các lỗi có thể là lỗi logic, các sự bất thường trong mã nguồn mà nó có thể dẫn đến các trạng thái lỗi (ví dụ: quên khởi tạo biến) hay không đáp ứng các tiêu chuẩn v.v...

Quy trình thanh tra

- Giới thiệu tổng quan về hệ thống cho nhóm thanh tra.
- Mã nguồn và các tài liệu liên quan được cung cấp trước cho nhóm thanh tra.
- Tiến hành việc thanh tra và các lỗi phát hiện được ghi nhận lại.
- Các chỉnh sửa, thay đổi để sửa chữa các lỗi phát hiện.
- Việc thanh tra lại có thể được thực hiện hoặc không cần thực hiện.

Checklists cho việc thanh tra

- Danh sách các lỗi thường gặp
- Danh sách lỗi thường phụ thuộc vào ngôn ngữ lập trình và phản ánh các đặc tính lỗi thường có thể xảy ra đối với ngôn ngữ lập trình đó.
- Một số ví dụ:
  - Giá trị khởi tạo
  - Phạm vi của mảng
  - Điều kiện dừng của vòng lặp v.v...

Hỗ trợ việc phân tích tự động

- Một số công cụ hỗ trợ việc phân tích tự động
- Các công cụ này phân tích các dòng lệnh và chỉ ra các điều kiện có thể xảy ra lỗi và cung cấp chúng cho nhóm thanh tra.
- Các công cụ này rất hiệu quả trong việc hỗ trợ thanh tra. Tuy nhiên, chúng không thể thay thế việc thanh tra.

Đặc tả hình thức

- Đặc tả hình thức được áp dụng khi hệ thống có thể được mô tả bằng các đặc tả toán học.
- Đặc tả hình thức là kỹ thuật rất tốt cho việc verification tĩnh.
- Kỹ thuật này phân tích toán học chi tiết trên các specification và đưa ra các tham số hình thức để đảm bảo chương trình đáp ứng với các đặc tả toán học.

Software testing: tiến trình

- Component testing
  - Kiểm tra từng component (unit test);
  - Thông thường người lập trình sẽ thực thi
  - Các test được tạo ra dựa trên kinh nghiệm của người phát triển phần mềm.
- System testing
  - Kiểm tra một nhóm các component được tích hợp.
  - Thông thường được thực hiện bởi nhóm độc lập.
  - Các kiểm tra dựa trên đặc tả của hệ thống.

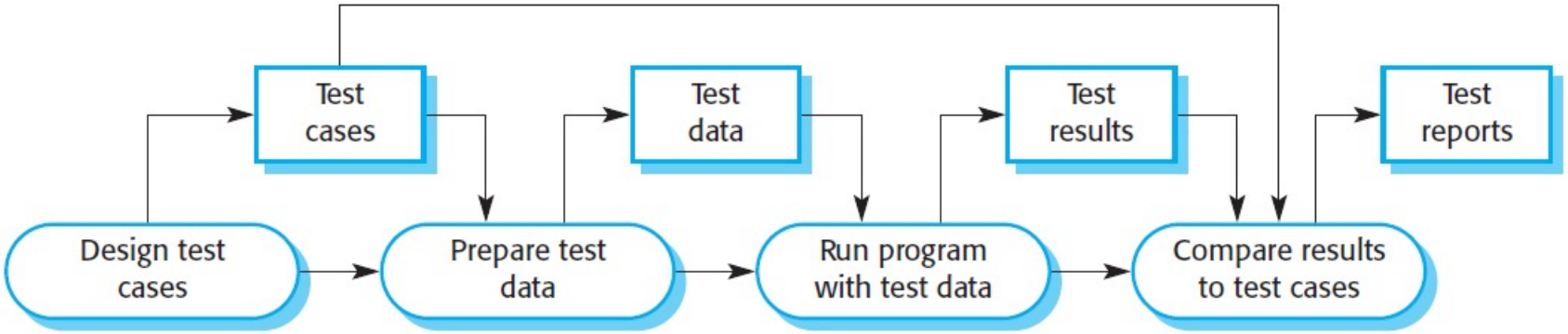
Defect testing

- Mục tiêu tìm ra các lỗi của chương trình.
- Sự thành công của defect testing là đưa ra một cách test để làm cho hệ thống chạy sai.

