Lý thuyết
Kiểm Tra Phần Mềm

Bài 04: ***Rà soát và lần bước chương trình***

GV: Nguyễn Ngọc Tú

Email: nntu@hoasen.edu.vn

Bộ môn: Kỹ thuật Phần mềm

Nội dung

- Duyệt, Lần bước và Rà soát/Thẩm tra
  Reviews, Walkthroughs, Inspections
- Lần bước và Rà soát chương trình
- Lợi ích của Duyệt/rà soát mã
- Rà soát mã
- Danh sách lỗi thông thường
- Kiểm lại
- Đánh giá ngang hàng

Duyệt, Lần bước và Rà soát

- Kỹ thuật kiểm tra chính trong giai đoạn đầu của quá trình phát triển
  - Phân tích yêu cầu, đặc tả và thiết kế
- Là tập các thủ tục và kỹ thuật phát hiện mã cho nhóm đọc mã
- Rà soát là cuộc họp của một nhóm người để đánh giá "lần bước" các tài liệu kỹ thuật một cách hệ thống
- Khi lỗi được xác định, ý kiến – phê bình đưa ra. Được phát hiện và quyết định trước khi xem xét khi nào tài liệu được chuyển tới các hành viên nhóm thực hiện duyệt lại.

Duyệt, Lần bước và Rà soát

- Sản phẩm
  - báo cáo được viết, trạng thái chất lượng và bất kỳ lỗi được phát hiện
  - thường không có hoạt động đề xuất
- *Lưu ý:*
  - *Kiểm tra – không phải gỡ rối*
  - *Tìm lỗi – không phải sửa lỗi*

Lần bước – Rà soát/Thẩm tra

Walkthroughs - Inspections

- Là hai kiểu chính của việc Rà soát:
  - walkthroughs
    - Không chính thức
    - nhóm đồng nghiệp, không có người quản lý
  - inspections
    - Tương tự
    - Chính thức
    - Nhóm của người thẩm tra

Lần bước – Rà soát

- Bao gồm một nhóm người
  - đọc
  - rà soát trực quan chương trình
- Dùng "Não Công" trong cuộc họp
- Ước chừng các công việc chuẩn bị trước
- Thường tốn: 90-120'

Lợi ích của Rà soát

- Đặc biệt cho việc kiểm chứng trong phân tích, thiết kế
- Bổ sung cho kiểm tra động trong giai đoạn kiểm chứng chương trình
  - có thể phát hiện sớm 30%-70% lỗi

Lợi ích của Rà soát

- Nhận diện tổng quát, các lỗi sớm được phát hiện .
  - trả giá thấp hơn nhưng lại đạt được khả năng sửa lỗi phù hợp hơn, tốt hơn
- Ví dụ:
  - *Giả sử tìm lỗi lúc thiết kế mất £1x để sửa, cùng lỗi đó nhưng ở giai đoạn sau:*
    - *trước khi kiểm thử £6.5x*
    - *thông qua kiểm thử £15x*
    - *Sau khi xuất xưởng £60x - £100x*

Lợi ích của Rà soát

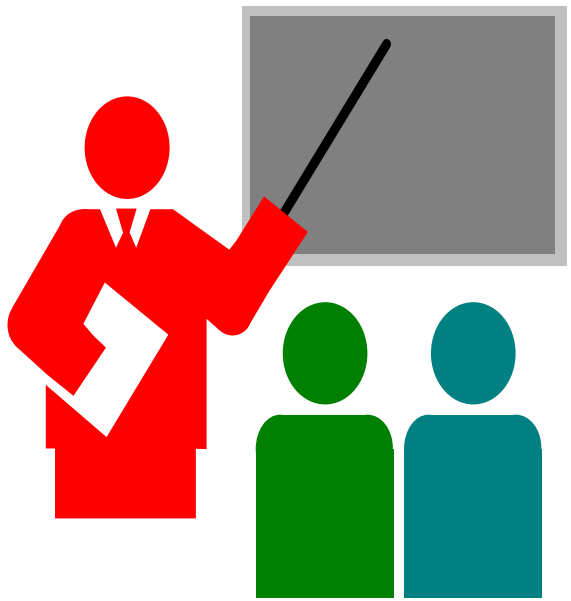
- Chỉ ra một "lô" các lỗi "liên quan",
  - sửa toàn thể (*một loạt*) lỗi sau đó
- Kiểm thử dựa trên máy tính thường chỉ cho thấy một triệu chứng của lỗi
- Hai cách kiểm thử bổ sung lẫn nhau
  - hiệu quả phát hiện lỗi sẽ giảm chất lượng nếu thiếu một trong hai phần này.

