ВОЕННО-КОСМИЧЕСКАЯ АКАДЕМИЯ ИМЕНИ А.Ф. МОЖАЙСКОГО
Кафедра информационно-вычислительных систем и сетей

УТВЕРЖДАЮ
ВрИО начальника 24 кафедры
полковник
А. Васильев
«_27_» февраля 2023 года

Тема 4. Оценивание надежности функционирования программного обеспечения

Практическое занятие

Оценивание надежности программного обеспечения с использованием метрик Холстеда

по дисциплине

Надежность автоматизированных систем

Автор: преподаватель 24 кафедры,
кандидат технических наук, доцент С. Баглюк

Обсуждено и одобрено на заседании 24 кафедры
« 27 » февраля 2023 года протокол № 6

Санкт - Петербург

2023

Цель занятия: сформировать у обучающимися практические навыки оценивания надежности программного обеспечения по исходному коду программы с помощью метрик Холстеда.

СОДЕРЖАНИЕ ЗАНЯТИЯ И ВРЕМЯ

Введение 5 мин.

1. Разбор типового примера 40 мин.
2. Выполнение индивидуального задания 135 мин.
3. Подготовка отчета 45 мин.
4. Защита отчета о выполненной работе 40 мин.

Заключение 5 мин.

1. Типовой пример

1.1. Метрики Холстеда

Одним из наиболее распространенных подходов к прогнозированию надежности программ основывается на анализе текста. Подсчитывается число различных операторов и операндов, частота их использования, и на основе этих данных выдается прогноз количества возможных ошибок, содержащихся в тексте программы. Этот подход был впервые описан в книге Холстеда «Начала науки о программах», поэтому оценки надежности, полученные таким образом называют *метриками Холстеда*:

η_1 – *количество отдельных различающихся операторов;*

η_2 – *количество отдельных различающихся операндов;*

N_1 – общее число операторов;

N_2– общее число операндов.

Через первичные характеристики выражается словарь

$$\eta = \eta_1 + \eta_2 \quad (1)$$

и длина программы

$$N = N_1 + N_2. \quad (2)$$

Холстед предложил оценивать длину программы по формуле

$$N^* = \eta_1 log_2 \eta_1 + \eta_2 log_2 \eta_2 . \quad (3)$$

Для больших программ корреляция между N и N^* достаточна велика.

Объем программы вычисляется по формуле:

$$V = N log_2 \eta = (N_1 + N_2) log_2 (\eta_1 + \eta_2). \quad (4)$$

Очевидно, что объем программы зависит от языка программирования. Поэтому Холстед вводит понятие потенциального объема, существующее для такого языка, в котором любая требуемая операция уже определена или реализована в виде процедуры или подпрограммы, тогда:

$$V^* = (N_1^* + N_2^*) log_2 (\eta_1^* + \eta_2^*) . \quad (5)$$

В минимальной форме ни операторы, ни операнды не требуют повторений, поэтому $N_1^* = \eta_1^*$ и $N_2^* = \eta_2^*$. Кроме того, минимально каждый алгоритм должен включать один оператор для имени функции и один оператор присвоения, то есть $\eta_1^* = 2$, тогда выражение для потенциального объема принимает вид:

$$V^* = (2 + \eta_2^*) log_2 (2 + \eta_2^*) . \quad (6)$$

Кроме того, Холстед предложил использовать метрику уровня языка, связанную с профессионализмом программиста и возможностью создания оптимального текста программы:

$$L = \frac{V^*}{V} \quad (7)$$

Если язык программирования не меняется, а меняется алгоритм, то произведение уровня программы на этот потенциальный объем должно быть постоянно и называется уровнем языка:

$$\lambda = LV^* = const . \quad (8)$$

Кроме того, в своих работах Холстед, используя материалы из области психологии работы человеческого мозга, определяет еще несколько характеристик, связанных с описанием работы программистов, например, общее число элементарных мысленных различий:

$$E = \frac{V}{L}. \quad (9)$$

Вводя еще ряд промежуточных показателей, получают выражение для метрики, названное Холстедом как число потенциальных ошибок, введенных в программу:

$$B^* = \frac{E^{2/3}}{3000} . \quad (10)$$

1.2. Пример выполнения индивидуального задания

Проводится анализ текста программы представленной в индивидуальном задании, например:

```
begin
   F:=1;
   read (N);
   for I:=1 to N do F:=F*I;
   writeln (F)
end.
```

Результаты анализа предлагается представить в виде таблицы:

η_1	оператор	количество	η_2	операнд	количество
1	:=	3	1	F	4
2	;	3	2	N	2
3	begin … end	1	3	I	2
4	read (…)	1	4	1	2
5	for … do	1			
6	writeln(…)	1			
7	*	1			
8	.	1			
η_1=8		N_1=12	η_2=4		N_2=10

Используя формулы (1) – (10), необходимо оценить число потенциальных ошибок в предложенном тексте программы, вычислив метрики Холстеда.

