「プログラムの実行履歴を用いた利用モデルの作成方法について」

2003.03.06

香川大学

高木 智彦，古川 善吾

<目次>

1. 背景
2. 目的
3. 統計的テスト（概要）
4. 統計的テスト（手順）
5. テストケース作成支援システム
6. 利用モデル　（概要）
7. 利用モデル　（作成手順）
8. 利用モデル　（プログラムとの対応付け）
9. 利用モデル　（適用例）
10. テストケースの作成
11. 本手法の効果と問題点
12. 今後の研究方針

1. 背景

- 統計的テストは，大量のテストケースを用いて，ソフトウェアの実際の利用状況における信頼性を評価できる．
- ただし，利用モデル（ソフトウェアの利用状況の要約）の作成にコストがかかる点が問題．

2. 目的

GCC（GNU Compiler Collection）およびテストカバレッジツール gcov の機能を利用して，効率的に利用モデルを作成する方法を確立する．

3. 統計的テスト（概要）

- ユーザの使用特性を外部仕様の観点から統計的に分析し，テストケースに反映させる．
- 使用頻度の高い機能にテストの重心を置くことによって，ユーザがよく使う機能に含まれるバグを重点的に検出．

＜従来のテスト＞
・全体を満遍なくテストする
・網羅基準 C_0, C_1）によるテスト十分性評価

＜統計的テスト＞
・使用頻度の高い機能にテストの 重点を置く
・MTTF（Mean Time To Failure）によるテスト十分性評価

4. 統計的テスト（手順）

1) 利用モデルの構築
・**利用モデル**：ソフトウェアの利用状況に関する要約．
マルコフ連鎖（遷移確率付き状態遷移図）で表される．

2) テストケースの生成
・**テストケース**：初期状態から終了状態に至る，状態と遷移の列．
マルコフ連鎖の遷移確率を満たす．

3) テストの実施

4) テストモデルの構築
・**テストモデル**：テスト履歴の要約．マルコフ連鎖で表される．

5) 信頼性（テスト十分性）の評価

5. テストケース作成支援システム

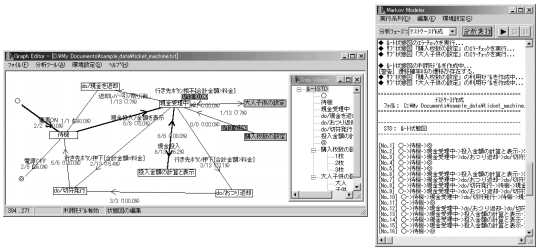


図10, 図11.　テストケース作成支援システム

6. 利用モデル (概要)

- 利用モデル：ソフトウェアの利用状況の要約.
- 実際の利用状況を反映したテストケースを得るためのモデル
- マルコフ連鎖 (遷移確率付き状態遷移図)で表す.
- ソフトウェアの仕様として作成される状態遷移図と，ユーザのソフトウェアに対する使用特性を用いて作成する.

⇩

ソフトウェアのプロトタイプや過去のバージョンの実行履歴

7. 利用モデル (作成手順)

1) 状態遷移図とプログラムを用意する.
   - ソフトウェアの外部仕様をまとめる.
   - 入出力に関する機能を漏れなく記述する.
2) プログラム中に標識コメントを挿入する.
   - 状態遷移図の状態や遷移に対応するプログラムのコード行に標識コメントを挿入する.
3) プログラムを実行する.
   - GCCを用いて，プログラムをコンパイルし実行.
   - gcovを用いて，プログラムの各コード行が実行される頻度 (累積の実行回数)を取得する.
4) 実行履歴を状態遷移図に関連付ける.
   - 標識コメントを手がかりに，コード行の実行回数を状態や遷移に関連付ける.
   - 遷移確率を計算し，利用モデルを完成させる.

8. 利用モデル (プログラムとの対応付け)

標識コメントの直後のコード行の実行回数を状態や遷移の実行回数とみなす.

