

Kiểm thử phần mềm

Kiểm thử trong vòng đời phần mềm

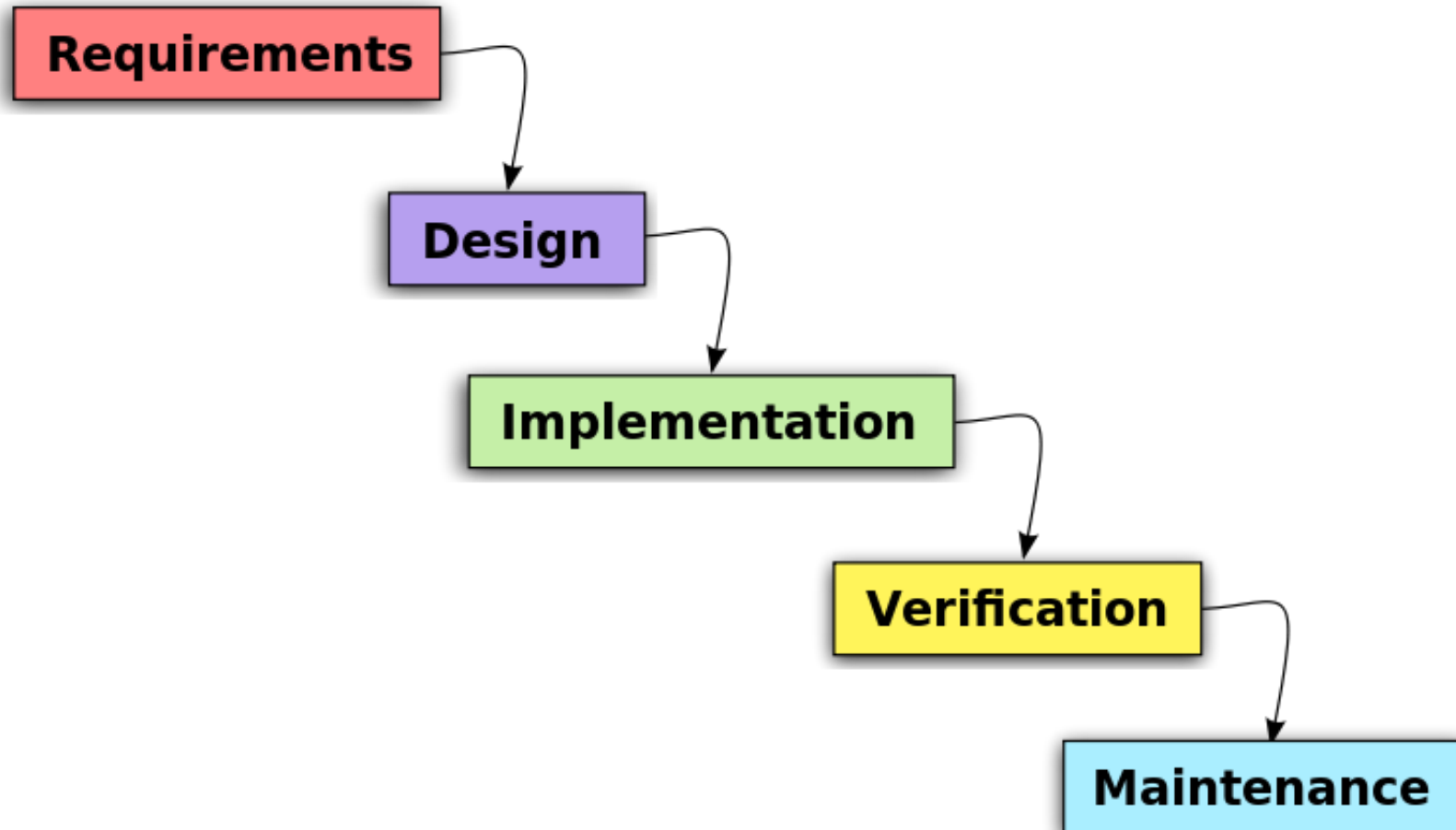


KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

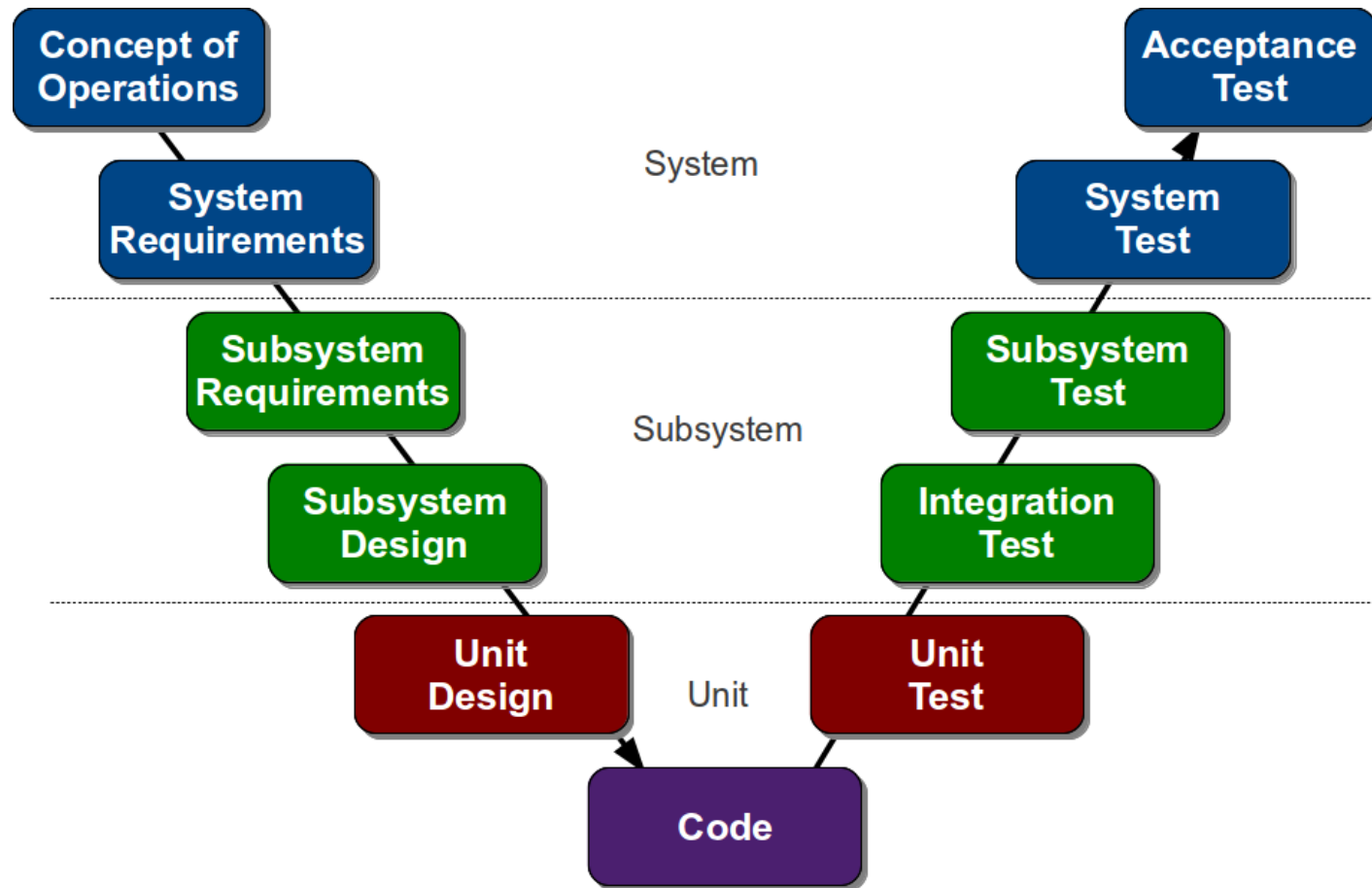
Nội dung

- ☐ **Kiểm thử trong vòng đời phần mềm**
- ☐ Các cấp độ kiểm thử
- ☐ Các loại kiểm thử