1) Словарь (1): $\eta = \eta_1 + \eta_2 = 8 + 4 = 12$.

2) Длина программы (2): $N = N_1 + N_2 = 12 + 10 = 22$.

3) Длина программы по Холстеду (3):

$$N^* = \eta_1 log_2 \eta_1 + \eta_2 log_2 \eta_2 = 8 \cdot log_2 8 + 4 \cdot log_2 4 = 32.$$

4) Объем программы в битах (4): $V = N\, log_2 \eta = 22 \cdot log_2 12 = 78{,}869 \approx 79$.

5) Потенциальный объем (6):

$$V^* = (N_1^* + N_2^*) log_2 (\eta_1 + \eta_2^*) = (2+4) log_2 (2+4) = 6\, log_2 6 = 15{,}51 \approx 16.$$

6) Уровень программы (6): $L = \frac{V^*}{V} \approx 0{,}203$.

7) Уровень языка (8): $\lambda = LV^* \approx 3{,}25$.

8) Общее число элементарных мысленных различий (9):

$$E = \frac{V}{L} \approx 389{,}2.$$

9) Число потенциальных ошибок, внесенных в программу (10):

$$B^* = \frac{E^{2/3}}{3000} \approx 0{,}0178.$$

Вероятность безотказной работы программы может быть вычислена по формуле

$$P_{\text{п}}(t) = e^{-\frac{B^*}{T_{\text{отл}}} t}, \qquad (11)$$

где $T_{\text{отл}}$ – время отладки программы;

t – время функционирования программы.

2. Порядок выполнения индивидуального задания

По аналогии с типовым примером, рассмотренным в п. 1, для фрагмента программы, приведенного в индивидуальном задании, рассчитать вероятность его безотказной работы.

Вариант 1

```
program R (Output);
var
   I, N ,X,Y: integer;
   procedure fibo (var First, Second: integer);
   var
      Fi : integer;
   begin
      Fi :=First;
      First := Second;
      Second := First+Fi;
   end;
begin
   X :=1;
   Y :=1;
   readln(N);
   writeln('Числа Фиббоначчи' : 35);
   write (X:1,',Y:1);
   for I :=1 to N-2 do
      begin
         fibo(X,Y);
         write(',Y:1);
      end;
end.
```

Вариант 2

```
program Day (Input, Output);
type
   DayOfWeek = (SUN, MON, TUE, WED, THU, FRI, SAT);
var
   D: DayOfWeek;
   Well: boolean;
   procedure ReadDW (var I: DayOfWeek; var Well: boolean);
   var
      C:char;
   begin
      read(C);
      Well:=false;
      case C of
         'S': begin
                 readln(C);
                 case C of
                    'U': begin Well:=true; I:= SUN end;
                    'A': begin Well:=true; I:= SAT end;
                 end;
              end;
         'M': begin
                 Well:=true;
                 I:=MON;
               end;
         'T': begin
                 readln(C);
                 case C of
                    'U': begin Well:=true; I:= TUE end;
                    'H': begin Well:=true; I:= THU end;
                 end;
              end;
         'W': begin Well:=true; I:= WED end;
         'F': begin Well:=true; I:= FRI end;
      end; {case}
      readln;
   end;
begin {main}
      ReadDW(D,Well);
      if not Well then
         writeln ('Ошибочный символ');
end. {main}
```

Вариант 3

```
program Fn (Input, Output);
var
   F,X,A,B,H,K,P : real;
   function CH(T:real):real;
   begin
      CH:=(exp(T)+exp(-T))/2;
   end;
begin
   read(A,B,H,K,P);
```

```
    writeln('H='H, 'K='K, 'P='P);
    X:=H;
    while X<=K do
    begin
        F:=A+B*CH(X);
            Writeln(X:5:1,' ',F:10:3);
            X:=X+P;
    end;
end.
```

