ショートノート

ファジィグラフを用いたクラス構造分析の有効性に関する実験的考察†

阿萬 裕久*¹, 矢鳴 虎夫*², 山田 宏之*¹, 野田 松太郎*¹

オブジェクト指向ソフトウェア開発において、クラス設計は最も重要な課題である。そのため、クラス設計に対して客観的な評価や予測を与えるためのソフトウェアメトリクスが研究されている。そういったメトリクスによる評価を受けて、設計者や開発者はクラスの設計に変更を加えることもある。しかしそれによって、逆に評価の低いクラスを作り出してしまふ恐れもある。それゆえ、そのクラスにおいて重要な位置にある構成要素、すなわちクラスメンバ——メソッドや属性——を把握しておくことは有用であると考えられる。そのようなクラスメンバをクラス構造の複雑さという観点から検出するため、ファジィグラフの連結構造分析法を用いた手法が提案されているが、その有効性についてはこれまで議論されていなかった。本論文では、その手法の有効性を確認するため、多数のクラス設計に対して疑似的な変更を施し、それに伴う複雑さメトリクス値の変化を調べるという実験を行った。その結果、片連結ならびに強連結に関して強化点となっているようなクラスメンバについて、上述の分析法は有効であることが確認された。

キーワード：オブジェクト指向、構造の複雑さ、メトリクス、ファジィグラフ

1 はじめに

オブジェクト指向ソフトウェア[1]の開発において、クラスの設計は最も重要な課題である。クラスは単に要求を満たすだけでなく、後で再利用しやすく、かつ、ソフトウェア障害[2]の発生に対しても対処しやすいように設計されなければならない。そのため、クラス設計に対して客観的な評価や予測を与えるためのソフトウェアメトリクスに関する研究が行われてきている[3]。例えば、クラスの規模やクラスの所有しているメソッドの複雑さ、クラス継承の深さといった観点での評価を与えるメトリクス[4]–[9]、テスト工数や残存バグ数などの予測を与えるメトリクス[10]等が挙げられる。そういったメトリクスによる評価を受けて、設計者や開発者はクラスの設計に変更を加えることがある。しかしながらそれによって、逆に評価の低いクラスを作り出してしまふ恐れもある。一般にメトリクスは、与えられたクラスやプログラムの状態・性質を測定するためのものであり、そこに変更を施すこ

とがどれだけの影響を持つかといった視点はとっていない。そこで、ファジィグラフの連結構造分析法[11]を応用してクラス構造の複雑さを分析する方法が提案されている[12]。これは、クラスを構成しているクラスメンバ（メソッドや属性）間での依存関係をファジィグラフのかたちで表現し、それに対して連結構造分析を行ない、クラス構造の複雑さに大きく影響を及ぼすことになるようなメンバを検出しようとするものである。しかしながら文献[12]では、その手法の提案に留まっており、手法の有効性に関しては議論してなかった。本論文では、多数のクラスに対して疑似的な変更を施し、その変更に伴う複雑さメトリクス値の変化を調べることで上述の分析法の有効性を実験的に示す。

2 ファジィグラフによるクラス構造の複雑さ分析

2.1 概要

ファジィグラフを用いたクラス構造分析法の概要を以下に示す。詳細は文献[12]を参照されたい。

- (1) 有向グラフによるクラス構造の表現
クラスを構成しているメンバ、すなわちメソッドと属性をそれぞれ有向グラフ上の頂点として表す。そして、それらの間での依存関係、すなわち情報の読み出し・書き込み、他のメソッドの呼び出しなどを頂点間を結ぶ有向辺で表す。

† An Empirical Study on the Effectiveness of Class Structural Analysis with Fuzzy Graph
Hirohisa AMAN, Torao YANARU, Hiroyuki YAMADA and Matu-Tarow NODA

*¹ 愛媛大学 工学部 情報工学科
Department of Computer Science, Faculty of Engineering, Ehime University

*² 東亜大学 工学部 システム工学科
Department of System Engineering, Faculty of Engineering, University of East Asia

