Môn học

KIỂM THỬ PHẦN MỀM

Chương III
CÁC KỸ THUẬT KIỂM THỬ TĨNH

Nội dung

1. Khái niệm kiểm thử tĩnh
2. Kỹ thuật rà soát phần mềm
3. Kỹ thuật phân tích tĩnh

III.1 Khái niệm kiểm thử tĩnh

- Có 2 phương pháp kiểm thử:
  - Kiểm thử tĩnh (Static Testing)
    - Thường được thực hiện bằng tay hoặc tools
    - Không thực thi code
  - Kiểm thử động (Dynamic Testing)
    - Thực thi code
    - Chỉ áp dụng để kiểm thử Code

Loại “bug” dễ tìm ra nhờ Kiểm thử tĩnh

- Sai lệch chuẩn
- Sai/thiếu yêu cầu
- Lỗi thiết kế
- Code không thể bảo trì
- Đặc tả giao diện không thống nhất

Kiểm thử tĩnh - Static Testing

- Gồm 2 kiểu:
  - People-based (manually): Review
  - Tool-based: Static Analysis

III.2 Kỹ thuật rà soát - Review

- Quá trình rà soát
  - Là cuộc họp mà các cá nhân phát triển dự án, quản lí dự án, và khách hàng tham gia xem xét, đánh giá, phê duyệt sản phẩm được phát triển ở mỗi giai đoạn.

(IEEE 1990)

Mục tiêu rà soát

- Mục tiêu trực tiếp
  - Tìm ra các khiếm khuyết và mục cần sữa chữa, thay đổi, hoàn thiện
  - Xác định rủi ro mới
  - Tìm ra những sai lệch so với mẫu chuẩn
  - Phê duyệt sản phẩm
- Mục tiêu gián tiếp
  - Tạo ra cuộc họp trao đổi kiến thức về phương pháp, công cụ và kĩ thuật phát triển

Lợi ích của rà soát

- Thực hiện giai đoạn sớm → Nhận "feedback" sớm
- Cải tiến quá trình phát triển PM, giảm thời gian phát triển
- Giảm thời gian và chi phí kiểm thử
- Tìm lỗi hiệu quả, giảm mức độ lỗi → tăng chất lượng sản phẩm

Lợi ích về chi phí

- Chi phí test được giảm từ 50% -> 80%
- Giảm chi phí bảo trì
- Loại bỏ khoảng 80%-95% các lỗi ở mỗi bước

Cái gì có thể được rà soát?

- Chiến lược, kế hoạch phát triển, hợp đồng, nghiên cứu khả thi
- Đặc tả yêu cầu, thiết kế
- Code
- Test plan, test cases, test results
- Tài liệu người dùng, tài liệu training
- Tiến độ dự án
- ….

Phương pháp rà soát

- Rà soát cá nhân (không chính thức, rà soát chéo)
- Rà soát kỹ thuật chính thức (Formal Technical Review)
- Rà soát ngang hàng (Peer review)
  - Inspection
  - Walkthrough

Rà soát kỹ thuật chính thức(FTR)

- Là phương pháp rà soát duy nhất cần thiết để phê duyệt sản phẩm ở mỗi giai đoạn.
- FTRs phổ biến:
  - Rà soát kế hoạch phát triển
  - Rà soát đặc tả yêu cầu
  - Rà soát thiết kế sơ bộ, thiết kế chi tiết, thiết kế DB
  - Rà soát kế hoạch test, thủ tục test
  - Rà soát mô tả phiên bản
  - Rà soát hướng dẫn sử dụng, bảo trì PM
  - Rà soát kế hoạch cài đặt

Rà soát kỹ thuật chính thức(FTR)

- Quy trình rà soát chính thức gồm 6 bước:
  - Vạch kế hoạch
  - Khởi đầu
  - Chuẩn bị
  - Cuộc họp xét duyệt
  - Làm lại, chỉnh sửa
  - Theo dõi

Rà soát kỹ thuật chính thức(FTR)

- Vạch kế hoạch: Moderator/leader
  - Lên lịch biểu
  - Thực hiện “entry check”
    - Không chứa số lượng lớn bug khi thực hiện short check
    - Tài liệu được đánh số dòng, tài liệu tham khảo có sẵn
    - Tác giả sẵn sàng tham gia vào review team và tự tin về chất lượng của tài liệu
  - Xác định tiêu chí dừng
  - Chọn đội ngũ tham gia rà soát: 3 đến 5 người

Rà soát kỹ thuật chính thức(FTR)

- Khởi đầu (Kick-off meeting)
  - Tác giả sẽ giới thiệu tài liệu cần rà soát và mục tiêu rà soát
  - Mối quan hệ giữa tài liệu cần rà soát và các tài liệu tham khảo được giải thích rõ ràng
  - Phân công vai trò, công việc cụ thể cho thành viên tham gia
  - Phân phối tài liệu rà soát và tài liệu liên quan