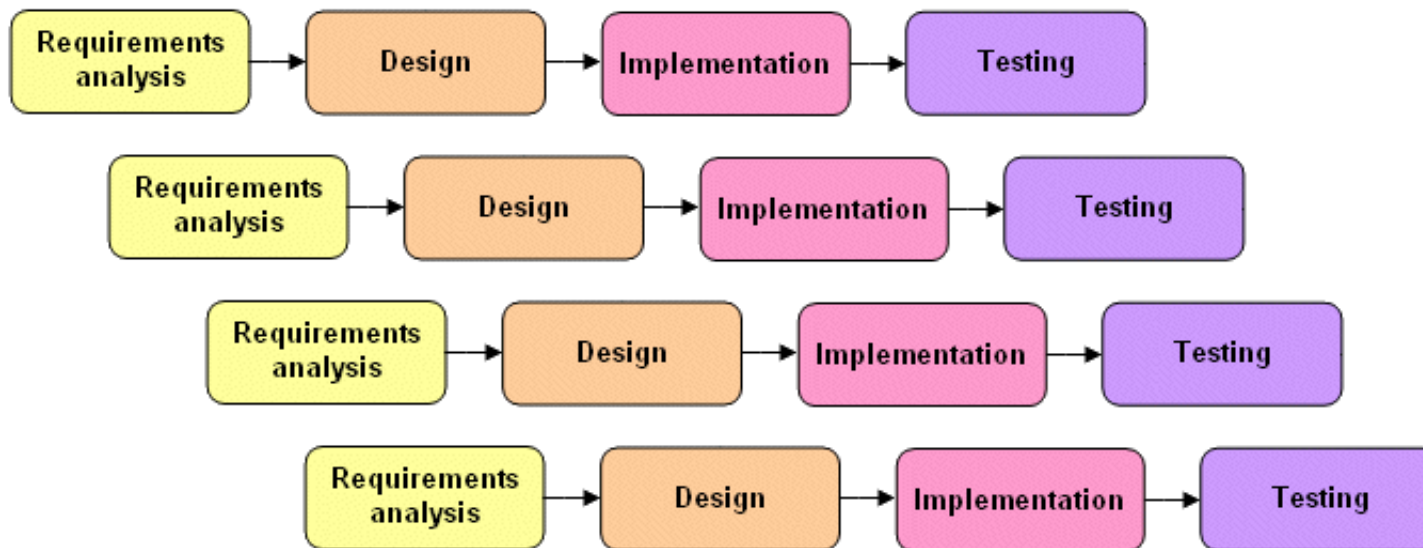
Mô hình thác nước



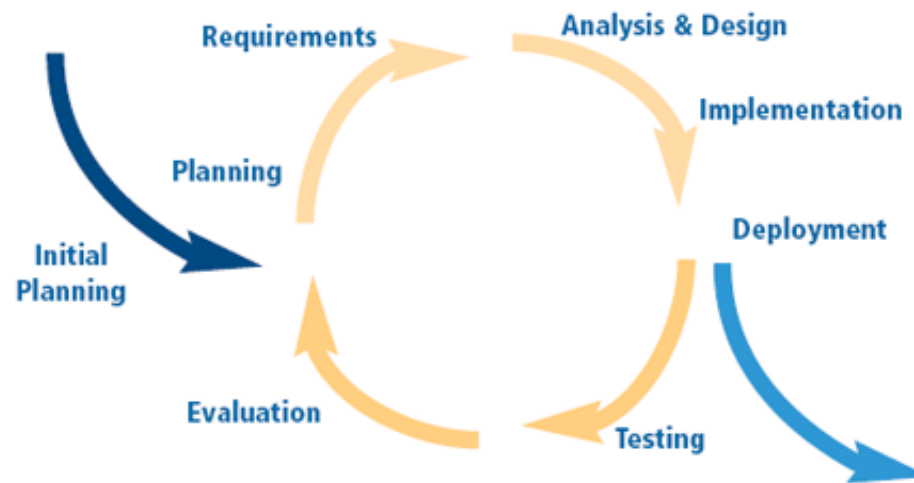
Mô hình chữ V



Mô hình gia tăng



Mô hình lập

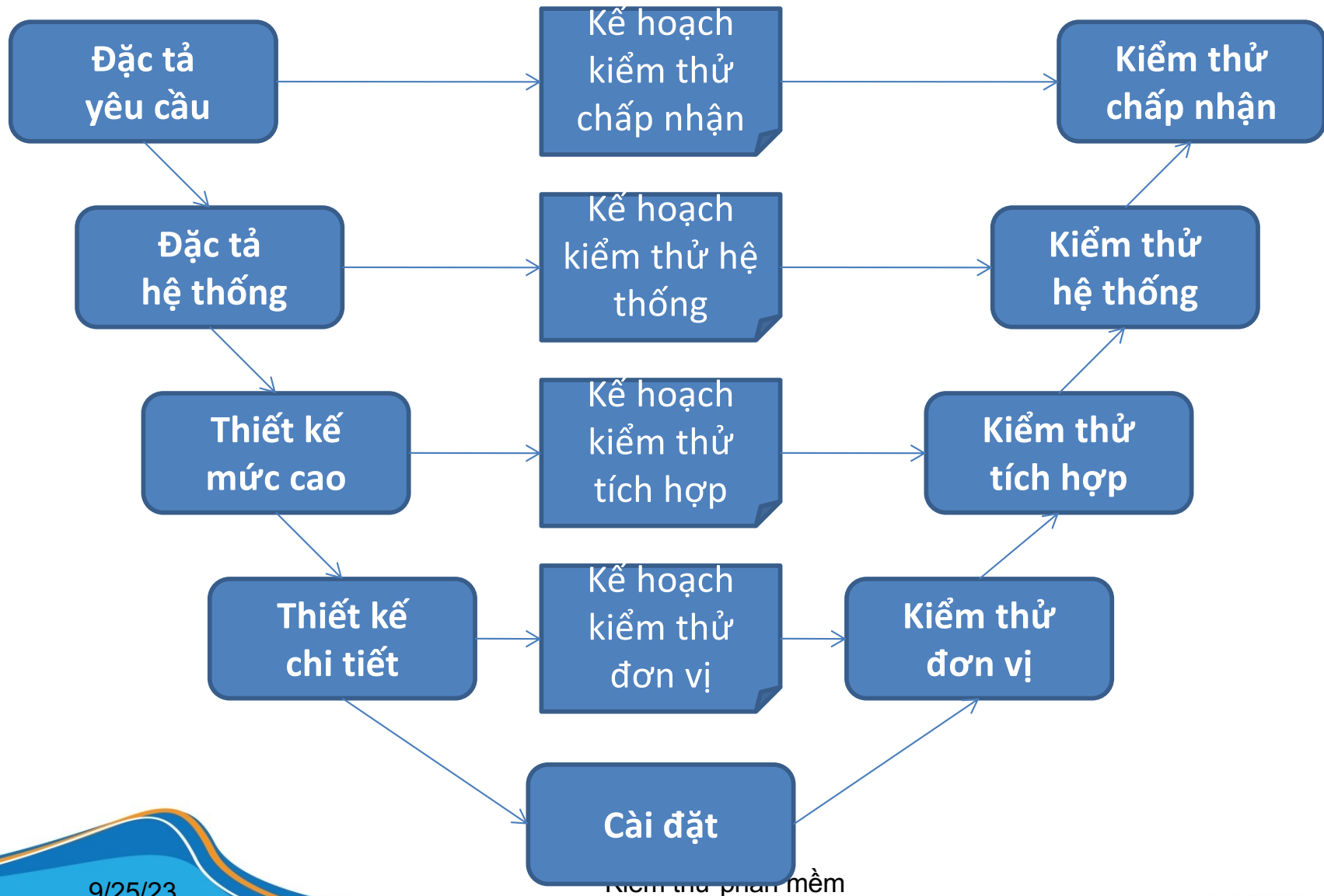


Iterative Development Model (Wikipedia)

Kiểm thử trong vòng đời phần mềm

- Đặc tính chung của kiểm thử tốt
 - ▣ Kiểm thử cho mỗi giai đoạn/phần phát triển
 - ▣ Các mức kiểm tra phối hợp liên tục, không trùng lặp
 - ▣ Phân tích, thiết kế bắt đầu sớm, ngăn ngừa lỗi

