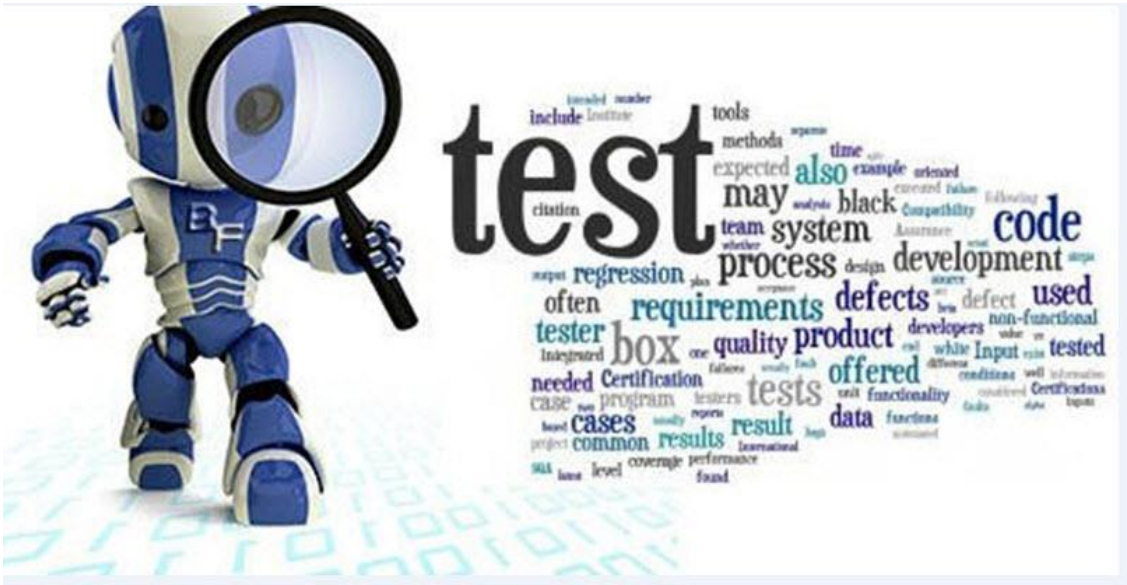
HUTECH
Đại học Công nghệ Tp.HCM

KIỂM THỬ PHẦN MỀM

Bài 7:

LỖI PHẦN MỀM

Thời gian: 6 tiết



NỘI DUNG

7.1. Tổng quan về lỗi phần mềm

7.2. Những nguyên nhân gây ra lỗi thường gặp

7.3. Các lỗi thường gặp trong phần mềm

7.4. Tìm lỗi và phân tích lỗi

7.1.TỔNG QUAN LỖI PHẦN MỀM

- ✓ Lỗi phần mềm thuộc nhiều loại.
- ✓ Điều quan trọng là phải hiểu được bản chất, ý nghĩa của nó và nguyên nhân để xử lý tốt hơn.
- ✓ Lỗi xuất phát từ sản phẩm hoặc thay đổi từ mong muốn người dùng.
- ✓ Các lỗi phần mềm phổ biến thì làm thế nào để xác định chúng trong quá trình kiểm thử

7.1.TỔNG QUAN LỖI PHẦN MỀM

- ✓ Lỗi phần mềm là một sai lệch dựa vào độ chính xác hoặc tính đúng đắn
- ✓ Lỗi phần mềm trong một chương trình tạo ra kết quả không chính xác hoặc không mong muốn:
  - Lỗi là sự khác nhau của kết quả thực tế và kết quả mong đợi
  - Lỗi là kết quả của việc không hoàn thành hoặc sai yêu cầu hoặc do vấn đề nhập dữ liệu.

