2P-05

Cソースプログラムの疲労度の一考察

○ 滝 陽一（日大院・工・情報） 須田 浩崇（日大工・情報） 鈴木 邦彦（日大院・工・情報）
金子 正人（日大工・情報） 武内 惇（日大工・情報） 藤本 洋（日大工・情報）

1．はじめに

プログラムの静的特性を測定し、プログラムの推敲を系統的に行うことにより、信頼性、変更容易性、再利用性の高いプログラムの設計法を提案してきた[1][2][3]。本稿では機能追加や障害対策のためにプログラムコードの変更作業が続けられる時、プログラムの性能や信頼性の劣化や変更作業工数の増大を招くため、プログラムの特性の変化を時系列で管理することにより当該プログラムを変更するか、あるいは新規に設計し直すかの判定（変更可否判定）を行うための方法を提案する。

プログラム変更に伴うプログラムの変更要求（外部要因）とプログラムの変更に伴うプログラムの静的特性・動的特性（内部特性）の変化を表す方法（メトリックス)、プログラムの時系列な変化に伴うメトリックス値の変化を解析する方法がないことから、プログラムの変更可否判定や変更作業の見積もりは作業担当者の能力に頼らざるをえない。このためプログラムの変更作業に多大の労力と費用を費やしたり、有力なソフトウェア資産を無駄にしていることが多い。本稿では、プログラムの変更可否判定や変更作業の見積もりを行うための科学的な方法論を考察する。

プログラムの変更可否を決定するためには、時間経過に伴うプログラム特性の変化を捉える必要があることから、現在適用されているプログラムコードのその時々のメトリックスでは不十分である。我々は「プログラムの疲労度」という概念を新しく導入することにより、プログラムの変更の外部要因とプログラムの内部特性の変化の関係モデルを作成する。その結果から、ライフサイクル全体に亘ってのソフトウェアの変更作業管理の仕組みを提案する。また、その関係モデルを基にプログラム疲労度の測定法、疲労限界の予測法、変更作業見積もりにおける疲労度の反映法、疲労を回復する方法、疲労を起こさないプログラム記法の実現を目指す。

2．プログラムの疲労度の考え方

2．1「もの」の疲労の考え方

“金属の疲労”を例に「もの」の疲労の特性を検討する(図1)。金属が壁に取り付けられているとする。この金属は最初は壁と垂直になっている。しかし金属の上から力を加えていくと、徐々に金属は変形していく。そして力を加え続けていくと、ある時点で金属は折れてしまう。金属が折れてしまった原因としては、マイクロクラックの蓄積によりヒビ割れが発生し、徐々にヒビ割れが増加したことで金属の強度が低下したことが挙げられる。従って、金属での疲労とは、マイクロクラックの量で示される。また、金属での疲労限界とは折れる直前のマイクロクラックの許容量であることがわかる。

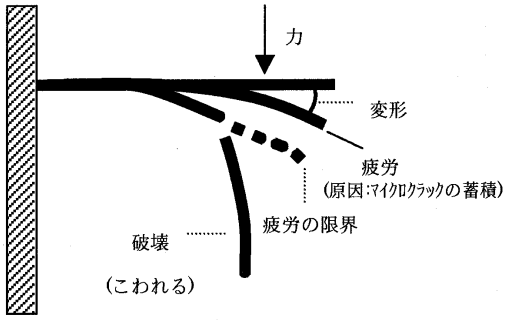

図1．「もの」の疲労 （例：金属）

2．2 「プログラム」の疲労の考え方

「もの」の疲労をプログラムの疲労に当てはめると、表1のようになる。図2に概要を示す。

表1．「もの」の疲労とプログラムの疲労の対応

「もの」の疲労	「プログラム」の疲労
力	プログラムの変更要求
変形	変更要求により変更されたプログラム
疲労	プログラムの変更に伴い、プログラムを悪くするもの
疲労の限界	プログラムの変更可の限界
破壊	もうプログラム変更ができない

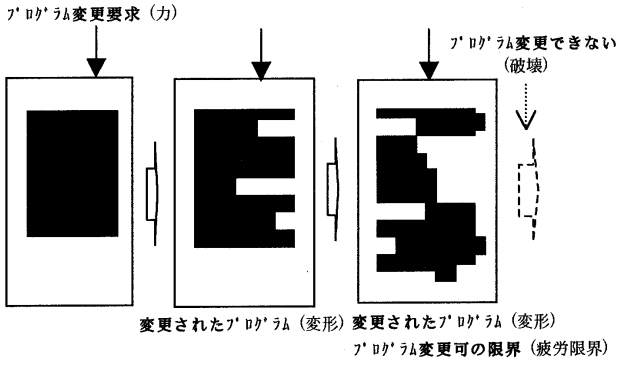

図2．「プログラム」の疲労

プログラムの疲労は信頼性の劣化や変更工数の増大という静的疲労と性能が劣化するという動的疲労の両面から検討する。静的疲労の原因はプログラムの構造の欠陥の増大であり、変更工数が莫大になったり信頼性が要求を満たさないという結果に陥る（図3、4)。動的疲労の原因はリソースの使用法の不適切さの増大であり、性能が要求を満たさないという結果に陥る。

A Consideration of C Source Program's Fatigue
Youichi Taki Hirotaka Suda Kunihiko Suzuki
Masato Kaneko Atushi Takeuchi Hiroshi Fujimoto
College of Engineering Nihon University

