ソフトウェア工学　136－9
（２００２．３．７）

Java クラスファイルにおける複雑さ計測ツールの開発

原田智文　　趙建軍

福岡工業大学　情報工学部　情報工学科
〒811-0295 福岡市東区和白東 3-30-1
Email：zhao@cs.fit.ac.jp

ソフトウェアの品質や生産性を評価・予測する一つの手段として、ソフトウェアの複雑さの尺度を評価する方法がある。ソフトウェア複雑さを評価し基準を持たせ比較することで、ソフトウェアの企画や開発、運用、保守などに役立てることができ、また、ソフトウェアの品質そのものを向上させることも可能となる。本研究では、Java クラスファイルにおける複雑さ尺度の提案とその計測ツールを開発することを目指す。Java クラスファイルの複雑さを測定し評価することにより、よりよい品質を持った Java ソフトウェアの開発の手助けとなることを考えられる。

A Metrics Collection Tool for Java Classfiles

Tomofumi Harada, Jianjun Zhao

Department of Computer Science and Engineering
Fukuoka Institute of Technology
3-30-1 wajiro-higashi, higashi-ku, Fukuoka 811-0295, Japan
Email：zhao@cs.fit.ac.jp

Abstract

In this paper, we introduce a metrics collection tool called Kafer that be used to collect various types of metrics for Java classfiles. We first describe some metrics for Java classfiles, and then show how to compute these metrics for a Java classfile using Kafer.

1.はじめに

ソフトウェアの品質や生産性を評価・予測する一つの手段として、ソフトウェアの複雑さの尺度を評価する方法がある。ソフトウェアの複雑さを詳細に評価することで、ソフトウェアの企画や開発、運用、保守などに役立てることができる。例えば、ソフトウェア製品の企画段階ではソフトウェア複雑度尺度を用いることで、スケジューリングや費用割り当てに役立てられ、また、生産されるソフトウェア製品をより詳細に評価・管理ができる。作成時にはソフトウェアの複雑さをツールとして用いることで、より簡単で品質の良いコードの作成をはかることができる。テストでは、ソフトウェア複雑さ尺度を用いることで、どの部分を入念にテストするかという基準を作ることができる[1]。このように、複雑さを評価することにより、ソフトウェアの品質を向上させることができる。

プログラムが特定の言語でコーディングされたとき、その複雑さは次の３つの観点から評価できる。すなわち、プログラムを構成する語彙の種類や量による語彙的複雑さ、プログラムの制御構造やデータ構造の形式などの構造的複雑さ、プログラムの内容そのものの論理的複雑さである[1]。これらを評価する代表的な要素としては、プログラムの大きさや制御構造、データ構造及びデータフローがある[2]。これらの要素を評価することで、プログラムの複雑さが示され、それを参考に、よりよい品質を持ったソフトウェアの開発が可能となる。

本研究では、Java クラスファイルよりプログラムの複雑さを測定し、評価することにより、よりよい品質を持ったソフトウェアの開発の手助けとなることを目指す。

本論文の構成を述べる。まず第 2 節では Java 仮想マシンについて説明したのち、なぜバイトコードを扱うのかを述べる。第 3 節でクラスファイルの複雑さ尺度についてどのようなものかを説明する。第 4 節では Java バイトコードに対する支援システムツール Kafer で複雑さ解析を行い結果を表示・検討する。そして第 5 節でまとめに入る。

2.Java 仮想マシン(Java virtual machine)

Java 仮想マシン(JVM)は Java プラットフォームの土台となる抽象的な計算機である[3]。Java 仮想マシンは特定のファイルフォーマット、すなわち class ファイルフォーマットの知識のみを保持しており、Java プログラミング言語の知識は保持していない。従って、class ファイルフォーマットに従った「正しい」class ファイルであればどの言語で記述されていたかは問題ではない。class ファイルとは、Java 仮想マシンの命令(もしくはバイトコード)とシンボルテーブル及びその他の付随的な情報を保持したものである。Java 仮想マシンは、中間コードである class ファイルのバイトコードの内容を正しく読み取り実行する。