7.2. NGUYÊN NHÂN GÂY RA LỖI

- **Kết quả từ lỗi thiết kế, lỗi logic hay lỗi chương trình**
  - **Lỗi thiết kế (Design)**
    - Không chính xác, chưa hoàn chỉnh, hiểu nhầm
  - **Lỗi logic**
    - Kết quả từ kiểm thử không hợp lệ, dữ liệu thử, thực thi không đúng thiết kế
  - **Lỗi cài đặt (coding)**
    - Thực thi không chính xác thiết kế

7.2. NGUYÊN NHÂN GÂY RA LỖI

Nguồn của sự thay đổi

- ✓ Những điều kiện kinh doanh và thị trường gây ra thay đổi yêu cầu sản phẩm và qui tắc nghiệp vụ (business rules)
- ✓ Khách hàng mới cần thay đổi nhu cầu dữ liệu, chức năng hay dịch vụ phân phối bởi hệ thống
- ✓ Tái tổ chức hay cắt giảm kinh doanh mà thay đổi ưu tiên và cấu trúc
- ✓ Ràng buộc ngân sách và lịch trình gây ra tái định nghĩa hệ thống

Hầu hết sự thay đổi là có lý do chính đáng

7.2. NGUYÊN NHÂN GÂY RA LỖI

Thay đổi thích nghi (Adaptive change)

- Để thích nghi phần mềm với thay đổi môi trường
- Kết quả khởi nguồn từ loại bỏ software, hardware khác biệt, phần mềm nền, trình biên dịch, hệ điều hành
- Kết quả từ thay đổi máy ảo, thêm tiện ích cho hệ điều hành, lưu không gian đĩa, khôi phục sự cố lỗi
- Thay đổi hệ thống phần mềm liên quan

7.2. NGUYÊN NHÂN GÂY RA LỖI

Thay đổi hoàn chỉnh (Perfective Change)

- ✓ Mở rộng những yêu cầu tồn tại của hệ thống
- ✓ Xuất phát từ thêm mới chức năng hay tinh chế chức năng đã cung cấp, một số chức năng có thể bị bỏ đi.

7.2. NGUYÊN NHÂN GÂY RA LỖI

Thay đổi ngăn ngừa (Preventive change)

- ✓ Đề cập đến vấn đề làm hỏng (giảm giá trị) cấu trúc
- ✓ Ngăn chức năng lạ hay cải thiện khả năng bảo trì của phần mềm. Gồm: tái cấu trúc mã nguồn, tối ưu mã nguồn và cập nhật sưu liệu

7.2. NGUYÊN NHÂN GÂY RA LỖI

Một số hiệu ứng lề

Sửa đổi phần mềm là công việc “nguy hiểm”, ta thường gặp ba loại chính của hiệu ứng lề

- ✓ Hiệu ứng lề của việc thay đổi mã nguồn
- ✓ Hiệu ứng lề của việc thay đổi dữ liệu
- ✓ Hiệu ứng lề của việc thay đổi tài liệu

7.2. NGUYÊN NHÂN GÂY RA LỖI

Hiệu ứng lề của việc thay đổi mã nguồn

Việc sửa lỗi luôn dẫn đến các vấn đề phức tạp, tập hợp các thay đổi có thể gây ra nhiều lỗi hơn

- Một chương trình con bị xóa hay thay đổi.
- Một dòng nhãn bị xóa hay thay đổi.
- Một biến bị xóa hay thay đổi.
- Các thay đổi để tăng khả năng thực hiện.
- Việc mở và đóng file bị thay đổi.
- Các phép toán logic bị thay đổi.
- Việc thay đổi thiết kế sẽ thay đổi lớn về chương trình.

7.2. NGUYÊN NHÂN GÂY RA LỖI

Hiệu ứng lề của việc thay đổi dữ liệu

- Định nghĩa lại cấu trúc bản ghi hay cấu trúc file.
- Tăng hoặc giảm kích thước một mảng.
- Thay đổi dữ liệu tổng thể.
- Xếp lại các tham số vào ra hay tham số của chương trình con.
- ….

