記憶力に基づくプログラム理解容易性評価尺度の追加実験

村上 優佳紗
近畿大学大学院
m.yukasa@gmail.com

角田 雅照
近畿大学大学院
tsunoda@info.kindai.ac.jp

中村 匡秀
神戸大学大学院
masa-n@cs.kobe-u.ac.jp

要旨

ソフトウェア開発の生産性を高めるためには，理解しやすいソースコードを書くことは非常に重要である．そのため，ソースコードの複雑度を評価する指標が，これまで数多く提案されてきた．例えばサイクロマチック数やCKメトリクスはソースコードの複雑度を評価する代表的な指標である．ただし，複雑度が低い場合でも，必ずしもプログラムが理解しやすいとは限らない．プログラムの理解のしやすさは，理解のために必要とする記憶量の多寡に基づくと仮定し，プログラム理解容易性評価尺度が提案されている．この尺度では，人間の短期記憶は，大きさの限られたキューになっていると仮定し，プログラムを読む時に，覚えるべき変数の個数がキューの大きさの上限を超える場合，理解が難しくなるとしてプログラムの理解容易性を評価している．ただし，指標を評価する実験では，学生を被験者としている．学生は年齢やプログラム経験が比較的均質であり，学生以外の，より幅広い年齢の被験者に適用した場合，必ずしも評価尺度がプログラムの理解容易性を表しているとは限らない．そこで本研究では，大学の教員を被験者実験とし，この評価尺度がプログラムの理解容易性を適切に表すかを確かめた．学生 24 人と大学教員 8 人を被験者として実験を行った結果，評価尺度がプログラムの理解容易性を適切に表していることがわかった．

1. はじめに

ソフトウェア開発の生産性を高めるためには，理解しやすいソースコードを書くことは非常に重要である．ソースコードが理解しにくい場合，ソフトウェアの保守性が低下しやすく，その結果，ソフトウェア開発の生産性が低下したり，ソフトウェアの品質が低下したりする可能性がある．そのため，ソースコードの複雑度を評価する指標が，これまで数多く提案されてきた．例えばサイクロマチック数[4]やCK メトリクス[1]はソースコードの複雑度を評価する代表的な指標である．

ただし，複雑度が低い場合でも，必ずしもプログラムが理解しやすいとは限らない．文献[3][5]ではプログラムの理解のしやすさは，理解のために必要とする記憶量の多寡に基づくと仮定し，プログラム理解容易性評価尺度を提案している．この尺度では，人間の短期記憶は，大きさの限られたキューになっていると仮定し，プログラムを読む時に，覚えるべき変数の個数がキューの大きさの上限を超える場合，理解が難しくなるとしてプログラムの理解容易性を評価している．

文献[3][5]では，プログラム理解容易性評価尺度の妥当性を確かめるために，学生を被験者として実験を行っている．ただし，学生は年齢やプログラム経験が比較的均質であり，学生以外の，より幅広い年齢の被験者に適用した場合，必ずしもこの評価尺度がプログラムの理解容易性を表しているとは限らない．そこで本研究では，大学の教員を被験者実験とし，この評価尺度がプログラムの理解容易性を適切に表すかを確かめる．

2. プログラム理解容易性評価尺度

プログラムの理解容易性を評価するために，文献[3][5]では，人間の記憶力に基づく尺度を提案している．これらの尺度では，プログラムを理解するためにメンタルシミュレーション[2]が行われることを前提としている．メンタルシミュレーションとは，プログラムの動作をコンピュータや筆記具を用いずに，思考しながら理解することであり，比較的規模の小さなコード片に対して用いられる．メンタルシミュレーションを行う場合，変数の値を記憶しておく必要があるが，多数の変数の値を同時に記憶しておくことは容易ではない．このため文献[3][5]では，多数の変数の値を記憶しておく必要のあるプログラムの場合，メンタルシミュレーションのコストが高くなり，プログラムの理解容易性が低下すると仮定している．