Kiểm thử trong vòng đời phần mềm



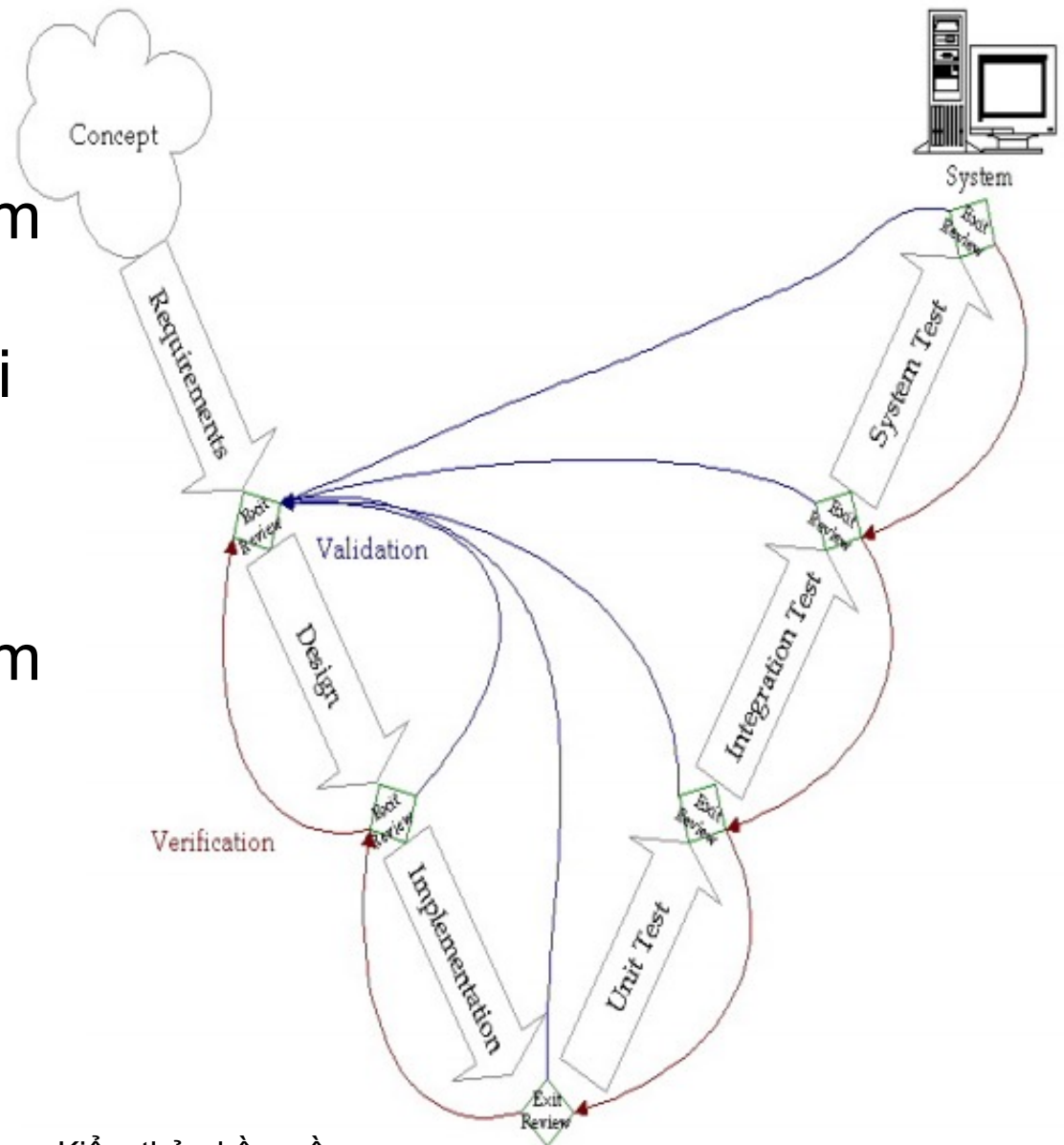
Kiểm thử trong vòng đời phần mềm

□ Verification

- Đảm bảo phần mềm được hiện thực đúng theo từng giai đoạn

□ Validation

- Đảm bảo phần mềm được xây dựng đúng theo yêu cầu khách hàng



Nội dung

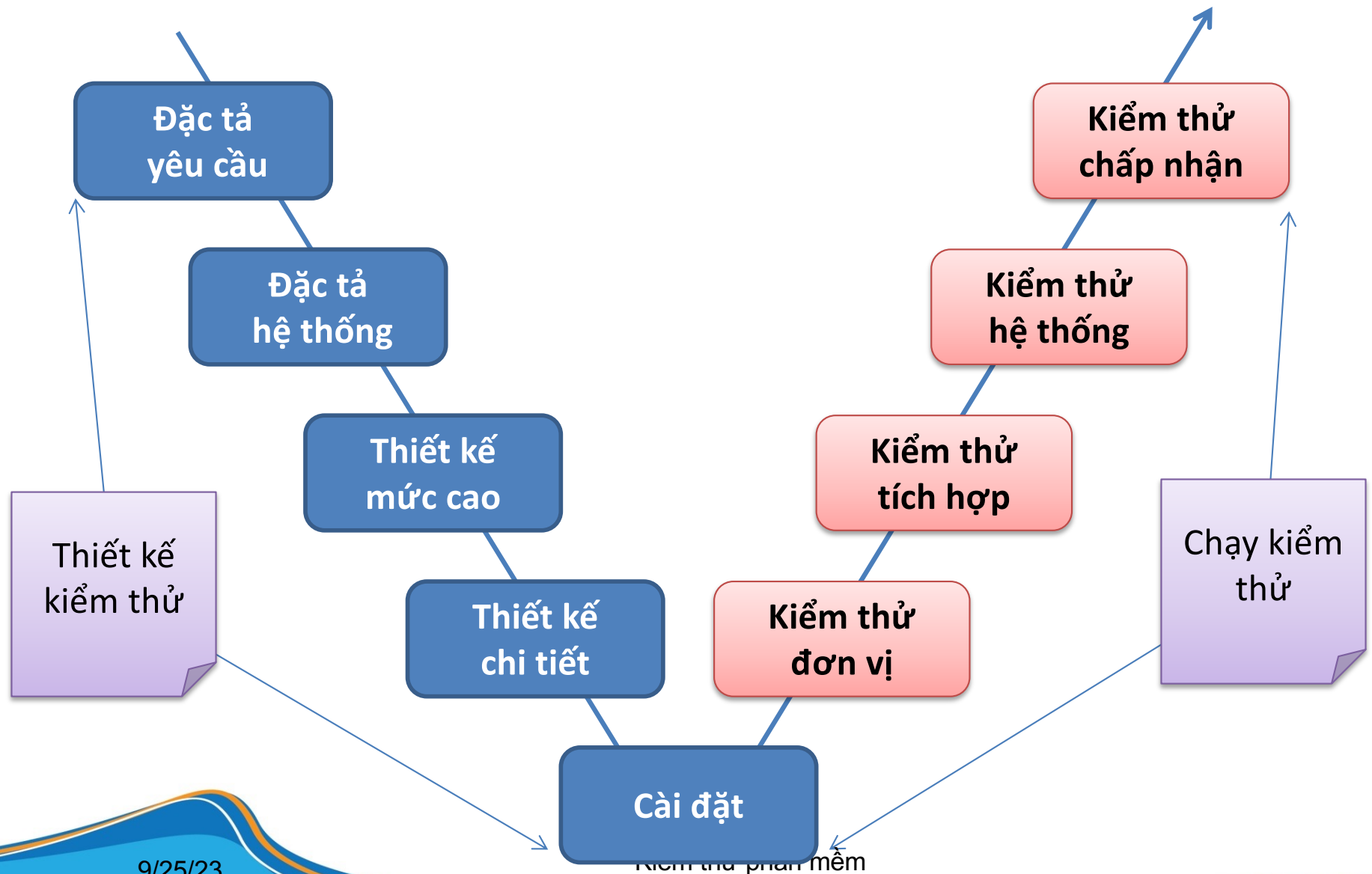
- ☐ Kiểm thử trong vòng đời phần mềm
- ☐ **Các cấp độ kiểm thử**
- ☐ Các loại kiểm thử

9/25/23

Kiểm thử phần mềm

10

Mô hình chữ V



Kiểm thử đơn vị – Unit testing

- ☐ Tên khác
 - ☐ Component testing
 - ☐ Module testing
 - ☐ Program testing