Hạn chế: bằng tài liệu thiết kế mô tả cấu trúc dữ liệu và cung cấp một lời chỉ dẫn tham khảo đến từng phần từ dữ liệu, các bản ghi, các file và các cấu trúc khác của các module phần mềm

7.2. NGUYÊN NHÂN GÂY RA LỖI

Hiệu ứng lề của việc thay đổi dữ liệu

- ✓ Thay đổi chương trình nguồn mà không thay đổi tài liệu thiết kế và tài liệu hướng dẫn sử dụng
- ✓ Hiệu ứng lề xảy ra trong các lần bảo trì sau đó, khi việc nghiên cứu không kỹ các tài liệu kỹ thuật, dẫn tới sự đánh giá sai về các đặc tính của phần mềm. Đối với người sử dụng, phần mềm tốt chỉ khi có tài liệu hướng dẫn sử dụng chúng.
- ✓ không được thay đổi thiết kế của phần mềm hoặc mã chương trình, mà chỉ cần chỉ ra sự thiếu rõ ràng trong tài liệu của người sử dụng

7.3. CÁC LỖI THƯỜNG GẶP TRONG PHẦN MỀM

[1]. Lỗi chức năng

[2]. Lỗi giao tiếp

[3]. Lỗi thiếu lệnh

[3]. Lỗi cú pháp

[4]. Lỗi lỗi xử lý

[5]. Lỗi tính toán

[6]. Lỗi dòng điều khiển

7.3. CÁC LỖI THƯỜNG GẶP TRONG PHẦN MỀM

[1]. Lỗi chức năng

Phần mềm có một lỗi chức năng nếu một cái gì đó mà ta mong muốn phần mềm làm là rắc rối, khó hiểu, không khả thi.

[2]. Lỗi giao tiếp

Bất cứ điều gì mà người dùng cuối cần biết để sử dụng nên được làm sẵn có trên màn hình. Ví dụ về lỗi giao tiếp: không cung cấp bảng hướng dẫn trợ giúp/menu.

[3]. Lỗi thiếu lệnh

Điều này xảy ra khi một lệnh dự kiến là thiếu.

7.3. CÁC LỖI THƯỜNG GẶP TRONG PHẦN MỀM

[4]. Lỗi cú pháp

Là chính tả hay ngữ pháp câu không chính xác. Trình biên dịch sẽ cảnh báo về bất kỳ lỗi cú pháp xuất hiện trong code.

[5] Lỗi "Lỗi xử lý"

Bất kỳ lỗi nào xảy ra khi người dùng đang tương tác với các phần mềm cần phải được xử lý một cách rõ ràng và có ý nghĩa. Nếu không, nó được gọi là một lỗi Xử lý lỗi.

7.3. CÁC LỖI THƯỜNG GẶP TRONG PHẦN MỀM

[6]. Lỗi tính toán

Những lỗi này xảy ra do các lý do sau đây: logic không tốt, công thức tính toán không chính xác, kiểu dữ liệu không phù hợp, lỗi lập trình.

[7]. Lỗi dòng điều khiển

Việc kiểm soát flow của phần mềm mô tả những gì sẽ làm gì tiếp theo và dựa trên điều kiện. Ví dụ, người dùng điền mẫu đơn và sử dụng: Save, Save và Close, và Cancel. Nếu người dùng nhấp “Save and Close”, thông tin trong form lưu và đóng form lại. Nếu nhấp vào button “Save and Close” mà không đóng form, thì đó là lỗi dòng điều khiển.

7.4. TÌM LỖI VÀ PHÂN TÍCH LỖI

7.4.1. Checklist kiểm tra mã nguồn

7.4.2. Qui tắc xác định lỗi phần mềm

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[1]. Các lỗi truy xuất dữ liệu (Data Reference Errors)

[2]. Các lỗi định nghĩa/khai báo dữ liệu (Data-Declaration Errors)

[3]. Các lỗi tính toán (Computation Errors)

[4]. Các lỗi so sánh (Comparison Errors)

[5]. Các lỗi luồng điều khiển (Control-Flow Errors)

[6]. Các lỗi giao tiếp (Interface Errors)

[7]. Các lỗi nhập/xuất (Input/Output Errors)

[8]. Các lỗi khác (Other Checks)

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[1]. Các lỗi truy xuất dữ liệu (Data Reference Errors)

1. Dùng biến chưa có giá trị xác định ?

```
int i, count;
for (i = 0; i < count; i++) {...}
```

2. Dùng chỉ số của biến array nằm ngoài phạm vi ?

```
int list[10];
If (list[10] == 0) {...}
```

3. Dùng chỉ số không thuộc kiểu nguyên của biến array?

```
int list[10];
double idx=3.1416;
if (list[idx] == 0) {...}
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[1]. Các lỗi truy xuất dữ liệu (Data Reference Errors)

4. Tham khảo đến dữ liệu không tồn tại (dangling references)?