Вариант 4

```
program gun;
var
    v0, alpha, t, dt, x, y, x0, y0,
    vx0, vy0, tc: extended;
    n, i: integer;
const g=9.81;
    begin
        ClrSer;
        writeln('введите начальную скорость в м/с');
        readln(v0);
        writeln;
        writeln('введите наклон траектории в градусах');
        readln(alpha);
        writeln;
        writeln('введите число точек');
        readln(N);
            alpha:=Pi*alpha/180;
            vx0:=v0*cos(alpha);
            vy0:=v0*sin(alpha);
            x0:=0; y0:=0; t:=2*vy0/g;
            dt:=t/(N-1); i:=1; tc:=0;
            while i<N do
                begin x:=x0+vx0*tc;
                    y:=y0+vy0*tc-g*sqr(tc)/2;
                Writeln(x,' ',y);
        end; end.
```

Вариант 5

```
program numint;
var
    n, i, k: integer;
    a, b, h, s: real;
    function F(x: real): real; {интегральная функция}
    begin
    F:=exp(-x*x) end;
        procedure TRAP (a, b: real; n: integer; F: real; var S: real);
        var i: integer;
        h: real;
        begin
            h:=(b-a)/n;
```

```
        S:=(F(a)+F(b))/2;
            for i:=1 to n-1 do S:=S+F(a+i*h);
            S:=S*h
    end;
begin {начало основного блока программы}
    a:=0; b:=2; n:=10;
    TRAP (a, b, n, F, S); S:=2*S/sqrt(Pi);
    writeln (‘Интеграл равен’, S)
end.
```

Вариант 6

```
program gorner2;
var
    a: array[0:3] of real;
    i: integer;
    n, x: real;
    begin
        readln (x);
            for i:=1 to 3 do read (a[i]);
            readln;
            n:=x/4+a[3];
            for i:=2 down to 0 do
            n:=x/n+a[i];
            writeln (‘n(x)=’, n)
        end;
    end.
```

Вариант 7

```
program M5;
var
    a: array[1..10, 1..10] of integer;
    i, k: byte;
    s: integer;
    begin
        s:=0;
        for i:=1 to 10 do
        begin
        for k:=1 to 10 do
        begin
            a[I, k]:=Trunc(Random*100)+1;
            write(a[i, k]:6);
            if k>i then s:=s+a[i, k]
        end;
        writeln
    end.
```

Вариант 8

```
program p1;
var
    z, x, b, xb, r1, v1, v2: real;
    procedure PA (var a, result: real); {описание процедуры}
    var w, v: real;
```

```
        begin
            v:=sqr(a)+1.0; w:=sin(a);
            result:=sqr(w)+cos(v)
        end; {раздел операторов основной программы}
    begin
        writeln (‘введите значения для X и B’);
        read (x, b);
        r1:=sqr(x)+5;
        PA(r1, v1); xb:=x*b;
        PA(xb, v2);
        z:=v1-sqr(v2);
        writeln (‘Результат’,z)
    end.
```

Вариант 9

```
uses crt,graph;
label 1,2;
    var
    rx,ry,r:integer;
    y0,x1,x2,x3,y1,y2,y3:integer;
    a,k,i:integer;
    p:real;
    c:char;
    gm,gd:integer;
    procedure sqare(col,p1,q1,p2,q2,p3,q3:integer);
    begin
        setcolor(col);
        line(p1,q1,p2,q2);
        line(p2,q2,p3,q3);
        line(p3,q3,p1,q1);
    end;
    procedure change1(var p1,q1,p2,q2,p3,q3:integer);
    begin
        p1:=i+50-round(60*sin((a+60)*p));
        q1:=200-round(60*cos((a+60)*p));
        p2:=i+50-round(60*sin(a*p));
        q2:=200-round(60*cos(a*p));
        p3:=i+50;
        q3:=200;
    end;
procedure change2(var p1,q1,p2,q2,p3,q3:integer);
    begin
        p1:=i;
        q1:=200;
        p2:=i-round(60*cos((a+60)*p));
        q2:=200-round(60*sin((a+60)*p));
        p3:=i+round(60*cos(a*p));
        q3:=200-round(60*sin(a*p));
    end;
    procedure change3(var p1,q1,p2,q2,p3,q3:integer);
    begin
        p1:=i+round(60*cos(a*p));
```

```
        q1:=round(60*sin(a*p));
        p2:=i-50;
        q2:=200;
        p3:=i-50+round(60*cos((a+60)*p));
        q3:=200-round(60*sin((a+60)*p));
    end;
    begin
        clrscr;
        k:=1;
        p:=pi/180;
        gd:=detect;
        initgraph(gd,gm,' ');
        line(0,201,getmaxx,201);
        i:=-50;
2: repeat
    if k=1 then
    for a:=0 downto -90 do
    begin
        change1(x1,y1,x2,y2,x3,y3);
        sqare(white,x1,y1,x2,y2,x3,y3);
        delay(1000);
        sqare(black,x1,y1,x2,y2,x3,y3);
    end
    else if k=-1 then for a:=0 to -180 do
    begin
        change2(x1,y1,x2,y2,x3,y3);
        sqare(white,x1,y1,x2,y2,x3,y3);
        delay(5);
        sqare(black,x1,y1,x2,y2,x3,y3);
    end;
    i:=i+k*50;
    until (i>getmaxx+50) or (i<-50) or (keypressed);
    if keypressed then c:=readkey
    else if i>getmaxx+50 then i:=-50
    else i:=getmaxx+50;
    if ord(c)=ord('p') then
    begin
        k:=k*(-1);
        c:='q';
    end
    else if ord(c)=ord('e') then goto 1 else c:='q';
    goto 2;
1: closegraph;
    end.
```

