B-024

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 Program Fatigue Caused by the Use of Variables in C Source Code

吉田勇輝* 金子正人** 武内　惇** 薗田孝造***
Yuki Yoshida　Masato Kaneko　Atsushi Takeuchi　Kouzou Sonoda

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した[1][2][3]．

さらに，プログラムの機能構造と変数の使用方法の関係に注目し，プログラム中の変数の使われ方の分類，ライフサイクル※２を提案し，関係図を用いることによって変数の使われ方とプログラム疲労の関係性を考察し実験を行い確認した[4]．

本稿ではこれまでの変数の使われ方の分類ではプログラムの処理の流れに考慮していないという問題があるため，プログラムの処理の流れに注目した分類を行い，同時に関係図の見やすさの改良も行う．また，変数の使われ方がプログラムへ与える影響を仕事量として数値化する．また，変数の使われ方の新たな分類と変数の仕事量について検討，分析し，変数の使われ方，変数の仕事量とプログラム疲労※１との関係について述べる．

２．変数の使われ方の再検討，関係図の改良

２．１　変数の使われ方の再検討

これまでの変数の使われ方の分類はプログラム変更作業への影響を重視し，考えたためにプログラムの処理の流れを考慮していなかった．そのため，実際のプログラム処理と「変数の重要度（以下，重要度）」の分類との間に不明慮な点が多く残った．

この問題を解決するため，変数内のデータの変更，使われ方の変更をすることによって，プログラム処理にどれだけ影響するのかに注目した変数の使われ方の分類を再検討した（表１）．

（１）重要度１：データの出力

データの出力動作に使われている場合，プログラム処理ではデータ出力後の処理に影響がないこと，変更作業では出力箇所を確認するということから重要度１とする．

（２）重要度２：戻り値のない関数の引数

戻り値のない関数の引数に使われている場合，プログラム処理では戻り値が無いために呼び出し側の関数の処理には影響が少ないこと，変更作業では関数間のデータのやり取りを確認するということから重要度２とする．

（３）重要度３：データを変更する動作

変数の初期設定や入力データの格納，値の変更，他の変数の値の変更への関与や戻り値のある関数の引数に使われている場合，プログラム処理では変数の値を変更することによって変更後のプログラムの処理に影響が出ること，変更作業では変更前と後の処理の流れ，関係した変数の動作，関数間のデータのやり取りを確認するということから重要度３とする．

（４）重要度４：条件分岐の条件判定

条件分岐の条件判定に使われている場合は，変数の値によって実行する処理としない処理が発生すること，変更作業では分岐によってプログラム処理の内容が変化するため分岐の数だけ重要度１～３の確認作業が増加するということから重要度４とする．

表１．新しい変数の使われ方と重要度

仕業、処理 ＼ 使われ方	実行結果の確認	関数間のデータのやり取り確認	データ変更後のプログラム処理の確認	入力データの確認	関係した変数の値の確認	分岐ごとの確認	条件判定の確認	実行結果	関数間のデータのやり取り	処理内容の変化	分岐	重要度
出力	○							○				1
関数の引数（戻り値無し）	○	○						○	○			2
初期設定	○		○	○				○		○		3
入力	○		○	○				○		○		
変数の値の変更	○		○	○				○		○		
他の変数の値の変更への関与	○		○	○	○			○		○		
関数の引数（戻り値有り）	○	○	○	○	○			○	○	○		
条件分岐の条件判定に使用	○	○	○	○	○	○	○	○	○	○	○	4

２．２　関係図の改良

関係図は変数の使われ方がどのようにプログラムへ影響しているのか，どの部分の変更が難しくなっているのかを図で見ることを目的としている．しかし，これまでの関係図はソースコードと図が別々なため，関係図とソースコードが見比べ辛いという問題がある．

そこで，新たな関係図はソースコードと変数が使われている範囲（以下，影響範囲）の向きを修正し，ソースコードと重ねることにより，一目で変更が難しい部分が分かるように見やすさを重点的に改良する（図１）．

関係図の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 重要度を横軸，ライフサイクルを縦軸と設定し，グラフを作成
⑤ ④で作成したグラフを基に変数の影響範囲を色分けし，ソースコードと重ねる
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を全ての変数について行う．

図１の２７行目から３３行目は複数の変数が使用されているため山が高くなり，変更作業が難しくなっていることを表す．

*日本大学大学院工学研究科情報工学専攻
**日本大学工学部情報工学科
***マイクロテクノ株式会社

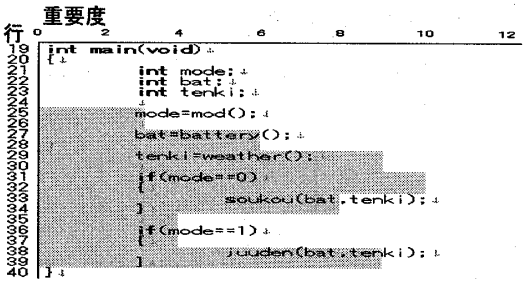

図１．関数ごとの関係図

３．変数の仕事量

３．１　仕事量の目的

変数の働きがプログラム疲労にどれだけ影響しているのかを定量的に確認するため，変数がプログラム中で働いている量を仕事量として算出する．また，関数ごとの仕事量の合計値，平均値，分布から仕事量とプログラム疲労度との関係を考察する．