図1 メンバシップ関数の一例

(2) 依存関係の定量化

(1)で得られた有向グラフの隣接行列を A とする。 A はクラスメンバ間の直接的な依存関係を表現している。そして、この A の n 乗 ($n \geq 2$) は、途中に $n-1$ 個のメンバを介した間接的な依存関係を表すことになる。これら A 及び A のべき乗に対し、分析者がそれらを考慮する度合いをそれぞれ重み係数として与え、その加重化合計でもって各メンバ間での依存関係を定量化する。そして、これらの依存関係の度合いが区間 $[0, 1]$ 内におさまるよう変換する。文献 [12] では、一つの定数を使って正規化しているが、例えば次式のようなメンバシップ関数を用いることもできる (図1)。

$$\mu(x) = \begin{cases} x/2 & (x \leq 2), \\ 1 & (x > 2). \end{cases} \quad (1)$$

こうして得られた依存関係の度合いをもとに、メンバ間の依存関係は一つのファジィグラフとして表現できる。

(3) ファジィグラフの連結構造分析

(2)で得られたファジィグラフにおいて、弱連結・片連結・強連結という3つの視点から各頂点が強化点・弱化点・中立点のいずれとなるのかを調べる。そして、強化点となるような頂点に対応しているメンバをクラスの分析者・設計者に報告し、再設計に関する一つの支援とする。

以上が文献 [12] で提案されている分析法の概要である。

2.2 例

上述の分析法を適用した一例を以下に示す。

(1) 有向グラフによるクラス構造の表現

いま、図2のクラスが与えられたとする。この場合、クラスのメンバとしては

- ・コンストラクタ[‡]: Stack
- ・メソッド: push, pop

```
public class Stack {
    int sp; // スタックポインタ
    int [ ] data= int [100]; // データ格納用配列
    public Stack() { sp = 0; } // コンストラクタ
    public void push(int x) { // データ格納
        if ( sp < 100 ) data [sp++] = x;
    }
    public int pop( ) { // データ抽出
        if ( sp > 0 ) return data [--sp];
        else return -1;
    }
}
```

図2 Java 言語によるスタックの構築例

・属性: sp, data

が挙げられる。そのため、これらを頂点に持った有向グラフを考える。

頂点間を結ぶ有向辺は次のようにして構築される。例えば、メソッド push に着目すると、これは

- (i) sp を読み出し、
- (ii) その値に応じて data へ書き込みを行い
- (iii) sp の値を +1 している。

それゆえ、

- (i) push $\leftarrow$ sp,
- (ii) push $\rightarrow$ data,
- (iii) push $\rightarrow$ sp,

という有向辺を設ける。その他の有向辺についても同様である。

この場合、結果として図3のような有向グラフが得られることになる。

(2) 依存関係の定量化

図3の場合、隣接行列 A は

	Stack	push	pop	sp	data
Stack	0	0	0	1	0
push	0	0	0	1	1
A=pop	0	0	0	1	0
sp	0	1	1	0	0
data	0	0	1	0	0

となる。ここで、 A 及び A のべき乗に対し、それらを考慮する度合いをそれぞれ重み係数として与え、その加重化合計でもって各メンバ間での依存関係を定量化する。 $A^n (n \geq 1)$ に対する重み係数を

[‡] 初期化を行うための特別なメソッド

図3 図2のStackクラスに対応する有向グラフ

w_n とすると、加重化合計(これを A_w^* と書く)は次式で与えられる:

$$A_w^* = \sum_{n=1}^{\infty} w_n A^n.$$

例えば、重み係数として

$$\begin{aligned} w_1 &= 1.0, \quad w_2 = 0.5, \quad w_3 = 0.2, \\ w_4 &= 0.1, \quad w_k = 0 (k \geq 5), \end{aligned} \quad (2)$$

を用い、加重化合計 A_w^* を求め、式(1)でもって値域を $[0, 1.0]$ へ変換すると、

$$f(A_w^*) = \begin{matrix} & \begin{matrix} \text{Stack} & \text{push} & \text{pop} & \text{sp} & \text{data} \end{matrix} \\ \begin{matrix} \text{Stack} \\ \text{push} \\ \text{pop} \\ \text{sp} \\ \text{data} \end{matrix} & \begin{bmatrix} 0 & 0.35 & 0.4 & 0.7 & 0.1 \\ 0 & 0.4 & 0.7 & 0.8 & 0.6 \\ 0 & 0.35 & 0.4 & 0.7 & 0.1 \\ 0 & 0.7 & 0.8 & 0.75 & 0.35 \\ 0 & 0.1 & 0.6 & 0.35 & 0.05 \end{bmatrix} \end{matrix}$$