文献[5]では，人間の短期記憶を FIFO キューとして，メンタルシミュレーションの仮想モデル（Virtual Mental Simulation Model; VMSM）を作成し，そのモデルに基づいて理解容易性評価尺度を定義している．基本的なアイディアは，ある変数の値を参照する際，（大きさに制限のある）キューに変数の値が記憶されている場合はコストが小さくなるとしている．逆にキューに値が記憶されていない場合，その値の変更箇所にまでバックトラックする必要が生じるため，コストが大きくなるとしている．

文献[5]では，VMSM に基づき，以下の 4 つの評価尺度を定義している．

- ASSIGN: 変数代入に関するコスト．
- RCL: 短期記憶内の変数を思い出すコスト．
- BT_CONST: 定数をバックトラックする回数．短期記憶にない定数を得るコスト．
- BT_VAR: 変数のバックトラックの距離．短期記憶にない変数を得るコスト．

文献[3]では，上記のメトリクスが変数の更新回数に基づく再計算コストを考慮していないことと，バックトラックに関するメトリクスがプログラムの行の入れ替えに敏感すぎることが問題点であるとし，新たな評価尺度を 2 つ提案している．具体的には，各変数の値の更新回数をベクトルの要素とし，ベクトルの要素の和に基づくメトリクス SUM_UPD とベクトルの要素の分散に基づくメトリクス SUM_VAR を定義している．

3. 実験

3.1. 概要

実験の目的は，学生以外（今回の実験では大学教員）の場合でも，プログラム理解容易性評価尺度が妥当に機能するかどうかを確かめることである．そのために，必要とする記憶力が異なる，複数のプログラムを用意し，被験者がコードを理解するために掛けた時間を計測した．

必要とする記憶量の異なるプログラムは，石黒らの研究[3]で示されているもの 4 つ（a0, a1, b0, b1．図 1，図 2 参照）を利用した．各プログラムは 20 行から 30 行の規模である．あらかじめ指定された変数の値が，プログラム実行後にどうなるかプログラムを読んで答え，それが正しかった場合，プログラムを理解できたとした．例えばプログラム a0 の場合，プログラム実行後の変数 i の値を答えさせた．プログラムの理解はメンタルシミュレーションにより行うこととし，メモなどは利用させないようにした．

各プログラムで理解のために必要とする記憶力の多寡

```
int i , t;

t = 11;
t = t - 1;
i = 2;
if(i < t){
        i = i + 2;
        if(i < t){
                i = i + 2;
        }
        if(i < t){
                i = i + 2;
                if(i < t){
                        i = i + 2;
                }
        }
        if(i < t){
                i = i + 2;
        }
}
System.out.println("i = " +i);
```

(a0)

```
int i , t;

t = 11;
t = t - 1;
i = 2;
if(i < t){
        t = t - 2;
        if(i < t){
                i = i + 2;
        }
        if(i < t){
                t = t - 2;
                if(i < t){
                        i = i + 2;
                }
        }
        if(i < t){
                i = i + 2;
        }
}
System.out.println("i = " +i);
```

(a1)

図 1 プログラム a0, a1[3]

```
int a, b, c, d, e, f, g;

a = 2;
b = 4;
c = 3;
d = 6;

c = c + 4;
d = d - 2;
if(c < 5)
        e = d + 5;
else
        e = d + 3;
a = a * 2;
b = b + 6;
if(a > 7)
        f = b - 3;
else
        f = b - 5;
g = e + f;

System.out.println("g = "+g);
```

(b0)

```
int a, b, c, d, e, f, g;

a = 2;
b = 4;
c = 3;
d = 6;

c = c + 4;
d = d - 2;
a = a * 2;
b = b + 6;
if(a > 7)
        f = b - 3;
else
        f = b - 5;
if(c < 5)
        e = d + 5;
else
        e = d + 3;
g = e + f;

System.out.println("g = "+g);
```

(b1)

図 2 プログラム b0, b1[3]