- ☐ Mỗi đơn vị được kiểm thử độc lập, trước khi tích hợp
- ☐ Mức thấp nhất và cụ thể, chi tiết nhất

Kiểm thử đơn vị – Unit testing

- Mục tiêu:
 - ▣ Đảm bảo mã nguồn từng đơn vị đúng theo đặc tả
 - ▣ Bao gồm chức năng và phi chức năng
- Dựa trên:
 - ▣ Yêu cầu
 - ▣ Thiết kế đơn vị
 - ▣ Mã nguồn

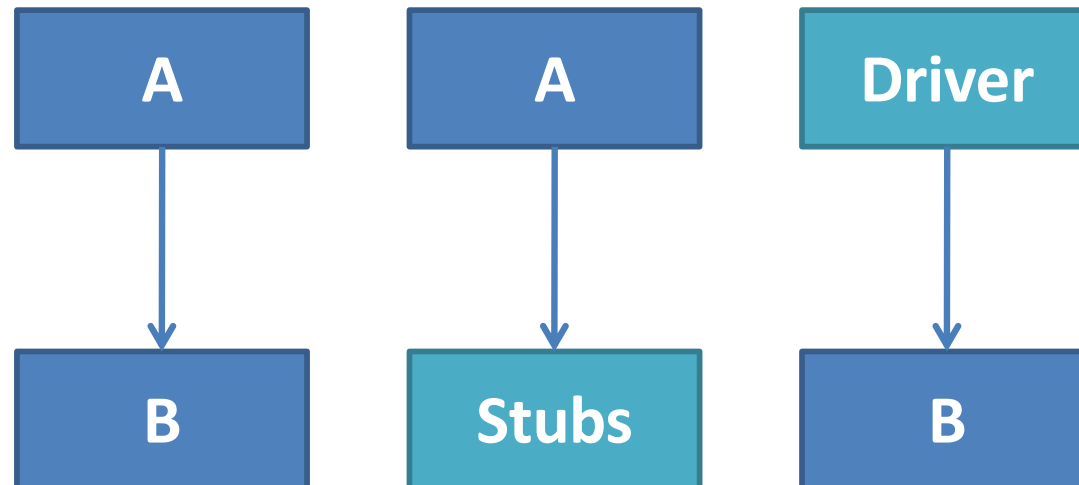


Kiểm thử đơn vị – Unit testing

- ☐ Ai thực hiện?
 - ☐ Lập trình viên
- ☐ Báo cáo
 - ☐ Lỗi được sửa ngay, không cần báo cáo
- ☐ Công cụ
 - ☐ Viết trực tiếp mã nguồn
 - ☐ Unit test framework
 - ☐ Mocking framework
 - ☐ Dependency Injection and IoC containers