Rà soát chương trình

- Kỹ thuật kiểm tra tĩnh bao gồm:
  - Kiểm tra bàn giấy (*desk checking – kiểm lại*)
    - Đọc lại mã trước khi kiểm thử
    - Lần bước qua mã tìm lỗi cú pháp - logic
  - Lần vết (*tracing*)
    - Đọc mã, mô phỏng thực hiện, ghi nhận giá trị
      - tìm lỗi logic tốt
  - So sánh chức năng – đặc tả
    - Cần thiết cho các đoạn mã "bẫy", không có lỗi rõ ràng

Rà soát chương trình

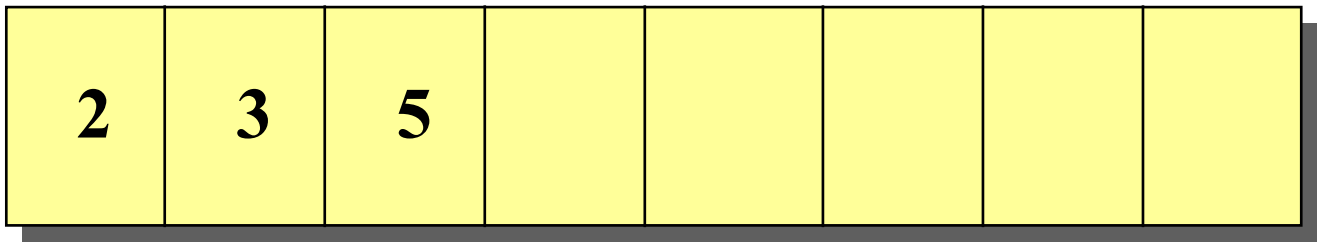
- Thường gồm 4 người
  - Điều phối (QC)
    - Phân phối tài liệu, lập lịch cho các phiên thẩm tra
    - Chủ trì phiên họp
    - Ghi nhận tất cả lỗi tìm kiếm
    - Bảo đảm lỗi sẽ được sửa
  - người lập trình,
  - người thiết kế chương trình,
  - chuyên viên kiểm thử



Rà soát chương trình

- Thực hiện
  - Trình bày mạch thực hiện qua từng hàng lệnh
    - Đưa ra các câu hỏi xác định lỗi
  - Được phân tích theo các chú ý trong danh mục ý kiểm tra.
- Người lập trình nhận danh sách lỗi
  - Sửa mã sai
- ***Danh sách lỗi được phân tích, phân loại, làm rõ***

Ví dụ: Sàng Erathosthenes



6 – bỏ qua bởi chia hết cho 2.
7 – được thêm vào *khe* kế tiếp.
8 – “lọt sàng” bởi chia hết cho 2
9 – “lọt sàng” bởi chia hết cho 3
...

Ví dụ: Sàng Erathosthenes

```
int P[101];
int V=1,  N=1, I;
main() {
  while (N<101) {
   for (I=1;I<=N;I++)
       if (!P[I]) {
           P[I]=V++ ;
           N++ ;
          }
```

```
    else if (V%P[I]==0)
          I=N+1;
      else
          I++;
  }  //while
}    //main
```

Chương trình đúng ?

Ví dụ: Sàng Erathosthenes

- Khai báo dữ liệu

Các biến ?

Giá trị mặc định ?

Khởi tạo dãy/mảng?

Biến cùng tên ?

Khởi tạo đúng ?

- Luồng điều khiển

Kết thúc mỗi vòng lặp ?

So trùng phát biểu DO/END ?

- Xuất / Nhập

Phát biểu OPEN đúng ?

Đặc tả định dạng đúng ?

Xử lý cuối / hết tập tin?

Danh mục ý kiểm tra
Checklist

Tạo các danh sách ý kiểm tra

- Các biến sẵn sàng khởi tạo ?
- Điều kiện trong phát biểu có đúng ?
- Mỗi vòng lặp chứa phát biểu kết thúc ?
- Dãy được xử lý nên:
  - giới hạn dưới là 0,1, … ?
  - Giới hạn trên là size hay size-1?
- Thực hiện gọi các hàm, chức năng với thông số đúng ?
- Cấu trúc liên kết bị sửa , có thay đổi trở lại ?
- …

Danh mục ý kiểm tra

- Lỗi tham chiếu dữ liệu
  - Biến tham chiếu chưa có thiết lập / khởi tạo trị ?
  - Chỉ số trong dãy nằm trong giới hạn cho phép ?
  - Mỗi chỉ số là một số nguyên ?
  - Xảy ra tham chiếu “treo” ?
  - Tham chiếu cùng một ô nhớ ?
  - Giá trị biến có kiểu khác khai báo/sử dụng ?
  - Địa chỉ ô nhớ tường minh / không tường minh ?
  - Tham chiếu khác kiểu ?
  - Liên kết thư viện với kiểu DL sử dụng ?
  - Chỉ mục chuỗi, toán tử, dãy
  - Trong OOP, kế thừa các lớp hiện thực ?