については，2 章で説明した 6 つのメトリクス（ASSIGN, RCL, BT_CONST, BT_VAR, SUM_UPD, VAR_UPD）に基づき評価した．これらのメトリクスに基づく各プログラムの理解容易性を表 1に示す（石黒らの研究[3]からの引用）．ASSIGN, BT_CONST, SUM_UPD より，プログラム b0, b1 を理解するためには，比較的記憶力が必要とされることがわかる．

被験者を学生グループと教員グループに分け，それぞれのグループの解答時間の平均値や中央値などを算出

表 1　各プログラムの理解容易性[3]

プログラム	ASSIGN	RCL	BT_CONST	BT_VAR	SUM_UPD	VAR_UPD
a0	12	6	0	80	7	1.25
a1	12	6	0	48	7	0.25
b0	18	8	1	30	11	0.24
b1	18	6	1	54	11	0.24

し比較した. 学生グループは, 近畿大学理工学部情報学科に所属する学部 4 年生から修士 2 年生の 24 名であり, 教員グループは同学科の教員 8 名である. 教員グループの平均年齢は 45 歳（最小 33 歳から最大 64 歳）である.

プログラムを読む順番が実験結果に影響することを避けるために, 4 つのプログラムを読む順番を被験者ごとに変更した. 例えばある被験者ではプログラムを a0, b1, a1, b0 の順に読むとし, 別の被験者では b1, a0, b0, a1 の順に読むなどとした.

3.2. 実験ツール

実験のために, 問題を出題し, 回答時間や誤回答の回数を計測するためのツールを作成した. 図 3 にツールのスクリーンショットを示す. 本ツールは表計算ソフトのマクロを用いて開発した. ツールの動作を以下に示す.

1. 「解答する」をクリックすると問題とテキストボックスを表示する.
2. 問題に正解するまでテキストボックスを表示し続ける.
3. テキストボックス（問題）が表示されてから正答するまでの回答時間と誤回答の回数を記録し, 被験者にも表示する.
4. 正解すると次の問題に自動的に遷移する.

3.3. 結果

表 2 に各グループの回答時間の平均値と中央値を示す. 教員グループ, 学生グループとも, 記憶力を比較的必要としないプログラム a0, a1 と比べ, プログラム b0,b1 のほうが, 解答時間が長くなっていた.

比較的記憶力を必要としないプログラム a0 と比べ, 何倍の回答時間が掛かっているかを調べた. 具体的には, プログラム a0 以外の回答時間÷プログラム a0 の回答時間を求めた. この値が大きいほど, a0 と比較して回答時間が多く掛かっていることを示している. 結果を表 3 に示す. 学生グループでは, b0 / a0, b1 / a0 の平均値と中央値のうち, b0 / a0 の平均値以外は 2 を下回っていたが, 教員グループでは, それらの値全てが 2 を上回ってい

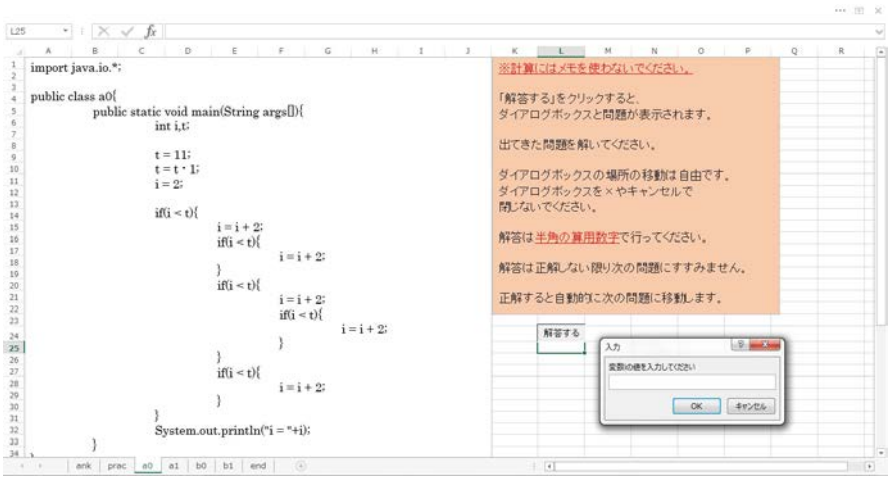

図 3　実験用ツールのスクリーンショット

表 2　各グループの回答時間（秒）

		a0	a1	b0	b1
学生	平均値	65	82	105	89
	中央値	58	75	100	70
教員	平均値	59	91	141	115
	中央値	51	51	144	109