プログラムをソースからではなく、バイトコードから解析すると様々な利点がある。まず、class ファイルフォーマットは明確に定義されており、ソースファイルよりも解析が容易である点が挙げられる。例えば、ソースプログラムにおいて繰り返し処理に for 文や while 文がある。これらは似たような動作をするが、ソースプログラムでは記述が明確に違うため、それぞれに対応した解析を行う必要が出てくる。それに対して Java バイトコードでは、これらの動作は goto 文と if 文で処理されるため、同じものと見なす事ができる。これは処理の軽減につながり、プログラムを解析する上で効率がよい。次に、プログラムが記述された言語に影響されないため、jasmin などのアセンブラや他の言

図 1　サンプルプログラムと Java バイトコードへの変換例

語で記述されたプログラムでも解析が可能となり、幅広く使うことができる。また、一般にプログラムはソースではなくコンパイル済みの class ファイルが配布されることが多いため、class ファイルを直接解析できたほうが利用価値が高いといえる。

図 1 にソースファイルを Java バイトコード命令に変換した例を示す。この例ではソースコード内の main メソッドと init メソッドをバイトコードに変換している。

3.Java クラスファイルの複雑さ尺度

ここでは、複雑さにはどのような要素があるのかを述べ、そこから評価する要素を決定し、評価方法を述べる。

3.1 プログラム複雑さ尺度

プログラムの複雑さを評価する際に測る要素は以下のようなものがある。

・サイズ
・データ構造
・データフロー
・制御構造

今回は、サイズと制御構造に注目する。

サイズは Line of Bytecode（バイトコード数）で評価する。図 2 のように全く同じ仕様書を渡しても、作成する人や状況によって、でき

あがるプログラムの行数にかなりのバラツキがでるようなケースもあり、どれが複雑かを判断する材料となりうる[4]。

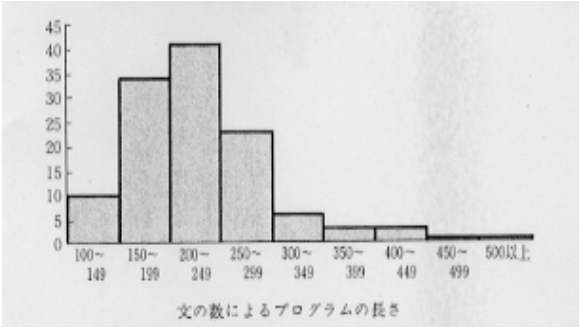

図２ 同一の仕様書から作られたプログラムコード量の分布

また、制御構造を評価する方法には次のようなものがある。

・MaCabe の Cyclomatic 数尺度
・Woodward の Knot の数
・Chen の尺度

この中で唯一プログラムの論理的な複雑さを表す尺度である Cyclomatic 数尺度を用いて、複雑さを判断する。

Cyclomatic 数尺度は、与えられたプログラムの制御流れグラフから、そのグラフ上の全ての経路の数を測定する方法である。このとき、プログラムから以下のような制御流れグラフを作成し、そのグラフのもとで複雑さを測定する。

制御流れグラフ G = (V,E) は、原始プログラムの文と制御流れから得られる有向グラフである。各頂点(∈V)は、原始プログラムにおいて分岐を含まない連続的な部分を１つのブロックとしてまとめたものであり、各有向辺(∈E)は頂点から頂点への制御流れを表す。制御流れグラフの例を図 3 に示す。

