社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS2004-32 (2004-11)

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

石黒 誉久† 井垣 宏† 中村 匡秀† 門田 暁人† 松本 健一†

†奈良先端科学技術大学院大学 情報科学研究科 〒630-0192 奈良県生駒市高山町 8916-5

E-mail: †{taka-is, hiro-iga, masa-n, akito-m, matumoto}@is.naist.jp

あらまし 理解しやすいプログラムを作成することは，プログラムの信頼性や保守性を向上させるうえで重要である．本稿では，プログラム理解性を評価するための一つの側面として，プログラムのメンタルシミュレーション に着目する．以前の研究でメンタルシミュレーションの仮想モデル(VMSM)を開発し，バックトラック距離が，メンタルシミュレーションのコストに大きく影響することが示された．しかし，後の実験でバックトラック距離だけではうまくコストが表せないような問題が存在する．本稿では，プログラム中の変数の更新がメンタルシミュレーションコストに大きく影響していることを考え，変数の更新頻度の総和と変数の更新頻度の分散による新たな動的メトリクスを提案する．実験の結果，提案メトリクスは，従来の VMSM で説明できなかったいくつかの問題を説明することができた．

キーワード プログラム理解性，メンタルシミュレーション，VMSM

Evaluating the Cost of Program Mental Simulation Based on the Number and Variance of Variable Updates

Takahisa ISHIGURO† Hiroshi IGAKI† Masahide NAKAMURA† Akito MONDEN†
Ken-ichi MATSUMOTO†

† Graduate School of Information Science, Nara Institute of Science and Technology

8916-5 Takayama-tyou, Ikoma-city, Nara, 630-0192 Japan

E-mail: †{taka-is, hiro-iga, masa-n, akito-m, matumoto}@is.naist.jp

Abstract This paper presents a method to estimate the cost of mental simulation of programs. In our previous research, we developed a model called virtual mental simulation model (VMSM). It showed that a distance of simulation backtrack significantly influences the cost of mental simulation. Through the subsequent experiments, however, we have found some cases that are not well justified by the distance only. In this paper, we show that updates of variables in a program also have significant correlation to the mental simulation. Then, we propose two new dynamic metrics: Sum in Number of Variable Updates and Variance in Number of Variable Updates. The experimental evaluation shows that the proposed metrics well explain the cases the previous VMSM cannot cover.

Keyword program comprehension, mental simulation, VMSM

1. はじめに

理解しやすいプログラムを作成することは，プログラムの信頼性や保守性を向上させるうえで重要である[8]．しかし，人間がプログラムを理解するという行為には，様々なヒューマンファクタが複雑に寄与する．従って，与えられたプログラムが理解しやすいかどうか(プログラム理解性と呼ぶ)を，単一の基準で定量的に評価することは一般的に困難である．

プログラム理解性を評価するための一つの側面として，本稿ではプログラムのメンタルシミュレーション [5] に着目する．メンタルシミュレーションとは，プログラムのソースコードPと入力Iに対して，人間が頭の中で計算機をシミュレートし，P の実行を追跡(トレース)する行為である．メンタルシミュレーションは，プログラムがどのように動作するのかを理解するための最もシンプルかつ効果的な手法である．デバッグや保守など様々なソフトウェア開発プロセスにおいて，比較的小さいコード断片に対して，頻繁に用いられる．

本研究では，プログラムがトレースしにくい，すなわちメンタルシミュレーションに高いコストがかかる時，そのプログラムが理解しにくいことの一要因になるのではないかと考える(勿論，それが理解性の全てを評価するものではない)．そして，メンタルシミュレー

ションのコストを定量的に見積もるための新たなソフトウェアメトリクスを提案する．

人間がメンタルシミュレーションを実行する際，変数の値を覚えたり，思い出したり，忘れたり，覚えなおしたりするために，短期記憶をあたかも計算機のレジスタやメモリのように用いる．計算機のメモリと異なり，人間の短期記憶は小容量，高揮発であるため，短期記憶への情報の出し入れのしやすさがメンタルシミュレーションのコストに大きな影響を与えると考えられる．

従来，多数のソフトウェアメトリクスが提案されている．まず，ソフトウェアの複雑さメトリクス[1][3]のうち，人間の記憶を意識したものに，生存変数の数(Number of Live Variables)[1] や Span[3]がある．しかし，これらはソースコード上の静的な性質を計測するもので，メンタルシミュレーションという動的なコード実行の評価には適さない．

