変数の使われ方に着目したプログラム疲労に関する一考察

A Study of Paid Attention to how to use the Variable for Program Fatigue

○吉田勇輝*　坂本裕司*　金子正人**　武内　惇**　薗田孝造***
Yuki Yoshida　Hiroshi Sakamoto　Masato Kaneko　Atsushi Takeuchi　Kouzou Sonoda
*日本大学大学院工学研究科　〒963-8642　福島県郡山市田村町徳定字中河1
**日本大学工学部　〒963-8642　福島県郡山市田村町徳定字中河1
***マイクロテクノ株式会社　〒211-0041　神奈川県川崎市中原区下小田中2-33-36

1．はじめに

プログラムに仕様変更要求があった場合，現在のプログラムを変更するか，新規にプログラムを作成するかのプログラムの変更可否判断（以下，変更可否判断）はプログラム開発者の経験によるため，プログラムの変更作業（以下，変更作業）の工数が予定を超えるという問題が起こることがある．プログラムの変更作業を行う前に，現在のプログラムの変更の難しさの状態を算定して，その結果から現在のプログラムを変更するか，新規にプログラムを作成するかを判断する手法が求められている．

これまで，プログラムモジュール（C言語における関数）を変更するにあたり，変更作業（追加，削除，書き換え）行数当たりの変更作業工数（変更作業にかかった時間　単位：分）をプログラム疲労度と呼び，プログラムソースコードのネストの構成と，条件判定式の構成に注目してプログラムソースコードの構成とプログラム疲労度の関係を分析した[1]．また，プログラムモジュール間の外部変数の使用法に注目してプログラムモジュール間の結合法とプログラム疲労度の関係を分析した[2][3]．

本稿ではプログラムの機能構造が複雑なプログラムモジュールの変更に労力を要することに注目し，プログラム中の変数の使われ方とプログラム疲労度の管駅を分析する．このため，

（1）変数のライフサイクル
（2）変数の使われ方の分析
（3）変数の使われ方の重要度

について考察し，変数の使われ方の可視化と疲労度の関係について述べる．

2．変数の使われ方と重要度

2．1　変数のライフサイクル

変数のライフサイクルとは，プログラムソースコード内で変数に値が設定された場所から，値の使用が終わるまでとする．

2．2　変数の使われ方

変数の使われ方はプログラム内で頻繁に用いられる変数の使い方であることから以下の６つに分類した．

① 初期値設定
② 戻り値のない関数の引数
③ 戻り値のある関数の引数
④ 変数内のデータ変更
⑤ 他の変数のデータ変更動作への関与
⑥ 条件分岐の条件判定に使用

ポインタや配列に関しては今後の研究課題とする．

2．3　変数の使われ方の重要度

変数の使われ方の分類を用いてプログラムへ与える影響の大きさ（以下，重要度）を次の４段階に分類する（表１）．重要度の算定理由を表１に示す．

表１．変数の使われ方と重要度

重要度	使われ方	理由
1	①初期設定 ②戻り値のない関数の引数	決められた場所の変数の値を変更するので変更作業が容易になり、プログラム疲労への影響が少ないと考えられるため
2	④変数内のデータ変更	変更箇所の決定、変数の値の変更と作業項目が増加することで変更作業が複雑になっていきプログラム疲労への影響も大きくなると考えられるため
3	③戻り値のある関数の引数 ⑤他の変数のデータ変更動作への関与	変更場所の決定、変数の値の変更を行うと、自動的に他の変数の値も変更することになるので変更作業が複雑になり、プログラム疲労への影響が大きくなると考えられるため
4	⑥条件分岐の条件判定に使用	変更場所の決定、変数の値の変更に加え、条件式の変更、分岐先の変更と変更作業がより複雑になり、プログラム疲労への影響が大きくなると考えられるため

3．変数の使われ方の可視化と疲労度の関係

２章で挙げた変数のライフサイクルと使われ方を用いてモジュール内の変数の働きや働いている範囲（以下、影響範囲）を可視化する．

3．1　変数の使われ方の可視化の手順

作成手順は以下の通りである．

① 変数のライフサイクルの確認
② 変数の使用箇所の確認
③ 変数の使用箇所ごとに重要度を分類
④ ライフサイクルをｘ軸、重要度をｙ軸と設定し、その関係から関係図を作成
⑤ ④で作成した図を基に変数の影響範囲を色分けする

①～⑤を全ての変数について行う

⑥ ①～⑤で作成した変数毎の図の影響範囲を重ね，モジュール全体の図を作成する

⑤の色分けでは重要度が低い方から青（／），緑（＼），黄色（グレー），赤（黒）とする．

3．2　変数の使われ方の可視化の事例

事例としてＣ言語により記述された約２００行のソーラーカー走行シミュレーションプログラム用いた．プログラムを複数の度関数に分けしたもの（以下，プログラムＡ）と，１つの関数にまとめたもの（以下，プログラムＢ）の２種類を作成し，可視化した結果の比較，検討を行う．

作成した図は以下の通りである．

- 図１はプログラムＡの変数毎の使われ方
- 図２はプログラムＡの main 関数全体
- 図１はプログラムＢの変数毎の使われ方
- 図２はプログラムＢの main 関数全体