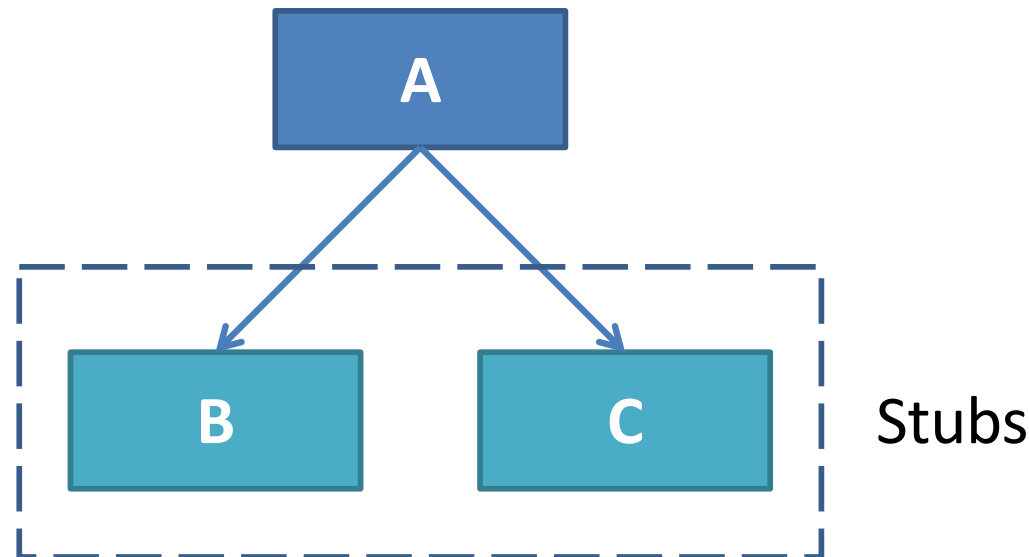
Kiểm thử đơn vị – Unit testing

- Stubs và Driver là các đơn vị giả lập
- Giá trị trả về
 - Cố định
 - Nhập vào



Kiểm thử đơn vị – Unit testing

- ☐ Đơn vị A có gọi đến đơn vị B và C
- ☐ Kiểm thử độc lập đơn vị A
 - ☒ Thay đơn vị B và C bằng các đơn vị giả lập (Stubs)



Kiểm thử đơn vị – Unit testing

- Test driven development
 - Test-first approach
 - Hướng tiếp cận phát triển phần mềm dựa trên Unit Test
 - Chuẩn bị và tự động hóa test case trước khi coding
 - Lập trình từng phần một → tất cả test case đều đạt

Kiểm thử tích hợp – Integration testing

- ☐ Kiểm tra hơn 2 đơn vị/hệ thống
- ☐ Mục tiêu:
 - ☐ Kiểm thử giao diện/sự tương tác giữa các đơn vị/hệ thống
 - ☐ Kiểm thử các tập không hoạt động độc lập
 - ☐ Kiểm thử chức năng và phi chức năng
- ☐ Dựa trên:
 - ☐ Thiết kế phần mềm
 - ☐ Kiến trúc phần mềm
 - ☐ Workflows/Use-cases



Kiểm thử tích hợp – Integration testing

- ☐ Hai cấp độ
 - ☐ Kiểm thử tích hợp đơn vị
 - ☐ Kiểm thử tích hợp hệ thống
- ☐ Ai thực hiện?
 - ☐ Người phát triển
 - ☐ Người thiết kế
 - ☐ Người kiểm thử độc lập



Kiểm thử tích hợp – Integration testing

- Chiến lược
 - Big-bang
 - Incremental (gia tăng)

9/25/23

Kiểm thử phần mềm

20

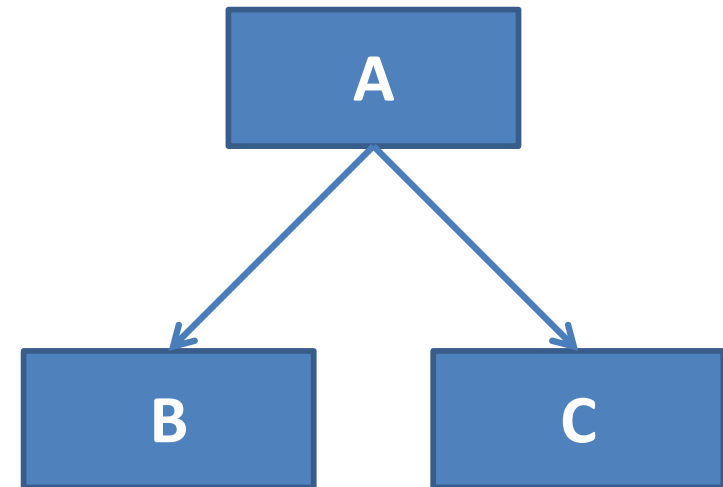
Kiểm thử tích hợp – Integration testing

□ Big-bang integration

□ Kiểm thử tích hợp mọi đơn vị một lần

□ Ví dụ:

- Kiểm thử đơn vị A, B, C
- Kiểm thử tích hợp A+B+C





Kiểm thử tích hợp – Integration testing

□ Big-bang integration

□ Ưu điểm

- Mọi đơn vị đã hoàn thành trước kiểm thử
- Không cần giả lập các đơn vị tích hợp phức tạp

□ Nhược điểm

- Tốn thời gian
- Khó định vị lỗi

Kiểm thử tích hợp – Integration testing

□ Incremental integration

- Bắt đầu với 1 đơn vị, thêm dần 1 đơn vị và kiểm thử nó theo một đường dẫn cơ sở (baseline)
- Ưu điểm
 - Dễ định vị lỗi và sửa chữa
 - Có thể bắt đầu sớm
- Nhược điểm
 - Khó khăn trong giả lập các đơn vị phức tạp
- Phân loại
 - Top-down
 - Bottom-up
 - Functional