この問題に対処するため，我々は以前の研究でメンタルシミュレーションの仮想モデル(VMSM)を開発し，4 つの動的メトリクスを提案した[7]．VMSM では，FIFO キューと変数の代入／参照の系列(Assignment and Reference Trace, AR-trace と呼ぶ)を用いて，メンタルシミュレーション中の短期記憶[4]の働きをモデル化する．人間は，変数の値を忘れた際に，その変数が最後に更新された場所までプログラムをバックトラックする．そのバックトラック距離が，メンタルシミュレーションのコストに大きく影響することが VMSM メトリクスによって示された．

しかし，その後の実験で，いくつかのプログラムにおいて，従来の VMSM でうまく説明できないケースが存在することがわかった．分析の結果，VMSM において以下の問題点があることがわかった．

(a) 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．

(b) プログラムの行の入替に敏感すぎる．

そこで，本研究では次の 2 種類の動的メトリクスを提案する：変数の更新頻度の総和(Sum in Number of Variable Updates, 以下 SUM_UPD)，変数の更新頻度の分散(Variance in Number of Variable Updates, 以下 VAR-UPD)．SUM_UPD および VAR-UPD はそれぞれ上記の問題点(a)(b)を解決するための新たな動的メトリクスである．実験の結果，提案する新たなメトリクスは，従来の VMSM で説明できなかったいくつかのケースを説明することができた．

2. 準備

2.1. メンタルシミュレーション

メンタルシミュレーションは，プログラムを頭の中でシミュレートしながら実行することである．例として以下のプログラム(断片)を考える：

$$a = b + 1;$$

この行をメンタルシミュレーションする際，頭の中で以下のステップを実行する．

Step1: b の値を思い出す

Step2: b の値と 1 を加算する

Step3: その結果を新たな a の値として覚える

この時，Step1 で変数 b の値を覚えているかどうかで，シミュレーションにかかるコストが大幅に変わってくる．もし b の値を覚えていたらシミュレーションは比較的簡単に進む．しかし b の値を忘れていたら，その値を何らかの手段で求める必要があり，シミュレーションは困難になると考えられる．メンタルシミュレーションを行う際，人間は覚える，思い出す，忘れる，覚えなおす，などの認知活動が行われる．そしてこれには人間の短期記憶が大きく関連する．

メンタルシミュレーション中，短期記憶に最も作用するのが，変数の参照および代入である．ここでは，プログラムを与えられた入力を基に実行した際に得られる，変数の代入／参照の実行系列(トレース[2])を AR-Trace と呼び，メンタルシミュレーションの解析に用いる．AR-trace の例として，図 1 にプログラム(a)と，そのトレース情報である AR-trace(b)を示す．

```
i = 1;
j = 2;
A[1] = i + 4;

j = A[i] - j;
```

(a)

i	1	代入
j	2	代入
i	1	参照
A[1]	5	代入
i	1	参照
A[1]	5	参照
j	2	参照
j	3	代入

(b)

図 1　AR-trace の例

2.2. 仮想メンタルシミュレーションモデル (VMSM)[7]

我々は以前の研究で，メンタルシミュレーションの仮想モデル(Virtual Mental Simulation Model, VMSM)を開発した．VMSM では，人間の短期記憶を FIFO キューであらわし，メンタルシミュレーションを以下のように仮想的にモデル化する．

VMSM 手順:

1. プログラム P と入力 I から AR-trace を得る．
2. AR-trace を前から順番にたどっていく．AR-trace の各行について，

2.1. 変数 x の参照に達したならば，x の値が FIFO キ

ューに存在するか調べる．

2.1.1. もし存在すれば，それを思い出し(recall)，その変数をキューの最後尾に入れる．

2.1.2. もし存在しなければ，シミュレーションを最近の代入(または参照)までバックトラック(backtrack)し，値を得る．最後に，その値をキューの最後尾に入れる．

2.2. 変数 y の代入に達したなら，左辺へ代入(assign)を行う．右辺値をキューの最後尾に入れる．

これに基づいて，メンタルシミュレーションのコストを見積もるための，以下の 4 種類のメトリクスを導出する．

ASSIGN: assign が行われた回数(変数代入にかかるコスト)

RCL: recall が行われた回数(短期記憶内の変数を思い出すコスト)

