**Tìm hiểu unit test**

Unit Test (UT) là một kỹ thuật quan trọng góp phần nâng cao chất lượng phần mềm (PM)

**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**

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | http://www.pcworld.com.vn/pcworld/info/misc/2005/12/A0512_CN_116.jpg |   Một thí dụ trong NUnit. Nút xanh thể hiện các trạng thái "pass", nút đỏ ứng với trạng thái "fail" và nút vàng ứng với trạng thái "ignore". |

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.

**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, yuy 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.

**Chiến lược viết mã hiệu quả với UT**  
  
• Phân tích các tình huống có thể xảy ra đối với mã. Đừng bỏ qua các tình huống tồi tệ nhất có thể xảy ra, thí dụ dữ liệu nhập làm một kết nối cơ sở dữ liệu thất bại, ứng dụng bị treo vì một phép toán chia cho không, các thủ tục đưa ra lỗi ngoại lệ sai có thể phá hỏng ứng dụng một cách bí ẩn...  
  
• Mọi UT phải bắt đầu với trạng thái "fail" và chuyển trạng thái "pass" sau một số thay đổi hợp lý đối với mã chính.   
  
• Mỗi khi viết một đoạn mã quan trọng, hãy viết các UT tương ứng cho đến khi bạn không thể nghĩ thêm tình huống nào nữa.  
  
• Nhập một số lượng đủ lớn các giá trị đầu vào để phát hiện điểm yếu của mã theo nguyên tắc:

- Nếu nhập giá trị đầu vào hợp lệ thì kết quả trả về cũng phải hợp lệ  
  
- Nếu nhập giá trị đầu vào không hợp lệ thì kết quả trả về phải không hợp lệ  
  
• Sớm nhận biết các đoạn mã không ổn định và có nguy cơ gây lỗi cao, viết UT tương ứng để khống chế.  
  
• Ứng với mỗi đối tượng nghiệp vụ (business object) hoặc đối tượng truy cập dữ liệu (data access object), nên tạo ra một lớp kiểm tra riêng vì những lỗi nghiêm trọng có thể phát sinh từ đối tượng này.

• Để ngăn chặn các lỗi có thể phát sinh trở lại thực thi tự động tất cả UT mỗi khi có một sự thay đổi quan trọng, hãy làm công việc này mỗi ngày. Các UT lỗi cho chúng ta biết thay đổi nào là nguyên nhân gây lỗi.  
  
• Để tăng hiệu quả và giảm rủi ro khi viết các UT, cần sử dụng nhiều phương thức kiểm tra khác nhau. Hãy viết càng đơn giản càng tốt.  
  
• Cuối cùng, viết UT cũng đòi hỏi sự nỗ lực, kinh nghiệm và sự sáng tạo như viết PM.

Link video hướng dẫn: <https://www.youtube.com/watch?v=T4WRsNxkyXY>