Kiểm thử tích hợp – Integration testing

□ Top-down Integration

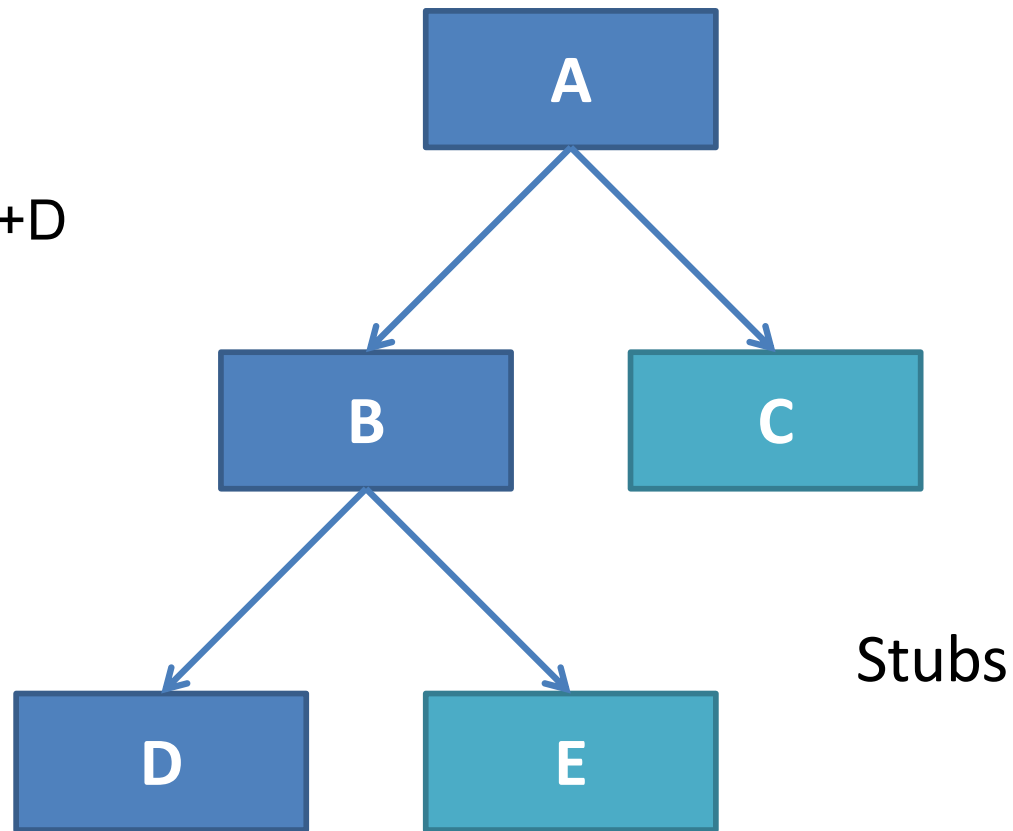
- Kiểm thử đơn vị ở mức cao trước, rồi tích hợp dần các đơn vị mức thấp hơn
- 2 cách
 - Tích hợp theo chiều sâu (breath-first)
 - Tích hợp theo chiều ngang (depth-first)

Kiểm thử tích hợp – Integration testing

□ Tích hợp theo chiều sâu (depth-first)

□ Ví dụ:

- Baseline0: A
- Baseline0: A+B
- Baseline0: A+B+D
- ...

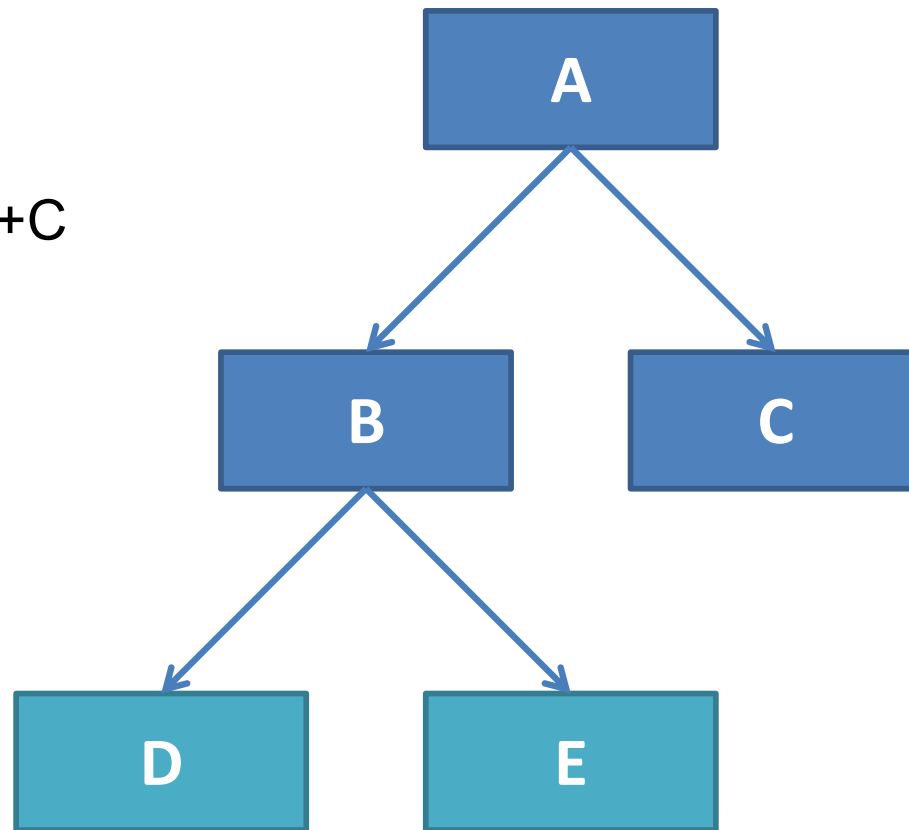


Kiểm thử tích hợp – Integration testing

□ Tích hợp theo chiều ngang (breath-first)

□ Ví dụ:

- Baseline0: A
- Baseline1: A+B
- Baseline2: A+B+C
- ...