BT_CONST: 定数を backtrack する回数(短期記憶にない定数を得るコスト)

BT_VAR: 変数の backtrack 距離．AR-trace 内で，現在の行から最新の代入(または参照)までの行数(短期記憶にない変数を得るコスト)．

実験の結果，BT_VAR がメンタルシミュレーションのコストに大きな相関を持つことが示されている．より詳細は[7]を参照されたい．

2.3. BT_VAR の問題点

前節までで述べたように，BT_VAR は，忘れた変数 x を思い出すためのコストを，単純に AR-trace 上のバックトラック距離で定義している．そのため，その後の実験で説明が困難な以下のような問題点(a)(b)が存在することが分かった．

(a) 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．

直感的に，変数 x の更新回数が多いほど，x の値の再計算コストは大きくなると考えられる．しかし，変数 x の更新回数が多いほど，AR-trace 上における最新代入(更新)までの距離が近くなるため，BT_VAR によるメンタルシミュレーションコストは小さくなってしまう．さらに，単一(あるいは少数)の変数の更新回数が飛び抜けて多いプログラムと頻繁に更新される変数の数がより多いプログラムとでは，後者の方がメンタルシミュレーション中の再計算がより困難になることが考えられるが，VMSM ではこれらのプログラムの間にシミュレーションコストの差は生じない．

図 2 にプログラム a0, a1 を示している．プログラム a0 では，変数 i が頻繁に更新され，変数 t はあまり更新されない．従って，メンタルシミュレーション中に t の値を忘れた場合の再計算コストは i と比較すると低くなることが考えられる．また，プログラム a1 では i と t 両方の変数が頻繁に更新される．そのために，a1 のメンタルシミュレーションコストは，t が固定的である a0 と比べて高くなると想像できる．

```
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 2 プログラム a0, a1

(b) プログラムの行の入替に敏感すぎる．

同じプログラムの一部のブロックを入れ替えるだけでも AR-trace 上のバックトラック距離が大きく変動し，BT_VAR で表されるコストに敏感に反映されてしまう．

例えば，図 3 のプログラム b1 は b0 のあるブロックを単純に移動したものである．これにより b1 では，変数 a における代入から参照までの距離が変わるため，VMSM の BT_VAR 値，すなわちメンタルシミュレーションにおけるコストが大きく変化する．しかしながら，b0 から b1 にプログラムを変化させたとしても，実際のシミュレーションコストが劇的に変化するとは考えられない．

```
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 3 プログラム b0, b1

以降では，変数の更新回数に基づく再計算コストを考慮したメトリクスを提案し，a0, a1, b0, b1 のプログラムに適用する．その後，被験者によるメンタルシミュレーションをそれらのプログラムに対して行って貰い，VMSM のシミュレーションコストと比較する．

3. 提案メトリクス

前節の問題点(a)(b)を解決するため，メンタルシミュレーションのコスト評価のための新たなメトリクスを提案する．

3.1. キーアイデア

いくつかの予備実験を通して，メンタルシミュレーション中にある変数 x の値を忘れた場合，x の再計算コストと x の更新頻度が大きく関わることが観察された．具体的には，

観測 1： 変数 x が頻繁に更新されるほど，x の再計算が難しくなる．

観測 2： 変数 x の他にも，頻繁に更新される変数が多く存在すればするほど，x の再計算が難しくなる．

例として，同じ仕様を満たし，同じ変数が使われるが，変数の更新頻度に差のあるプログラム x0, x1 を考え，変数 m, n の更新回数をそれぞれ二次元整数ベクトルで書くと，以下のようになるものとする．

x0 における m, n の更新回数 = (7, 1)

x1 における m, n の更新回数 = (4, 4)

すると，観測 1 は，x0 のメンタルシミュレーションにおいては，m の値を再計算するほうが n の再計算よりも難しいことを説明できる．また，変数更新のトータルな回数は，x0 と x1 ともに 8 である．しかし，各変数の更新回数に着目すると，x0 では m のみが頻繁に更新されるので，比較的楽にシミュレーションできるのに対し，x1 では m も n もおしなべて更新されるので難しい．即ち，変数の更新回数の分散が低いほど，多くの変数の動的な変化を把握する必要があるため，シミュレーションが難しくなる(観測 2 による)．