が得られる。この (i, j) 成分[§]は、「 i 番目のメンバが j 番目のメンバに影響を及ぼす」というファジィ命題に対応するメンバーシップ値である。なお、ここでいう「影響」とは、 i 番目のメンバにおいて何らかの障害(誤動作や予期されないデータの保持)が発生したとすると、それが直接的もしくは間接的な原因となって j 番目のメンバでも何らかの障害が発生するということを意味する。

これは、ファジィグラフとして図4のように表現される。

(3) ファジィグラフの連結構造分析

ファジィグラフでは、各辺にそれぞれ一つのメンバーシップ値が対応している。ここで、メンバーシップ値が定数 c 以上であるような辺のみで作られたクリस्पグラフ(つまり、メンバーシップ値は考えずに辺の有無のみを考慮したグラフ)を c -

カットグラフ[11]という。いま、 $c=0.6$ とすると、図4のファジィグラフに対する c -カットグラフは図5のようになる。図5において、辺の向きを無視すると、任意の2頂点間に経路が存在している。つまり、このグラフは弱連結なグラフ[11]である。ここで $c > 0.6$ とすると、頂点「data」が孤立してしまい、 c -カットグラフは弱連結でなくなる。それゆえ、 c -カットグラフが弱連結となるような c の上限は0.6であるといえる。この値を弱連結度という。つまり、図4のファジィグラフの弱連結度は0.6である。

ここで、頂点「data」とそれに接続する辺を取り除いた場合を想定する。すると、弱連結度は0.7と高くなる。つまり、頂点「data」の存在は、ファジィグラフの弱連結度をより低いものとするような働きをもつといえる。このような頂点を弱化点[11]という。より一般的には、ファジィグラフから一つの頂点とその頂点に接続する辺を取り除いてサブグラフを作り、そこでの弱連結度が強まるとき、その頂点を弱化点という。また、逆に弱

図4 図2のStackクラスに対応するファジィグラフ

図5 図4のファジィグラフに対する c -カットグラフ ($c=0.6$)

§ 正確には「第 i 行、第 j 列成分」であるが、ここではこのように書く。

連結度が弱まるときはその頂点を強化点といい、弱連結度が変わらないときは中立点という [11]–[13].

結果として図4の場合、頂点「sp」が強化点、頂点「data」が弱化点[¶]、その他が中立点となる。同様の手順で、片連結及び強連結についても強化点・弱化点・中立点の検出が可能である。ただし、片連結とは「任意の2頂点間において、少なくとも一方から他方への経路(辺の方向は無視しない)が存在すること」を意味し、強連結とは「任意の2頂点間において、双方向の経路が存在すること」を意味する。

この例の場合、sp と push が片連結に関する強化点となる。強連結に関する強化点は存在しない。

したがって、文献[12]に従うと、メソッド「push」や属性「sp」に関係するような部分に変更を施すことは、他のメンバにも何らかの影響を与える可能性が高く、注意が必要である。特に、規模の大きなクラスや構造の複雑なクラスに変更を施す場合、そのような個所を把握しておくことは重要である。

3 実験

前節では、文献[12]で提案されている分析法の概要を示した。これには、ファジィグラフにおいて強化点に対応しているようなメンバは、そのクラス構造の複雑さに関して重要な位置にあり、クラス設計の見直しや修正を行う際に特に注意が必要であるという主張が込められている。しかしながら、この手法を用いることによる有効性については議論されていない。そこで本研究では、この分析法の有効性を確認するための実験を行った。具体的には、クラス設計に対して擬似的な変更を施し、その変更を通じてクラス構造の複雑さの変化を測定する実験を行った。実験の方法、対象、結果の集計方法、ならびに実験の持つ意味を以下に示す。