Stubs

Kiểm thử phần mềm



Kiểm thử tích hợp – Integration testing

□ Top-down Integration

□ Ưu điểm

- Phát hiện sớm các lỗi thiết kế
- Có phiên bản hoạt động sớm

□ Khuyết điểm

- Khó mô phỏng các đơn vị cấp thấp có chức năng phức tạp
- Không kiểm thử đầy đủ các chức năng chi tiết

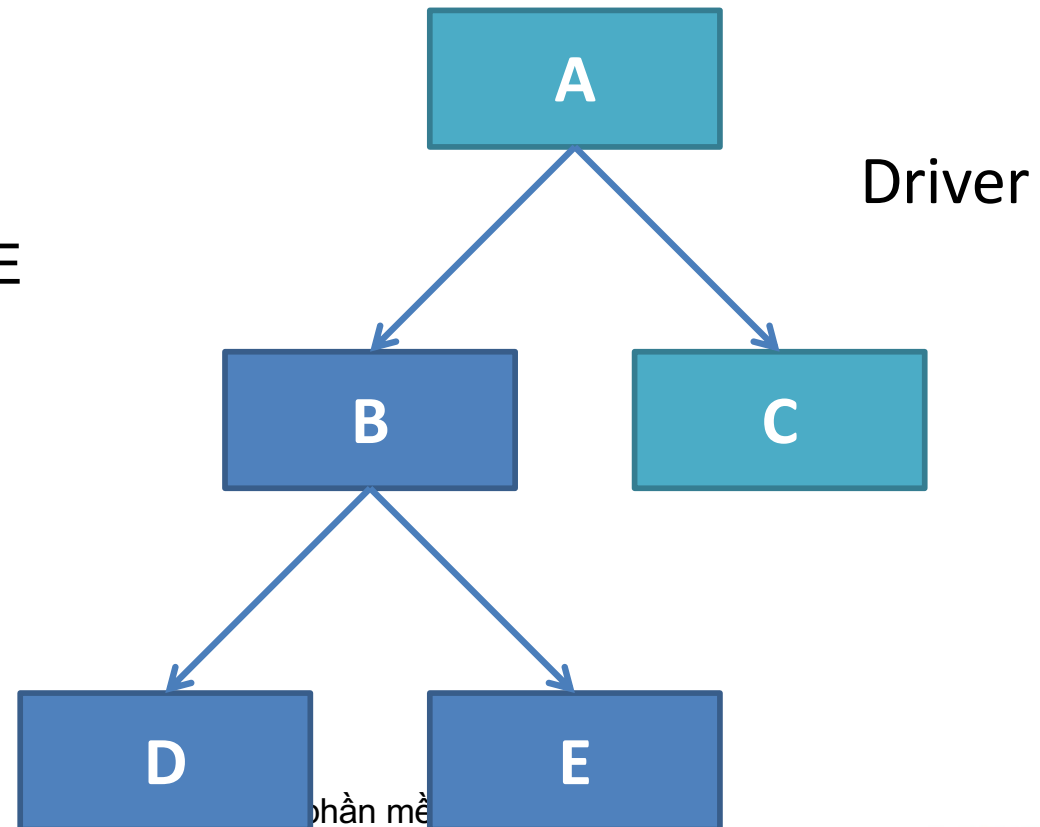
Kiểm thử tích hợp – Integration testing

□ Bottom-up Integration

□ Kiểm thử đơn vị ở mức thấp trước, rồi tích hợp dần các đơn vị mức cao hơn

□ Ví dụ:

- Baseline0: D
- Baseline1: D,B
- Baseline2: D,B,E
- ...





Kiểm thử tích hợp – Integration testing

- Bottom-up Integration
 - ▣ Các đơn vị ở mức thấp nhất được tích hợp thành các nhóm thể hiện một chức năng của phần mềm
 - ▣ Một driver được tạo ra để thao tác các test case
 - ▣ Nhóm đơn vị được kiểm nghiệm
 - ▣ Driver được bỏ đi và các nhóm đơn vị được tích hợp dần lên phía trên trong sơ đồ phân cấp chương trình



Kiểm thử tích hợp – Integration testing

□ Bottom-up Integration

□ Ưu điểm

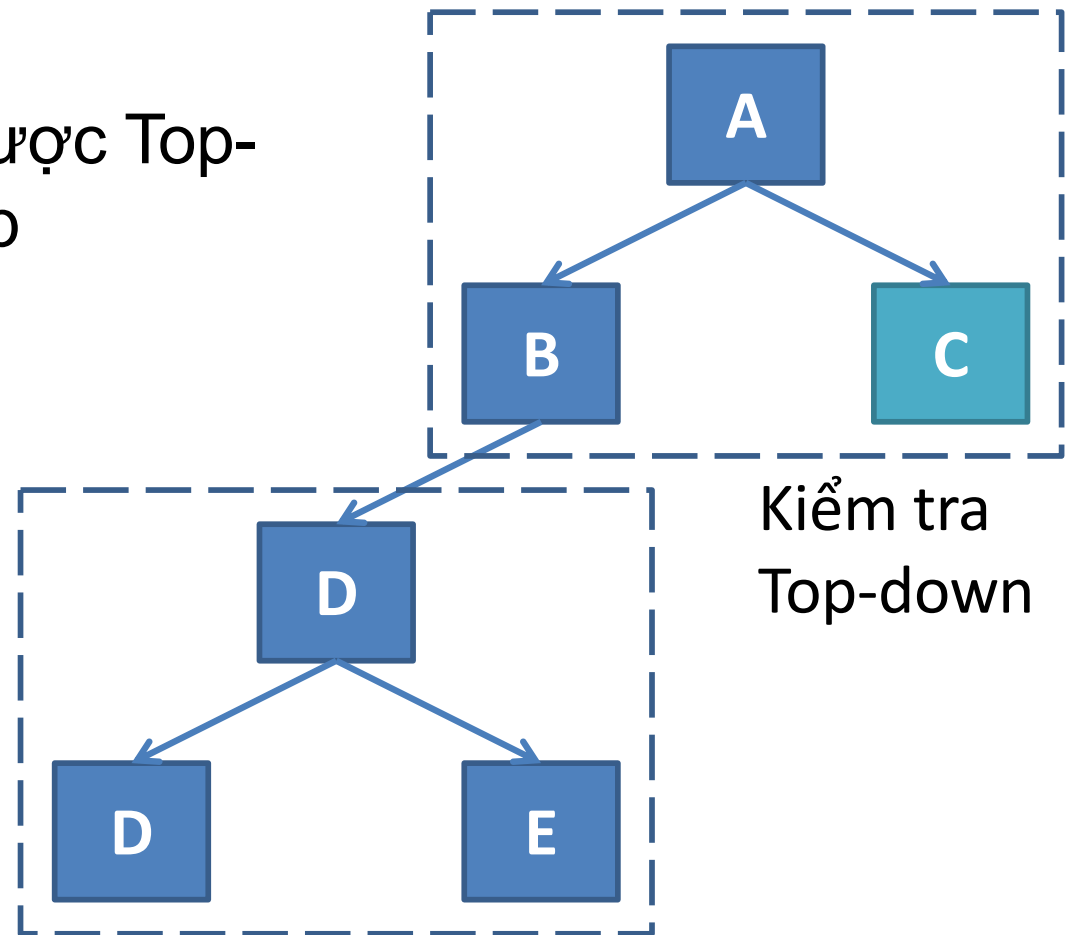
- Trách tạo đơn vị giả lập có chức năng phức tạp
- Thuận tiện phát triển các đơn vị cấp thấp dùng lại được

