プログラマの視線を用いたバグ特定プロセスの分析

奈良先端科学技術大学院大学
情報科学研究科
ソフトウェア工学講座 M2　上野 秀剛

研究の背景

- ソフトウェアの開発コストの削減，品質の向上のためにはバグを早期の取り除くことが重要
- バグを取り除くためにテスト工程前にコードレビューが行われている

研究の背景(cont.)

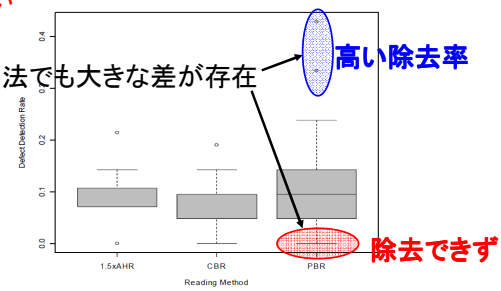
- いろいろなレビュー手法が提案されている
  - CBR（Checklist-Based Reading）
  - PBR（Perspective-Based Reading）
  - TCBR（Test Case Based Reading）
- 各手法のレビュー効率（バグ発見時間など）を比較した研究も盛んに行われている[1]

[1]Laitenberger, O., Rombach, D. (Session Chairs), “Software Inspections, Reviews & Walkthroughs,” ICSE2002 IMPACT Presentations, 2002

手法の差と個人の差

- 個人ごとの効率の差についてはあまり研究されていない

同じ手法でも大きな差が存在



[2]岡本，丸山，鈴木，高橋，西山，野中，“要求仕様書の特性に着目した個人レビュー手法の実験的評価”，第20年度ソフトウェア品質管理研究会分科会報告書，日本科学技術連盟，pp.191-209，2004

個人の差の要因

- レビュー効率の差はレビュー実施者のさまざまな要因により発生する
  - プログラムの読解能力
  - コードを読み進める順序
  - 理解戦略
  - etc
- 詳細なレビュー行動の分析はされていない

本研究の目的

- コードレビュー実施者がバグを発見するまでのレビュー行動を観察し，被験者間の差の要因を分析する

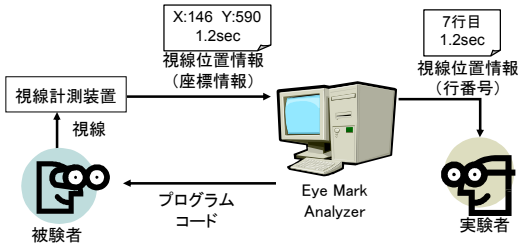
本研究のアプローチ

- バグの入ったコードをレビューしてもらい被験者の視線の動きを計測することで，レビュー行動を分析する
  - 視線を計測することで判断や,意図を見ることができる
- レビューは1行を読む，特定の行へジャンプするという動作を含む
  - コード上を動く視線を行単位で計測

7

実験ツール：Eye Mark Analyzer

- コード上の視線の動きを分析するツール
  - 視線計測装置の出力する座標位置情報をスクロール状態を考慮して行番号に対応付ける



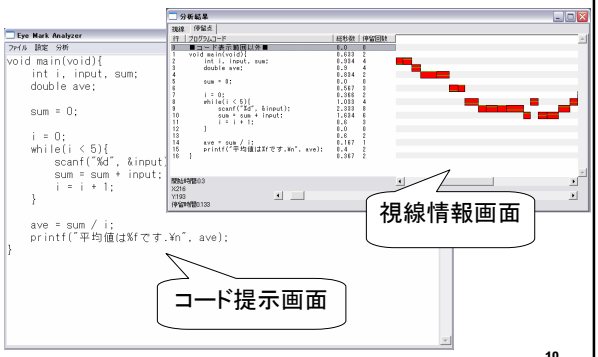
8

EMAの機能

- 視線位置情報を行単位に変換
  - コードのスクロールを考慮
- 視線の動きを再生
  - コード上の点として表示
- 分析用のデータの出力
  - 視線の動きの履歴（行単位）

9

EMAの実行画面



10

実験

- コードレビューをしてバグを発見してもらう
- 被験者
  - 大学院生5名（1名は実務経験者）
  - 全員が1度以上のコードレビュー経験があった
  - プログラム経験は3～4年
- レビューするプログラム
  - C言語で書かれた12～23行のプログラム6問
  - バグは1つだけ存在する
    - 被験者にはバグが1つだけあることを事前に伝えた

11

実験の流れ

1. 被験者に提示するプログラムの仕様を伝える
2. プログラムをレビューし，バグを探してもらう
   - 発見したら作業を中断しバグを口頭で伝えてもらう
   - 正しければ終了，誤っていたら作業を再開する
   - レビュー時間の合計が5分を超えたら任意で終了できる
3. レビュー終了後，レビュー中に考えたことについてインタビューを行う
   1. コードだけを見せてインタビュー
   2. コードと視線データを見せてインタビュー