・実験方法

クラス構造を表現している有向グラフにおいて、有向辺を1つだけ追加・削除するという操作を可能な限り行った。その際、操作の前後におけるクラス構造の複雑さの変化を測定した。なお、複雑さの測定には、クラス構造複雑さメトリクスの一つである SC [6]を利用した。SC は、一つのクラスが与えられたとき、そこに含まれる各メンバの

複雑さとメンバ間の依存関係の強さとを複合することで、クラス構造の総合的な複雑さについて評価値を与えるものである。今回の実験では、各メンバの複雑さ測定には LOC (コメント文と空行を除いたプログラム行数) [3], [9]を用い、メンバ間の依存関係の定量化には(2)式のパラメータを用いた。

・実験対象

Java 言語で記述されたクラスを95個用意し、実験対象とした。なお、これらのクラスは Sun Microsystems 社から無償で配布されている Java 2 SDK, Standard Edition v1.2.2に含まれていたクラスから無作為に抽出したものである。実際には、これらを有向グラフで表すと複数の連結成分に別れてしまうものもあり、結果的に338個の連結成分について実験を行うことになった。

・結果の集計方法

(1) 各メンバごとの集計：

各メンバに対し、それに接続するような有向辺が追加されたとき、あるいはそれに接続していた有向辺が削除されたときに、その操作の前後における複雑さメトリクス値の変化量を求め、各メンバごとにその平均値を算出した。

(2) 連結成分ごとの集計：

(1)で得られた平均値でもって各メンバを順位付けし、その中での強化点メンバの順位を記録した。なお、順位は平均値の降順によるものである。つまり、 N 個のメンバからなる連結成分が与えられたとき、強化点メンバはその N 個の中で(変化量の)平均値が第何位の大きさであるかを記録した。

ただし、強化点メンバの検出は、2.2節と同じ手順・同じパラメータで行った。

・実験の持つ意味

有向辺の追加・削除は、そのクラスにおけるメソッド-属性間、もしくはメソッド間での依存関係の変更を意味しており、間接的にクラス設計の変更を表している。この変更操作の前後における複雑さメトリクス値の変化量は、その変更操作がクラス構造の複雑さに及ぼした影響の大きさを表している。そして、これらの変化量をメンバごとに集計することで、そのメンバの他への影響力が表現される。ここでは、強化点メンバがその他のメンバに対して高い影響力を持っていることを示せ

¶ sp を無視すると弱連結度が0.4に低下する。

|| 個々の結果に関しては、紙面の都合上割愛する。

ばよい。

一例を挙げると、実験対象の一つに AbstractCollection というクラスがあり、それを有向グラフで表現すると11個の連結成分に分割されることになった。そのうちの一つに12個のメンバからなる連結成分があり、強化点も1個だけ含まれていた。そこで各メンバに対し、その有向グラフ上で有向辺の追加・削除を可能な数だけ実施し、複雑さメトリクスによる値の変化量を測定した。その結果を表1に示す。ここでは「size」が弱連結・片連結・強連結に関する強化点である。表1から、size に対して変更を施した場合のメトリクス値変化の平均値は、この中で2番目に大きい（第2位）ことがわかる。

同様に、実験対象におけるすべての強化点について、その平均値順位を集計した。結果を表2に示す¹⁾。

結果として、弱連結に関する強化点メンバの約28%、片連結に関する強化点メンバの約82%、強連結に関する強化点メンバの約79%がそれぞれ上位3位を占めていた。片連結・強連結に関する強化点メンバの個数は、弱連結に関するそれに比べれば決して多い数ではないが、サンプルを無作為に抽出したことを考慮すれば、

表1 AbstractCollection クラスの一部に対する実験結果

メンバ名	平均値	メンバ名	平均値
iterator	248.5	remove	100.5
size	128.3	removeAll	82.8
toString	123.8	retainAll	82.8
toArray ₁	120.2	clear	71.2
contains	110.5	containsAll	65.0
toArray ₂	105.8	isEmpty	41.6

表2 強化点メンバの平均値順位の分布

順位	弱連結での個数	片連結での個数	強連結での個数
1	41	14	7
2	36	7	5
3	19	6	3
4	8	2	1
5	15	1	0
6	17	0	0
7	14	0	0
8	11	0	0
9	10	0	1
10～	169	3	2