□ Nhược điểm

- Phát hiện chậm lỗi thiết kế
- Chậm có phiên bản thực hiện được

Kiểm thử tích hợp – Integration testing

- Sandwich testing
 - ▣ Phối hợp 2 chiến lược Top-down và Bottom-up



Các đơn vị được tích
hợp theo chức năng

Kiểm thử phần mềm



Kiểm thử tích hợp – Integration testing

- Functional incremental
 - ▣ Tích hợp và kiểm thử các đơn vị theo thứ tự thực hiện của một chức năng

9/25/23

Kiểm thử phần mềm

32

Kiểm thử hệ thống – System testing

- ☐ Là bước cuối cùng của kiểm thử tích hợp
- ☐ Kiểm thử hệ thống như một tổng thể
- ☐ Mục tiêu:
 - ☐ Phát hiện sai sót trong toàn bộ hệ thống chạy trên môi trường
 - ☐ Kiểm thử chức năng và phi chức năng
- ☐ Dựa trên:
 - ☐ Đặc tả yêu cầu phần mềm
 - ☐ Use case
 - ☐ Tài liệu hướng dẫn sử dụng



Kiểm thử hệ thống – System testing

- Ai thực hiện?
 - ▣ Thường và nên nhóm kiểm thử độc lập
- Phân loại
 - ▣ Kiểm thử chức năng – Functional testing
 - ▣ Kiểm thử phi chức năng – Non-functional testing

Kiểm thử chấp nhận – Acceptance testing

- ☐ Bước cuối cùng của validation
- ☐ Mục tiêu:
 - ☐ Xác nhận từ phía người dùng hệ thống đáp ứng đúng mong đợi của người dùng
- ☐ Dựa trên
 - ☐ Đặc tả yêu cầu
- ☐ Ai thực hiện?
 - ☐ Khách hàng/Người sử dụng
 - ☐ Có thể bao gồm kiểm thử viên

□ Alpha testing và Beta testing

□ Giống

- Khi phần mềm đã ổn định
- Nhận phản hồi về lỗi, mong đợi, đề xuất

□ Khác

- Alpha testing thực hiện tại môi trường phát triển
- Beta testing thực hiện tại môi trường thực tế

Nội dung

- ☐ Kiểm thử trong vòng đời phần mềm
- ☐ Các cấp độ kiểm thử
- ☐ **Các loại kiểm thử**

Các loại kiểm thử

- Kiểm thử chức năng
 - ▣ Functional testing/Black-box testing
- Kiểm thử phi chức năng
 - ▣ Non-functional testing
- Kiểm thử cấu trúc
 - ▣ Structural testing/White-box testing
- Kiểm thử liên quan thay đổi
 - ▣ Confirmation testing/Re-testing & Regression testing

Kiểm thử chức năng

- ☐ Functional testing/Black-box testing
- ☐ Dựa trên đặc tả chức năng
- ☐ Phát hiện sai sót về chức năng
- ☐ Không quan tâm đến cách cài đặt

Kiểm thử chức năng

- Các kỹ thuật
 - ▣ 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)
 - ▣ Kiểm thử dựa trên use case (Use case testing)

Kiểm thử phi chức năng

- ☐ Kiểm thử hiệu năng – Performance testing
- ☐ Kiểm thử tính tiện dụng – Usability testing
- ☐ Kiểm thử bảo mật – Security testing
- ☐ Kiểm thử cấu hình/cài đặt –
Configuration/Installation testing
- ☐ Kiểm thử sao lưu/khôi phục – Back-
up/Recovery testing

Kiểm thử phi chức năng

- Kiểm thử hiệu năng – Performance testing
 - Kiểm thử khối lượng – Volume testing
 - Kiểm tra khả năng xử lý dữ liệu lớn của hệ thống
 - Kiểm thử tải/quá tải – Load/Stress testing
 - Kiểm tra yêu cầu về thời gian đáp ứng của hệ thống

Kiểm thử phi chức năng

- Kiểm thử tính tiện dụng – Usability testing
 - Dễ học, sử dụng đơn giản
 - Hiệu quả khi sử dụng
 - Giao diện đơn giản, đồng nhất
 - Hỗ trợ thông tin phản hồi
 - Ngăn ngừa lỗi
 - Liên kết tắt
 - Thông điệp báo lỗi tốt
 - ...

Kiểm thử phi chức năng

- Kiểm thử bảo mật – Security testing
 - ▣ Kiểm tra tính hợp lệ của việc truy xuất trong và ngoài chương trình

Kiểm thử phi chức năng

- Kiểm thử cấu hình/cài đặt –
Configuration/Installation testing
 - Kiểm tra cấu hình
 - Phần cứng, môi trường phần mềm khác nhau
 - Cấu hình bản thân phần mềm
 - Độ độ nâng cấp phiên bản
 - Kiểm tra cài đặt
 - Gói cài đặt (CD, mạng, ...)
 - Uninstall

Kiểm thử phi chức năng

- Kiểm thử sao lưu/khôi phục – Back-up/Recovery testing
 - ▣ Kiểm tra khả năng sao lưu và khôi phục hệ thống từ sự cố

Kiểm thử cấu trúc

- ☐ Có nghiên cứu mã nguồn
- ☐ Phân tích thứ tự thực hiện các lệnh

Kiểm thử cấu trúc

- Phương pháp bao phủ mã lệnh (code coverage)
 - ▣ Bao phủ câu lệnh (Statement coverage)
 - ▣ Bao phủ nhánh (Decision coverage)
 - ▣ Bao phủ điều kiện (Condition coverage)
 - ▣ Bao phủ quyết định đa điều kiện (Multiple condition coverage)
 - ▣ Bao phủ lặp (Loop coverage)

Kiểm thử liên quan thay đổi

- ☐ Kiểm tra sau khi lỗi được sửa chữa
- ☐ Kiểm thử lại – Re-testing/Confirmation testing
 - ☐ Kiểm tra lại chính xác trường hợp kiểm thử đã phát hiện ra lỗi
 - ☐ Xác nhận lỗi đã được sửa chữa
 - ☐ → Không bảo đảm lỗi mới không phát sinh
- ☐ Kiểm thử hồi qui – Regression testing
 - ☐ Kiểm tra lại tất cả các trường hợp kiểm thử đã thỏa trước đó
 - ☐ Tìm ra các lỗi mới phát sinh

Thảo luận