12

実験結果

- 30試行（6問×5名）中，3件で視線データが取れなかった
- 27件中，3件がリタイヤ
- インタビューを含んだ実験時間58～90分
- インタビューでは視線情報を見せた後のほうがレビュー時の思考を思い出すことができた

13

分析結果

- 4つの特徴的な動きが見られた
  - 結果全体に見られた動き
    - 変数宣言，代入の確認
    - コードのスキャン
  - バグの発見が遅かった結果に見られた動き
    - 短いスキャン時間
  - バグを発見できなかった結果に見られた動き
    - 特定の行への集中

14

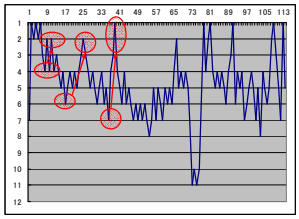
結果全体に見られた動き：変数宣言，代入の確認

- レビュー中に新たな変数を見つけるとその変数の宣言か初期値代入部分を見る
  - 56.3%(63／112)が3秒以内に見ている
  - 92.0%が最終的には見ている

15

結果全体に見られた動き：変数宣言，代入の確認

```
01 void main(void){
02    int i, input, sum;
03
04    i = 0;
05    while(i < 5){
06       scanf("%d", &input);
07       sum = sum + input;
08       i = i + 1;
09    }
10
11    printf("合計は%dです.¥n", sum);
12 }
```



16

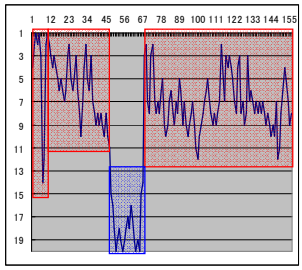
結果全体に見られた動き：コードのスキャン

- コード全体を眺めるような動作（スキャン）を行ってから特定の行を精読する
  - レビューに要した時間の内，初めの30%でコード全体の72.8%を見ている
  - インタビューからも確認された
  - 結果を表示する行はほとんど見ていない

17

結果全体に見られた動き：コードのスキャン

```
01 int makeSum(int max){
02    int i, sum;
03    sum = 0;
04
05    i = 0;
06    while(i < max){
07       sum = sum + i;
08       i = i + 1;
09    }
10    return sum;
11 }
12
13 void main(void)
14 {
15    int input, sum;
16
17    scanf("%d",&input);
18    sum = makeSum(input);
19    printf("合計は%dです.¥n", sum);
20 }
```



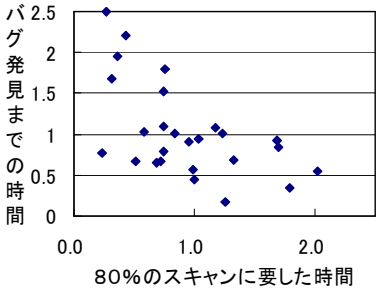
18

バグ発見が遅かった結果に見られた動き：短いスキャン時間

- コード全体をスキャンする時間が平均よりも短い場合，バグ発見までの時間が延びる傾向があった
  - スキャン時間：レビュー開始からコードの80%を読むまでの時間

19

バグ発見が遅かった結果に見られた動き：短いスキャン時間



※被験者の平均レビュー時間で正規化

20

バグを発見できない結果に見られた動き：特定の行への集中

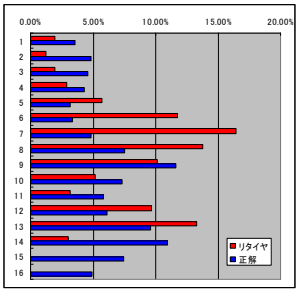
- リタイア（バグを発見できずにレビューを終了）した3件は特定の行に視線が集中していた
  - 視線が集中した行はバグに関係がある場合とない場合があった

21

バグを発見できない結果に見られた動き：特定の行への集中

```
01 void main(void){
02    int i, input, sum;
03    double ave;
04
05    sum = 0;
06
07    i = 0;
08    while(i < 5){
09       scanf("%d", &input);
10       sum = sum + input;
11       i = i + 1;
12    }
13
14    ave = sum / i;
15    printf("平均値は%fです.¥n", ave);
16 }
```

バグ



22

まとめと今後の課題

- コードレビュー時の視線を計測し，4つの特徴的動作が見つかった
  - 変数宣言，および代入の確認
  - コードのスキャン
  - 短いスキャン時間
  - 特定の行への集中
- より大規模なプログラムでも同じ結果になるのか
- より多くの被験者での実験を行う必要がある

23

補足資料

24

視線による判断，意図の分析

- 何らかの作業をしようとする人間の判断や意図は，視線の動きに表れる[3]
- コードレビューでは視線の動きにバグを見つける過程の判断や意図が表れると思われる

[3]苧阪良二，中溝幸夫，古賀一男，“眼球運動の実験心理学”，名古屋大学出版社，1993.