表 3　プログラム a0 との回答時間の比

		a1 / a0	b0 / a0	b1 / a0
学生	平均値	1.6	2.0	1.4
	中央値	1.2	1.5	1.4
教員	平均値	1.5	2.4	2.2
	中央値	1.0	2.4	2.4

表 4　誤回答数の基本統計量

		a0	a1	b0	b1
学生	平均値	1.0	0.5	1.3	0.8
	中央値	0.0	0.0	0.0	0.0
	標準偏差	2.5	0.8	2.2	2.3
教員	平均値	0.1	0.4	1.1	0.5
	中央値	0.0	0.0	1.0	0.0
	標準偏差	0.4	0.7	1.0	0.8

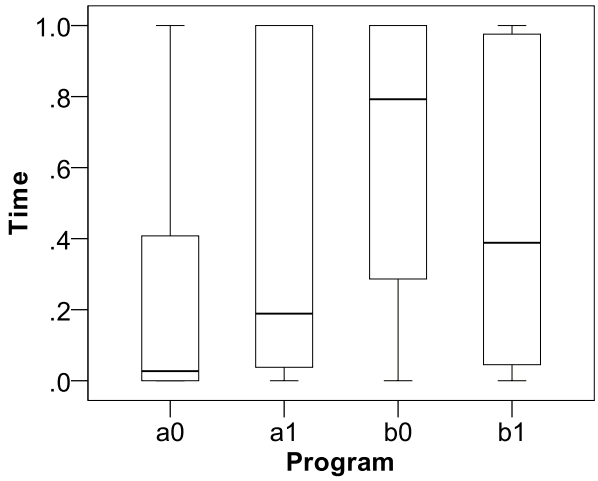

図 4　学生グループの回答時間正規化

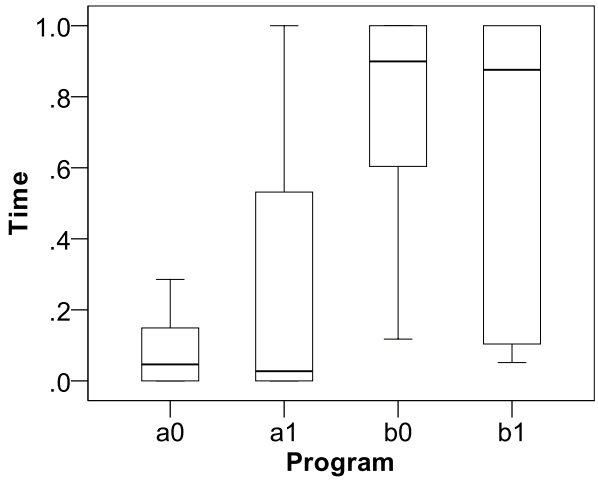

図 5　教員グループの回答時間正規化

た.

被験者ごとに解答時間を正規化して比較した. 正規化は(回答時間 – 最小解答時間)÷(最大回答時間 – 最小回答時間)により行った. 正規化を行うことで被験者個人の問題の得手不得手を判別することができるためである. 箱ひげ図を図 4, 図 5 に示す. 学生グループの場合, プログラム a0 以外はばらつきが大きい, すなわち記憶力を比較的多く必要とするプログラム b0, b1 の場合でも, 相対的に回答時間が長くなるとは限らないといえる. 教員グループの場合, 特にプログラム b0 において相対的に回答時間が長い傾向があった. これらの結果より, 教員グループのほうが, プログラム理解容易性評価尺度がより明確に機能しているといえる.