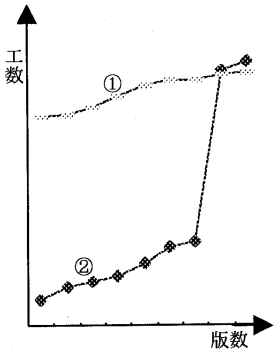

図3.工数に関する疲労度

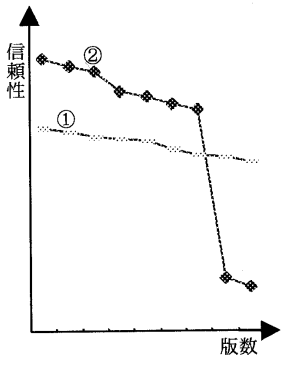

図4.信頼性に関する疲労度

2．3 プログラムの疲労限界の考え方

プログラムの疲労は静的疲労に関する工数・信頼性と動的疲労に関する性能の3つの視点で測る

図3のグラフは、プログラム変更の際の変更作業工数の推移を時系列で管理したものである。このグラフでは、プログラム①のグラフは最初から工数高い。しかし、グラフの上がり方を見てみるとあまり急激な上昇は見られない。結論としてプログラム①は最初から工数のかかるプログラムであり、変更回数が増加してプログラムがその都度変更されても、最初のプログラムと比較してみてそれほど急激な疲労はしていないということがいえる。一方、プログラム②のグラフでは最初の工数は低い。しかし、変更回数を増やしていくとある時点で著しく上昇してしまっている。この急激な上昇をしている個所で明らかにプログラムの構造が悪くなる変更をしたために、プログラムが疲労してしまい工数が莫大に増加したことがいえる。ここが疲労限界である。我々はこの箇所の変更内容から急激な作業工数増加の原因を見つけることでプログラム疲労の要素を定義できると考えている。

図4のグラフは、プログラム変更の際の信頼性の推移を時系列で管理したものである。このグラフでは、プログラム①のグラフは最初から信頼性が低い。しかし、グラフの下がり方を見てみるとあまり急激な減少は見られない。結果として、プログラム①は最初から信頼性の低いプログラムであり、変更回数が増加してプログラムがその都度変更されても、最初のプログラムと比較してみてそれほど急激な疲労はしていないということがいえる。一方、プログラム②のグラフでは最初の信頼性は高い。しかし、変更回数を増やしていくとある時点で著しく減少してしまっている。この急激な減少をしている個所で明らかにプログラムの構造が悪くなる変更をしたために、プログラムが疲労してしまい信頼性が急激に低下したことがいえる。ここが疲労限界である。我々はこの箇所の変更内容から急激な信頼性低下の原因を見つけることでプログラム疲労の要素を定義できると考えている。

動的疲労の性能ついての疲労限界の見方は、図4のグラフの横軸がリソース使用法の悪さ、縦軸が性能に変わっただけであり、図4と同様である。

疲労限界を基準にすることでプログラムの変更可否の判断を可能にする。すなわち、疲労限界に達していない時点ではプログラムが変更可であり、疲労限界に達した時点以降では新しいプログラムを開発した方が効率的であると考えられる。

3．プログラム疲労度と品質メトリックス

プログラムの疲労度には構造の悪さとリソース使用法の悪さの2つが関わりを持っている。その構造の悪さとリソースの使用法をどのようにして計測するかということで、我々はISOソフトウェア品質特性図[4]とBohemの品質特性図[4]を基に、プログラム疲労度に対応するメトリックスを抽出した。その結果を表2に示す。

今回抽出したメトリックスはQAC[5]で測定することができる。現在、プログラム疲労に関するメトリックス、メトリックスの疲労限界基準値などを多数のプログラムの変更版を基にデータ収集し、調査中である。

表2．プログラム疲労度に関するメトリックス

計測尺度	内容		対応するメトリックス（計測方法）
プログラムの構造	関数ﾍﾞｰｽ	手続き	ｱﾙｺﾞﾘｽﾞﾑ全体の経路複雑度
			経路複雑度の拡張指標
			制御構造の最大ネスティング数
			関数の中のgoto文の数
			関数の中の可能性のあるﾊﾟｽｶｳﾝﾄ数
		ﾃﾞｰﾀ	未使用変数の数
			構造体の複雑度
	ﾌｧｲﾙﾍﾞｰｽ		ﾌﾟﾛｸﾞﾗﾑ難易度
			ｺﾒﾝﾄ中の非ﾌﾞﾗﾝｸ文字数とｺｰﾄﾞ中の非ﾌﾞﾗﾝｸ文字数の比率
リソースの使用法			ﾒﾓﾘの使用法 同期のﾀｲﾐﾝｸﾞ

4．おわりに

プログラムの変更可否判断の仕組みを確立するために、我々は実世界に存在する「もの」の疲労度に対応させて、「プログラム疲労度」という概念を用いた。この仕組みを実現することにより、プログラムの変更可否判断が可能になるだけでなく、現在のプログラムの疲労度合、疲労限界の予測、変更作業の見積もり、疲労しているプログラムの疲労回復、疲労を起こさないプログラム記法について検討を進める。今後の課題としては、変更時の工数・性能・信頼性が疲労度のメトリックスの値とどのように関連し、メトリックスの値がどのような傾向をもっているのかを調査することが挙げられる。それにより、疲労の限界値が設定できると考えられる。

5．謝辞

本研究を進めるに当たり、適用事例等について御検討を願った、(株) 東陽テクニカの二上課長、奥村氏、熊野氏に感謝致します。

－参考文献－
[1] 野宮 将己 他：Cソースプログラムの推敲技術、1996年度卒業研究報告書
[2] 高山 優典 他：Cソースプログラムの推敲技術、1997年度卒業研究報告書
[3] 橋本 哲雄 他：Cソースプログラムの疲労度、1998年度卒業研究報告書
[4] 森口 繁一：ソフトウェア品質管理ガイドブック、日本規格協会
[5] Cソースコード静的解析ツール：(株)東陽テクニカ