保守　ロセス　対するプログラムスライスの実験的評価

西松　　顯[†]　　楠本　真二[†]　正員）
井上　克郎[†]（正員）

An Experimental Evaluation of Program Slicing on Software Maintenance Process

Akira NISHIMATSU[†], *Nonmember*, Shinji KUSUMOTO[†], and Katsuro INOUE[†], *Members*

[†] 大阪大学 大学院基礎工学研究科, 豊中市
Graduate School of Engineering Science, Osaka University, Toyonaka-shi, 560-8531 Japan

あらまし　本論文　は　ログラムスライスが実際の保守作業に有効であるかどうかを実験的に評価した．実験の結果，プログラムスライスを用いた場合の方が，プログラムスライスを用いない場合よりも効率よく保守作業が行えることが確認できた．

キ　ワ　ド　プログラムスライス，保守プロセス，プログラム理解

1. ま え が き

ソフトウェア開発におけるコストの大部分は，テスト，デバッグと保守プロセスに費されている．保守プロセスの主な時間はプログラム理解の時間と指摘されている [6]．そのためプログラム理解を支援する方法の開発が重要となっている．プログラム理解を支援するための方法の一つとして，プログラムスライス (Program Slice，スライスと略す) を利用した手法が提案されている．スライスを用いることにより，プログラム中で変更を行うべき部分をスライスとして抽出し，スライスに含まれる部分にのみ注目することで，プログラム理解を効率良く行うことが期待される．

本論文では，文献 [3] で開発したスライシングツール (ツールと呼ぶ) を用いて，プログラムスライスが実際の保守プロセスに有効であるかどうかを実験的に評価をした(注1)．

2. 適 用 実 験

2.1 実 験 概 要

実験の概要を表1に示す．具体的には，酒屋問題 [8] に対する仕様書 (S1) とレストランの座席予約システム問題 (予約問題と呼ぶ) に対する仕様書 (S2) と，それぞれの仕様書に対応するプログラムを2種類ずつ用意する (S1に対応する2種類のプログラムを，それぞれP1.1，P1.2，S2に対応する2種類のプログラムを，それぞれP2.1，P2.2とする)．また，被験者6人を2つのグループG1とG2に分ける．まず，G1の被験者 (A1，A2，A3) がP1.1およびP1.2を，G2(B1，B2，B3) の被験者がP2.1およびP2.2を，それぞれスライスを用いずに保守作業を行う．次に，被験者に対してスライスの講義を行ない，実際にスライスの保守への適用例を示す事で，被験者のスライスへの理解を深める．そして，G1の被験者がP2.1及びP2.2を，G2の被験者がP1.1及びP1.2を，それぞれスライスを用いて保守作業を行う．上記のように被験者は各問題に対して2回，全体では計4回の保守作業を行う．

表1　実 験 概 要
Table 1　Overview of experiment

	G1(3人)	G2(3人)
適用実験1	**Exp1.1** 酒屋問題 (P1.1,P1.2) (スライス利用不可)	**Exp1.2** 予約問題 (P2.1,P2.2) (スライス利用不可)
講義	スライス	
適用実験2	**Exp2.1** 予約システム問題 (P2.1,P2.2) (スライス利用)	**Exp2.2** 酒屋問題 (P1.1,P1.2) (スライス利用)

2.2 保 守 作 業

本実験における保守作業とは被験者に対してプログラムと仕様書を与え，その仕様書に対する変更要求を被験者にプログラムに対して反映してもらうというものである．ここで被験者の行う作業は，まず，与えられた仕様書の内容を理解し，プログラムがどのような機能を実現しているのかを調べ (仕様書理解 [Step1])，仕様書に書かれているある機能がプログラムのどの部分で実現されているかを対応づける (仕様書とプログラムの対応 [Step2])．仕様書を理解した後に，変更部分認識として与えられた仕様書の変更内容を理解し (変更内容理解 [Step3])，変更するべきプログラムの部分を認識する (変更内容とプログラム変更部分の対応 [Step4])．プログラム中の変更すべき部分を認識できれば，被験者は実際にプログラムを変更する (変更 [Step5])．Step5では，プログラムの編集，コンパイル，変更内容が正しく実現できているかを確認するテスト，そして，仮にフォールトが存在する場合はデバッグを行う．

2.3 実 験 結 果

本実験で計測するデータは2.2節で述べたStep2〜Step5に要した時間である．本来，本実験ではスライス抽出機能の利用により保守作業が効率良く行える

(注1)：Control Experiment [9]であり，データ収集・分析はGQMパラダイム [1]を用いた．

Step2の部分のみの時間を計測し，評価するべきであるが，被験者がプログラムの理解，仕様書の変更の理解，プログラムの変更部分の認識を正しく行ったかどうかを正確に判断できないと考えたため(例えば，非常に短い時間でプログラム理解を終了した被験者が，プログラム変更に多くの時間を費した場合，被験者のプログラム理解が不十分である事が言える)，実際にプログラムを正しく変更する作業までの時間を計測，評価する事にした．