Mục tiêu của tiến trình testing

- **Validation testing**
  - Cho developer và khách hàng thấy được sản phẩm đáp ứng các yêu cầu và hoạt động theo như mong muốn.
- **Defect testing**
  - Phát hiện các lỗi trong chương trình so với đặc tả.

Tiến trình software testing



System testing

- Tích hợp các component lại thành sub-system hay system và kiểm tra
- Có thể kiểm tra từng sản phẩm trung gian trước khi giao cho khách hàng.
- Bao gồm 2 giai đoạn:
  - **Integration testing**: nhóm test sẽ truy cập vào source code và kiểm tra sub-system hay toàn bộ system khi tích hợp
  - **Release testing**: nhóm test sẽ kiểm tra toàn bộ hệ thống sẽ được giao cho khách hàng như một black-box.

Integration testing

- Liên quan đến việc hệ thống được xây dựng từ các thành phần khác nhau. Việc kiểm tra sẽ xem xét sự tương tác giữa các thành phần
- Top-down integration và bottom-up integration
- Để đơn giản hóa việc xác định các lỗi, hệ thống cần được tích hợp từ từ và từng phần.

Release testing

- Tiến trình kiểm tra sản phẩm cuối cùng trước khi giao cho khách hàng.
- Mục tiêu để đảm bảo hệ thống thật sự đáp ứng các yêu cầu của khách hàng.
- Release testing thông thường là kiểm tra black-box hay kiểm tra chức năng (functional testing)
  - Dựa trên đặc tả của hệ thống.
  - Các tester không cần biết việc hệ thống được triển khai ra sao.

Testing guidelines

- Các chỉ dẫn có thể gợi ý cho nhóm test trong việc chọn các dữ liệu test có thể gây ra lỗi trong hệ thống:
  - Các dạng dữ liệu mà hệ thống sẽ xuất ra thông báo lỗi.
  - Dữ liệu có thể làm tràn vùng đệm
  - Lặp đi lặp lại các chuỗi dữ liệu nhập nhiều lần
  - Tìm cách “ép” hệ thống cho ra các kết quả khoong phù hợp
  - “Ép” việc tính toán lên thật lớn hay xuống thật nhỏ.

Use cases

- Use case có thể là nền tảng để sinh ra các test cho hệ thống. Các use case giúp xác định các hoạt động nào cần test và giúp thiết kế ra các test case.
- Từ các lược đồ sequence, nhóm test sẽ xác định được các input và output của các test

Performance testing

- Một phần trong release test có thể có thêm performance test và reliability testing.
- Performance test thông thường liên quan đến việc có danh sách các test với mức độ tăng dần đều cho đến khi hệ thống không thể đáp ứng được nữa.

Stress testing

- Là việc ép hệ thống chạy hơn giới hạn cao nhất của thiết kế. Việc ép hệ thống chạy như vậy có thể giúp chỉ ra thêm một số lỗi.
- Stress test cũng giúp cho thấy tình trạng của hệ thống trong trường hợp không đáp ứng được. Hệ thống phải thể hiện trạng thái này tốt. Stress test cũng giúp phát hiện việc mất dữ liệu hay kết nối có ok không trong các trường hợp quá tải như thế này.
- Stress testing rất phù hợp đối với các hệ thống phân tán khi chúng phải chịu tải lớn có thể dẫn đến việc hoạt động trở nên kém hiệu quả nhiều.

Component testing hay unit test

- Component hay unit testing là tiến trình kiểm tra độc lập các component.
- Nó là tiến trình tìm ra các lỗi (defect)
- Component có thể là:
  - Một hàm bên trong đối tượng
  - Một lớp với nhiều hàm và thuộc tính.
  - Composite components.

