Best Practices in Code Inspection for Safety-Critical Software

IEEE Software (May/June 2003)

Jorge Rady de Almeida Jr., Joao Batista Camargo Jr., Bruno Abrantes Basseto, and Sergio Miranda Paz, Escola Politecnica da USP
Email: jorge.almeida@poli.usp.br

Checklist topics	Error (%)
1. routine return methods	5
2. interrupt-handling routines and critical regions	5
3. repetitive-loop control	5
4. I/O tests	20
5. program flow control	10
6. unused source code	10
7. variables and constants	10
8. source code comments	20
9. source code legibility	10
10. preprocessor directives	5
11. code optimization	0

```
/* 1. routine return methods */
int a (int b)
{
       if (b == 0)
       {
              d = 1;
              return 0;
       }
       else
       {
              if (b == 1)
                     d = 3;
              // this path does not return a value
              else
                     return 1;
       }
}
```

```
/* 2. interrupt-handling routines and critical regions */
void
interrupt int_rx(void)
{
       if (n_queue == MAX) return;
       // this path is dangerous
       queue[n_queue++] = inport (P_ADDRESS);
       outport(EOI, EOIVALUE);
}

interrupt void insert(void)
{
       char   new;
       new = inp(0xf3);
       queue fila[pos] = new;
       pos++;
}
char remove(void)
{
       char   ret;
       if (pos == 0) return -1;
       ret = file[pos];
       pos--;
       return ret;
}
```

```
/* 3. repetitive-loop control */
void a (int b)
{
        char i;
        static c[256];
        for (i = b; i > 0; i--)
                c[i] = 0;
}


void a (int b)
{
        char i;
        for (i = 0; i < b; i++)
                c[i] = 0;
}


/* 4. I/O tests */
void
routine(void)
{
        static int      IO_Test = 0;

        if (IO_Test)  fail();
        IO_Test = 1;
        /* routine body */
        if (!IO_Test) fail();
        IO_Test = 0;
}


/* 5. program flow control */
        if sel > 3 then
                call fail;
        else do case sel;
                do;     /* case 0 */
                end;
                do;     /* case 1 */
                end;
                do;     /* case 2 */
                end;
        end;

switch (a)
{
        case 0:
                b = 1;
                break;
        case 2:
                b = 6;
        case 3:
                b = 7;
                break;
}
```

/* 7. variables and constants */

```
ex:1
       if (i > MAX) fail();
              // lower limit not verified
       queue[i] = queue[i+1];
              // i+1 may be out of bounds
ex:2
file #1:
       MAX EQU      $2F
       ; assembly declaration in decl.equ

file #2:
       #define      MAX    48
       // C declaration i ext.h

ex:3
extern char *test;
       //declaration in file test.h

int    test[80];
       // declaration in file test.c
```

/* 9. source code legibility */

```
BYTE Mode;
       #define      MIN_Temperature 32

BYTE   *pMode;
       #define      MAX_Temperature 212
```