Danh mục ý kiểm tra

- Lỗi khai báo
  - Khai báo biến tường minh
  - Giá trị mặc định nếu không gán tường minh ?
  - Biến được khởi tạo ở đâu ? Khởi tạo chưa ?
  - Mỗi biến gán đúng độ dài và kiểu DL ?
  - Khởi tạo biến đồng nhất với kiểu bộ nhớ
  - Sử dụng *nhầm* biến với tên giống nhau

Danh mục ý kiểm tra

- Lỗi tính toán
  - Kiểu không đồng nhất
  - Tính toán tự ép kiểu, không ép kiểu → sai dữ liệu
    - x= 1, y =2, z = x/y: (z = 0 / 0.5 ?)
  - Tính toán với các biến Cùng kiểu khác kích cỡ, độ dài
  - Kiểu dl biến được gán không đủ "không gian" chứa kết quả
  - Tràn trên hay dưới trong quá trình tính toán các phần của biểu thức (dù kết quả không tràn)
    - Vd: $10^{-1000} * 10^{700} = 10^{-300}$
  - Xảy ra trường hợp số bị chia là 0 ?
  - Máy biểu diễn dưới dạng nhị phân – có những trường hợp kết quả sẽ không chính xác
    - 10 * 0.1 == 1.0 ???
  - Nơi nào trong ứng dụng, có thể xảy ra biến có giá trị vô nghĩa
    - Xác suất P : [0..1] , P=2 ?
  - Thứ tự đánh giá và xử lý ưu tiên toán hạng
  - Sử dụng phép toán trong các trường hợp nguyên
    - 2*i/2 == I ? (I chẵn, I lẻ, xử lý nhân trước, …)

Danh mục ý kiểm tra

- Lỗi so sánh
  - So sánh có kiểu khác nhau
  - So sánh biến khác kích cỡ, so sánh có kèm phép toán, luật ép kiểu có thích hợp
  - Các toán hạng so sánh đúng, biên: >, >=, <, <=, ..
  - Biểu thức, phép toán luận lý dùng đúng/hiểu rõ
  - Các toán hạng trong biểu thức luận lý
  - So sánh giữa các phân số hay số thực: biểu diễn gần đúng ở cơ số 2
  - Thứ tự ưu tiên của các phép toán luận lý
  - Cách thực hiện tổ hợp luận lý
    - If (x!=0 && y/x>z) { …. }

Danh mục ý kiểm tra

- Lỗi luồng điều khiển
  - Chương trình có phân nhiều nhánh, nhất là dùng GOTO, có nhảy đến đúng nơi cần thực hiện ?
  - Mỗi vòng lặp sẽ kết thúc ?
  - Chương trình, module, thủ tục có kết thúc
  - Có xảy ra một vòng lặp không bao giờ thực hiện
  - Vòng lặp kết hợp biến lặp và điều kiện luận lý: lặp đúng không nếu đk luận lý luôn đúng
  - Lỗi dư/thiếu 1 do bỏ sót giá trị biên: 0
  - Khối lệnh được giới hạn đúng ?
  - Có quyết định nào không phủ hết các trường hợp ? (bỏ sót)

Danh mục ý kiểm tra

- Lỗi giao tiếp (API)
  - Số thông số bằng số đối số gửi tới lúc gọi module
  - Trùng kiểu với thông số
  - Kiểu đơn vị đo thống nhất
    - Gửi tới trị có độ đo là Radian cho đối số tính bằng độ (degree)
  - Truyền thông số giữa các module
  - Sử dụng hàm dựng sẵn có đúng
  - Điểm nhập hay thông số có dạng tham chiếu hay không
  - Cho phép tham chiếu hàm
  - Sử dụng chung biến toàn cục: có cùng chung kiểu ?
  - Hằng có thể chuyển như đối số

Danh mục ý kiểm tra

- Lỗi xuất nhập
  - Khai báo tập tin rõ ràng ? Sử dụng đúng thuộc tính ?
  - Thuôc tính cho việc mở tập tin đúng ?
  - Định dạng phù hợp với việc xuất nhập
  - Đủ bộ nhớ sẵn sàng cho việc đọc tập tin
  - Các tập tin đã mở trước khi sử dụng
  - Các tập tin được đóng sau khi sử dụng
  - Phát hiện kết thúc tập tin và xử lý đúng
  - Lỗi IO được xử lý
  - Lỗi chính tả, ngữ pháp trong văn bản in , hiển thị