＜標識コメントの挿入位置＞

① 関数宣言の直前 (状態に対応)
② 関数呼び出しの直前 (遷移に対応)
③ 入出力ステートメントの直前 (状態に対応)
④ その他

9. 利用モデル (適用例)

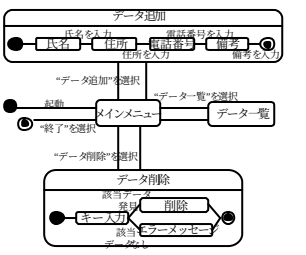


図1. アドレス帳プログラムの状態遷移図

```
/* std:メインメニュー */
int menu_screen(void) {
  int num;
  do {
    printf("********************************\n");
    printf("*  アドレス帳メインメニュー  *\n");
    printf("********************************\n");
    printf("○ 機能を選択して下さい \n");
    printf("データ追加 : 1   データ削除 : 2\n");
    printf("データ一覧 : 3   終了       : 4\n");
    scanf("%d", &num);
  } while (num < 1 || num > 4);
  return num;
}

================================================

while ((menu_item = menu_screen()) != END) {
  switch (menu_item) {
    case ADD:
      /* std: "データ追加"を選択 */
      add_item();
      break;
    case DELETE:
      /* std: "データ削除"を選択 */
      delete_item();
      break;
    case DISPLAY:
      /* std: "データ一覧"を選択 */
      display();
      break;
  }
}
```

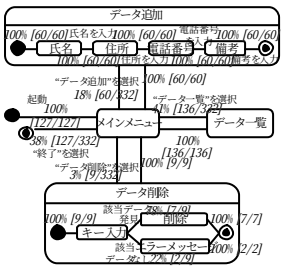


図6. アドレス帳プログラムの利用モデル例

```
/* std:メインメニュー */
int menu_screen(void) {
332   int num;
332   do {
332     printf("********************************\n");
332     printf("*  アドレス帳メインメニュー  *\n");
332     printf("********************************\n");
332     printf("○ 機能を選択して下さい。\n");
332     printf("データ追加 : 1   データ削除 : 2\n");
332     printf("データ一覧 : 3   終了       : 4\n");
332     scanf("%d", &num);
332   } while (num < 1 || num > 4);
332   return num;
332 }

================================================

332 while ((menu_item = menu_screen()) != END) {
205   switch (menu_item) {
 60     case ADD:
          /* std: "データ追加"を選択 */
 60       add_item();
 60       break;
        case DELETE:
          /* std: "データ削除"を選択 */
  9       delete_item();
  9       break;
        case DISPLAY:
          /* std: "データ一覧"を選択 */
136       display();
136       break;
205   }
205 }
```

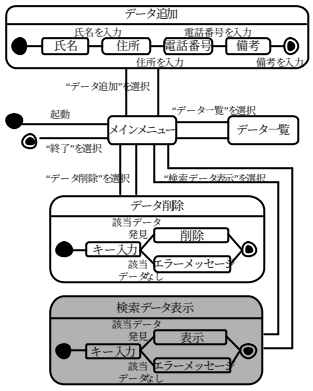


図7. 新機能を追加したアドレス帳プログラムの状態遷移図

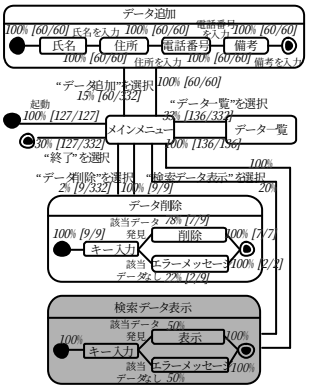
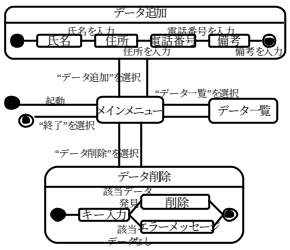


図8. 新機能を追加したアドレス帳プログラムの利用モデル例

10. テストケースの作成



初期状態」→起動→「メインメニュー」→"データ追加"を選択→「データ追加」(初期状態」→ラムダ遷移→ 氏名」→氏名を入力→ 住所」→住所を入力→ 電話番号」→電話番号を入力→ 備考」→備考を入力→ 終了状態」)→ラムダ遷移→ 「メインメニュー」→"データ一覧"を選択→ 「データ一覧」→ラムダ遷移→ 「メインメニュー」→" 終了"を選択→ 終了状態」

11. 本手法の効果と問題点

＜効果＞

- 図の流用が可能
- 過去のバージョンが利用可能
- 仕様とプログラム間の矛盾点の発見

＜問題点＞

- 標識コメントの挿入が煩雑
- テストモデルの作成が不可能

12. 今後の研究方針

- テストの実施や信頼性評価の考察
- テスト支援システムの試作
- 本手法の客観的な有効性の調査