5C−5

プログラムスライスに基づいたデバッグ支援システムの評価のための追証実験

芦田佳行　西松顯　楠本真二　井上克郎

大阪大学大学院基礎工学研究科

1 まえがき

デバッグにおけるフォールト位置特定を効率良く行うための手法としてプログラムスライス[3]が提案されている．プログラムスライスとは，プログラム内のある文のある変数(スライシング基準)に影響を与える全ての文の集合であり，プログラムスライスを利用することでプログラムの参照範囲を限定できる．これまでに我々はプログラムスライスがフォールト位置特定に有効であることを実験的に評価した[1]．具体的には，被験者をプログラムスライスを利用する者と利用しない者に分けてフォールト位置特定を行い，それに要した時間を比較するというものであった．本研究では，この実験の被験者数を増やし，より正確な評価を試みた．

1.1 デバッグ支援ツール

デバッグ支援ツール[2]は大きく分けて2つの機能($Fc1$,$Fc2$)を持つ．

($Fc1$) 通常のデバッガと同様にプログラムのステップ実行，変数の参照，ブレークポイントの設定などが行える．

($Fc2$) 与えられたスライシング基準から静的スライスを抽出する．

対象となる言語は，次のような仕様を持つPascalのサブセットである．(1)文として，代入文，条件文，繰り返し文，手続き呼出文，begin–endで囲まれる複合文を扱う．(2)手続きの再帰呼び出しも扱う．手続きの引数には，値渡しと変数渡しの2種類がある．(3)変数のデータ型はスカラー型のみでポインタ型は扱わない．具体的には整数型，文字型，論理型およびそれらを要素に持つ配列型である．

2 前回の実験

文献[1]で行った実験について述べる．実験の目的は，デバッグ支援ツールを使い，プログラムスライスのフォールト位置特定に対する有効性を確認することである．

An Additional Experiment for evaluating the usefulness of debugging system using program slicing technique
Yoshiyuki Ashida, Akira Nishimatsu, Shinji Kusumoto and Katsuro Inoue
Graduate School of Engineering Science, Osaka University, 1-3 Machikaneyama, Toyonaka, Osaka, 560-8531 JAPAN.

2.1 被験者

実験は大阪大学基礎工学部情報科学科の学生6人に対して行った．6人の被験者は，2つのグループG1とG2にそれぞれ3人ずつ分け，G1に含まれる被験者をA1,A2,A3，G2に含まれる被験者をB1,B2,B3と呼ぶ．

2.2 対象プログラム

酒屋問題[4]に対するプログラム($P1$)をデバッグの対象とする．$P1$に対して$F1.1$〜$F1.8$の8つのフォールトを準備し，それを用いて$P1.1$〜$P1.8$の8つのプログラムを作成した．$P1.i$は$F1.i$〜$F1.8$のフォールトを含んでいる．また，$data1.1$〜$data1.8$の8つのテストデータがあり，$data1.i$を用いると$F1.i$のみを発見できる．

2.3 実験方法

G1に属する被験者は，$P1.1$〜$P1.8$に対してツールの($Fc1$)と($Fc2$)の両方を使ってフォールト位置を特定する．G2に属する被験者は，$P1.1$〜$P1.8$に対して$Fc1$のみを使ってフォールト位置を特定する．

実際の手順は次の通りである．

Step0 :$i=1$とする．

Step1 :$data1.i$を用いてプログラム$P1.i$のフォールト位置の特定を行う．

Step2 :特定したフォールトとその位置を実験監督者に申告する．
正しい場合は**Step3**へ．
間違った場合には，**Step1**へ戻る．

Step3 :$i==8$の場合実験終了．$i<8$の場合，$i=i+1$として**Step1**へ戻る．

2.4 分析

G1が全てのフォールト位置特定に要した時間は平均で122分(A1:119分，A2:128分，A3:120分)であり，G2が全てのフォールト位置特定に要した時間は平均で165分(B1:154分，B2:175分，B3:166分)であった．このデータに対して有意水準1%で平均値の差の検定を行うと有意な差が見られた．

また，フォールトごとに平均値の差を調べると，$F1.3$,

$F1.5$, $F1.6$ の 3 つのフォールトで特に有意な差が見られた.

