ソフトウェア開発論

プログラミング局面

第2.16版

- 構造化プログラミング
- コード・インスペクション

ソフトウェア開発論・スケジュール

- はじめに
- 要求分析
- 外部設計
- 内部設計
- **プログラミング局面**
- 単体試験
- 結合試験
- 総合試験
- 信頼性評価
- まとめ

構造化プログラミング

- 高信頼性、保守性の鍵となる技術
- 構造化プログラミングすれば、効率の良いコード・インスペクションおよび試験(テスト)が出来るようになる。

注:プログラム・コードのことを、以下コードと略す。

構造化プログラミング

- 作法(HOW)
  - go to文を使用しない。文としては、**順次、条件、繰り返し**＝入り口１個、出口１個の物**のみ**を用いてコードを作成する。
    - 注意: javaはgotoがない。
  - begin..endや{..}で字下げ(２字もしくは3字)を行う。
- なぜ(WHY)
  - 正当性定理:順次、条件、繰り返し文のみで構成されている場合は、正当な(欠陥のない)プログラムを作成可能である。[H.D.Mills, 1987]

構造化プログラミングされたコード例

```
import java.util.*;
import java.io.*;
//返済シミュレーション・ソフト hensai2.java
//倍精度計算して見れば
class hensai2 {
  public static void main(String[] args) {
    try{
      FileWriter F2= new FileWriter("hensai2.dat");
      long m=1000000L;//元金　（単位：円）
      double r=23;　　//年利　（単位：%）
      int n=10;　　//期間　（単位：年）
      F2.write("Java2 hemsai2.java: "+ (long)(m * Math.pow( 1+(r/100) ,n))+"¥r¥n");
      double sum=m;
      for (int i=1; i<=10; i++){
        sum= sum + sum*(r/100);
        F2.write("year "+i+" "+sum+"¥r¥n");
      }
      F2.close();
    }
    catch (Exception e) {
      System.out.println("Exception in main: " + e);
    }
  }
  //--------------
}
```

コード・インスペクション

- 方法(HOW)：変数置き換え法による検証、チェックリストによる方法
- 目的(WHY)：内部設計書とコードの差異がないことを確認する。また、内部設計書の不足がないかどうか検査する。
  このことにより、
  内部設計書、コードの欠陥を試験に持ち込まない。
- 効果1：これにより、開発、保守コストの削減が期待できる。
  - 1 ： 10 ： 100の法則
- 効果2：検査カバレージが、試験よりも格段に高い。
  - 100 : 10 : 1　の法則

コード・インスペクションの達人になる。

- コード・インスペクション＆テストを効率的にやるための準備＝構造化プログラミング
  - 順次
  - 条件
  - 繰り返し(for, while文を使い、do文は使わないこと)
- 構造化プログラミングをやれば、100%欠陥は取れる可能性があることが証明されている。（正当性定理）

変数置き換え法による検証

順次文の検証

1. **コード**

```
TEMP = 元金
元金 = 返済額
返済額 = TEMP
```

初期状態：元金o、TEMPo、返済額o

2. トレーステーブル作成(コードから)

	元金	TEMP	返済額
一行目	元金o	元金o	返済額o
二行目	返済額o	元金o	返済額o
三行目	返済額o	元金o	元金o

内部設計書
元金と返済額の入れ替え

3. **検証 （検証対象コードが内部設計書と等価か？）**
   - 最後（三行目）で、元金と返済額の数値が返済額o，元金oと入れ代わっている。 内部設計書と同じ動きをするのでOK

順次文を試験すると…

1. **コード**

```
TEMP = 元金
元金 = 返済金
返済額 = TEMP
```

本当は返済額だけども、タイプミスした。

2. 試験とは、すなわち具体的な特定の数値を初期値として、コードを実行すること
   - 元金o = 2
   - 返済額o = 0
   - （返済金oはたまたま、0）

内部設計書
元金と返済額の入れ替え

3. 試験結果
   - 元金 = 0，返済額 = 2で、入れ替わっているので、コードは正しいように見える！

条件文の検証

1. コード

```
if (顧客ID.equals("234")) {
    返済額 = 元金+1;
}
```

2. トレーステーブル作成(コードから)