25

関連研究

- レビュー
  - レビュー手法の比較[4]
- 視線
  - 英語例文検索システム利用者の「迷い」を検出[5]
  - バグ症状を伝えてのデバッグ[6]
    - モジュール（関数）単位での視線分析

[4]Laitenberger, O., Rombach, D. (Session Chairs), “Software Inspections, Reviews & Walkthroughs,” ICSE2002 IMPACT Presentations, 2002.
[5]高木啓伸，“視線の移動パターンに基づくユーザの迷いの検出”，情報処理学会論文誌，Vol.41，No.5，May 2000.
[6]門田暁人，高田義広，鳥居宏次，"視線追跡装置を用いたデバッグプロセスの観察実験，" 信学技報，ソフトウェアサイエンス，SS96-5，pp.1-8，July 1996.

26

停留点の定義

- 注視点
  - 被験者の視線とディスプレイの交点
  - 同じ場所を見ていても視線が動くこと（固視微動）による測定誤差
- 停留点
  - 一定時間以上，一定範囲に視線が停留した点
  - 本実験では停留点による分析を行った
  - 本実験では50ms以上，30ピクセル以内と定義

27

実験環境

- 視線計測装置
  - NAC社製 非接触型アイカメラEMR-NC
- ディスプレイ
  - 21インチ液晶ディスプレイ
  - 解像度1024×768
  - 表示したコードは高さ18ピクセル
  - 行間は5ピクセル

28

問題の詳細

問題	プログラムの仕様	バグの内容
0	整数5つを入力し合計を返す	合計を入力する変数が0クリアされていない
1	入力値Nに対して1～Nの合計を返す	while文の終了条件の間違い
2	整数5つを入力し平均値を返す	平均値をintとして計算するため，小数点が丸められる
3	0が入力されるまで最大255個の入力を受け付け，その平均値を返す	平均値の計算を実際に入力された数ではなく最大入力数で計算している
4	入力A，Bを入れ替え，入れ替え後のA，Bを出力する	入れ替えを行う関数へ値渡しで変数を渡しているため入れ替えが行われない
5	入力された値が素数かどうかを判定する	素数とそうでない値を逆に判断する

29

問題0

```
01  void main(void){
02      int i, input, sum;
03
04      i = 0;
05      while(i < 5){
06          scanf("%d", &input);
07          sum = sum + input;
08          i = i + 1;
09      }
10
11      printf("合計値は%dです.¥n", sum);
12  }
```

30

問題1

```
01 int makeSum(int max){
02     int i, sum;
03     sum = 0;
04
05     i = 0;
06     while(i < max){
07         sum = sum + i;
08         i = i + 1;
09     }
10     return sum;
11 }
12
13 void main(void)
14 {
15     int input, sum;
16
17     scanf("%d",&input);
18     sum = makeSum(input);
19     printf("1から%dの合計は%d", input, sum);
20 }
```

31

問題2

```
01 void main(void){
02     int i, input, sum;
03     double ave;
04
05     sum = 0;
06
07     i = 0;
08     while(i < 5){
09         scanf("%d", &input);
10         sum = sum + input;
11         i = i + 1;
12     }
13
14     ave = sum / i;
15     printf("平均値は%fです.¥n", ave);
16 }
```

32

問題3

```
01 #define MAX 255
02 void main(void){
03     int i, num, list[MAX];
04     double sum=0;
05
06     i = 0;
07     while(i < MAX){
08         scanf("%d", &list[i]);
09         if(list[i] == 0)break;
10         i = i + 1;
11     }
12
13     num = i;
14
15     i = 0;
16     while(i < MAX){
17         sum = sum + list[i];
18         i = i + 1;
19     }
20
21     printf("%d個の平均値は%fです.¥n", i, sum/i);
22 }
```

33

問題4

```
01 void swap(int a, int b){
02     int tmp;
03     tmp = a; a = b; b = tmp;
04 }
05
06 void main(void){
07     int list[2], i;
08
09     i = 0;
10     while(i < 2){
11         printf("%dつ目:", i+1);
12         scanf("%d", &list[i]);
13         i = i + 1;
14     }
15
16     swap(list[0], list[1]);
17
18     i = 0;
19     while(i < 2){
20         printf("交換後の%dつ目:%d¥n", i+1, list[i]);
21         i = i + 1;
22     }
23 }
```

34

問題5

```
01 void main(void){
02     int i, num, isPrime = 0;
03
04     printf("判定したい整数を入力してください:");
05     scanf("%d", &num);
06
07     i = 2;
08     while(i < num){
09         if(num%i == 0)
10             isPrime = 1;
11         i = i + 1;
12     }
13
14     if(isPrime == 1)
15         printf("%dは素数です.", num);
16     else
17         printf("%dは素数ではありません.", num);
18 }
```

35