まずグラフで用いられている各要素をそれぞれ定義する。

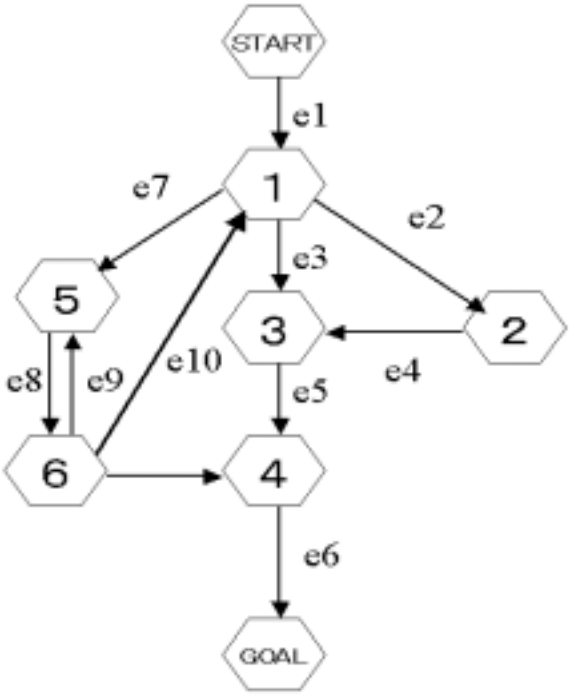

図３ 制御流れグラフ

C (G) ：複雑さ
e = | E | ：制御流れグラフの有向辺の数
n = | V | ：制御流れグラフの頂点の数
すると、制御流れグラフ G = (V,E) よりＣは

$$C(G) = e - n + 2$$

で求められる。

図 3 の場合 e = 11, n = 8, よって

$$C(G) = 11 - 8 + 2 = 5$$

となる。これが Cyclomatic 数尺度である。

実際にグラフから経路を求めてみる。ここで対象となる経路は、

・ある頂点から出発して、再び同じ頂点へ戻ってくる閉路
・START から GOAL へ向かうシーケンシャルなルート

の二種類がある。前者は

・１→５→３→１→……
・５→６→５→……

であり、後者は

・S→１→５→６→４→G

・S→１→３→４→G
・S→１→２→３→４→G
の計５つあることがわかる。

Cyclomatic 数尺度はテストや保守作業のし辛さを表す。複雑さの目安としては、
10 以下・・・よい構造
30 以上・・・構造に疑問
50 以上・・・テストが不可能
75 以上・・・いかなる変更も誤修正を生む原因を作る
とされ[5]、実際には 32 を超えると急にエラーが潜在している可能性が増えるとされている[6]。

3.2 バイトコード複雑さ評価尺度

バイトコード複雑さの尺度を以下に示す。以下の６つの項目で評価する。

・number of methods
　メソッドの数。
・number of fields
　クラス全体の変数の数
・number of attributes
　クラスファイル内にある属性の数。ソースファイルからは見出せない。
・loc of class
　クラスのバイトコード数。
・average loc of method
　メソッドの平均バイトコード数。
・number of Constant pool entries
　コンスタント・プールの数。
・Cyclomatic Complexity
　Cyclomatic 数尺度。

4. クラスファイル複雑さの評価

ここでは、クラスファイル複雑さの解析に用いたツール Kafer の説明を行い、Kafer を実際に使用した結果を紹介する。

4.1 Java 理解支援システム Kafer

Kafer は Java バイトコードに対する理解支援システムである。Java バイトコードの解析や複雑さの尺度測定などの機能がある。図４に Kafer を示す。

図４　解析ツール Kafer

4.2 実行例

実際に複雑さの評価の方法を紹介する。この説明で使うサンプルプログラムは図１のプログラムを使用する。

Kafer の機能の一つである「Matric Tools」より、以下の３つの方法でクラスファイルの複雑さの解析を行う。

・　Line of Bytecode

バイトコードの数をクラスファイルより読み取り、メソッドごとに出力する。図 5 の左側の値がメソッド名で、その右側がそのメソッドのバイトコード数である。下部にはメソッド一つ辺りのバイトコード数の平均が表示される。

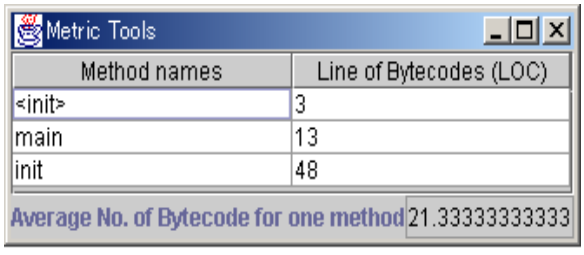

図 5 Line of Bytecode

・ McCabe's Metric

クラスファイルの Cyclomatic 数尺度をメソッドごとに評価する。バイトコードより制御流れグラフを作成し、それをもとにして計測している。左側がメソッド名で、右側が Cyclomatic 数尺度である。