以上のことから，「全ての変数の更新回数の総数」および「変数の更新回数の分散」の二つを，新たなメトリクスとして提案することにする．

3.2. メトリクスの定義

プログラム P に対して，P の全ての変数の集合を

$$V = \{v_1, v_2, \cdots, v_n\}$$

とする．また，

$$upd(v) \quad (v \in V)$$

を P を入力 I に関して実行した時に，v が更新される回数とする（即ち，P の AR-trace 内で v への代入として出現した数）．この時，

$$UPD(p) = [upd(v_1), upd(v_2), \cdots, upd(v_n)]$$

を p の変数更新ベクトルと呼ぶ．

プログラム p に対する新たなメトリクス(SUM_UPD, および VAR_UPD)を以下のように定義する．

$$SUM\_UPD(p) = \sum upd(v_i)$$

(p の更新ベクトルの要素の和)

$$VAR\_UPD(p) = Var(upd(v_1), \cdots, upd(v_n))$$

(p の更新ベクトルの要素の分散)

この時，SUM_UPD(p)，VAR_UPD(p)をそれぞれ p の更新総和メトリクス，更新分散メトリクスと呼ぶ．

今，プログラム p と q を考え，それぞれの更新総和メトリクス，更新分散メトリクスが得られたとする．3.1 節の観測 1, 2 から，以下のことを説明できる．

SUM_UPD(p)<SUM_UPD(q)ならば，q の方がメンタルシミュレーションのコストが高くなる可能性がある．

VAR_UPD(p)<VAR_UPD(q)ならば，q の方がメンタルシミュレーションのコストが低くなる可能性がある．

さらに，これらのメトリクスのどちらかが同じである場合，プログラムのメンタルシミュレーションコストは同じでない方のメトリクスに依存する．

SUM_UPD(p) = SUM_UPD(q) であるようなプログラム p, q において，VAR_UPD(p) < VAR_UPD(q)ならば，q の方がメンタルシミュレーションのコストが低くなると考えられる．

VAR_UPD(p) = VAR_UPD(p)であるようなプログラム p, q において，SUM_UPD(p) < SUM_UPD(q)ならば，q のほうがメンタルシミュレーションのコストが高くなると考えられる．

この 2 つのメトリクスにより，3.1 節の例題 a0，a1 を説明することができる．

4. 評価実験

2 章で上げた BT_VAR の問題点，3 章で提案したメトリクスについての評価実験を行った．

4.1. 実験設定

与えられたプログラムについての理解を 22 名の被験者にのべ 44 回試みてもらった．22 名とも本学の情報科学研究科の学生である．実際に使用したプログラムは 2.3 節で述べた以下の 4 つで，このプログラムは全て Java 言語によって記述されている．

表 1　実験結果

Program	VMSM				提案メトリクス		実験結果		
	ASSIGN	RCL	BT_CONST	BT_VAR	SUM_UPD	VAR_UPD	Miss	理解コスト	有意差
a0	12	6	0	80	7	1.25	0.09	1.52	2.5%
a1	12	6	0	48	7	0.25	0.27	2.25	
b0	18	8	1	30	11	0.24	0.27	2.63	25%
b1	18	6	1	54	11	0.24	0.09	2.13	

a0, a1:主に更新分散メトリクスを評価する目的で作成した．a0, a1 では，変数の数，変数の代入回数(すなわち更新総和メトリクス)，AR-trace の長さは同じで，変数の更新回数の分散(すなわち更新分散メトリクス)の差と被験者による理解速度の差を比較する(図 2)．

b0, b1:b0 のプログラムの一部のブロックをプログラムの仕様と AR-trace の長さを変化させないように移動し，b1 を作成した．BT_VAR によるメンタルシミュレーションコストの差と実際の被験者による理解に要する時間の差を比較する(図 3)．

実験では，被験者 1 人につき a0, a1 のどちらか 1 つ，b0, b1 のどちらか 1 つの計 2 問を均等に全ての被験者に割り当てた．メンタルシミュレーションに対する慣れが実験結果に影響を及ぼさないように，実験のプログラムにとりかかる前に練習問題を全員に解いて貰った．さらに，a, b の順番でプログラムにとりかかる被験者と b, a の順番でとりかかる被験者が同数になるように配慮した．各プログラムにおいて，被験者に行って貰った作業は以下の通りである．

