1. ***Tìm Hiểu Công Cụ Unit Test***

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).

**1.Khái Niệm:**

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.

**2.Vòng Đời Của Unit Test:**

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

**3.Thiết Kế Unit Test:**  
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.

**4.Ứng Dụng Unit Test:**

• 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).

***B. Tìm Hiểu Công Cụ Automation Test***

**1.Khái Niệm**

Trong lĩnh vực kiểm thử phần mềm, thì kiểm thử tự động hay còn gọi là Automation testing đóng một vai trò quan trọng góp phần nâng cao năng suất kiểm thử, giảm thiểu lỗi cũng như sự nhàm chán với việc kiểm thử bằng tay trong một thời gian dài hoặc lặp đi lặp lại.

**2.Ưu Điểm:**

- Độ tin cậy cao: công cụ kiểm thử tự động có sự ổn định cao hơn so với con người, đặc biệt trong trường hợp nhiều test cases, nên độ tin cậy cao hơn so với kiểm thử thủ công.  
- Khả năng lặp: công cụ kiểm thử tự động ra đời là để giúp cho các tester không phải lặp đi lặp lại các thao tác (ví dụ: nhập dữ liệu, click, check kết quả…) một cách nhàm chán với độ tin cậy và ổn định cao.  
- Khả năng tái sử dụng: với một bộ kiểm thử tự động, người ta có thể sử dụng cho nhiều phiên bản ứng dụng khác nhau, đây được gọi là tính tái sử dụng.  
- Tốc độ cao: do thực thi bởi máy nên tốc độ của kiểm thử tự động nhanh hơn nhiều so với tốc độ của con người. Nếu cần 5 phú để thực thi một test case một cách thủ công thì có thể người ta chỉ cần khoảng 30s để thực thi một cách tự động.  
- Chi phí thấp: nếu áp dụng kiểm thử tự động đúng cách, người ta có thể tiết kiệm được nhiều chi phí, thời gian và nhân lực, do kiểm thử tự động nhanh hơn nhiều so với kiểm thử thủ công, đồng thời nhân lực cần để thực thi và bảo trì scripts không nhiều.

**3.Nhược Điểm:**

- Khó mở rộng, khó bảo trì: trong cùng một dự án, để mở rộng phạm vi cho kiểm thử tự động khó hơn nhiều so với kiểm thử thủ công vì cập nhật hay chỉnh sửa yêu cầu nhiều công việc như debug, thay đổi dữ liệu đầu vào và cập nhật code mới.  
- Khả năng bao phủ thấp: do khó mở rộng và đòi hỏi nhiều kỹ năng lập trình nên độ bao phủ của kiểm thử tự động thấp xét trên góc nhìn toàn dự án.  
- Vấn đề công cụ và nhân lực: hiện nay cũng có nhiều công cụ hỗ trợ kiểm thử tự động khá tốt nhưng chúng vẫn còn nhiều hạn chế. Ngoài ra nhân lực đạt yêu cầu (có thể sử dụng thành thạo các công cụ này) cũng không nhiều.

Khi nào nên sử dụng công cụ kiểm thử tự động? Một số công cụ kiểm thử tự động phổ biến.