Danh mục ý kiểm tra

- Kiểm tra dạng khác
  - Trình biên dịch liệt kê các định danh mà ở đó không sử dụng hoặc chỉ 1 lần
  - Trình biên dịch liệt kê các thuộc tính được gán trị không kỳ vọng
  - Biên dịch thành công nhưng có vài cảnh báo và thông tin “đính kèm”
  - Chương trình hoặc module đủ hiệu quả
  - Lỗi sai biệt hay thiếu hàm/thủ tục

“Thẩm tra” chương trình

- Chủ yếu cho kiểm thử đơn vị. Một phiên “thẩm tra”:
  - 2 giờ
  - tối đa 200 phát biểu của chương trình
- Hiệu quả.
- Những lỗi lộ ra “không thể” được phát hiện bởi cá nhân người hiện thực trong các điều kiện không chính thức

Lần bước (Walkthrough)

- Nhóm 3-5 người
  - Điều phối
  - Thư ký: ghi nhận lỗi
  - Người Kiểm thử
- Đề nghị
  - Người lập trình cấp cao/KN
  - Chuyên gia NNLT
  - Người mới lập trình
  - Người sẽ bảo trì
  - Người từ dự án khác
  - Người cùng nhóm lập trình

Lần bước

- Phiên họp
  - Với các thành viên đóng vai trò như “máy tính”
  - Mọi người được xem như là Tester và chuẩn bị trước các tập nhỏ Mẫu kiểm thử (*test cases*)
    - Dữ liệu vào
    - Kết quả kỳ vọng
  - Thông qua cuộc họp, các Mẫu thử sẽ được “thực thi” từng bước trong chương trình theo một cách “tổng quát”
  - Phần lớn các lỗi
    - Phát hiện thông qua hỏi Người lập trình

Kiểm lại (Desk checking)

- Người đọc chương trình, kiểm tra nó một cách chi tiết để ghi lại danh sách lỗi và/hoặc các bước kiểm thử dữ liệu trong quá trình kiểm lại.
- Quá trình không hữu ích do:
  - Quá trình kiểm thử hoàn toàn vô kỷ luật.
  - Trái ngược với nguyên lý kiểm thử ở chương 2
- Vì vậy Desk checking tốt nhất được thực hiện theo kịch bản của một người khác không phải là tác giả của chương trình.

Kiểm lại (Desk checking)

- Desk checking kém hiệu quả hơn quá trình walktrough và inspector vì không có sự phối hợp làm việc theo nhóm.
  - Quá trình làm việc theo nhóm thúc đẩy sự cạnh tranh lành mạnh
  - con người thích phô trương các lỗi mà họ tìm thấy.
- Trong quá trình desk-checking có thể sẽ không tìm thấy lỗi nào.
- Tóm lại Desk-checking cũng có giá trị còn hơn là không làm gì cả. Tuy nhiên nó kém hiệu quả hơn walkthrough và inspection.

Đánh giá ngang hàng
Peer Rating

- Không liên quan kiểm thử
- Lượng giá chương trình trong các ảnh hưởng
  - Chất lượng tổng thể, tính bảo trì, tính mở rộng, khả dụng và rõ ràng-sáng sủa
  - Giúp người lập trình tự đánh giá
  - Đánh giá từ các đồng nghiệp cùng trình độ
  - Trên các chương trình ngẫu nhiên được chọn (cả tốt lẫn xấu)
- Điểm lượng giá qua các câu hỏi (1..7)
  - Dễ hiểu
  - Thấy rõ thiết kế mức cao và suy luận được
  - Thấy rõ thiết kế mức thấp và suy luận được
  - Dễ sửa chương trình
  - Đáng tự hào khi viết chương trình này

Bài tập - đọc hiểu

- BT Nhóm
  - Mỗi nhóm đưa ra 2 ví dụ (mã ngắn hơn 10 dòng) cho mỗi nhóm lỗi xảy ra

- Xét giải thuật loại bỏ một phần tử ra khỏi danh sách liên kết đơn có thứ tự với đoạn mã như sau:

```
entry *  delete_linked_list(entry *
current_head, int key_to_delete,
entry_ptr * deleted_entry) {

    entry * current_element, *
previous_element;
    if (current_head == NULL) {
        *deleted_entry = NULL;
        return NULL;
    }
    if (current_head->key ==
key_to_delete) {
        *deleted_entry = current_head;
        return current_head->next;
    }
    previous_element = current_head;
    current_element = current_head->next;
    while (current_element != NULL) {
        //  {*}
        if (current_element->key ==
key_to_delete) {
            previous_element->next =
current_element->next;

            // {**}
            *deleted_entry = current_element;
            return current_head;
        }
        current_element = current_element-
>next;
    }
    *deleted_entry = NULL;
    return current_head;
 }
```

Đọc thêm

- [1]. Chapter 03

Q/A

- http://www.mediafire.com/?sharekey=1c3651a15901b57c8c9e7c56ba37815fbf01f83de0545d2f