1. プログラムの実行を頭の中でたどる．
2. 1.を実行する際にメモを取らない．
3. プログラムが出力する値を解答として入力する．
4. 解答した値が正解と一致するまで繰り返す．

実験では，プログラム毎に被験者が間違えた回数と理解に要した時間の 2 つのデータを取得した．正確な時間を測定するため，Macromedia Flash を用いて実験ツールを作成した(図 4)．

4.2. 実験結果

実験結果では，被験者が理解に要した時間を集計する際に，各被験者が練習問題を理解するのに要した時間を利用して正規化を行った．これは，メンタルシミュレーションを行う際の各被験者の経験やスキル等の個人差による偏りを減らすためである．表 1 に，被験者の間違い回数，正規化を行って算出した理解コストに加えて，プログラム a0, a1 ならびに b0, b1 の間の理解コストにおける優位差を有意水準 5%の t 検定にかけ，その結果を記述した．

まず，a0, a1 の間では，t 検定により，理解コストは a1 のほうが大きくなる結果となった．この結果も 3.1 節で述べた観測 2 に一致する．つまり，変数の数と更新総和メトリクスの値が等しければ，更新分散メトリクスが低い方がメンタルシミュレーションのコストは増大するという実験結果となった．一方，VMSM メトリクスでは，変数の更新回数ではなく，距離によってコストを評価しているため，a0 のほうが逆にシミュレーションコストが大きくなるという値を示す結果となった．

さらに，このプログラムでは，全ての被験者に対し，自分が理解を進めたプログラムとは違う方のプログラムを実験終了後に見せ(a0 を解いた被験者には a1 を，a1 を解いた被験者には a0 を見せた)，どちらの問題が難しいと感じるかとその理由をアンケートとして質問した．その結果，22 人のうち有効回答が 16 名で，有効回答 16 名のうちの 12 名が a1 のほうが難しいと答え，残りの 4 名がどちらも同じではないかと答えた．また，a0 のほうが難しいと答えた被験者はいなかった．a1 のほうが難しいと答えた理由は，a0 では片方の変数の値がほとんど変わらないのに対し，a1 では i も t も変化するために，変数の値を記憶することが困難であるから，というものがほとんどだった．

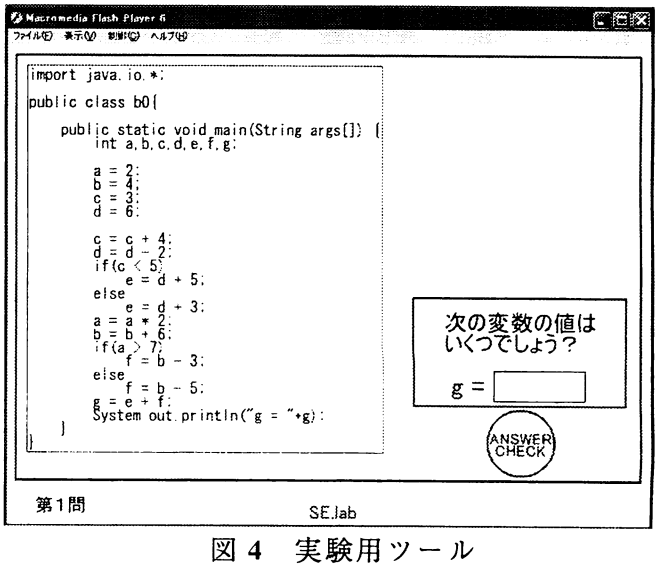

図 4　実験用ツール

次に，プログラム b0, b1 の間には，プログラムの理解コストの差が無いことが t 検定の結果から分かった．これは 2.3 節で述べた直感的な観測に一致する．すなわち，今回のプログラムのようなあるプログラムブロックの位置が変わることなどによる距離の変化は，必ずしもそのプログラムのメンタルシミュレーションコストを左右しないという直感に沿う結果となった．