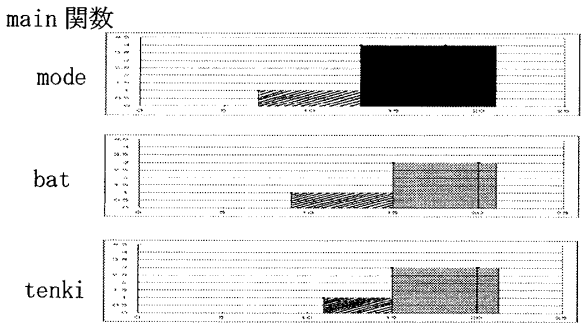

図１．プログラムＡ変数ごとの関係図

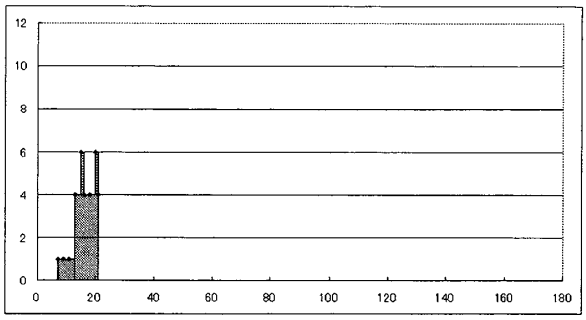

図２．プログラムＡ全ての変数の変数ごとの関係図

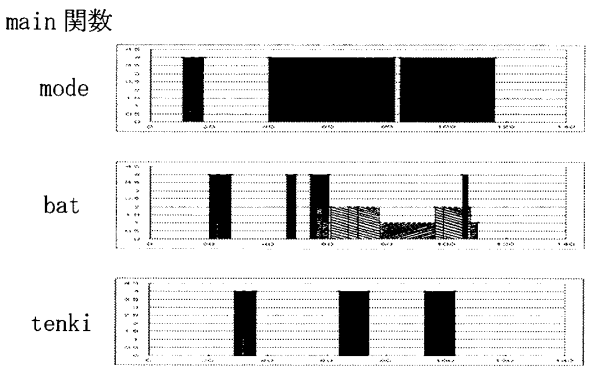

図３．　プログラムＢ変数毎の変数ごとの関係図

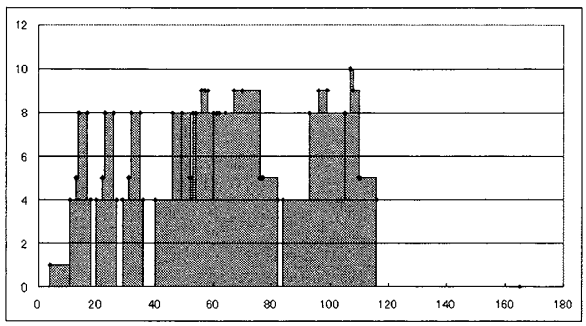

図４．プログラムＢ全ての変数の変数ごとの関係図

３．３　プログラム疲労度との関係

図１と図３を比べると，図１の方が関数に分割したプログラムＡを基に作成したため，影響範囲が狭い，図の凹凸が少ないといったことが分かる．これは，図３と図４を比較した場合も同様のことが言える．

また，図３では，bat 変数（２段目）は色が頻繁に変化しているほか，重要度４「⑥条件分岐の条件判定に使用」の影響範囲が広く，プログラム全体に及んでいる．

これらのことから，図４のように凹凸が多い，図３のように色が頻繁に変わる場合は，変数の使用方法がプログラム内で頻繁に変わっているのでプログラムが複雑になっていく傾向にある．よって，プログラムプログラム疲労が蓄積していると考えられる．

他に，変数ごとのライフサイクル，影響範囲が狭い方がその変数はあまり動作していない，プログラムへの影響が小さいということになり，変更が容易なり，プログラム疲労が蓄積していないと考えられる．また，モジュールの機能構造が複雑なプログラムＢの変数の影響範囲が広い傾向にある．よって，機能構造が複雑なモジュールの変数は影響範囲が広いと考えられる．

また，図３の mode 変数様に変数の使われ方が「⑥条件分岐の条件判定に使用」が主な変数は，プログラムにおいて処理を制御する変数であると考えられるので変数の内容，使い方を変更する場合は，変更箇所と影響範囲の確認をしながらの作業になる．よって，プログラムの変更が難しくなる．よって，このような変数の使い方もプログラム疲労が蓄積しやすいと考えられる．

図１，２の様に変数の使われ方が単純，影響範囲が狭い場合はプログラム疲労が蓄積していないと考えられる．

さらに，図３や図４を見ると，いくつかの大きな塊になっている部分がある．この塊の部分は一つの機能として独立しているとすることができ，関数として区切ることができる部分でないかと考えられる．実際にプログラムＡ，プログラムＢのソースコードと図２，４を合わせて検証すると，プログラムＡで関数に分割している範囲の処理と図２，４の塊の部分の処理の内容がほぼ一致した．このことから，今回可視化するために用いた図から，変数の使われ方，影響範囲の他に，どの部分をどのように関数に分割した方がよいかということも分かることが判明した．よって，モジュール強度が低いモジュールを判別，分割するための方法としても用いることができると考えられる．

４．おわりに

本稿では，変数のライフサイクル，変数の使われ方，を可視化し，可視化した結果がどのような状態になるとプログラム疲労度への影響が大きいのかを考察した．

今後の課題として，ポインタや配列の使用も含め、変数の使われ方の詳細化と図の改良，考察内容を実験を用いて検証することとなっている．

【参考文献】
[1]室屋ほか：“Ｃプログラム疲労の疲労に関する一考察”，電気関係学会東北支部連合大会，2004
[2]中島ほか：“プログラム疲労度に関する一考察”第 49 回日本大学工学部学術研究報告会，2006
[3]坂本ほか：“モジュール結合度に基づくプログラム疲労度の測定方法の一考察”情報処理学会全国大会，2008