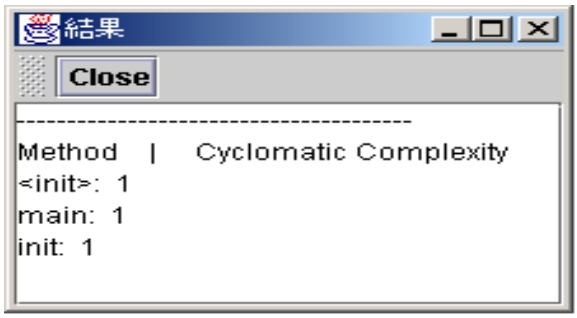

図６ McCabe's Metric

図５では main と init バイトコードの量に３倍の開きがあることが示されたが、複雑さは大差ないことが図６より明らかとなった。

・ A Metrics Suite

各々の結果をまとめて出力する。まずメソッドとその Cyclomatic 数尺度が出力される。それからバイトコードの各評価尺度が出力される。

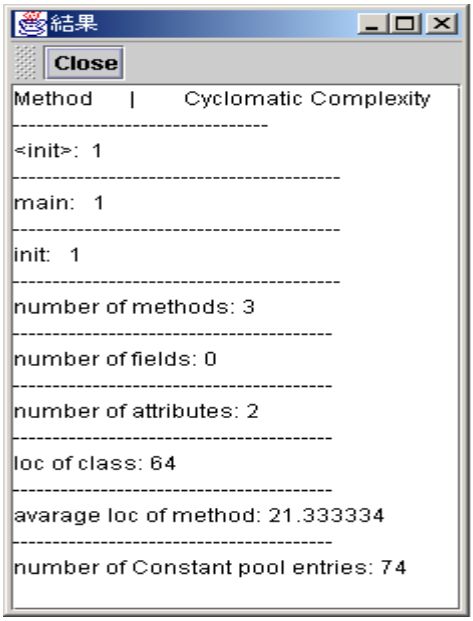

図７ A Metrics Suite

5.まとめ

本研究では、ソフトウェア複雑さ尺度の提案と、その計測を行うツールを開発した。そしてプログラムの複雑さを測定し評価することにより、よりよい品質を持ったソフトウェアの開発の手助けとなることを目指した。その結果、バイトコードから各クラスファイルの複雑さを測定することができた。このことにより、ソフトウェアの品質の向上に役立てることができるといえる。

今後の課題だが、まず、今回は制御構造とサイズにのみ注目しているが、その他の要素に注目した複雑さ測定法も使えるようにする必要がある。今回、複雑さ評価に用いた Cyclomatic 数尺度での判断は、制御構造以外の要因、例えば変数の代入や参照の変化があっても、複雑さの評価には表れないという問題を抱えているためである。したがって、複雑さの判断をより一般的なものにするためにも、他の尺度を併せて用いる必要がある。Kafer はまだ未完成であり、この辺りを拡張していく必要があるといえる。また、大規模な応用システムに対し、このツールがどれだけ有効であるかを確かめる必要性がある。

参考文献

[1] 梁 海述, 辻野 嘉宏, 都倉 信樹, “モジュール間依存度を考慮したプログラムの複雑度”, 電子情報通信学会論文誌, Vol.J73-D-I No.11 , pp.882-890, 1990 年 11 月

[2] 梁 海述, 辻野 嘉宏, 都倉 信樹, “ データフロー情報を考慮したソフトウェアの複雑度について ”, 電子情報通信学会論文誌, Vol.J72-D-I No.11 , pp789-796, 1989 年 11 月

[3] ティム・リンドボルム, フランク・イェリン著,野崎 裕子 訳, “ The Java 仮想マシン 仕様 ”,アジソン・ウェスレイ・パブリシャーズ・ジャパン,1997 年 12 月

[4] トム・デマルコ著, 大野 徇郎 訳, “ デマルコ、大いに語る ”,日科技連, 1998 年 12 月

[5] C.Jones,“ ソフトウェア開発の定量化法 ”,共立出版, 1991 年

[6] 大場 充 著, “ ソフトウェア・プロジェクトの実績データ収集・分析技法 ”,ソフトリサーチセンター, 1993 年