4.3. 実験考察

a0 と a1 の実験結果の差は，変数値のリハーサル（rehearsal：繰り返し覚える，もしくは，思い出すことによる忘却の回避）[6]が影響していると思われる．プログラム a0 では，変数 i は頻繁に更新されるが，変数 t は一度更新された後，参照のみが繰り返し行われる．そのため，変数 i ではリハーサルが起こり，記憶にとどまりやすくなると考えられる．一方，プログラム a1 では，変数 i と t の両方が頻繁に更新されるため，リハーサルの働く余地が a0 と比べて少なくなる．結果として変数 i, 変数 t, それぞれの値を記憶しておくことが a0 と比べて困難になり，シミュレーションコストの増加をまねいたと考えられる．リハーサルの起こりやすさは，更新分散メトリクスにより計測できると考えられる．プログラム全体での変数の更新回数の総和が一定であるとすると，更新分散メトリクスが大きくなるほど，リハーサルが起こりにくくなる変数が増えるためである．

b0 と b1 では，変数のリハーサルの起こりやすさはほぼ同等であったと考えられる．そのことは，本稿で提案した更新総和メトリクスと更新分散メトリクスによって評価できた．また，このプログラムでは，被験者の大半(22 人中 20 人)がプログラムの制御構造や変数の位置を最初に理解してから値の計算を行っていた．このような手法でメンタルシミュレーションを行う場合，変数の再計算時にその変数の最新代入(更新)位置を被験者は既に知っているため，バックトラック距離(BT_VAR)はシミュレーションコストとはなりにくいと考えられる．また，このプログラムでは，被験者の変数記憶パターンとしては以下の 3 つが存在した．

(A) 変数の値をほとんど記憶せず，対象の変数の値が必要となる段階で計算をした被験者(22 人中 14 人)

(B) 必要な変数とその値のみを順番に記憶していき，メンタルシミュレーションを行った被験者(22 人中 6 人)

(C) 最初から順番にできる限り多くの変数を覚えていき，忘れた時点で再計算を行うという過程を繰り返した被験者(22 人中 2 人)が存在した．

(A)のタイプでは，変数の値をほとんど記憶しないため，メンタルシミュレーションのコストは再計算を行う際の変数の更新回数と計算を同時に行う必要のある変数の数(VAR_UPD)に依存すると考えられる．(B)のタイプでは，変数の記憶のしやすさは上記の a0, a1 の考察でも述べたように，VAR_UPD に影響を受ける．(C)は効率が悪くなったタイプ(B)と考えることができるために，(C)のタイプの変数記憶パターンにおいても VAR_UPD が大きな意味を持つと言える．

5. おわりに

本研究では，先行研究である VMSM によるコスト評価の問題点をあげ，それを解決するために新たなメトリクスを提案した．このメトリクスは変数の更新頻度に注目したものであり，実験によりその有効性を確かめた．

今回の実験では，メンタルシミュレーションの際に制御構造を最初に確認し，必要な変数のみを選別して記憶や再計算を行う被験者が多いことが分かった．また，変数の参照や更新を行う過程で，リハーサルという現象が変数の記憶のしやすさに影響を与えている可能性があることが判明した．今後は，リハーサルや変数の記憶パターン等を重視したメンタルシミュレーションにおけるプログラムのコスト評価モデルについての実験と観測を進めていくことを考えている．

文　献

[1] Aho, Alfred V, Sethi, Ravi, and Ullman, Jeffrey D. Compilers: Principles, Techniques, and Tools, Addison-Wesley Publich Company, 1986.

[2] Ball, T. and Larus, J. R, "Optimally Profiling and Tracing Programs," ACM Transactions on Programming Languages and Systems, val.16, No.3, pp.1319-1360, July, 1994.

[3] Conte, S. D. Dunsmore, H. E. and Shen V. Y. Software Engineering Metrics and Models, The Benjamin-Cummings Publishing Co, Redwood, 1986.

[4] Cowan, N. "The Magical Number 4 in Short-Term Memory: A Reconsideration of Mental Storage Capacity," Behavioral and Brain Sciences, vol.24, pp.87-185, 2001.

[5] Dunsmore, A. and Roper, M. "A Comparative Evaluation of Program Comprehension Measures," The Journal of Systems and Software, vol.52, Issue 3, pp.121-129, June, 2000.

[6] Miller, G. A. "The magical number seven, plus or minus two: Some limits on our capacity for processing information.," Psychological Review, pp81-97, 1956.

[7] Nakamura, M. and Monden, A. Satoh, H. Itoh, T. Matsumoto, K. Kanzaki, Y. "Queue-based Cost Evaluation of Mental Simulation Process in Program Comprehension," Proc. Of 9th International Software Metrics Symposium, pp.351-360, Sep. 2003.

[8] Spencer, R. "Program Comprehension," Encyclopedia of Computer Science and Technology, April, 1995.