強化点メンバが上位に位置しやすいという傾向は一般的なものであると考えられる。これから、片連結あるいは強連結に関して強化点であるようなメンバにおいて、他のメンバとの関係構造を変更するような修正を施すことは、クラス構造の複雑さに大きく影響することになるという傾向が見てとれる。しかしながら、弱連結に関する強化点については、必ずしもそうではないといえる。弱連結ではメンバ間での結合関係の方向性を無視しているため、メンバ間における間接的な依存関係を十分に考慮できないことがその理由として考えられる。

4 まとめ

本論文では、文献[12]で提案されている手法の有効性を確認するため、多数のクラス設計に対して疑似的な変更を施し、それに伴う複雑さメトリクス値の変化を調べるという実験を行った。その結果、片連結ならびに強連結に関して強化点となっているようなクラスメンバにおいて、その関係構造を変更することはクラス全体の構造の複雑さに大きく影響を及ぼすような傾向にあることが確認できた。すなわち、これらはそのクラス構造の複雑さに関して重要な位置にあることが多く、クラス設計の見直しや修正を行う際に特に注意が必要であるといえる。それゆえ、文献[12]で提案されている手法のうち、片連結ならびに強連結に関する部分は一つのクラス構造分析法として有効であると考えられる。

謝辞

本論文について有益なコメントを下された査読者の方に感謝いたします。

参考文献

- [1] J. Rumbaugh, "Object-Oriented Modeling and Design," Prentice Hall, New Jersey, 1991 (羽生田栄一監訳, "オブジェクト指向方法論 OMT," トップラン, 1992)。
- [2] 玉井哲雄, 三嶋良武, 松田茂広, "ソフトウェアのテスト技法," 共立出版, 1988。
- [3] 松本健一, "解説ソフトウェアメトリクス," 日本ファジィ学会誌, vol.10, no.5, pp.796--803, Oct. 1998。
- [4] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," IEEE Trans. Software Eng., vol.20, no.6, pp.476-493, June 1994。
- [5] M. Lorenz, J. Kidd, "Object-Oriented Software Metrics," PTR Prentice Hall, New Jersey, 1994 (宇治邦明 監訳, "オブジェクト指向ソフトウェアメトリクス," プレンティスホール出版, 東京, 1995)。
- [6] H. Aman, H. Yamada, N. T. Noda, T. Yanaru, "A

- graph-based class structural complexity metric and its evaluation," IEICE Trans. Inf. & Syst., vol. E85-D, no.4, pp.674-684, Apr.2002.
- [7] 金恩美, 椿元, 楠本真二, 菊野亨, "C++プログラムに対する複雑さメトリクスの提案と大学環境での実験的評価," 電子情報通信学会論文誌 (D-I), vol.J79-D-I, no.10, pp.729-737, 1996.
- [8] 白田光有, "オブジェクト指向設計のためのソフトウェア評価尺度の考察," 電子情報通信学会論文誌 (D-I), vol.J79-D-I, no.1, pp.9-17, 1996.
- [9] M Takehara, T. Kamiya, S. Kusumoto and K. Inoue, "Empirical evaluation of method complexity for C++ program," IEICE Trans. Inf. & Syst., vol.E83-D, no.8, pp.1698-1700, Aug. 2000.
- [10] S. Yamada and S. Osaki, "Software reliability growth modeling : models and applications," IEEE Trans. Software Eng., vol.SE-11, no.12, pp. 1431-1437, 1985.
- [11] 山下元, "ファジィ情報分析," 共立出版, 1995.
- [12] 阿萬裕久, 矢鳴虎夫, 永松正博, 宮本和典, "ファジィグラフの連結構造分析を用いたオブジェクト指向ソフトウェアにおけるクラス構造の複雑さに関する考察," 日本ファジィ学会誌, vol.11, no.3, pp.521-527, May1999.
- [13] 西田俊夫, 竹田英二, "ファジィ集合とその応用," 森北出版, 1978.
- (2002年9月6日 受付)
(2002年10月27日 採録)
- [問い合わせ先]
〒790-8577 松山市文京町3
愛媛大学 工学部 情報工学科
阿萬 裕久
TEL : 089-927-8525
FAX : 089-927-9973
E-mail : aman@cs.ehime-u.ac.jp