Rà soát kỹ thuật chính thức(FTR)

- Chuẩn bị
  - Thành viên đội rà soát làm việc độc lập: rà soát tài liệu sử dụng checklist, viết ghi chú
- Cuộc họp rà soát: gồm 3 phiên
  - Ghi nhật ký
  - Thảo luận
  - Đưa ra quyết định: chấp nhận, chấp nhận một phần, không chấp nhận

Rà soát kỹ thuật chính thức(FTR)

- Làm lại, chỉnh sửa
  - Dựa trên các bug được tìm thấy → tác giả thực hiện fix bug, thay đổi cần thiết
- Theo dõi
  - Moderator theo dõi để quyết định từng mục đã thỏa mãn yêu cầu
  - Phân công người rà soát lại
  - Cho phép chuyển sang pha tiếp

Phương châm FTR

- Lập checklist cho từng sản phẩm cần rà soát
- Đào tạo cho các thành viên rà soát
- Phân tích rà soát trước để cải tiến phương pháp rà soát
- Lập lịch rà soát trong kế hoạch dự án và cấp phát nguồn lực cho rà soát
- Giới hạn số người tham gia rà soát
- Phiên rà soát không quá 2 giờ
- Hạn chế tranh luận, bác bỏ
- Trình bày rõ ràng vấn đề, không gượng ép giải quyết
- Ghi chú lên bảng tường

III.3 Phân tích tĩnh - Static Analysis

- Tìm khiếm khuyết trong code, tài liệu thiết kế, yêu cầu
- Có nhiều công cụ hỗ trợ phân tích tĩnh, và đa số tập trung vào code
- Là 1 dạng của kiểm thử tự động
  - Kiểm tra vi phạm các chuẩn
  - Kiểm tra mọi thứ có thể gây ra lỗi

Cái gì có thể phân tích?

- Coding standards
- Code Structure
  - Luồng điều khiển - Control Flow Analysis
  - Luồng dữ liệu - Data Flow Analysis
  - Data Structure
- Code metrics (độ đo)
  - Mức lồng nhau
  - Độ phức tạp Cyclomatic
  - Lines of code (LOC)
  - Operands & operators (Halstead's metrics)

Coding standards

- Chuẩn mã nguồn
  - Tập hợp các quy tắc lập trình. Mỗi ngôn ngữ sẽ có chuẩn riêng.
- Ví dụ:
  - Luôn kiểm tra biên của mảng khi thực hiện copy dữ liệu vào mảng đó
  - Quy ước đặt tên
  - Thụt lề
  - …

Phân tích luồng dữ liệu

- Xem xét các biến trong chương trình
  - Khai báo, định nghĩa biến
  - Sử dụng biến
  - Biến phải được khai báo trước khi sử dụng
  - Tham chiếu tới biến chưa được gán giá trị(defined)
  - Biến chưa bao giờ được sử dụng
  - Phạm vi tồn tại của biến

Phân tích luồng dữ liệu

```
1.  n := 0
2.  read (x)
3.  n := 1
4.  while x > y do
5.      begin
6.          read (y)
7.          write( n*y)
8.          x := x - n
9.      end
```

Luồng dữ liệu bất thường
n : được định nghĩa lại và ko được sử dụng

Lỗi luồng dữ liệu
y: được sử dụng mà chưa được định nghĩa(gán giá trị)

Phân tích luồng điều khiển

- Vòng lặp vô hạn, đệ quy vô hạn
- Mã “chết”
- Nhảy đến nhãn không xác định
- Độ phức tạp của lưu đồ
- ….

Unreachable code

```
int f(int x, int y)
{
    return x+y;
    int z=x*y;
}
```

```
double x = sqrt(2);
if (x > 5)
{
    printf(“%d”,x);
}
```

Ưu điểm và nhược điểm

- Ưu điểm:
  - Tìm ra khiếm khuyết khó thấy ở kiểm thử động
  - Đưa ra những đánh giá về chất lượng code và thiết kế → giúp cải tiến hơn
- Khuyết điểm:
  - Không phân biệt được “fail-safe” code với lỗi thực khi lập trình → nhiều fail message
  - Không thực thi code

Công cụ

- **Multi-language**
  - Moose : C/C++, Java, Smalltalk, .NET,…
  - Copy/Paste Detector (CPD): Java, JSP, C, C++ and PHP code.
  - ChecKing: Java, JSP, Javascript, HTML, XML, .NET, PL/SQL, embedded SQL, C/C++, Cobol,…

Công cụ

- **C/C++**
  - Cppcheck, Frama-C, …
- **Java**
  - Checkstyle, FindBugs, Jtest,…