```
int *pi;
if (*pi == 10) {...} //pi đang tham khảo đến địa chỉ không hợp lệ - Null
int *pi = new int;
...
delete (pi);
if (*pi = 10) {...} //pi đang tham khảo đến địa chỉ
 //mà không còn dùng để chứa số nguyên
```

5.Truy xuất dữ liệu thông qua alias có đảm bảo thuộc tính dữ liệu đúng?

```
int pi[10];
pi[1] = 25;
char* pc = pi;
if (pc[1] == 25) {...} //pc[1] khác với pi[1];
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[1]. Các lỗi truy xuất dữ liệu (Data Reference Errors)

6. Dùng chỉ số bị lệch ?

```
int i, pi[10];

for (i = 1; i <= 10; i++) pi [i] = i;
```

7. Class có hiện thực đủ các tác vụ trong interface mà nó hiện thực không ?

8. Class có hiện thực đủ các tác vụ "pure virtual" của class cha mà nó thừa kế không?

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[2]. Các lỗi khai báo dữ liệu

1. Tất cả các biến đều được định nghĩa hay khai báo tường minh chưa?

```
int i;
extern double d;
d = i*10;
```

2. Định nghĩa hay khai báo đầy đủ các thuộc tính của biến dữ liệu chưa?

```
static int i = 10;
```

3. Biến array hay biến chuỗi được khởi động đúng chưa ?

```
int pi[10] = {1, 5,7,9} ;
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[2]. Các lỗi khai báo dữ liệu

4. Kiểu và độ dài từng biến đã được định nghĩa đúng theo yêu cầu chưa ?

```
short IPAddress;
byte Port;
```

5. Giá trị khởi động có tương thích với kiểu biến ?

```
short IPAddress = inet_addr("203.7.85.98");

byte Port = 65535;
```

6. Có dùng các biến ý nghĩa khác nhau nhưng tên rất giống nhau không?

```
int count, counts;
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[3]. Các lỗi tính toán (Computation Errors)

1. Thực hiện phép toán trên toán hạng không phải là số?

```
CString s1, s2;
int ketqua = s1/s2;
```

2. Thực hiện phép toán trên các toán hạng có kiểu không tương thích?

```
byte b;
int i;
double d;
b = i * d;
```

3. Thực hiện phép toán trên các toán hạng có độ dài khác nhau ?

```
byte b;
int i;
b = i * 500;
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[3]. Các lỗi tính toán (Computation Errors)

4. Gán dữ liệu vào biến có độ dài nhỏ hơn ?

```
byte b;
int i;
b = i * 500;
```

5. Kết quả trung gian bị tràn?

```
byte i, j, k;
i = 100;
j = 4;
k = i * j / 5;
```

6. Phép chia có mẫu bằng 0 ?

```
byte i, k;
i = 100 / k;
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[3]. Các lỗi tính toán (Computation Errors)

7.Mất độ chính xác khi mã hóa/giải mã số thập phân/số nhị phân?

8. Giá trị biến nằm ngoài phạm vi ngữ nghĩa ?

```
int tuoi = 3450;
tuoi = -80;
```

9. Thứ tự thực hiện các phép toán trong biểu thức có tương thích với thứ tự mà máy thực hiện? Người lập trình hiểu đúng về thứ tự ưu tiên các phép toán chưa?

```
double x1 = (-b-sqrt(delta)) / 2*a;
```

10. Kết quả phép chia nguyên có chính xác theo yêu cầu không ?

```
int i = 3;
if (i/2*2) == i) {...}
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[4]. Các lỗi so sánh (Comparison Errors)

1. So sánh 2 dữ liệu có kiểu không tương thích ?

```
int ival;
char sval[20];
if (ival == sval) {...}
```

2. So sánh 2 dữ liệu có kiểu không cùng độ dài ?