Вариант 10

```
program _tst;
uses crt;
const sega000=$a000;
const grc_addr:word=$3ce;
const NVertx=24;
const x_max=10.0;
```

```
const y_max=7.0;
const Xp_max=639;
const Yp_max=479;
var
    j: integer;
    ch: char;
    color: byte;
    X1,Y1: integer;
    horfact,verfact: double;
    x,y: array [0..NVertx] of double;
    function iX( x:double):integer;
    begin
        iX:=round((x+x_max/2.0)*horfact+0.5);
    end;
    function iY( y:double):integer;
    begin
        iY:=Yp_max-round((y+y_max/2)*verfact+0.5);
    end;
    procedure moveTo_(x,y: double);
    begin
        X1:=iX(x);
        Y1:=iY(y);
    end;
    procedure SetVM(mode_:word);
    begin
        asm
            mov ax,mode_
            int 10h
        end;
    end;
    procedure lineTo_(x, y :double; col:byte);
    var X2,Y2: integer ;
    begin
        X2:=iX(x);
        Y2:=iY(y);
        line_(X1,Y1,X2,Y2,col);
        X1:=X2; Y1:=Y2;
    end;
    begin
        horfact:=(Xp_max/x_max);
        verfact:=(Yp_max/y_max);
        x[0]:=0.0;
        y[0]:=0.0;
        x[1]:=1.0;
        y[1]:=0.0;
        x[2]:=1.0;
        y[2]:=1.0;
        x[3]:=0.0;
        y[3]:=1.0;
        SetVM($12);
        color:=12;
        moveTo_(x[0],y[0]);
```

```
    for j:=1 to 3 do
    lineTo_(x[j],y[j],12);
    lineTo_(x[0],y[0],12);
    ch:=readkey;
    SetVM($3)
end.
```

3. Содержание отчета

По выполнению задания каждый курсант должен представить отчет. Отчет должен содержать:

- название практического занятия;
- цель занятия;
- индивидуальное задание к работе;
- перечень основных характеристик надежности программного обеспечения;
- результаты выполнения индивидуального задания;
- выводы по работе.

Отчетный материал представляется преподавателю, и результаты защищаются с выставлением оценки.

4. Критерии для оценивания выполнения индивидуального задания

«Отлично», если обучающийся правильно выполнил индивидуальное задание и правильно ответил на заданные преподавателем контрольные вопросы.

«Хорошо», если обучающийся правильно выполнил индивидуальное задание и правильно ответил не на все заданные преподавателем контрольные вопросы.

«Удовлетворительно», если обучающийся неправильно выполнил индивидуальное задание, но правильно ответил на большинство заданных преподавателем контрольных вопросов.

«Неудовлетворительно», если обучающийся неправильно выполнил индивидуальное задание и не ответил на заданные преподавателем контрольные вопросы.

5. Контрольные вопросы

1. Что понимается под надежностью программного обеспечения?
2. Что является причиной отказа программного обеспечения?

3. Чем отличается отказ программного обеспечения от отказа аппаратуры?

4. Как оценивается надежность программного обеспечения?

5. Чем отличаются аналитические от эмпирических моделей надежности программного обеспечения?

6. Перечислите основные метрики Холстеда.

7. Каким образом можно повысить надежность программного обеспечения?

С. Баглюк

(воинское звание, подпись, инициал имени, фамилия автора)

« 20» февраля 2023 г.