３．２　仕事量の算出方法

変数の仕事量は関係図の影響範囲の面積を求めることによって算出する．よって変数ごとの仕事量の計算式は以下の通りである．

$$A = \sum_{i=1}^{4} i x_i$$

A：仕事量　ｉ：重要度　x_i：ライフサイクル（行数）

３．３　変数の仕事量とプログラム疲労の関係

変数の仕事量とプログラム疲労の関係を調べるために１つの関数で構成されているプログラムＡと複数の関数で構成されているプログラムＢについて考察する．２つは同じ機能のプログラムである．プログラムＡの仕事量を表２に，プログラムＢの仕事量を表３に示す．

表２．プログラムＡ（main 関数）の仕事量

関数名	変数名	仕事量	合計	平均	一行あたりの仕事量
main	mode	443	1780	254.29	15.21
	bat	260			
	tenki	315			
	kyori	210			
	sokudo	87			
	syouhi	91			
	stime	374			

表３．プログラムＢ（main 関数）の仕事量

関数名	変数名	仕事量	合計	平均	一行あたりの仕事量
main	mode	54	100	33.33	4.55
	bat	26			
	tenki	20			
mod	mo	51	51	51	3.40
battery	ba	51	51	51	3.40
weathar	we	51	51	51	3.40
soukou	stime	164	603	100.50	12.31
	batt (batの派生)	103			
	wea (tenkiの派生)	44			
	kyori	127			
	sokudo	81			
	syouhi	84			
juuden	stime	136	281	93.67	7.39
	batte (batの派生)	57			
	weat (tenkiの派生)	88			
syuturyoku	kyo (kyoriの派生)	3	5	2.50	1
	batter (batt , batteの派生)	2			

プログラムＡとプログラムＢの仕事量を比べると，プログラムＢの方が仕事量の合計値，平均値ともに低くなった．これはプログラム内の機能を複数の関数に分けることによって仕事量が分散したためと考えられる．また，両方のプログラムに仕事量の偏りが見られた．仕事量の一番大きい変数と小さい変数の差はプログラムＡでは最大３５６，プログラムＢでは最大１２０になった．また，１行当たりの仕事量を求めると，プログラムＡの方がプログラムＢより小さくなった．

この結果から，変数ごとに考えた場合，変数の仕事量が大きいとその変数はプログラムにとって重要な変数のため変更作業を慎重に行わなければならない．そのため変更作業の工数が増加し，プログラム疲労への影響が大きいと考えられる．

関数ごとに考えた場合，変数の仕事量の合計値，平均値が大きいとその関数の構成は複雑になる傾向があり，プログラム疲労への影響が大きいと考えられる．

１行ごとの仕事量が多い場合はその関数はプログラムにおいて重要な役割を担っている変数が集まっている，もしくは，重要な処理を行っていると考えられる．そのため，変更作業を行う際に，関数や変数の分析，テスト作業などを入念に行わなければならないため，変更作業が難しくなり工数が増加しプログラム疲労への影響が大きいと考えられる．

プログラム全体，関数全体の仕事量の合計値，平均値が小さいが，特定の変数へ偏っている場合，その変数はプログラムにとって重要な変数であると考えられる．そのため，仕事量が偏った変数やその変数が使われている部分の変更作業が難しくなり工数が増加し，プログラム疲労への影響が大きいと考えられる．

また，プログラムの規模が大きくなると仕事量も大きくなる傾向にあると考えられる．

４．おわりに

本稿では，変数の使われ方の違いや変数の仕事量がプログラム疲労へ影響していること，変数の仕事量がプログラム疲労度を予測するパラメータとして用いることができる見込みを得た．

今後の課題として，今回分析した内容を実験を用いて確認すること，変数の使われ方の分類，関係図のさらなる改良，仕事量がプログラム疲労へ与える影響のさらなる分析，疲労予測式※３[1]のパラメータ設定，適用を行っていく．

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
[4]吉田ほか：“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”，情報処理学会全国大会，2009

用語解説

※１プログラムの変更の難しさを表す
　プログラム疲労が高い場合はプログラムの変更が難しい
※２ソースコードにおいて変数の内部にデータが格納され使用開始した行から最後に使用した行までの行数
※３変更作業後のプログラム疲労度を予測する式