著者紹介

あまん ひろひさ
阿萬 裕久 [正会員]

愛媛大学 工学部 情報工学科

平成13年 九州工業大学大学院工学研究科博士後期課程修了。博士(工学)。同年より愛媛大学工学部助手(情報工学科)。現在に至る。主としてオブジェクト指向ソフトウェアメトリクスに関する研究に従事。日本知能情報ファジィ学会, 電子情報通信学会, 情報処理学会, 日本ソフトウェア科学会, IEEE 各会員。

やなる とらお
矢鳴 虎夫 [正会員]

東亜大学 工学部 システム工学科

昭和44年 九州工業大学大学院工学研究科修士課程修了。同年 九州工業大学工学部助手(電子工学科)。同大学情報処理教育センター講師, 助教授, 同大学情報科学センター助教授を経て, 平成元年 同大学情報工学科教授(知能情報工学科), 平成3年 同大学工学部教授(電気工学科), 平成13年より東亜大学工学部教授(システム工学科), 現在に至る。工学博士。主にソフトコンピューティングに関する研究に従事。日本知能情報ファジィ学会, 電子情報通信学会, 情報処理学会, 人工知能学会, バイオメディカルファジィシステム学会各会員。

やまだ ひろゆき
山田 宏之 [非会員]

愛媛大学 工学部 情報工学科

昭和63年 大阪大学大学院工学研究科博士後期課程修了。工学博士。同年愛媛大学工学部助手(情報工学科)。同大学講師を経て, 平成8年より同大学助教授。主にソフトウェアの要求獲得やソフトウェア保守に関する研究に従事。電子情報通信学会, 情報処理学会, 日本ソフトウェア科学会, 人工知能学会, IEEE, ACM 各会員。

の だまつ たろう
野田松太郎 [非会員]

愛媛大学 工学部 情報工学科

昭和44年 大阪市立大学大学院理学研究科博士課程修了。理学博士。昭和45年愛媛大学工学部助手(電子工学科)。同大学講師, 助教授を経て, 平成3年より同大学教授(情報工学科)。数式処理, 特に数値数式融合計算のアルゴリズムとシステム開発に関する研究に従事。最近では, Web コンピューティングやネットワークセキュリティに関する研究に興味を持っている。著書に『情報処理ハンドブック』(第1編5章「数値的方法」担当, 情報処理学会, オーム社), 『数式処理と数値計算—いかに結合させるか』(情報処理学会誌, 39巻2号)など。数式処理・数値数式融合計算に関する論文多数。平成4年情報処理学会論文賞受賞。日本数式処理学会会長, 情報処理学会, 日本応用数理学会, ACM, IMACS 各会員。

An Empirical Study on the Effectiveness of Class Structural Analysis with Fuzzy Graph

by

Hirohisa AMAN, Torao YANARU, Hiroyuki YAMADA and Matu-Tarow NODA

Abstract :

In object-oriented software development, class design is one of the most significant factors. In order to provide assessments or predictions of class designs, software metrics have been studied in software engineering discipline. When a software metric does not give a reasonable evaluation to a class, we could modify the class structure, and our modification could lead to simplify the class design quality. In order to support such modifications in terms of class structural complexity, an analysis-method with fuzzy graph has been proposed in the past. This is a method for detecting a class member —a method or an attribute— such that it may have a significant impact on the class structural complexity. However, the method effectiveness has never been discussed. The contribution of this paper is to present an empirical study on the effectiveness of that method. In the empirical study, many object-oriented programs (Java classes) are made pseudo modifications, and impacts of those modifications are evaluated by a software metric. The empirical results show that the class members corresponding to the strengthen points in terms of unilaterally connected graph or strongly connected graph could have significant impacts on the class structural complexity. Thus, the above method is useful for class structural analysis.

Keywords : object-orientation, structural complexity, metrics, fuzzy graph

Contact Address : **Hirohisa AMAN**

Department of Computer Science, Faculty of Engineering, Ehime University

3 Bunkyo-cho, Matsuyama, Ehime 790-8577, Japan

TEL : +81-89-927-8525

FAX : +81-89-927-9973

E-mail : aman@cs.ehime-u.ac.jp