- 顧客ID==234 の時

	返済額
二行目	元金o+1

- 顧客IDが234でない時

	返済額
四行目	返済額o

内部設計書

もし、顧客ID==234なら、返済額は元金に1を足す。

3. 検証(トレーステーブルの動きは、内部設計書と同じか？)
   - 顧客ID==234 の時は内部設計書と同じ動作をするのでOK。
     そうでないときは正しいかどうか不明

繰り返し文の検証

- コード
  - long 返済額=元金; double r=本日の金利/36500;
  - for (int i=1; i<=返済予定日数; i++){
  - 返済額 = (long)(返済額*(1+r));
  - }
- 繰り返し回数0の時の検査
  - トレーステーブル作成

	返済額
ループ前	元金
ループ後	元金

  - 検証
    - 返済額=元金で抜けて、内部設計書通りか？
- 無限ループの可能性の検査
  - 返済予定日数が無限大に近いとどうなるか
- 繰り返し回数有限の時の検査(n回目に見つかったと考える)
  - トレーステーブル作成

	返済額
ループ前	元金
ループ1	返済額1=元金*(1+r)
ループ2	返済額2=返済額1*(1+r) =元金*(1+r)*(1+r)=元金*$(1+r)^2$
・・・	
ループn	元金*$(1+r)^n$
ループ後	元金*$(1+r)^n$

  - 検証

内部設計書
日歩利子計算

    - 返済額n=元金*$(1+r)^n$ で抜けていいのか？

条件、繰り返しの複合文の検証

- コード
  - long 返済額=元金; double r=本日の金利/36500;
  - if (顧客ID.equals("234")) {
  - 返済額 = 元金+1;
  - }
  - else for (int i=1; i<=返済予定日数; i++){
  - 返済額 = (long)(返済額*(1+r));
  - }
- 検査項目(4つのトレーステーブルを、コードだけ見て作成する。)
  - ifがtrueの時
  - ifがfalseの時
    - 繰り返し回数0の時
    - 無限ループの可能性
    - 繰り返し回数有限の時
- 検証: 検査結果と内部設計書を比較する。

内部設計書
顧客IDが234の時は、返済額は元金＋1、そうでなければ日歩利子計算

演習：変数置き換え法による検証

- 神様：内部設計書(HIPO：元金取得メソッドより)

  - 顧客IDを手がかりに顧客DBより、お客さまのお名前、元金を取得
  - IF 顧客IDが顧客DBに未登録ならば、 エラーコード=1を返す。

- インスペクション対象：コード

```
private static 元金取得結果 元金取得メソッド(String 顧客ID){
元金取得結果 out=new 元金取得結果();
if (顧客ID.equals("234")) {
  out.お客様のお名前="Satoh Kazuo";
  out.元金=1000000;
}
return out;
}
```

チェックリスト（例）

1. プログラムの全変数は、その値を使用する前に初期化してあるか？
2. すべての定数に名前を付けたか？
3. 条件文中の条件は正しいか？
4. 繰り返しはすべて確実に停止するのか？
5. 配列の処理では、その下限の境界は0か1かあるいはその他の値か？
   上限の境界は配列の大きさもしくは大きさ−1に等しくなっているか？
6. 文字列が使用されているときには、区切り文字を陽に指定しているか？
7. 動的記憶が用いられているときには、記憶域は正しく割り付けられているか？
8. すべての関数と手続き呼び出しは、パラメータの数が正しいか？
9. （型指定言語を指定していないとき）仮パラメータ、パラメータ宣言形式と実パラメータの型は一致するか？
10. リンク、バインド形式の構造を変更するときには、すべてのリンクを正しく再指定しているか？
11. 複合文ではカッコの用法は正しいか？
12. 生じうる誤り条件をすべて考慮しているか？
13. キーワードのチェックに関する部品では、すべてのキーワードをチェックしたか？

出典：SOFTWARE ENGINEERING fourth edition, Ian Sommerville, Addison-Wesley, 1991/07.

おわり