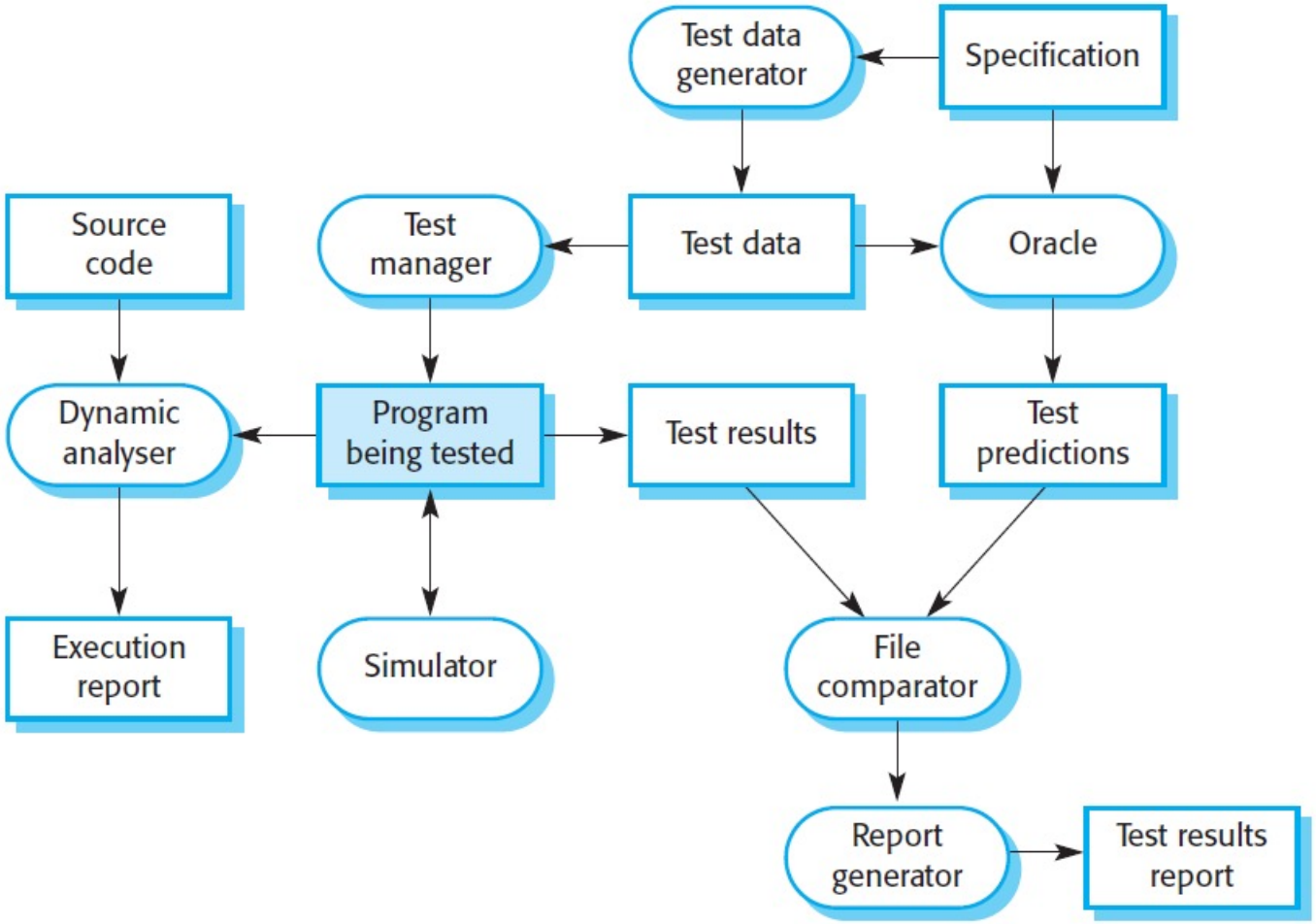
Thiết kế test case

- Thiết kế các test case (input & output) sẽ được sử dụng để kiểm tra hệ thống.
- Mục tiêu: tạo ra một tập hợp các bộ test để validate và phát hiện các lỗi.
- Một số cách tiếp cận
  - Requirements-based testing
  - Partition testing
  - Structural testing (white-box)

Test automation

- Testing là tiến trình tốn nhiều công sức.
- Có một số công cụ, chẳng hạn Junit, hỗ trợ việc thực thi các test một cách tự động.
- Tuy nhiên, đôi khi rất khó tích hợp các công cụ test tự động vì đặc tính “đóng” của hệ thống cần test.

A testing workbench



Một số thay đổi cho các công cụ test

- Các script có thể được áp dụng để mô phỏng tương tác người dùng và các patterns khi sinh dữ liệu test.
- Các test output có thể được chuẩn bị bằng tay để so sánh.
- Công cụ so sánh file có thể được viết thêm để so sánh theo yêu cầu.