誤回答数の基本統計量を表 4 に示す. プログラム b0 を除き中央値は 0 であり, どちらのグループも誤回答を多く行った被験者は少ないことがわかる. プログラム b0 については, 教員グループの中央値が 1 となったことから, このプログラムは教員グループにとって, 特に理解が難しかったと考えられる. なお, 学生グループでは, 一部の被験者の誤回答数が多く, プログラム a1 を除き, 標準偏差が 2 を超えていた. ただし, 教員グループと比較して, 平均値と中央値に大きな差はないため, 全体としては正答率に教員グループと差はないといえる.

表 5　アンケート結果

	問 I			問 II		
	1	2	3	1	2	3
学生	43%	43%	14%	33%	50%	17%
教員	57%	29%	14%	46%	46%	8%

3.4. 考察

教員グループの被験者が，実験に集中して取り組んでいなかった可能性は低いと考えられる．これは，プログラム a0 の出題順序が被験者によって異なっていたにも関わらず，a0 の回答時間が学生グループよりも低かったためである．

実験後のアンケートで出題したプログラムについてアンケートを行った．その内容は以下の 2 つである．

I. 1 問目と 3 問目について，具体的にどの部分がどのように変更されたかについて気づいたか．
II. 2 問目と 4 問目について，具体的にどの部分がどのように変更されたかについて気づいたか．

回答は以下の 3 つとした．

1. 気づいたうえで早く読めたと思う
2. 気づいたが早く読むのに役立たなかった
3. 気づかなかった

学生グループの有効解答数は 24，教員グループの有効解答数は 7 であった．アンケート結果を表 5 に示す．学生グループと比較して，教員グループは問 I，問 II とは「1:気づいたうえで早く読めたと思う」と解答した被験者が多かった．図 5 において，教員グループのプログラム a1 の回答時間のばらつきが比較的小さかったのは，このことが影響している可能性がある．記憶力を必要とするプログラム b0，b1 について，もし回答 2，3 が多かった場合，回答時間がより長くなった可能性がある．

4. おわりに

本研究では，記憶力に基づくプログラム理解容易性評価尺度[3][5]に着目し，指標の妥当性を評価するために，被験者の対象を変えて実験を行った．この尺度では，人間の短期記憶は，大きさの限られたキューになっていると仮定し，プログラムを読む時に，覚えるべき変数の個数がキューの大きさの上限を超える場合，理解が難しくなるとしてプログラムの理解容易性を評価している．文献[3][5]の実験では，評価尺度の妥当性を確かめるために，学生を被験者として実験を行っている．

ただし，学生は年齢やプログラム経験が比較的均質であり，学生以外の，より幅広い年齢の被験者に適用した場合，必ずしもこの評価尺度がプログラムの理解容易性を表しているとは限らない．そこで本研究では，大学の教員を被験者実験とし，この評価尺度がプログラムの理解容易性を適切に表すかを確かめた．その結果，学生の被験者よりもより明確に，プログラム理解容易性評価尺度が適切に機能した．

謝辞　本研究の一部は，文部科学省科学研究補助費（挑戦的萌芽：課題番号 26540029，基盤 C：課題番号 25330090）による助成を受けた．

参考文献

[1] Chidamber, S. and Kemerer, C: A metrics suite for object oriented design, IEEE Transactions on Software Engineering, vol.20, no.6, pp.476-493, 1994.
[2] Dunsmore, A., Roper, M. and Wood, M.: The role of comprehension in software inspection, Journal of Systems and Software, vol.52, no.2–3, pp.121-129, 2000.
[3] 石黒誉久，井垣宏，中村匡秀，門田暁人，松本健一：変数更新の回数と分散に基づくプログラムのメンタルシミュレーションコスト評価，電子情報通信学会技術報告，SS2004-32，pp.37-42, 2004.
[4] McCabe, T: A Complexity Measure, IEEE Transactions on Software Engineering, vol.SE-2, no.4, pp.308-320, 1976.
[5] Nakamura, M., Monden, A., Satoh, H., Itoh, T., Matsumoto, K., and Kanzaki, Y.: Queue-based Cost

Evaluation of Mental Simulation Process in Program Comprehension, Proc. of International Software Metrics Symposium, pp.351-360 (2003).