```
int ival;
char cval;
if (ival == cval) {...}
```

3. Toán tử so sánh đúng ngữ nghĩa mong muốn? Dễ lộn giữa

= và ≠, <= và >=, and và or...

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[4]. Các lỗi so sánh (Comparison Errors)

4. Có nhầm lẫn giữa biểu thức Bool và biểu thức so sánh?

```
if (2 < i < 10) {...}
if (2 < i && i < 10) {...}
```

5. Có hiểu rõ thứ tự ưu tiên các phép toán?

```
if(a==2 && b==2 || c==3) {...}
```

6. Cách thức tính biểu thức Bool của chương trình dịch như thế nào?

```
if(y==0 || (x/y > z))
```

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[5]. Các lỗi luồng điều khiển (Control-Flow Errors)

1. Thiếu thực hiện 1 số nhánh trong lệnh quyết định theo điều kiện số học ?

```
switch (i) {
case 1: ... //cần hay không cần lệnh break;
case 2: ...
case 3: ...
}
```

2. Mỗi vòng lặp thực hiện ít nhất 1 lần hay sẽ kết thúc ?

```
for (i=x ; i<=z; i++) {...} //nếu x > z ngay từ đầu thì sao ?
for (i = 1; i <= 10; i--) {...} //có dừng được không ?
```

3. Biên của vòng lặp có bị lệch ?

```
for (i = 0; i <= 10; i++) {...} //hay i < 10 ?
```

4. Có đủ và đúng cặp token begin/end, {} ?

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[6]. Các lỗi giao tiếp (Interface Errors)

1. Số lượng tham số cụ thể được truyền có = số tham số hình thức của hàm được gọi ?
2. Thứ tự các tham số có đúng không ?
3. Thuộc tính của từng tham số thực có tương thích với thuộc tính của tham số hình thức tương ứng của hàm được gọi ?
4. Đơn vị đo lường của tham số thực giống với tham số hình thức ?
5. Tham số read-only có bị thay đổi nội dung bởi hàm không?
6. Định nghĩa biến toàn cục có tương thích giữa các module chức năng không ?

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[7]. Các lỗi nhập/xuất (Input/Output Errors)

1. Lệnh mở/tạo file có đúng chế độ và định dạng truy xuất file?
2. Kích thước của buffer có đủ chứa dữ liệu đọc vào không ?
3. Có mở file trước khi truy xuất không ?
4. Có đóng file lại sau khi dùng không ? Có xử lý điều kiện hết file?
5. Có xử lý lỗi khi truy xuất file không ?
6. Chuỗi xuất có bị lỗi từ vựng và cú pháp không ?

7.4.1. CHECKLIST KIỂM TRA MÃ NGUỒN

[8]. Các lỗi khác (Other Checks)

- ✓ Có biến nào không được tham khảo trong danh sách tham khảo chéo (cross-reference)?
- ✓ Cái gì được kỳ vọng trong danh sách thuộc tính ?
- ✓ Có các cảnh báo hay thông báo thông tin ?
- ✓ Có kiểm tra tính xác thực của dữ liệu nhập chưa ?
- ✓ Có thiếu hàm chức năng ?

7.4.2. QUY TẮC XÁC ĐỊNH LỖI PHẦN MỀM

- ✓ Quy tắc 1: Phần mềm không thực hiện một số thứ giống như mô tả trong bản đặc tả phần mềm
- ✓ Quy tắc 2: Phần mềm thực hiện một số việc mà bản đặc tả yêu cầu nó không được thực hiện
- ✓ Quy tắc 3: Phần mềm thực hiện một số chức năng mà bản đặc tả không đề cập
- ✓ Quy tắc 4: Phần mềm không thực hiện một số việc đặc tả không đề cập tới, nhưng là những việc nên làm
- ✓ Quy tắc 5: Trong mắt của người kiểm thử, phần mềm là khó hiểu, khó sử dụng, chậm đối với người sử dụng.

HUTECH
Đại học Công nghệ Tp.HCM

KIỂM THỬ PHẦN MỀM

Bài 7:

LỖI PHẦN MỀM

Thời gian: 6 tiết

Q&A