3 追証実験

3.1 実験の目的

前回の実験では, スライスを利用したグループの方がフォールト位置の特定に要した時間は少ないという結果が得られた. しかし, サンプル数は 6 人と少ないものであった. そこで本研究では, 新たな被験者に同じ実験を行ってもらい, そこで得られたデータと前回の実験データを合わせて分析し, より正確な評価を行うことを目的とする.

3.2 実験方法

今回の実験は大阪大学基礎工学部情報科学科の学生 6 人に対して行った. 6 人の被験者を 2 つのグループに分け, A'1,A'2,A'3 の 3 人の被験者を G1 に, B'1,B'2,B'3 の 3 人の被験者を G2 に含める. 実験の方法は前回の実験と全く同じとした.

3.3 分析

今回の実験データを前回の実験データに追加して評価する. G1 の実験データを表 1 に, G2 の実験データを表 2 にそれぞれ示す.

G1 が全てのフォールト位置特定に要した時間は平均 115 分 (A1:119 分, A2:128 分, A3:120 分, A'1:127 分, A'2:102 分, A'3:93 分), G2 は平均 159 分 (B1:154 分, B2:175 分, B3:166 分, B'1:151 分, B'2:169 分, B'3:140 分) であった. このデータに対して有意水準 1%で平均値の差の検定を行うと有意な差が見られた.

各フォールトごとに平均値の差の検定を行うと, 前回の実験では $F1.3$, $F1.5$, $F1.6$ で有意な差が見られたが, これらのフォールトに関しては, 今回の実験においても同様の結果が得られた.

3.4 考察

ここでは, 有意な差が出たフォールト, 全く差が見られなかったフォールトについて考察する.

全く差が見られなかったフォールトは $F1.4$ である. $F1.4$ はある必要な代入文の欠如というフォールトである. 文が欠如しているフォールトの場合, 文があるべき場所の周辺がプログラムスライスに含まれないことがあり, その場合はプログラムスライスがあまり有効でないと考えられる.

有意な差が見られたものは $F1.3$, $F1.5$, $F1.6$ である. $F1.5$, $F1.6$ は, 誤った出力変数に対するプログラムスライスを見るだけで容易にフォールトを発見できるものであった. また, $F1.3$ は関数呼び出し文が欠如しているものであった. しかし, 同じような処理を行う関数が 2 つあり, その一方に文の欠如というフォールトが含まれており, 出力変数に対するプログラムスライスにその 2 つの関数が含まれるため, 2 つの関数を比較することで容易にフォールトを発見できた.

4 まとめ

今回, プログラムスライスがフォールト位置の特定に有効であるかを調べるための追証実験を行った. その結果, プログラムスライスを利用したグループとそうでないグループでは有意な差があることが確認できた.

表 1: G1(スライス利用)

	A1	A2	A3	A'1	A'2	A'3
$F1.1$	31	26	17	28	16	18
$F1.2$	8	10	20	12	16	7
$F1.3$	15	15	13	13	7	18
$F1.4$	25	20	22	24	13	21
$F1.5$	14	26	18	15	20	9
$F1.6$	15	10	10	15	11	8
$F1.7$	4	12	8	7	12	5
$F1.8$	7	9	12	13	7	7
total	119	128	120	127	102	93

表 2: G2(スライスなし)

	B1	B2	B3	B'1	B'2	B'3
$F1.1$	17	28	23	40	55	18
$F1.2$	10	18	12	13	10	12
$F1.3$	26	36	28	31	20	31
$F1.4$	27	17	32	17	11	18
$F1.5$	35	25	41	15	26	28
$F1.6$	17	23	17	17	21	15
$F1.7$	17	16	7	5	13	6
$F1.8$	15	12	6	13	13	12
total	154	175	166	151	169	140

参考文献

[1] 西江, 神谷, 楠本, 井上:"プログラムスライスに基づくデバッグ支援ツールの実験的評価", ソフトウェアシンポジウム 97 予稿集, pp.142-147(1997).

[2] 佐藤, 飯田, 井上:"プログラムの依存解析に基づくデバッグ支援ツールの試作", 情報処理学会論文誌, Vol. 37, No.4, pp.536-545(1996).

[3] Weiser, M.:"Programmers use slices when debugging", *Communications of the ACM*, Vol. 25, No.7, pp. 446-452(1982).

[4] 山崎利治:"共通問題によるプログラム設計技法解説", 情報処理学会誌, 25, 9, p.934(1984).