各仕様書の変更をプログラムへ反映させるのに要した時間(単位:分)のデータを表2と表3に示す．

表2 酒屋問題
Table 2 S1(P1.1, P1.2)

	P1.1	P1.2	合計
A1	75	50	125
A2	133	89	223
A3	188	301	489
平均	132.0	146.0	279.0
B1	126	87	213
B2	110	60	170
B3	90	64	154
平均	108.7	70.3	179.0

表3 予約問題
Table 3 S2(P2.1, P2.2)

	P2.1	P2.2	合計
B1	105	95	200
B2	99	106	205
B3	112	126	238
平均	105.3	109.0	214.3
A1	44	37	81
A2	82	69	151
A3	78	74	152
平均	68.0	60.0	128.0

3. 分析・評価

酒屋問題の2種類の仕様変更に要した平均時間は，スライスを利用したG2では179.0分(B1:213 B2:170 B3:154)，スライスを利用しなかったG1では279.0分(A1:125 A2:223 A3:489)となっている．平均時間を見るとG2の方がG1よりも仕様変更に要した時間が100分短くなっているが，有意差(注2)が見られなかった．

予約問題の2種類の仕様変更に要した平均時間は，スライスを利用したG1では128.0分(A1:81 A2:151 A3:152)，スライスを利用しなかったG2では214.3分(B1:200 B2:205 B3:238)となっている．平均時間を見るとG1の方がG2よりも仕様変更に要した時間が約86分短くなっており，有意差も確認できた．

4. 考　　察

仕様変更の内容によってスライスの有効性に差が生じた事について考察する．一般にスライスは，文献[4]で分類されている4種類のフォールトのうち，文の欠如(missing statement)には有効ではないと言われている(我々は，これらの結果を[2]において実際に確認している)．本実験では仕様書変更要求として，機能変更と機能追加の2種類を用意している．機能変更とは元の仕様書に既にある機能を変更するもので，機能追加とは 元の仕様書への新たな機能の追加を実現するものである．機能変更においては，既に存在するコードから変更部分をスライスとして抽出できるが，機能追加においては，新たにコードを追加する必要があり，追加されるコードはスライス抽出時には存在しないため，変更部分認識時にはスライス抽出機能は有効ではないと考えられる．実際に各仕様変更ごとに分析すると，スライス抽出機能は予約問題においては機能の追加よりも機能の変更の方がより有効であることが確認できた．

(注2)：平均値の差の検定(ウェルチの検定)を有意水準5%で行なった[5].

5. む す び

本研究では，スライスが実際の保守作業に有効であるかどうかを実験的に評価した．実験の結果，スライスを用いた方が，スライスを用いない場合よりも効率よく保守作業が行えることが確認できた．本実験で用意したプログラムのサイズは300〜400行程度と小規模ではあるが，スライスを用いることで参照範囲を45%〜52%に限定できる(この数値は被験者が実験において抽出した典型的なスライスのサイズである)．大規模なプログラムにスライス抽出技法を適用する場合においても本実験と同様に参照範囲を限定することができ，保守作業を効率良く行えると考えられる．

謝辞 本研究は，一部文部省科学研究費補助金特定領域研究(A)(2)(課題番号: 10139223)の補助を受けている．

文　　献

[1] Basili, V. R., Caldiera, G. and Rombach, H. D., *Goal Question Metric Paradigm*, in John J. Marciniak, editor, Encyclopedia of Software Engineering, vol.1, John Wiley & Sons, pp.528-532, 1994.
[2] 西松, 楠本, 井上, "フォールト位置特定におけるプログラムスライスの実験的評価", 信学技報, SS98-3, May 1998.
[3] 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", 情処学論, Vol.37, No.4, pp.536-545, 1996.
[4] 下村, "変数値エラーにおける Critical Slice に基づくバグ究明戦略", 情処学論, Vol.33, No.4, pp.501–511, 1992.
[5] 芝, 渡部, 石塚 編: 統計用語辞典, 新曜社, 1984.
[6] Robson, D.J et al., "Approaches to Program Comprehension", J.System Software, Vol.14, No.1, 1991.
[7] Weiser, M., "Program Slicing", *IEEE Trans. on Soft. Eng.*, Vol.10, No.4, pp. 352-357, 1984.
[8] 山崎利治, "共通問題によるプログラム設計技法解説", 情処学誌, Vol.25, No.9, p.934, 1984.
[9] Zelkowitz, M. and Wallace, D. R., "Experimental models for validating technology", *IEEE Software*, Vol.31, No.5, pp.23-31, 1998.

（平成5年10月25日受付，8年9月17日再受付）