# Nhóm :

1. Lê Xuân Thành
2. Phan Thị Châu Quỳnh
3. Trần Văn Phông

# CÔNG CỤ UNIT TEST

Đề tài: PHẦN MỀM QUẢN LÝ PHÒNG VÉ.

Unit Test (UT) là một kỹ thuật quan trọng góp phần nâng cao chất lượng phần mềm (PM), nhưng có nhiều quan điểm trái ngược nhau về việc đưa UT vào quy trình phát triển PM. Bài viết này giới thiệu một cái nhìn tổng quan về UT và mô hình phát triển phần mềm hiện đại TDD (Test-Driven Development).

**UNIT TEST**  
UT là kỹ thuật kiểm nghiệm các hoạt động của mọi chi tiết mã (code) với một quy trình tách biệt với quy trình phát triển PM, giúp phát hiện sai sót kịp thời. UT còn có thể giúp phát hiện các vấn đề tiềm ẩn và các lỗi thời gian thực ngay cả trước khi chuyên viên kiểm định chất lượng (QA - Quality Assurance) tìm ra, thậm chí có thể sửa lỗi ngay từ ý tưởng thiết kế.  
  
UT là các đoạn mã có cấu trúc giống như các đối tượng được xây dựng để kiểm tra từng bộ phận trong hệ thống. Mỗi UT sẽ gửi đi một thông điệp và kiểm tra câu trả lời nhận được đúng hay không, bao gồm:   
  
• Các kết quả trả về mong muốn  
  
• Các lỗi ngoại lệ mong muốn  
  
Các đoạn mã UT hoạt động liên tục hoặc định kỳ để thăm dò và phát hiện các lỗi kỹ thuật trong suốt quá trình phát triển, do đó UT còn được gọi là kỹ thuật kiểm nghiệm tự động.  
  
UT có các đặc điểm sau:  
  
• Đóng vai trò như những người sử dụng đầu tiên của hệ thống.  
  
• Chỉ có giá trị khi chúng có thể phát hiện các vấn đề tiềm ẩn hoặc lỗi kỹ thuật.

**Vòng đời của UT**

UT có 3 trạng thái cơ bản:  
  
• Fail (trạng thái lỗi)  
  
• Ignore (tạm ngừng thực hiện)  
  
• Pass (trạng thái làm việc)  
  
Toàn bộ UT được vận hành trong một hệ thống tách biệt. Có rất nhiều PM hỗ trợ thực thi UT với giao diện trực quan. Thông thường, trạng thái của UT được biểu hiện bằng các màu khác nhau: màu xanh (pass), màu vàng (ignore) và màu đỏ (fail).  
  
UT chỉ thực sự đem lại hiệu quả khi:  
  
• Được vận hành lặp lại nhiều lần  
  
• Tự động hoàn toàn  
  
• Độc lập với các UT khác.

**Thiết kế UT**  
  
Mỗi UT đều được tiết kế theo trình tự sau:  
  
• Thiết lập các điều kiện cần thiết: khởi tạo các đối tượng, xác định tài nguyên cần thiết, xây dựng các dữ liệu giả...  
  
• Triệu gọi các phương thức cần kiểm tra.  
  
• Kiểm tra sự hoạt động đúng đắn của các phương thức.  
  
• Dọn dẹp tài nguyên sau khi kết thúc kiểm tra.  
  
**Ứng dụng của UT**  
• Kiểm tra mọi đơn vị nhỏ nhất là các thuộc tính, sự kiện, thủ tục và hàm.   
  
• Kiểm tra các trạng thái và ràng buộc của đối tượng ở các mức sâu hơn mà thông thường chúng ta không thể truy cập được.  
  
• Kiểm tra các quy trình (process) và mở rộng hơn là các khung làm việc(workflow - tập hợp của nhiều quy trình).

**Lợi ích của UT**

Thời gian đầu, người ta thường do dự khi phải viết UT thay vì tập trung vào viết mã cho các chức năng nghiệp vụ. Công việc viết UT có thể ngốn nhiều thời gian, Tuy nhiên UT đem lại lợi ích to lớn như:  
  
• Tạo ra môi trường lý tưởng để kiểm tra bất kỳ đoạn mã nào, có khả năng thăm dò và phát hiện lỗi chính xác, duy trì sự ổn định của toàn bộ PM và giúp tiết kiệm thời gian so với công việc gỡ rối truyền thống.  
  
• Phát hiện các thuật toán thực thi không hiệu quả, các thủ tục chạy vượt quá giới hạn thời gian.

• Phát hiện các vấn đề về thiết kế, xử lý hệ thống, thậm chí các mô hình thiết kế.  
  
• Phát hiện các lỗi nghiêm trọng có thể xảy ra trong những tình huống rất hẹp.  
  
• Tạo hàng rào an toàn cho các khối mã: Bất kỳ sự thay đổi nào cũng có thể tác động đến hàng rào này và thông báo những nguy hiểm tiềm tàng.

  UT là môi trường lý tưởng để tiếp cận các thư viện API bên ngoài một cách tốt nhất. Sẽ rất nguy hiểm nếu chúng ta ứng dụng ngay các thư viện này mà không kiểm tra kỹ lưỡng công dụng của các thủ tục trong thư viện. Dành ra thời gian viết UT kiểm tra từng thủ tục là phương pháp tốt nhất để khẳng định sự hiểu đúng đắn về cách sử dụng thư viện đó. Ngoài ra, UT cũng được sử dụng để phát hiện sự khác biệt giữa phiên bản mới và phiên bản cũ của cùng một thư viện.

Trong môi trường làm việc cạnh tranh, UT còn có tác dụng rất lớn đến năng suất làm việc:

 Giải phóng chuyên viên QA khỏi các công việc kiểm tra phức tạp.  
  
• Tăng sự tự tin khi hoàn thành một công việc. Chúng ta thường có cảm giác không chắc chắn về các đoạn mã của mình như liệu các lỗi có quay lại không, hoạt động của module hiện hành có bị tác động không, hoặc liệu công việc hiệu chỉnh mã có gây hư hỏng đâu đó...

• Là công cụ đánh giá năng lực của bạn. Số lượng các tình huống kiểm tra (test case) chuyển trạng thái "pass" sẽ thể hiện tốc độ làm việc, năng suất của bạn.